Automated Tail Bound Analysis for Probabilistic Recurrence Relations ⋆

Yican Sun[1], Hongfei Fu[2] ⋆⋆, Krishnendu Chatterjee[3], and Amir Kafshdar Goharshady[4]

[1] School of Computer Science, Peking University, China
sycpku@pku.edu.cn
[2] Department of Computer Science, Shanghai Jiao Tong University, China
fuhf@cs.sjtu.edu.cn
[3] IST Austria
krishnendu.chatterjee@ist.ac.at
[4] The Hong Kong University of Science and Technology
goharshady@cse.ust.hk

Abstract. Probabilistic recurrence relations (PRRs) are a standard formalism for describing the runtime of a randomized algorithm. Given a PRR and a time limit κ, we consider the classical concept of tail probability $\Pr[T \geq \kappa]$, i.e., the probability that the randomized runtime T of the PRR exceeds the time limit κ. Our focus is the formal analysis of tail bounds that aims at finding a tight asymptotic upper bound $u \geq \Pr[T \geq \kappa]$ in the time limit κ. To address this problem, the classical and most well-known approach is the cookbook method by Karp (JACM 1994), while other approaches are mostly limited to deriving tail bounds of specific PRRs via involved custom analysis.

In this work, we propose a novel approach for deriving exponentially-decreasing tail bounds (a common type of tail bounds) for PRRs whose preprocessing time and random passed sizes observe discrete or (piecewise) uniform distribution and whose recursive call is either a single procedure call or a divide-and-conquer. We first establish a theoretical approach via Markov's inequality, and then instantiate the theoretical approach with a template-based algorithmic approach via a refined treatment of exponentiation. Experimental evaluation shows that our algorithmic approach is capable of deriving tail bounds that are (i) asymptotically tighter than Karp's method, (ii) match the best-known manually-derived asymptotic tail bound for QuickSelect, and (iii) is only slightly worse (with a $\log \log n$ factor) than the manually-proven optimal asymptotic tail bound for QuickSort. Moreover, our algorithmic approach handles all examples (including realistic PRRs such as QuickSort, QuickSelect, DiameterComputation, etc.) in less than 20 seconds, showing that our approach is efficient in practice.

⋆ Mainland Chinese authors are ordered by contribution, while other authors are ordered alphabetically.

⋆⋆ Corresponding Author

1 Introduction

Probabilistic program verification is a fundamental area in formal verification [4]. It extends the classical (non-probabilistic) program verification by considering randomized computation in a program and hence can be applied to the formal analysis of probabilistic computations such as probabilistic models [16], randomized algorithms [33,11,32,3], etc. In this line of research, verifying the time complexity of probabilistic recurrence relations (PRRs) is an important subject [33,11]. PRRs are a simplified form of recursive probabilistic programs and extend recurrence relations by incorporating randomization such as randomized preprocessing and divide-and-conquer. They are widely used in analyzing the time complexity of randomized algorithms (e.g., QuickSort [18], QuickSelect [19], and DiameterComputation [29, Chapter 9]). Compared with probabilistic programs, PRRs abstract away detailed computational aspects , such as problem-specific divide-and-conquer and data-structure manipulations, and include only key information on the runtime of the underlying randomized algorithm. Hence, PRRs provide a clean model for time-complexity analysis of randomized algorithms and randomized computations in a general sense.

In this work, we focus on the formal analysis of PRRs and consider the fundamental problem of tail bound analysis that aims at bounding the probability that a given PRR does not terminate within a prescribed time limit. In the literature, prominent works on tail bound analysis include the following. First, Karp proposed a classic "cookbook" formula [23] similar to Master Theorem. This method is further improved, extended, and mechanized by follow-up works [15,33,6]. While Karp's method has a clean form and is easy to use and automate, the bounds from the method are known to be not tight (see e.g. [28,17]). Second, the works [28] and resp. [17] performed ad-hoc custom analysis to derive asymptotically tight tail bounds for the PRRs of QuickSort and resp. QuickSelect, respectively. These methods require manual effort and do not have the generality to handle a wide class of PRRs.

From the literature, an algorithmic approach capable of deriving tight tail bounds over a wide class of PRRs is a major unresolved problem. Motivated by this challenge, we have the following contributions to this work:

- Based on Markov's inequality, we propose a novel theoretical approach to derive exponentially-decreasing tail bounds, a common type for many randomized algorithms. We further show that our theoretical approach can always derive an exponentially-decreasing tail bound at least as tight as Karp's method under mild assumptions.
- From our theoretical approach, we propose a template-based algorithmic approach for a wide class of PRRs that have (i) common probability distributions such as (piecewise) uniform distribution and discrete probability distributions and (ii) either a single call or a divide-and-conquer for the form of the recursive call. The technical novelties in our algorithm lie in a refined treatment of the estimation of the exponential term arising from our theoretical approach via integrals, suitable over-approximation, and the monotonicity of the template function.
- Experiments show that our algorithmic approach derives asymptotically tighter tail bounds when compared with Karp's method. Furthermore, the tail bounds derived from our approach match the best-known bound for

QuickSelect [17], and are only slightly worse by a $\log \log n$ factor against the optimal manually-derived bound for QuickSort [28]. Moreover, our algorithm synthesizes each of these tail bounds in less than 20 seconds and is efficient in practice.

A limitation of our approach is that we do not consider the transformation from a realistic implementation of a randomized algorithm into its PRR representation. However, such a transformation would require examining a diversified number of randomization patterns (e.g., randomized divide-and-conquer) in randomized algorithms and thus is an orthogonal direction. In this work, we focus on the tail bound analysis and present a novel approach to address this problem.

2 Preliminaries

Below we present necessary background in probability theory and the tail bound analysis problem we consider.

A *probability space* is a triple $(\Omega, \mathcal{F}, \Pr)$ such that Ω is a non-empty set termed as the *sample space*, $\mathcal{F}$ is a *σ-algebra* over Ω (i.e., a collection of subsets of Ω that contains the empty set $\emptyset$ and is closed under complement and countable union), and $\Pr(\cdot)$ is a *probability measure* on $\mathcal{F}$ (i.e., a function $\mathcal{F} \to [0,1]$ such that $\Pr(\Omega) = 1$ and for every pairwise disjoint set-sequence $A_1, A_2, \ldots$ in $\mathcal{F}$, we have that $\sum_{i \geq 1} \Pr(A_i) = \Pr\left(\bigcup_{i \geq 1} A_i\right)$.

A *random variable* X from a probability space $(\Omega, \mathcal{F}, \Pr)$ is an $\mathcal{F}$-measurable function $X : \Omega \to \mathbb{R}$, i.e., for every $d \in \mathbb{R}$, we have that $\{\omega \in \Omega \mid X(\omega) < d\} \in \mathcal{F}$. We denote $\mathbb{E}[X]$ as its expected value; formally, we have $\mathbb{E}[X] := \int X \, \mathrm{dPr}$. A *discrete probability distribution* (DPD) over a countable set U is a function $\eta : U \to [0,1]$, such that $\sum_{u \in U} \eta(u) = 1$. The *support* of the DPD is defined as $\mathrm{supp}(\eta) := \{u \in U \mid \eta(u) > 0\}$. We abbreviate finite-support DPD as FSDPD.

A *filtration* of probability space $(\Omega, \mathcal{F}, \Pr)$ is an infinite sequence of $\{\mathcal{F}_n\}_{n \geq 0}$ of σ-algebra over Ω such that $\mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq \mathcal{F}$ for every $n \geq 0$. Intuitively, it models the information at the n-th step. A *discrete-time stochastic process* is an infinite sequence $\Gamma = \{X_n\}_{n \geq 0}$ of random variables from the probability space $(\Omega, \mathcal{F}, \Pr)$. The process Γ is *adapted* to a filtration $\{\mathcal{F}_n\}_{n \geq 0}$ if for all $n \geq 0$, X_n is $\mathcal{F}_n$-measurable. Given a filtration $\{\mathcal{F}_n\}_{n \geq 0}$, a *stopping time* is a random variable $\tau : \Omega \to \mathbb{N}$, such that for every $n \geq 0$, $\{\omega \in \Omega \mid \tau(\omega) \leq n\} \in \mathcal{F}_n$.

A discrete-time stochastic process $\Gamma = \{X_n\}_{n \in \mathbb{N}}$ adapted to a filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}}$ is a *martingale* (resp. *supermartingale*) if for every $n \in \mathbb{N}$, $\mathbb{E}[|X_n|] < \infty$ and it holds a.s. that $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n] = X_n$ (resp. $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n] \leq X_n$). Intuitively, a martingale (resp. supermartingale) is a discrete-time stochastic process in which for an observer who has seen the values of $X_0, \ldots, X_n$, the expected value at the next step, i.e. $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n]$, is equal to (resp. no more than) the last observed value X_n. Also, note that in a martingale, the observed values for $X_0, \ldots, X_{n-1}$ do not matter given that $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n] = X_n$. In contrast, in a supermartingale, the only requirement is that $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n] \leq X_n$ and hence $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n]$ may depend on $X_0, \ldots, X_{n-1}$. Also, note that $\mathcal{F}_n$ might contain more information than just the observations of X_i's.

Example 1. Consider the classical gambler's ruin: a gambler starts with Y_0 dollars of money and bets continuously until he loses all of his money. If the bets are

unfair, i.e. the expected value of his money after a bet is less than its expected value before the bet, then the sequence $\{Y_n\}_{n\in\mathbb{N}_0}$ is a supermartingale. In this case, Y_n is the gambler's total money after n bets. On the other hand, if the bets are fair, then $\{Y_n\}_{n\in\mathbb{N}_0}$ is a martingale. See Appendix A for another illustrative example of martingales. □

We refer to standard textbooks (such as [37,7]) for a detailed treatment of all the concepts illustrated above.

2.1 Probabilistic Recurrence Relations

In this work, we focus on probabilistic recurrence relations (PRRs) that describe the runtime behaviour of a single recursive procedure. Instead of having a direct syntax for a PRR, we propose a mini programming language *LRec* that captures a wide class of PRRs that have common probability distributions such as (piecewise) uniform distributions and discrete probability distributions, and whose recursive call consists of either a procedure call or two procedure calls in a divide-and-conquer style. We present the grammar of *LRec* in Figure 1.

$$
\begin{array}{rrl}
\textbf{(PRR)} & \mathsf{proc} & ::= \texttt{def}\ p(n; c_p) = \{\mathsf{comm}\} \\
\textbf{(Command)} & \mathsf{comm} & ::= \texttt{sample}\ v \leftarrow \mathsf{dist}\ \texttt{in}\ \{\mathsf{body}\} \mid \bigoplus_{i=1}^{k} c_i : \mathsf{comm}_i \\
\textbf{(Recursive Body)} & \mathsf{body} & ::= \texttt{pre}(\mathsf{expr}); \texttt{invoke}\ \mathsf{call} \\
\textbf{(Recurive Call)} & \mathsf{call} & ::= p(v); p(\mathsf{size} - v) \mid p(v) \mid p(\mathsf{size} - v) \\
 & & \quad (\text{where } \mathsf{size} \text{ is either } \lfloor \tfrac{n}{b} \rfloor + c \text{ or } \lceil \tfrac{n}{b} \rceil + c) \\
\textbf{(Distribution)} & \mathsf{dist} & ::= \texttt{uniform}(n) \mid \texttt{muniform}(n) \mid \texttt{discrete} \mid \dots \\
\textbf{(Expression)} & \mathsf{expr} & ::= v \mid v^{-1} \mid \ln v \mid n \mid \ln n \mid n^{-1} \mid c \\
 & & \quad \mid \mathsf{expr} + \mathsf{expr} \mid \mathsf{expr} - \mathsf{expr} \mid \mathsf{expr} \times \mathsf{expr}
\end{array}
$$

Fig. 1. The Grammar of *LRec*

In the grammar, we have two positive-integer valued variables n, v which stand for the input size and the sampled value in the randomization of the passed size to the recursive calls of a procedure, respectively. We use $b > 0, c, c_p$ to denote integer constants, and use p to denote the name of the single procedure in the PRR. We consider arithmetic expressions expr as polynomials over $v, v^{-1}, \ln v$ and $n, n^{-1}, \ln n$ (which we call *pseudo-polynomials* in this work) and common probability distributions, including (i) the uniform distribution $\texttt{uniform}(n)$ over $\{0, 1, \dots, n-1\}$, (ii) the piecewise uniform distribution $\texttt{muniform}(n)$ that returns $\max\{i, n-i-1\}$ where i observes the uniform distribution $\texttt{uniform}(n)$, and (iii) any FSDPD (indicated by discrete) whose probabilities and values are constants and pseudo-polynomials, respectively. We also support other piecewise uniform distribution, e.g, the distribution that each $v \in \{0, \dots, n/2\}$ has probability $\frac{2}{3n}$ and each $v \in \{n/2+1, \dots, n-1\}$ has probability $\frac{4}{3n}$.

The nonterminal proc generates the PRR in the form $\texttt{def}\ p(n; c_p) = \{\mathsf{comm}\}$, for which c_p is an integer constant as the threshold of recursion, meaning that the procedure halts immediately when $n < c_p$, and comm is the function body of the procedure. The nonterminal comm generates all statements with one of the two forms as follows.

- A sampling statement (indicated by sample) followed by first a preprocessing time of expr amount (i.e., $\mathtt{pre}(\mathsf{expr})$) and then the recursive calls generated by the nonterminal call.
- A probabilistic choice in the form $\bigoplus_{i=1}^{k} c_i : \mathsf{comm}_i$ where each statement comm_i is executed with probability c_i.

We restrict the recursive calls to be either a single recursive call $p(v)$ or $p(\mathsf{size} - v)$, or a divide-and-conquer composed of two consecutive recursive calls $p(v)$ and $p(\mathsf{size} - v)$, for which we consider a general setting that the relevant overall size size is in the form of the input size n divided by some positive integer b with possibly an offset c. Choosing $b = 1, c = -1$ means the normal situation that the overall size is $n - 1$, i.e., removing one element from the original input.

Given a PRR p, we use $\mathsf{func}(p)$ to represent its function body. We always assume that the given PRR is *well-formed*, i.e., every c_i in a probabilistic choice is within $[0, 1]$ and every random passed size (e.g. $v, \mathsf{size} - v$) falls in $[0, n]$. Below, we present two examples for PRRs.

Example 2 (QuickSelect). Consider the problem of finding the d-th smallest element in an unordered array of n distinct elements. A classical randomized algorithm for solving this problem is QuickSelect [19] with $O(n)$ expected running time. We present the detail of this algorithm in Appendix B. We model the algorithm as the following PRR:

$$\mathtt{def}\ p(n;2) = \{\mathtt{sample}\ v \leftarrow \mathtt{muniform}(n)\ \mathtt{in}\ \{\mathtt{pre}(n);\ \mathtt{invoke}\ p(v);\}\}$$

Here, we use $p(n; 2)$ to represent the number of comparisons performed by QuickSelect over an input of size n, and v is the variable that captures the size of the remaining array that has to be searched recursively. It observes as the value $\max\{i, n-1-i\}$ where the value of i is sampled uniformly from $\{0, \ldots, n-1\}$, we use $\mathtt{muniform}(n)$ to represent this distribution. □

Example 3 (QuickSort). Consider the classical problem of sorting an array of n distinct elements. A well-known randomized algorithm for solving this problem is QuickSort [18]. We model the algorithm as the following PRR. The detail of this algorithm is also presented in Appendix B.

$$\mathtt{def}\ p(n;2) = \{\mathtt{sample}\ v \leftarrow \mathtt{uniform}(n)\ \mathtt{in}\ \{\mathtt{pre}(n);\ \mathtt{invoke}\ p(v); p(n-1-v);\}\}$$

Here, v and $n - 1 - v$ capture the sizes of the two sub-arrays. □

Below we present the semantics of a PRR in a nutshell. We relegate the details of the semantics in Appendix C. Consider a PRR generated by *LRec* with the procedure name p, a *configuration* σ is a pair $\sigma = (comm, \widehat{n})$ where *comm* represents the current statement to be executed and $\widehat{n} \geq c_p$ is the current value for the variable n. A *PRR state* μ is a triple $\langle \sigma, C, \mathbf{K} \rangle$ for which:

- σ is either a configuration, or halt for the termination of the whole PRR.
- $C \geq 0$ records the cumulative preprocessing time so far.
- $\mathbf{K}$ is a stack of configurations that remain to be executed.

We use emp to denote an empty stack, and say that a PRR state $\langle \sigma, C, \mathbf{K} \rangle$ is *final* if $\mathbf{K} = \mathsf{emp}$ and $\sigma = \mathsf{halt}$. Note that in a final PRR state $\langle \mathsf{halt}, C, \mathsf{emp} \rangle$, the value C represents the total execution runtime of the PRR. The semantics of the PRR is defined as a discrete-time Markov chain whose state space is the set of all PRR

states and whose transition function $\mathbf{P}$, where $\mathbf{P}(\mu, \mu')$ is the probability that the next PRR state is μ' given the current PRR state is $\mu = ((\mathit{comm}, \widehat{n}), C, \mathbf{K})$. The probability is determined by the following cases.

- For final PRR states μ, $\mathbf{P}(\mu, \mu) := 1$ and $\mathbf{P}(\mu, \mu') := 0$ for other $\mu' \neq \mu$. This means that the PRR stays at termination once it terminates.
- In the divide-and-conquer case $\mathit{comm} = \texttt{sample}\ v \leftarrow \mathit{dist}\ \texttt{in}\ \{\texttt{pre}(e);\ \texttt{invoke}\ p(v); p(s-v)\}$, we first sample v from the distribution *dist*. Then, with probability $\mathit{dist}(v)$, we accumulate the preprocessing time e into the cumulative processing time C, then we recursively invoke $p(v)$ and push the remaining task $p(s-v)$ into the stack. The probability for the single recursion case is defined analogously. The only difference is that there is no need to push some recursive call into the stack in the single recursion case.
- In the case $\mathit{comm} = \bigoplus_{i=1}^{k} c_i : \mathit{comm}_i$, we have that $\mathbf{P}(\mu, \mu_i) = c_i$ for each $1 \leq i \leq k$ for which we have $\mu_i := ((\mathit{comm}_i, \widehat{n}), C, \mathbf{K})$.

With an initial PRR state $((\mathsf{func}(p), n^*), 0, \mathsf{emp})$ where $n^* \geq c_p$ is the input size, the Markov chain induces a probability space where the sample space is the set of all infinite sequences of PRR states, the σ-algebra is generated by all *cylinder sets* over infinite sequences of PRR states, and the probability measure is uniquely determined by the transition function $\mathbf{P}$. We refer to [4] for details. We use $\Pr_{n^*}$ for the probability measure where $n^* \geq c_p$ is the input size.

We further define the random variable τ such that for any infinite sequence of PRR states $\rho = \mu_0, \mu_1, \ldots, \mu_t, \ldots$ with each $\mu_t = ((\mathit{comm}_t, \widehat{n}_t), C_t, \mathbf{K}_t)$, $\tau(\rho)$ equals the first moment that the sequence reaches a final PRR state, i.e., $\tau(\rho) = \inf\{t \mid \text{the PRR state } \mu_t \text{ is final}\}$, for which $\inf \emptyset = \infty$. We will always ensure that τ is almost-surely finite, i.e., $\Pr_{n^*}(\tau < \infty) = 1$). Note that the random cumulative processing time C_τ in the PRR state $\mu_\tau \in \rho$ is the total execution time of the given PRR.

We formulate the tail bound analysis over PRRs as follows. Given a time limit $\alpha \cdot \kappa(n^*)$ symbolic in the initial input n^* and the coefficient α, the goal of tail bound analysis is to infer an upper bound $u(\alpha, n^*)$ symbolic in n^* and α such that for every input size n^* and plausible value for α, we have that

$$\Pr_{n^*}[C_\tau \geq \alpha \cdot \kappa(n^*)] \leq u(\alpha, n^*). \tag{1}$$

As tails bounds are often evaluated asymptotically, we focus on deriving tight $u(\alpha, n^*)$ when α, n^* are sufficiently large. To compare the magnitude of two tail bounds, we follow the straightforward way that first treats α as a fixed constant and compares the bounds over n^*, and then if the magnitude over n^* is identical, we take a further comparison over the magnitude on the coefficient α.

3 Exponential Tail Bounds via Markov's Inequality

In this section, we demonstrate our theoretical approach for deriving exponentially decreasing tail bounds based on Markov's inequality. Due to the space limitation, We relegate all proof details into Appendix D.

Before illustrating our approach, we first translate a PRR in the language *LRec* with the single procedure p into the canonical form as follows.

$$p(n; c_p) = \texttt{pre}(S(n)); \texttt{invoke}\ p(\mathsf{size}_1(n)); \ldots; p(\mathsf{size}_r(n)) \tag{2}$$

where (i) $S(n)$ is a random value related to the input size n that represents the randomized pre-processing time and observes a probability distribution resulting from a discrete probability choice of piecewise uniform distributions, and (ii) $\texttt{invoke}\ p(\mathsf{size}_1(n)); \ldots; p(\mathsf{size}_r(n))$ is a random statement that is either a single recursive call $p(\mathsf{size}_1(n))$ or a divide-and-conquer $p(\mathsf{size}_1(n)); p(\mathsf{size}_2(n))$ upon the resolution of the randomization. For the latter, we use a random value r (which is either 1 or 2) to represent the number of recursive calls.

The translation can be implemented by a straightforward recursive procedure $\mathsf{Tf}(n, Prog)$ that takes a positive integer n (as the input size) and a statement $Prog$ (generated by the nonterminal comm) to be processed, see Appendix D for details. Note that the procedure $\mathsf{Tf}(n, Prog)$ outputs the *joint* distribution of the random value $S(n)$ and the random recursive call $p(\mathsf{size}_1(n)); \ldots; p(\mathsf{size}_r(n))$ i.e., they may not be independent.

Our theoretical approach then works directly on the canonical form (2). It consists of two major steps to derive an exponentially-decreasing tail bound. In the first step, we apply Markov's inequality and reduce the tail bound analysis problem to the over-approximation of the moment generating function $\mathbb{E}[\exp(t \cdot C_\tau)]$ where C_τ is the cumulative pre-processing time defined previously and $t > 0$ is a scaling factor that aids the derivation of the tail bound. In the second step, we apply Optional Stopping Theorem (a classical theorem in martingale theory) to over-approximate the expected value $\mathbb{E}[\exp(t \cdot C_\tau)]$. Below we fix an PRR with procedure p in the canonical form (2), and a time limit $\alpha \cdot \kappa(n^*)$.

Our first step applies Markov's inequality. Our approach relies on the well-known exponential form of Markov's inequality below.

Theorem 1. *For every random variable X and any scaling factor $t > 0$, we have that* $\Pr[X \geq d] \leq \mathbb{E}[\exp(t \cdot X)] / \exp(t \cdot d)$.

The detailed application of Markov's inequality to tail bound analysis requires to choose a scaling factor $t := t(\alpha, n)$ symbolic in α and n. After choosing the scaling factor, Markov's inequality gives the following tail bound:

$$\Pr[C_\tau \geq \alpha \cdot \kappa(n^*)] \leq \mathbb{E}[\exp(t(\alpha, n^*) \cdot C_\tau)] / \exp(t(\alpha, n^*) \cdot \alpha \cdot \kappa(n^*)). \tag{3}$$

The role of the scaling factor $t(\alpha, n^*)$ is to scale the exponent in the term $\exp(\kappa(\alpha, n^*))$, and this is in many cases necessary as a tail bound may not be exponentially decreasing directly in the time limit $\alpha \cdot \kappa(n^*)$.

An unsolved part in the tail bound above is the estimation of the expected value $\mathbb{E}[\exp(t(\alpha, n^*) \cdot C_\tau)]$. Our second step over-approximates the expected value $\mathbb{E}[\exp(t(\alpha, n^*) \cdot C_\tau)]$. To achieve this goal, we impose a constraint on the scaling factor $t(\alpha, n)$ and an extra function $f(\alpha, n)$ and show that once the constraint is fulfilled, then one can derive an upper bound for $\mathbb{E}[\exp(t(\alpha, n^*) \cdot C_\tau)]$ from $t(\alpha, n)$ and $f(\alpha, n)$. The theorem is proved via Optional Stopping Theorem.

Theorem 2. *Suppose we have functions $t, f : [0,\infty) \times \mathbb{N} \to [0,\infty)$ such that*

$$\mathbb{E}[\exp(t(\alpha,n)\cdot \mathsf{Ex}(n \mid f))] \leq \exp(t(\alpha,n)\cdot f(\alpha,n)) \tag{4}$$

for all sufficiently large $\alpha, n^* > 0$ and all $c_p \leq n \leq n^*$, where

$$\mathsf{Ex}(n \mid f) := S(n) + \textstyle\sum_{i=1}^{r} f(\alpha, \mathsf{size}_i(n)).$$

Then for $t_*(\alpha,n^*) := \min_{c_p \leq n \leq n^*} t(\alpha,n)$, we have that

$$\mathbb{E}[\exp(t_*(\alpha,n^*)\cdot C_\tau)] \leq \mathbb{E}[\exp(t_*(\alpha,n^*)\cdot f(\alpha,n^*))].$$

Thus, we obtain the upper bound $u(\alpha,n^*) := \exp(t_*(\alpha,n^*)\cdot(f(\alpha,n^*) - \alpha\cdot\kappa(n^*)))$ for the tail bound in (1).

Proof sketch. We fix a procedure p, and some sufficiently large α and n^*. In general, we apply the martingale theory to prove this theorem. To construct a martingale, we need to make two preparations.

First, by the convexity of $\exp(\cdot)$, substituting $t(\alpha,n)$ with $t_*(\alpha,n^*)$ in (4) does not affect the validity of (4).

Second, given an infinite sequence of the PRR states $\rho = \mu_0, \mu_1, \ldots$ in the sample space, we consider the subsequence $\rho' = \mu'_0, \mu'_1, \ldots$ as follows, where we represent μ'_i as $((\mathsf{func}(p), \hat{n}'_i), C'_i, \mathbf{K}'_i)$. It only contains states that are either final or at the entry of p, i.e., $comm = \mathsf{func}(p)$. We define $\tau' := \inf\{t : \mu'_t \text{ is final}\}$, then it is straightforward that $C'_{\tau'} = C_\tau$. We observe that μ'_{i+1} represents the recursive calls of μ'_i. Thus, we can characterize the conditional distribution $\mu'_{i+1} \mid \mu_i$ by the transformation function $\mathsf{Tf}(\hat{n}, \mathsf{func}(p))$ that models the distribution of randomized pre-processing costs and recursive calls. In detail, $\mu'_{i+1} \mid \mu_i$ is the distribution as follows.

- We first draw $(S, \mathsf{size}_1, \mathsf{size}_2, r)$ from $\mathsf{Tf}(\hat{n}'_i, \mathsf{func}(p))$.
- We accumulate S into the global cost. If there is a single recursion ($r = 1$), we invoke this sub-procedure. If there are two recursive calls, we push the second call $p(\mathsf{size}_2)$ into the stack and invoke the first one $p(\mathsf{size}_1)$.

Now we construct the super-martingale as follows. For each $i \geq 0$, we first represent the stack $\mathbf{K}'_i$ for μ'_i as $(\mathsf{func}(p), \mathsf{s}_{i,1}) \cdots (\mathsf{func}(p), \mathsf{s}_{i,q_i})$, where q_i represent the stack size. We prove that another stochastic process $y_0, y_1, \ldots$ that forms a super-martingale, where $y_i := \exp\left(t_*(\alpha,n^*)\cdot\left(C'_i + f(\alpha,\hat{n}'_i) + \sum_{j=1}^{q_i} f(\alpha,\mathsf{s}_{i,j})\right)\right)$. Note that $y_0 = \exp(t_*(\alpha,n^*)\cdot f(\alpha,n^*))$, and $y_{\tau'} = \exp(t_*(\alpha,n^*)\cdot C'_{\tau'}) = \exp(t_*(\alpha,n^*)\cdot C_\tau)$. Thus we informally have that $\mathbb{E}[\exp(t_*(\alpha,n^*)\cdot C_\tau)] = \mathbb{E}[y_{\tau'}] \leq \mathbb{E}[y_0] = \exp(t_*(\alpha,n^*)\cdot f(\alpha,n^*))$ and the theorem follows. □

It is natural to ask whether our theoretical approach can always find an exponential-decreasing tail bound over PRRs. We answer this question by showing that under a difference boundedness and a monotone condition, the answer is yes. We first present the difference boundedness condition (A1) and the monotone condition (A2) for a PRR Δ in the canonical form (2) as follows.

(A1) Δ is *difference-bounded* if there exist two real constants $M' \leq M$, such that for every $n \geq c_p$, and every possible value $(V, s_1, \ldots, s_k)$ in the support of the probability distribution $\mathsf{Tf}(n, \mathsf{func}(p))$, we have that

$$M' \cdot \mathbb{E}[S(n)] \leq V + (\textstyle\sum_{i=1}^{k} \mathbb{E}[p(s_i)]) - \mathbb{E}[p(n)] \leq M \cdot \mathbb{E}[S(n)].$$

(A2) Δ is *expected non-decreasing* if $\mathbb{E}[S(n)]$ does not decrease as n increases.

In other words, (A1) says that for any possible concrete pre-processing time V and passed sizes $s_1, \ldots, s_k$, the difference between the expected runtime before and after the recursive call is bounded by the magnitude of the expected pre-processing time. (A2) simply specifies that the expected pre-processing time be monotonically non-decreasing. These conditions are fulfilled by most randomized algorithms, see Appendix G for details.

With the conditions (A1) and (A2), our theoretical approach guarantees a tail bound that is exponentially decreasing in the coefficient α and the ratio $\mathbb{E}[p(n^*)]/\mathbb{E}[S(n^*)]$. The theorem statement is as follows.

Theorem 3. *Let Δ be a PRR in the canonical form (2). If Δ satisfies (A1) and (A2), then for any function $w : [1, \infty) \to (1, \infty)$, the functions f, t given by*

$$f(\alpha, n) := w(\alpha) \cdot \mathbb{E}[p(n)] \quad \textit{and} \qquad t(\alpha, n) := \frac{\lambda(\alpha)}{\mathbb{E}[S(n)]}$$
$$\textit{with} \qquad \lambda(\alpha) := \frac{8(w(\alpha)-1)}{w(\alpha)^2(M_2-M_1)^2}$$

fulfill the constraint (4) in Theorem 2. Furthermore, by choosing $w(\alpha) := \frac{2\alpha}{1+\alpha}$ in the functions f, t above and $\kappa(\alpha, n^*) := \alpha \cdot \mathbb{E}[p(n^*)]$, one obtains the tail bound

$$\Pr[C_\tau \geq \alpha \mathbb{E}[p(n^*)]] \leq \exp\left(-\tfrac{2(\alpha-1)^2}{\alpha(M_2-M_1)^2} \cdot \tfrac{\mathbb{E}[p(n^*)]}{\mathbb{E}[S(n^*)]}\right).$$

Proof sketch. We first rephrase the constraint (4) as

$$\mathbb{E}\left[\exp\left(t(\alpha, n) \cdot \left(S(n) + \textstyle\sum_{i=1}^{r} f(\alpha, \mathsf{size}_i(n)) - f(\alpha, n)\right)\right)\right] \leq 1$$

Then we focus on the exponent in the $\exp(\cdot)$, by (A1), the exponent is a bounded random variable. By further calculating its expectation and applying Hoeffiding's Lemma [20], we obtain the theorem above. □

Note that since $\mathbb{E}[p(n)] \geq \mathbb{E}[S(n)]$ when $n \geq c_p$, the tail bound is at least exponentially-decreasing with respect to the coefficient α. This implies that our theoretical approach derives tail bounds that are at least as tight as Karp's method when (A1) and (A2) holds. When $\mathbb{E}[p(n)]$ is of a strictly greater magnitude than $\mathbb{E}[S(n)]$, our approach derives asymptotically tighter bounds.

Below, we apply the theorem above to prove tail bounds for Quickselect (Example 2) and Quicksort (Example 3).

Example 4. For QuickSelect, its canonical form is $p(n; 2) = n + p(\mathsf{size}_1(n))$, where $\mathsf{size}_1(n)$ observes as $\texttt{muniform}(n)$. Solving the recurrence relation, we obtain that $\mathbb{E}[p(n)] = 4 \cdot n$. We further find that this PRR satisfies (A1) with two constants $M' = -1, M = 1$. Note that the PRR satisfies (A2) obviously. Hence, we apply Theorem 3 and derive the tail bound for every sufficiently large α:

$$\Pr[C_\tau \geq 4 \cdot \alpha \cdot n^*] \leq \exp\left(-\tfrac{2(\alpha-1)^2}{\alpha}\right).$$

On the other hand, Karp's cookbook has the tail bound

$$\Pr[C_\tau \geq 4 \cdot \alpha \cdot n^*] \leq \exp\left(1.15 - 1.12 \cdot \alpha\right).$$

Our bound is asymptotically the same as Karp's but has a better coefficient. □

Example 5. For QuickSort, its canonical form is $p(n;2) = n + p(\mathsf{size}_1(n)) + p(\mathsf{size}_2(n))$, where $\mathsf{size}_1(n)$ observes as $\texttt{muniform}(n)$ and $\mathsf{size}_2(n) = n-1-\mathsf{size}_1(n)$. Similar to the example above, we first calculate $\mathbb{E}[p(n)] = 2 \cdot n \cdot \ln n$. Note that this PRR also satisfies two assumptions above with two constants $M' = -2\log 2, M = 1$. Hence, for every sufficiently large α, we can derive the tail bound as follows:

$$Pr[C_\tau \geq 2 \cdot \alpha \cdot n^* \cdot \ln n^*] \leq \exp\left(-\tfrac{0.7(\alpha-1)^2}{\alpha} \cdot \ln n^*\right).$$

On the other hand, Karp's cookbook has the tail bound

$$Pr[C_\tau \geq 2 \cdot \alpha \cdot n^* \cdot \ln n^*] \leq \exp\left(-\alpha + 0.5\right).$$

Note that our tail bound is tighter than Karp's with a $\ln n$ factor. □

From the generality of Markov's inequality, our theoretical approach can handle to general PRRs with three or more sub-procedure calls. However, the tail bounds derived from Theorem 3 is still not tight since the theorem only uses the expectation and bound of the given distribution. For example, for QuickSelect, the tightest known bound $\exp(-\Theta(\alpha \cdot \ln \alpha))$ [17], is tighter than that derived from Theorem 3. In the next section, we present an algorithmic approach that fully utilizes the distribution information and derives tight tail bounds that can match [17].

4 An Algorithmic Approach

In this section, we demonstrate an algorithmic implementation for our theoretical approach (Theorem 2). Our algorithm synthesizes the functions t, f through template and a refined estimation on the exponential terms from the inequality (4). The estimation is via integration and the monotonicity of the template. Below we fix a PRR $p(n; c_p)$ in the canonical form (2) and a time limit $\alpha \cdot \kappa(n^*)$.

Recall that to apply Theorem 2, one needs to find functions t, f that satisfy the constraint (4). Thus, the first step of our algorithm is to have pseudo-monomial template for $f(\alpha, n)$ and $t(\alpha, n)$ in the following form:

$$f(\alpha, n) := c_f \cdot \alpha^{p_f} \cdot \ln^{q_f} \alpha \cdot n^{u_f} \cdot \ln^{v_f} n \tag{5}$$

$$t(\alpha, n) := c_t \cdot \alpha^{p_t} \cdot \ln^{q_t} \alpha \cdot n^{u_t} \cdot \ln^{v_t} n \tag{6}$$

In the template, we have $p_f, q_f, u_f, v_f, p_t, q_t, u_t, v_t$ are given integers, and $c_f, c_t > 0$ are unknown positive coefficients to be solved. For several compatibility reasons (see Proposition 1 and 2 in the following), we require that $u_f, v_f \geq 0$ and $u_t, v_t \leq 0$. We say that the concrete values $\overline{c_f}, \overline{c_t}$ for the unknown coefficients $c_f, c_t > 0$ are *valid* if the concrete functions $\overline{f}, \overline{t}$ obtained by substituting $\overline{c_f}, \overline{c_t}$ for c_f, c_t in the template (5) and (6) satisfy the constraint (4) for every sufficiently large $\alpha, n^* \geq 0$ and all $c_p \leq n \leq n^*$.

We consider the pseudo-polynomial template since the runtime behavior of randomized algorithms can be mostly captured by pseudo-polynomials. We choose monomial templates since our interest is the asymptotic magnitude of the tail bound. Thus, only the monomial with the highest degrees matter.

Our algorithm searches the values for $p_f, q_f, u_f, v_f, p_t, q_t, u_t, v_t$ by an enumeration within a bounded range $\{-B, \ldots, B\}$, where B is a manually specified positive integer. To avoid exhaustive enumeration, we use the following proposition to prune the search space.

Proposition 1. *Suppose that we have functions $t, f : [0,\infty) \times \mathbb{N} \to [0,\infty)$ that fulfill the constraint (4). Then it holds that (i) $(p_f, q_f) \leq (1,0)$ and $(p_t, q_t) \geq (-1,0)$, and (ii) $f(\alpha, n) = \Omega(\mathbb{E}[p(n)])$, $f(\alpha, n) = O(\kappa(n))$ and $t(\alpha, n) = \Omega(\kappa(n)^{-1})$ for any fixed $\alpha > 0$, where we write $(a,b) \leq (c,d)$ for the lexicographic order, i.e., $(a \leq c) \wedge (a = c \to b \leq d)$.*

Proof. Except for the constraint that $f(\alpha, n) = \Omega(\mathbb{E}[p(n)])$, the other constraints simply ensure that the tail bound is exponentially-decreasing. To see why $f(\alpha, n) = \Omega(\mathbb{E}[p(n)])$, we apply Jensen's inequality [31] to (4) and obtain $f(n) \geq \mathbb{E}[\mathsf{Ex}(n|f)] = \mathbb{E}[S(n) + \sum_{i=1}^{r} f(\mathsf{size}_i(n))]$. Then we imitate the proof of Theorem 2 and derive that $f(n) \geq \mathbb{E}[p(n)]$. □

Proposition 1 shows that it suffices to consider (i) the choice of u_f, v_f that makes the magnitude of f to be within $\mathbb{E}[p(n)]$ and $\kappa(n)$, (ii) the choice of u_t, v_t that makes the magnitude of t^{-1} within $\kappa(n)$, and (iii) the choice of p_f, q_f, p_t, q_t that fulfills $(p_f, q_f) \leq (1,0), (p_t, q_t) \geq (-1,0)$. Note that an over-approximation of $\mathbb{E}[p(n)]$ can be either obtained manually or derived from automated approaches [11].

Example 6. Consider the quickselect example (Example 2), suppose we are interested in the tail bound $\Pr[C_\tau \geq \alpha \cdot n]$, and we enumerate the eight integers in the template from -1 to 1. Since $\mathbb{E}[p(n)] = 4 \cdot n$, by the proposition above, we must have that $(u_f, v_f) = (1,0)$, $(u_t, v_t) \geq (-1,0)$, $(p_t, q_t) \geq (-1,0)$, $(p_f, q_f) \leq (1,0)$. This reduces the number of choices for the template from 1296 to 128. A choice is $f(\alpha, n) := c_f \cdot \alpha \cdot (\ln \alpha)^{-1} \cdot n$ and $t(\alpha, n) := c_t \cdot \ln \alpha \cdot n^{-1}$. □

In the second step, our algorithm solves the unknown coefficients c_t, c_f in the template. Once they are solved, our algorithm applies Theorem 2 to obtain the tail bound. In detail, our algorithm computes $t_*(\alpha, n^*)$ as the minimum of $t(\alpha, n)$ over $c_p \leq n \leq n^*$, and by $u_t, v_t \leq 0$, $t_*(\alpha, n^*)$ is simply $t(\alpha, n^*)$, so that we obtain the tail bound $u(\alpha, n^*) = \exp(t(\alpha, n^*) \cdot (f(\alpha, n^*) - \alpha \cdot \kappa(n^*)))$.

Example 7. Continue with Example 6. Suppose we have successfully found that $\overline{c_f} = 2, \overline{c_t} = 1$ is a valid concrete choice for the unknown coefficients in the template. Then $t_*(\alpha, n^*)$ is $t(\alpha, n^*) = \ln \alpha \cdot (n^*)^{-1}$, and we have the tail bound $u(\alpha, n^*) = \exp(2 \cdot \alpha - \alpha \cdot \ln \alpha)$, which has better magnitude than the tail bound by Karp's method and our Theorem 3 (See Example 4). □

Our algorithm follows the guess-and-check paradigm. The guess procedure explores possible values $\overline{c_f}, \overline{c_t}$ for c_f, c_t and invokes the check procedure to verify whether the current choice is valid. Below we present the guess procedure in Section 4.1, and the check procedure in Section 4.2.

4.1 The Guess Procedure Guess(f, t)

The pseudocode for our guess procedure is given in Algorithm 1. In detail, the guess procedure $\mathsf{Guess}(f, t)$ first receives a positive integer M as the doubling

and halving number (Line 1), then iteratively enumerates possible values for the unknown coefficients c_f and c_t by doubling and halving for M times (Line 3 – Line 4), and finally calls the check procedure (Line 5). The guess procedure is justified by the following theorem (proved in Appendix E).

Theorem 4. *Given the template for $f(\alpha, n)$ and $t(\alpha, n)$ as in (5) and (6), if $\overline{c_f}, \overline{c_t}$ are valid choices, then (i) for every $k > 1$, $k \cdot \overline{c_f}, \overline{c_t}$ remains to be valid, and (ii) for every $0 < k < 1$, $\overline{c_f}, k \cdot \overline{c_t}$ remains to be valid.*

Algorithm 1: The Guess Procedure

```
   Input  : Template for f(α, n) and
            t(α, n) as in (5) and (6)
   Output: values c_f, c_t > 0 for (5) and (6)
1  Parameter: M for the maximum steps
     of doubling and halving.
2  Procedure Guess(f, t):
3    for c_f := 1/2, 1, 2, ..., 2^{M-1} do
4      for c_t := 1, 2^{-1}, ..., 2^{-M} do
5        if CheckCond(c_f, c_t) then
6          Return (c_f, c_t)
```

Theorem 4 above implies that if the check procedure is sound and complete (i.e., CheckCond always terminates and $\overline{c_f}, \overline{c_t}$ fulfills the constraint (4) iff CheckCond$(\overline{c_f}, \overline{c_t})$ returns true), then the guess procedure guarantees to find a solution $\overline{c_f}, \overline{c_t}$ (if it exists) when the parameter M is large enough.

Example 8. Continued with Example 6, suppose $M = 2$, we enumerate $\overline{c_f}$ from $\{\frac{1}{2}, 1, 2\}$, and $\overline{c_t}$ from $\{1, \frac{1}{2}, \frac{1}{4}\}$. We try every possible combination, and we find that CheckCond$(2, 1)$ returns true. Thus, we return $(2, 1)$ as the result. In Section 4.2, we will show how to conclude that CheckCond$(2, 1)$ is true. □

4.2 The Check Procedure CheckCond$(\overline{c_f}, \overline{c_t})$

The check procedure takes as input the concrete values $\overline{c_f}, \overline{c_t}$ for the unknown coefficients in the template, and outputs whether they are valid. It is the most involved part in our algorithm due to the difficulty to tackle the validity of the constraint (4) that involves the composition of polynomials, exponentiation and logarithms. The existence of a sound and complete decision procedure for such validity is extremely difficult and is a long-standing open problem [1,36]. Hence, we propose a sound procedure that may reject valid values $\overline{c_f}, \overline{c_t}$.

To circumvent the difficulty from exponentiation and logarithm, the check procedure first strengthens the constraint (4) into a canonical constraint, and then applies a decision algorithm that is sound and complete up to any additive error. We also discuss possible extensions for the check procedure in Remark 1.

Below we fix a PRR with procedure p in the canonical form (2) and illustrate the details of the algorithm. We first present the canonical constraint $Q(\alpha, n)$ and how to decide the canonical constraint. The constraint is given by

$$Q(\alpha, n) := \left[\sum_{i=1}^{k} \gamma_i \cdot \exp(f_i(\alpha) + g_i(n)) \leq 1\right] \tag{7}$$

subject to:

(C1) For each $1 \leq i \leq k$, we have $\gamma_i > 0$ is a positive constant, $f_i(\alpha)$ is a pseudo-polynomial in α, and $g_i(n)$ is a pseudo-polynomial in n.

(C2) For each $1 \leq i \leq k$, the exponents for n and $\ln n$ in $g_i(n)$ are non-negative.

We use $Q_L(\alpha, n)$ to represent the LHS term in (7). We concern about the property that whether $Q(\alpha, n)$ holds for all sufficiently large α and all $n \geq c_p$, and show that this property can be checked by the following algorithm *Decide* up to any additive error. Below we provide a sketch of this algorithm. The full details are relegated to Appendix E.

— First, the algorithm computes an integer L such that instead of considering every $n \geq c_p$, it suffices to consider every $c_p \leq n \leq L$.

— Second, for every integer $c_p \leq k \leq L$, the algorithm substitute n with k to eliminate the variable n in the exponent of the canonical constraint (7). Then, each exponent $f_i(\alpha) + g_i(k)$ becomes a pseudo-polynomial in α. Since we only consider the situation when α is sufficiently large, the algorithm computes the limit R_k for $Q_L(\alpha, k)$ as $\alpha \to \infty$. The algorithm takes the decision based on the comparison between the limit R_k and 1. □

Algorithm *Decide* has the soundness and completeness up to any additive error, as is illustrated by the following theorem. The proof is conducted directly via the definition of limit and is relegated to Appendix E.

Theorem 5. *Algorithm Decide has the following properties:*
- *(Completeness) If $Q(\alpha, n)$ does not hold for infinitely many α and some $n \geq c_p$, then the algorithm returns false.*
- *(Soundness) For every $\varepsilon > 0$, we have that if $Q_L(\alpha, n) \leq 1 - \varepsilon$ for all sufficiently large α and all $n \geq c_p$, then the algorithm returns true.*

Then we show how to strengthen the constraint (4) into the canonical constraint (7), so that Algorithm *Decide* applies. We rephrase (4) as

$$\mathbb{E}\left[\exp(t(\alpha, n) \cdot (S(n) + \textstyle\sum_{i=1}^{r} f(\alpha, \mathsf{size}_i(n)) - f(\alpha, n))\right] \leq 1 \tag{8}$$

and consider two functions $\overline{f}, \underline{t}$ obtained by substituting the concrete values $\overline{c_f}, \overline{c_f}$ for unknown coefficients into the template (5) and (6). We observe that the joint-distribution of the random quantities $S(n), r \in \{1, 2\}$ and $\mathsf{size}_1(n), \ldots, \mathsf{size}_r(n)$ in the canonical form (2) over PRRs can be described by several probabilistic branches $\{c_1 : B_1, \ldots, c_k : B_k\}$, which corresponds to the probabilistic choice commands in the PRR. Each probabilistic branch B_i has a constant probability c_i, a deterministic pre-processing time $S_i(n)$, a fixed number of subprocedure calls r_i, and a probability distribution for the variable v. The strengthening first handles each probabilistic branch, and then combines the strengthening results of every branch into a single canonical constraint.

The strengthening of each branch is an application of a set of rewriting rules. Intuitively, each rewriting step over-approximates and simplifies the expectation term in the LHS of (8). Through multiple steps of rewriting, we eventually obtain the final canonical constraint. Below we present the details of the strengthening for a probabilistic branch by distinguishing the single recursion and the divide-and-conquer case.

Consider the single recursion case $r = 1$ where a probabilistic branch has deterministic pre-processing time $S(n)$, distribution dist for the variable v and passed size $H(v, n)$ for the recursive call. We have a case analysis on the distribution dist as follows.

— *Case I*: dist is a FSDPD $\texttt{discrete}\{c'_1 : \mathsf{expr}_1, \ldots, c'_k : \mathsf{expr}_k\}$, where v observes as expr_i with probability c'_i. Then the expectation in (8) is exactly:

$$\textstyle\sum_{i=1}^{k} c'_i \cdot \exp\left(t(\alpha,n)\cdot S(n) + t(\alpha,n)\cdot f(\alpha, H(\mathsf{expr}_i, n)) - t(\alpha,n)\cdot f(\alpha,n)\right)$$

Thus it suffices to over-approximate the exponent $X_i(\alpha,n) := t(\alpha,n)\cdot S(n) + t(\alpha,n)\cdot f(\alpha,H(\mathsf{expr}_i,n)) - t(\alpha,n)\cdot f(n)$ into the form subject to (C1)–(C2). For this purpose, our strengthening repeatedly applies the following rewriting rules (R1)–(R4) for which $0 < a < 1$ and $b > 0$:

(R1) $f(\alpha, H(\mathsf{expr}_i, n)) \le f(\alpha, n)$

(R2) $\ln(an-b) \le \ln n + \ln a \quad \ln(an+b) \le \ln n + \ln(\min\{1, a + \frac{b}{c_p}\})$

(R3) $0 \le n^{-1} \le c_p^{-1} \quad 0 \le \ln^{-1} n \le \ln^{-1} c_p$ (R4) $\lfloor \frac{n}{b} \rfloor \le \frac{n}{b} \quad \lceil \frac{n}{b} \rceil \le \frac{n}{b} + \frac{b-1}{b}$

(R1) follows from the well-formedness $0 \le H(\mathsf{size}_i, n) \le n$ and the monotonicity of $f(\alpha,n)$ with respect to n. (R2)–(R4) are straightforward. Intuitively, (R1) can be used to cancel the term $f(\alpha, H(\mathsf{size}_i,n)) - f(\alpha,n)$, (R2) simplifies the sub-expression in ln, (R3) is used to remove floors and ceils, and (R4) to remove n^{-c} and $\ln^{-c} n$ to satisfy the restriction (C2) of the canonical constraint. To apply these rules, we consider two strategies below.

(S1-D) Apply (R1) and over-approximate $X_i(\alpha,n)$ as $t(\alpha,n)\cdot S(n)$. Then, we repeatedly apply (R3) to remove terms n^{-c} and $\ln^{-c} n$.

(S2-D) Substitute f and t with the concrete functions $\overline{f}, \overline{t}$ and expand $H(\mathsf{expr}_i, n)$. Then we first apply (R4) to remove all floors and ceils, and repeatedly apply (R2) to replace all occurrences of $\ln(an+b)$ with $\ln n + \ln C$ for some constant C. By the previous replacement, the whole term $X_i(\alpha,n)$ will be over-approximated as a pseudo-polynomial over α and n. Finally, we eagerly apply (R3) to remove all terms n^{-c} and $\ln^{-c} n$.

Our algorithm first tries to apply (S2-D), if it fails to derive a canonical constraint, then we apply the alternative (S1-D) to the original constraint. If both the strategies fails, we report failure and exit the check procedure.

Example 9. Suppose v observes as $\{0.5 : n-1, 0.5 : n-2\}, S(n) := \ln n, t(\alpha,n) := \frac{\ln\alpha}{\ln n}, f(\alpha,n) := 4\cdot\frac{\alpha}{\ln\alpha}\cdot n\cdot\ln n, H(v,n) := v$. We consider applying both strategies to the first term $\mathsf{expr}_1 := n-1$ and $X_1(\alpha,n) := t(\alpha,n)\cdot(S(n) + f(\alpha,n-1) - f(\alpha,n))$. If we apply (S1-D) to X_1, it will be approximated as $\exp(\ln\alpha)$. If we apply (S2-D) to X_1, it will be first over-approximated as $\frac{\ln\alpha}{\ln n}\cdot(\ln n + 4\cdot\frac{\alpha}{\ln\alpha}\cdot v\cdot\ln n - 4\cdot\frac{\alpha}{\ln\alpha}\cdot n\cdot\ln n)$, then we substitute $v = n-1$ and derive the final result $\exp(\ln\alpha - 4\cdot\alpha)$. Hence, both the strategies succeed. □

— *Case II*: dist is $\texttt{uniform}(n)$ or $\texttt{muniform}(n)$. Note that $H(v,n)$ is linear with respect to v, thus $H(v,n)$ is a bijection over v for every fixed n. Hence, if v observes as $\texttt{uniform}(n)$, then

$$\mathbb{E}[\exp(t(\alpha,n)\cdot f(\alpha, H(v,n)))] \le \tfrac{1}{n}\textstyle\sum_{v=0}^{n-1}\exp(t(\alpha,n)\cdot f(\alpha,v)) \tag{9}$$

If v observes as $\texttt{muniform}(n)$, a similar inequality holds by replacing $\frac{1}{n}$ with $\frac{2}{n}$.Since $f(\alpha,v)$ is a non-decreasing function with respect to v, we further over-approximate the summation in (9) by the integral $\int_0^n \exp(t(\alpha,n)\cdot f(\alpha,v))\mathrm{d}v$.

Example 10. Continue with Example 8, we need to check $\overline{t}(\alpha, n) = \frac{\ln \alpha}{n}$ and $\overline{f}(\alpha, n) = \frac{2\cdot\alpha}{\ln \alpha} \cdot n$. By the inequality (9), we expand the constraint (8) into $\frac{2}{n} \cdot \exp(\ln \alpha - 2\cdot\alpha) \cdot \sum_{v=0}^{n-1} \exp(\frac{2\cdot\alpha\cdot i}{n})$. By integration, it is further over-approximated as $\frac{2}{n} \cdot \exp(\ln \alpha - 2 \cdot \alpha) \cdot \int_0^n \exp(\frac{2\cdot\alpha\cdot v}{n})\mathrm{d}v$. □

Note that we still need to resolve the integration of an exponential function whose exponent is a pseudo-monomial over α, n, v. Below we denote by d_v the degree on the variable v and by ℓ_v the degree of $\ln v$. We first list the situations where the integral can be computed exactly.

- If $(d_v, \ell_v) = (1, 0)$, then the exponent could be expressed as $W(\alpha, n)\cdot v$,where $W(\alpha, n)$ is a pseudo-monomial over α and n. We can compute the integral as $\frac{\exp(n\cdot W(\alpha,n))-1}{W(\alpha,n)}$ and over-approximate it as $\frac{\exp(n\cdot W(\alpha,n))}{W(\alpha,n)}$ by removing -1 in the numerator.
- If $(d_v, \ell_v) = (0, 1)$, then the exponent is of the form $W(\alpha, n) \cdot \ln v$. We follow a similar procedure with the case above and obtain the over-approximation $\frac{n\cdot\exp(\ln n\cdot W(\alpha,n))}{W(\alpha,n)}$.
- If $(d_v, \ell_v) = (0, 0)$, then the result is trivially $n \cdot \exp(W(\alpha, n))$.

Then we handle the situation where the exact computation of the integral is infeasible. In this situation, the strengthening further over-approximates the integral into simpler forms by first replacing $\ln v$ with $\ln n$, and then replacing v with n to reduce the degrees ℓ_v and d_v. Eventually, the exponent in the integral bows down to one of the three situations (where the integral can be computed exactly) above, and the strengthening returns the exact value of the integral.

Example 11. Continue with Example 10. We express the exponent as $\frac{2\cdot\alpha}{n} \cdot v$. Thus, we can plug $\frac{2\cdot\alpha}{n}$ into $W(\alpha, n)$ and obtain the integration result $\frac{\exp(2\cdot\alpha)}{2\cdot\alpha/n}$. Furthermore, we can simplify the formula in Example 10 as $\frac{\exp(\ln \alpha)}{\alpha}$. □

To finish the strengthening for the single recursion case, we move the term $\frac{1}{n}$ (or $\frac{2}{n}$) that comes from the `uniform` (or `muniform`) distribution and the coefficient term $W(\alpha, n)$ into the exponent. If we move these terms directly, it may produce $\ln \ln n$ and $\ln \ln \alpha$ that comes from taking the logarithm of $\ln n$ and $\ln \alpha$. Hence, we first apply $\ln c_p \le \ln n \le n$ and $1 \le \ln \alpha \le \alpha$ to remove all terms $\ln n$ and $\ln \alpha$ outside the exponent (e.g., $\frac{\ln \alpha}{\ln n}$ is over-approximated as $\frac{\alpha}{\ln c_p}$). After the over-approximation, the terms outside the exponentiation form a polynomial over α and n, we can trivially move these terms into the exponent by taking the logarithm. Finally, we apply (R4) in Case I to remove n^{-c} and $\ln^{-c} n$. If we fail to obtain the canonical constraint, the strengthening reports failure.

Example 12. Continue with Example 11, we move the term α into the exponentiation and simplify the over-approximation result as $\exp(\ln \alpha - \ln \alpha) = 1$. As a result, we over-approximate the LHS of (8) as 1 and we conclude that $\mathsf{CheckCond}(2, 1)$ holds. □

Having finished the single recursion case, we illustrate the divide-and-conquer case. We represent the branch as $\mathtt{pre}(S(n)); p(v); p(H_2(n) - v);$, where $H_2(n)$ is of the form $\lceil \frac{n}{b} \rceil + c$ or $\lfloor \frac{n}{b} \rfloor + c$. W.l.o.g., we assume that $H_2(n)$ is $n - 1$ or n, and other choices of $H_2(n)$ follows the same treatment. Similar to the single

recursion case, we perform a case analysis on dist. When dist is an FSDPD, we follow the same strategy as (S1-D) and (S2-D). In detail, for (S1-D), we observe that $f(\alpha, v) + f(\alpha, H_2(n) - v) \leq f(\alpha, H_2(n)) \leq f(\alpha, n)$, thus we can still over-approximate the expectation in (8) as $t(\alpha, n) \cdot S(n)$. For (S2-D), we only need to further consider to over-approximate the extra term $t(\alpha, n) \cdot f(\alpha, H_2(n) - v)$ by the same procedure.

When dist is $\texttt{uniform}(n)$ (or $\texttt{muniform}(n)$), then we can also over-approximate the subterm in the expectation (8) $L = \mathbb{E}[\exp(t(\alpha, n) \cdot (f(\alpha, v) + f(\alpha, H_2(n) - v)))]$ through the summation as follows:

$$L \leq \frac{1}{n} \sum_{i=0}^{n-1} \exp(t(\alpha, n) \cdot (f(\alpha, v) + f(\alpha, H_2(n) - v))) \tag{10}$$

When dist is $\texttt{muniform}(n)$, we simply replace $\frac{1}{n}$ by $\frac{2}{n}$. Since $f(\alpha, v) + f(\alpha, H_2(n) - v)$ is non-decreasing when $v \in [0, \lceil \frac{H_2(n)}{2} \rceil - 1]$ and is non-increasing when $v \in [\lfloor \frac{H_2(n)}{2} \rfloor + 1, H_2(n)]$. Hence, we can over-approximate the summation in (10) as two integrals:

$$\begin{aligned}
&\left(\int_0^{\lceil \frac{H_2(n)}{2} \rceil} + \int_{\lfloor \frac{H_2(n)}{2} \rfloor}^{H_2(n)} \right) \exp(t(\alpha, n) \cdot (f(\alpha, v) + f(\alpha, H_2(n) - v)) \mathrm{d}v \\
&= 2 \int_0^{\lceil \frac{H_2(n)}{2} \rceil} \exp(t(\alpha, n) \cdot (f(\alpha, v) + f(\alpha, H_2(n) - v)) \mathrm{d}v
\end{aligned}$$

The integral is more complex than the single recursion case, since we need to integrate the exponential function whose exponent is a pseudo-polynoimal over n, α, v and $H_2(n) - v$. Thus, it is hard to apply our approach in the single recursion case. Instead, we cut the integral into Q parts (where Q is also a parameter) with equal length and over-approximate each part using the maximum value. Intuitively, the over-approximation will be more precise when Q increases. In this way, we can over-approximate the integral as:

$$2 \cdot \lceil \frac{H_2(n)}{2 \cdot Q} \rceil \sum_{i=1}^{Q} \exp(t(\alpha, n) \cdot (f(\alpha, \frac{i}{Q} \cdot \lceil \frac{H_2(n)}{2} \rceil) + f(\alpha, H_2(n) - \frac{i}{Q} \cdot \lceil \frac{H_2(n)}{2} \rceil)))$$

Note that the formula above is similar to the case of FSDPD, we apply the same strategy for FSDPDs to over-approximate the formula above into the canonical constraint.

In the end, we present how to combine the strengthening results for different branches into a single canonical constraint. Suppose for every probabilistic branch B_i, we have successfully obtained the canonical constraint $Q_{L,i}(\alpha, n) \leq 1$ satisfying the restrictions (C1)–(C2), which serves as the strengthening of the original constraint (8). Then, the canonical constraint for the whole distribution is $\sum_{i=1}^{k} c_i \cdot Q_{L,i}(\alpha, n) \leq 1$. Intuitively, there is probability c_i for the branch B_i, thus the combination follows by simply expanding the expectation term.

A natural question is to ask whether our algorithm can always succeed to obtain the canonical constraint. We have the proposition as follows.

Proposition 2. *If the template for t has a lower magnitude than $S(n)^{-1}$ for every branch, then the rewriting always succeeds.*

Proof. Under the condition above, we first consider the single recursion case. When dist is FSDPD, we can apply (S1-D) to over-approximate the exponent

as $t(\alpha, n) \cdot S(n)$. Since $t(\alpha, n)$ has a lower magnitude than $S(n)^{-1}$, by further applying (R3) to eliminate n^{-c} and $\ln^{-c} n$, we obtain the canonical constraint. If dist is $\texttt{uniform}(n)$ or $\texttt{muniform}(n)$, by case analysis of the degree d_v for v and the degree ℓ_v for $\ln v$, we observe that the over-approximation result for the integral is either $\frac{\exp(f(\alpha,n))}{f(\alpha,n)\cdot t(\alpha,n)}$ (when $d_v > 0$) or $\frac{\ln n \cdot \exp(f(\alpha,n))}{f(\alpha,n)\cdot t(\alpha,n)}$ (when $d_v = 0$). Thus, we can cancel the term $f(\alpha, n)$ in the exponent and obtain the canonical constraint by the subsequent steps. The proof is the same for the divide-and-conquer case. □

To ensure our algorithm never fails, we restrict $u_t, v_t \leq 0$ in the template.

Remark 1. Our algorithm can be extended to support piecewise uniform distributions (e.g. each of $0, \ldots, n/2$ with probability $\frac{2}{3n}$ and each of $n/2+1, \ldots, n-1$ with probability $\frac{4}{3n}$) by handling each piece separately.

5 Experimental Results

In this section, we evaluated our algorithm over classical randomized algorithms such as QuickSort (Example 3), QuickSelect (Example 2), DiameterComputation [29, Chapter 9], RandomizedSearch [27, Chapter 9], ChannelConflictResolution [24, Chapter 13], examples such as Rdwalk and Rdadder in the literature [8], and four manually-crafted examples (MC1 – MC4). For each example, we manually compute its expected running time for the prunning. A detailed description of all these examples is provided in Appendix B.

We implemented our algorithm in C++, for which we choose $B = 2$ (as the bounded range for the template), $M = 4$ (in the guess procedure), $Q = 8$ (for the number of parts in the integral), and prune the search space by Theorem 1. All results were obtained on an Ubuntu 18.04 machine with an 8-Core Intel i7-7900x Processor (13.75M cache, up to 4.30 GHz) and 40 GB of RAM.

We report the tail bound derived by our algorithm in Table 1, for which "Benchmark" lists the benchmarks we consider, "$\alpha \cdot \kappa(n^*)$" lists the time limit of interest, "Our bound" lists the tail bound by our approach, "Time(s)" lists the runtime of our approach measured in seconds, and "Karp's bound" lists the bounds given by Karp's cookbook method. From the table, one observes that our algorithm constantly derives asymtotically tighter tail bounds than Karp's method. Moreover, all these bounds are obtained in a few seconds, demonstrating the efficiency of our algorithm. Furthermore, our algorithm obtains bounds with tighter magnitude than our completeness theorem (Theorem 3) in 9 benchmarks, and bounds with the same magnitude for the others, see Appendix G for details.

6 Related Work

Below we compare our approach with the most related results in the literature.

Karp's Cookbook. Our approach is orthogonal to Karp's cookbook method [23] since we base our approach on Markov's inequality, and the core of Karp's method is a dedicated proof for establishing that an intricate tail bound function is a prefixed point of the higher order operator derived from the given PRR. Furthermore, our automated approach can derive asymptotically tighter tail bounds

Table 1. Experimental Result

Benchmark	$\alpha \cdot \kappa(n^*)$ in (1)	Our bound	Time(s)	Karp's bound
QuickSelect	$\alpha \cdot n^*$	$\exp(2 \cdot \alpha - \alpha \cdot \ln \alpha)$	2.90	$\exp(1.15 - 0.28 \cdot \alpha)$
QuickSort	$\alpha \cdot n^* \cdot \ln n^*$	$\exp((4 - \alpha) \cdot \ln n^*)$	17.38	$\exp(0.5 - 0.5 \cdot \alpha)$
L1Diameter	$\alpha \cdot n^*$	$\exp(\alpha - \alpha \cdot \ln \alpha)$	2.81	$\exp(1.39 - 0.69 \cdot \alpha)$
L2Diameter	$\alpha \cdot n^* \cdot \ln n^*$	$\exp(\alpha - \alpha \cdot \ln \alpha)$	2.98	$\exp(1.39 - 0.69 \cdot \alpha)$
RandSearch	$\alpha \cdot \ln n^*$	$\exp((2 \cdot \alpha - \alpha \cdot \ln \alpha) \cdot \ln n^*)$	2.94	$\exp(-0.29 \cdot \alpha \cdot \ln n^*)$
Channel	$\alpha \cdot n^*$	$\exp(0.5 \cdot (2 - \alpha) \cdot n^*)$	17.47	$\exp(-1 + 0.37 \cdot \alpha)$
Rdwalk	$\alpha \cdot n^*$	$\exp(0.25 \cdot (1 - \alpha) \cdot n^*)$	17.18	$\exp(0.60 - 0.41 \cdot \alpha)$
Rdadder	$\alpha \cdot n^*$	$\exp(0.25 \cdot (1 - \alpha) \cdot n^*)$	17.76	Not applicable
MC1	$\alpha \cdot \ln n^*$	$\exp((\alpha - \alpha \cdot \ln \alpha) \cdot \ln n^*)$	3.06	$\exp(-0.69 \cdot \alpha \cdot \ln n^*)$
MC2	$\alpha \cdot \ln^2 n^*$	$\exp((\alpha - \alpha \cdot \ln \alpha) \cdot \ln n^*)$	2.87	$\exp(-0.69 \cdot \alpha \cdot \ln n^*)$
MC3	$\alpha \cdot n^* \cdot \ln^2 n^*$	$\exp(2 \cdot \alpha - \alpha \cdot \ln \alpha)$	3.00	$\exp(1.15 - 0.28 \cdot \alpha)$
MC4	$\alpha \cdot n^*$	$\exp(\alpha - \alpha \cdot \ln \alpha)$	2.98	Not applicable

than Karp's method over all 12 PRRs in our benchmark. Our approach could also handle randomized preprocessing times, which is beyond the reach of Karp's method. Since Karp's proof of prefixed point is ad-hoc, it is non-trivial to extend his method to handle the randomized cost. Nevertheless, there are PRRs (e.g., Coupon-Collector, see Appendix H) that can be handled by Karp's method but not by ours. Thus, our approach provides a novel way to obtain asymptotically tighter tail bounds than Karp's method.

The recent work [33] extends Karp's method for deriving tail bounds of work-and-span runtime over parallel randomized algorithms. This method derives the same tail bounds as Karp's method over PRRs with a single recursive call (such as QuickSelect) and cannot handle randomized pre-processing time. Compared with this approach, our approach derives tail bounds with tighter magnitude on 11/12(92.67%) benchmarks and derives the tail-bound with the same magnitude on QuickSort.

Custom Analysis. Custom analysis of PRRs [17,28] has successfully derived tight tail bounds for QuickSelect and QuickSort. Compared with the custom analysis that requires ad-hoc proofs, our approach is automated, has the generality from Markov's inequality, and is capable of deriving bounds identical or very close to the tail bounds from the custom analysis.

Probabilistic Programs. There are also relevant approaches in probabilistic program verification. These approaches are either based on martingale concentration inequalities (for exponentially-decreasing tail bounds) [8,14,12,21,13], Markov's inequality (for polynomially-decreasing tail bounds) [9,25,34], fixed-point synthesis [35], or weakest precondition reasoning [22,5]. Compared with these approaches, our approach is dedicated to PRRs (a light-weight representation of recursive probabilistic programs) and involves specific treatment of common recursive patterns (such as randomized pivoting and divide-and-conquer) in randomized algorithms, while these approaches usually do not consider common recursion patterns in randomized algorithms. Below we have detailed technical comparisons with these approaches.

– Compared with the approaches based on martingale concentration inequalities [8,13,14,12,21], our approach has the same root as them, since martingale concentration inequalities are often proved via Markov's inequality. However, those approaches have more accuracy loss since these martingale concentration inequalities usually make further relaxations after applying Markov's inequality. In contrast, our automated approach directly handles the constraint after applying Markov's inequality by having a refined treatment of exponentiation and hence has better accuracy in deriving tail bounds.
– Compared with the approaches [9,25,34] that derive polynomially-decreasing tail bounds, our approach targets the sharper exponentially-decreasing tail bounds and hence is orthogonal.
– Compared with the fixed-point synthesis approach [35] that synthesizes exponentially decreasing tail bounds via either the synthesis of polynomial repulsing ranking supermartingales [14] or convex programming for the restricted case of linear exponents, our approach is orthogonal as it is based on Markov's inequality. Note that the approach [35] can only handle 3/12 (25%) of our benchmarks, and synthesize the tail bound with the same magnitude as ours on these three benchmarks.
– Compared with weakest precondition reasoning [22,5] that requires first specifying the bound functions and then verifying the bound functions by proof rules related to fixed-point conditions, mainly with manual efforts, our approach can be automated and is based on Markov's inequality rather than fixed point theorems. Although Karp's method is also based on a particular tail bound function as a prefixed point and thus can be embedded into the weakest precondition framework, Karp's proof of prefixed point requires deep insight, which is beyond the reach of existing proof rules. Moreover, even a slight relaxation of the tail bound function into a simpler form in Karp's method no longer keeps the bound function to be a prefixed point (see Appendix H for details). Hence, the approach of the weakest precondition may not be suitable for deriving tail bounds.

References

1. Achatz, M., McCallum, S., Weispfenning, V.: Deciding polynomial-exponential problems. In: Sendra, J.R., González-Vega, L. (eds.) Symbolic and Algebraic Computation, International Symposium, ISSAC 2008, Linz/Hagenberg, Austria, July 20-23, 2008, Proceedings. pp. 215–222. ACM (2008). https://doi.org/10.1145/1390768.1390799
2. Agrawal, S., Chatterjee, K., Novotný, P.: Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. PACMPL **2**(POPL), 34:1–34:32 (2018)
3. Aguirre, A., Barthe, G., Hsu, J., Kaminski, B.L., Katoen, J.P., Matheja, C.: A pre-expectation calculus for probabilistic sensitivity. Proc. ACM Program. Lang. **5**(POPL) (jan 2021). https://doi.org/10.1145/3434333
4. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)
5. Batz, K., Kaminski, B.L., Katoen, J.P., Matheja, C., Verscht, L.: A calculus for amortized expected runtimes. Proc. ACM Program. Lang. **7**(POPL), 1957–1986 (2023). https://doi.org/10.1145/3571260
6. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. An EATCS Series, Springer (2004)

7. Billingsley, P.: Probability and Measure. Wiley, 3rd edn. (1995)
8. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: CAV. pp. 511–526 (2013)
9. Chatterjee, K., Fu, H.: Termination of nondeterministic recursive probabilistic programs. CoRR **abs/1701.02944** (2017)
10. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through positivstellensatz's. In: CAV 2016. pp. 3–22 (2016)
11. Chatterjee, K., Fu, H., Murhekar, A.: Automated recurrence analysis for almost-linear expected-runtime bounds. In: Majumdar, R., Kunčak, V. (eds.) Computer Aided Verification. pp. 118–139. Springer International Publishing, Cham (2017)
12. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. TOPLAS **40**(2), 7:1–7:45 (2018)
13. Chatterjee, K., Goharshady, A.K., Meggendorfer, T., Zikelic, D.: Sound and complete certificates for quantitative termination analysis of probabilistic programs. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13371, pp. 55–78. Springer (2022). https://doi.org/10.1007/978-3-031-13185-1_4
14. Chatterjee, K., Novotný, P., Žikelić, Đ.: Stochastic invariants for probabilistic termination. In: POPL 2017. pp. 145–160 (2017)
15. Chaudhuri, S., Dubhashi, D.P.: Probabilistic recurrence relations revisited. Theoretical Computer Science **181**(1), 45–56 (1997)
16. Goodman, N.D., Mansinghka, V.K., Roy, D., Bonawitz, K., Tenenbaum, J.B.: Church: a language for generative models. In: UAI 2008. pp. 220–229. AUAI Press (2008)
17. Grübel, R.: Hoare's selection algorithm: A markov chain approach. Journal of Applied Probability **35**(1), 36–45 (1998), http://www.jstor.org/stable/3215544
18. Hoare, C.A.R.: Algorithm 64: Quicksort. Commun. ACM **4**(7), 321 (1961)
19. Hoare, C.A.R.: Algorithm 65: find. Commun. ACM **4**(7), 321–322 (1961)
20. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**(301), 13–30 (1963)
21. Huang, M., Fu, H., Chatterjee, K.: New approaches for almost-sure termination of probabilistic programs. In: APLAS. pp. 181–201 (2018)
22. Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected runtimes of randomized algorithms. J. ACM **65**(5), 30:1–30:68 (2018). https://doi.org/10.1145/3208102
23. Karp, R.M.: Probabilistic recurrence relations. Journal of the ACM **41**(6), 1136–1150 (1994)
24. Kleinberg, J.M., Tardos, É.: Algorithm design. Addison-Wesley (2006)
25. Kura, S., Urabe, N., Hasuo, I.: Tail probabilities for randomized program runtimes via martingales for higher moments. In: Vojnar, T., Zhang, L. (eds.) TACAS. LNCS, vol. 11428, pp. 135–153. Springer (2019)
26. Mahmoud, H.: Pólya urn models. CRC press (2008)
27. McConnell, J.J. (ed.): The Analysis of Algorithms: An Active Learning Approach. Jones & Bartlett Learning (2001)
28. McDiarmid, C., Hayward, R.: Large deviations for quicksort. Journal of Algorithms **21**(3), 476–507 (1996)
29. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
30. Preparata, F.P., Muller, D.E.: Finding the intersection of n half-spaces in time o(n log n). Theor. Comput. Sci. **8**, 45–55 (1979). https://doi.org/10.1016/0304-3975(79)90055-0, https://doi.org/10.1016/0304-3975(79)90055-0

31. Rudin, W.: Real and Complex Analysis, 3rd Ed. McGraw-Hill, Inc., USA (1987)
32. Smith, C., Hsu, J., Albarghouthi, A.: Trace abstraction modulo probability. Proc. ACM Program. Lang. **3**(POPL) (jan 2019). https://doi.org/10.1145/3290352, https://doi.org/10.1145/3290352
33. Tassarotti, J., Harper, R.: Verified tail bounds for randomized programs. In: ITP. pp. 560–578 (2018)
34. Wang, D., Hoffmann, J., Reps, T.W.: Central moment analysis for cost accumulators in probabilistic programs. In: Freund, S.N., Yahav, E. (eds.) PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021. pp. 559–573. ACM (2021). https://doi.org/10.1145/3453483.3454062, https://doi.org/10.1145/3453483.3454062
35. Wang, J., Sun, Y., Fu, H., Chatterjee, K., Goharshady, A.K.: Quantitative analysis of assertion violations in probabilistic programs. In: Freund, S.N., Yahav, E. (eds.) PLDI. pp. 1171–1186. ACM (2021)
36. Wilkie, A.J.: Schanuel's Conjecture and the Decidability of the Real Exponential Field, pp. 223–230. Springer Netherlands, Dordrecht (1997). https://doi.org/10.1007/978-94-015-8923-9_11
37. Williams, D.: Probability with Martingales. Cambridge university press (1991)

A An Illustrative Example of a Martingale

Example 13. Consider an unbiased and discrete random walk, in which we start at a position X_0, and at each second walk one step to either left or right with equal probability. Let X_n denote our position after n seconds. It is easy to verify that $\mathbb{E}(X_{n+1}|X_0,\ldots,X_n) = \frac{1}{2}(X_n-1)+\frac{1}{2}(X_n+1) = X_n$. Hence, this random walk is a martingale. Note that by definition, every martingale is also a supermartingale. As another example, consider the classical gambler's ruin: a gambler starts with Y_0 dollars of money and bets continuously until he loses all of his money. If the bets are unfair, i.e. the expected value of his money after a bet is less than its expected value before the bet, then the sequence $\{Y_n\}_{n\in\mathbb{N}_0}$ is a supermartingale. In this case, Y_n is the gambler's total money after n bets. On the other hand, if the bets are fair, then $\{Y_n\}_{n\in\mathbb{N}_0}$ is a martingale. See Appendix A for another illustrative example of martingales. □

Example 14 (Pólya's Urn [26]). As a more interesting example, consider an urn that initially contains R_0 red and B_0 blue marbles ($R_0+B_0>0$). At each step, we take one marble from the urn, chosen uniformly at random, look at its color and then add two marbles of that color to the urn. Let B_n, R_n and M_n respectively be the number of red, blue and all marbles after n steps. Also, let $\beta_n = \frac{B_n}{M_n}$ and $\rho_n = \frac{R_n}{M_n}$ be the proportion of marbles that are blue (resp. red) after n steps. Let $\mathcal{F}_n$ model the observations until the n-th step. The process described above leads to the following equations:

$$M_{n+1} = 1 + M_n,$$

$$\mathbb{E}[B_{n+1} \mid \mathcal{F}_n] = \mathbb{E}[B_{n+1} \mid B_1,\ldots,B_n] = \frac{B_n}{M_n}\cdot(B_n+1) + \frac{R_n}{M_n}\cdot B_n,$$

$$\mathbb{E}[R_{n+1} \mid \mathcal{F}_n] = \mathbb{E}[R_{n+1} \mid B_1,\ldots,B_n] = \frac{R_n}{M_n}\cdot(R_n+1) + \frac{B_n}{M_n}\cdot R_n.$$

Note that we did not need to care about observing R_i's, M_i's, β_i's or ρ_i's, because they can be uniquely computed in terms of B_i's. More generally, an observer can observe only B_i's, or only R_i's, or only β_i's or ρ_i's and can then compute the rest using this information. Based on the equations above, we have:

$$\mathbb{E}[\beta_{n+1} \mid \mathcal{F}_n] = \frac{B_n}{M_n}\cdot\frac{B_n+1}{M_n+1} + \frac{M_n-B_n}{M_n}\cdot\frac{B_n}{M_n+1} = \frac{B_n}{M_n} = \beta_n,$$

$$\mathbb{E}[\rho_{n+1} \mid \mathcal{F}_n] = \frac{R_n}{M_n}\cdot\frac{R_n+1}{M_n+1} + \frac{M_n-R_n}{M_n}\cdot\frac{R_n}{M_n+1} = \frac{R_n}{M_n} = \rho_n.$$

Hence, both $\{\beta_n\}_{n\in\mathbb{N}_0}$ and $\{\rho_n\}_{n\in\mathbb{N}_0}$ are martingales. Informally, this means that the expected proportion of blue marbles in the next step is exactly equal to their observed proportion in the current step. This might be counter-intuitive. For example, consider a state where 99% of the marbles are blue. Then, it is more likely that we will add a blue marble in the next state. However, this is mitigated by the fact that adding a blue marble changes the proportions much less dramatically than adding a red marble.

B Examples of Probabilistic Recurrence Relations

Example 15 (QuickSelect). Consider the problem of finding the d-th smallest element in an unordered array of n distinct elements. A classical randomized algorithm for solving this problem is QuickSelect [19]. It begins by choosing a pivot element u of the array uniformly at random. It then compares all the other $n-1$ elements of the array with u and divides them into two parts: (i) those that are smaller than u and (ii) those that are larger. Suppose that there are d' elements in part (i). If $d' < d-1$, then the algorithm recursively searches for the $(d-d'-1)$-th smallest element of part (ii). If $d' = d-1$, the algorithm terminates by returning u as the desired answer. Finally, if $d' > d-1$, the algorithm recursively finds the $d-$th smallest element in part (i). Note that the classical median selection algorithm is a special case of QuickSelect. While there is an involved deterministic algorithm for solving the same problem, more involved linear-time non-randomized algorithms exist for the same problem, QuickSelect provides a simple randomized variant matching the best-known $O(n)$ deterministic runtime. Since there is only one recursive procedure in QuickSelect, we model the algorithm as the following PRR.

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{muniform}(n)\ \texttt{in}\ \{\texttt{pre}(n);\ \texttt{invoke}\ p(v);\}\}$$

Here, $p(n)$ represents the number of comparisons performed by QuickSelect over an input of size n, and v is the random variable that captures the size of the remaining array that has to be searched recursively. It can be derived that $v = \max\{i, n-i-1\}$ where i is sampled uniformly from $\{0,\ldots,n-1\}$. □

Example 16 (QuickSort). Consider the classical problem of sorting an array of n distinct elements. A well-known randomized algorithm for solving this problem is QuickSort. Similar to QuickSelect, QuickSort begins by choosing a pivot element u of the array uniformly at random. It then compares all the other $n-1$ elements of the array with u and divides them into two sub-arrays: (i) those that are smaller than u and (ii) those that are larger. The algorithm then runs recursively on both sub-arrays. Compared with its deterministic counterpart, MergeSort, QuickSort provides a simple randomized variant whose expected runtime matches the best-known deterministic runtime $O(n \log n)$.

We model the algorithm as the following PRR:

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{uniform}(n)\ \texttt{in}\ \{\texttt{pre}(n);\ \texttt{invoke}\ p(v); p(n-1-v);\}\}$$

Here, v and $n-1-v$ are random variables that capture the sizes of the two sub-arrays, where i is uniformly sampled from $\{0,\ldots,n-1\}$. □

Example 17 (DiameterComputation). Consider the DiameterComputation algorithm [29, Chapter 9] to compute the diameter of an input finite set S of three-dimensional points. Depending on the metric, i.e. Eucledian or L_1, we obtain two different recurrence relations. For the Euclidean L_2 metric, we have the following PRR:

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{uniform}(n)\ \texttt{in}\ \{\texttt{pre}(n \cdot \ln n);\ \texttt{invoke}\ p(v);\}\}$$

For the L_1 metric, the PRR is as follows:

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{uniform}(n)\ \texttt{in}\ \{\texttt{pre}(n);\ \texttt{invoke}\ p(v);\}\}$$

Note that here we use deterministic versions for the subroutine HalfspaceIntersection as required in DiameterComputation, see [30] for the Euclidean case and [29, Problem 9.6, Page 276] for the L_1 case. □

Example 18 (RandomizedSearch). Consider Sherwood's RandomizedSearch algorithm (cf. [27, Chapter 9]). The algorithm checks whether an integer value d is present within the index range $[i, j]$ $(0 \leq i \leq j)$ in an integer array ar which is sorted in increasing order and is without duplicate entries. The algorithm outputs either the index of the item or -1 if d is not present in the index range $[i, j]$. The PRR for this example contains only one recurrence equation*:

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{muniform}(n)\ \texttt{in}\ \{\texttt{pre}(1);\ \texttt{invoke}\ p(v);\}\}$$

□

Example 19 (ChannelConflictResolution). We consider two network scenarios in which n clients are trying to get access to a network channel. This problem is also called Resource-Contention Resolution [24, Chapter 13]. In this problem, if more than one client tries to access the channel, then no client can access it, and if exactly one client requests access to the channel, then the request is granted. In the concurrent setting, the clients share one variable, which is the number of clients which have not yet been granted access. Also in this scenario, once a client gets an access the client does not request for access again. For this problem, we obtain an over-approximating recurrence relation

$$\texttt{def}\ p(n;2) = \bigoplus \begin{cases} \frac{1}{e} : \texttt{pre}(1); \texttt{invoke}\ p(n-1); \\ 1 - \frac{1}{e} : \texttt{pre}(1); \texttt{invoke}\ p(n); \end{cases}$$

Example 20 (RandomWalk). We consider the classic random walk cases appears in many previous literatures [10,14,8,35],

$$\texttt{def}\ p(n;1) = \bigoplus \begin{cases} 0.5 : \texttt{pre}(1); \texttt{invoke}\ p(n); \\ 0.5 : \texttt{pre}(1); \texttt{invoke}\ p(n-3); \end{cases}$$

Example 21 (RandomAdder). We consider the random accumulator from previous literatures [8,35],

$$\texttt{def}\ p(n;1) = \bigoplus \begin{cases} 0.5 : \texttt{pre}(2); \texttt{invoke}\ p(n-1); \\ 0.5 : \texttt{pre}(1); \texttt{invoke}\ p(n-1); \end{cases}$$

□

Example 22 (MC1). We consider the following manually-crafted recurrence relation,

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{uniform}(n)\ \texttt{in}\ \{\texttt{pre}(1);\ \texttt{invoke}\ p(v);\}\}$$

□

Example 23 (MC2). We consider the following manually-crafted recurrence relation,

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{uniform}(n)\ \texttt{in}\ \{\texttt{pre}(\ln n);\ \texttt{invoke}\ p(v);\}\}$$

□

Example 24 (MC3). We consider the following manually-crafted recurrence relation,

$$\texttt{def}\ p(n;2) = \{\texttt{sample}\ v \leftarrow \texttt{muniform}(n)\ \texttt{in}\ \{\texttt{pre}(n \cdot \ln n);\ \texttt{invoke}\ p(v);\}\}$$

□

Example 25 (MC4). We consider the following manually-crafted recurrence relation,

$$\texttt{def}\ p(n;2) = \bigoplus \begin{cases} 0.5 : \{\texttt{sample}\ v \leftarrow \texttt{uniform}(n)\ \texttt{in} \\ \qquad\qquad \{\texttt{pre}(1);\ \texttt{invoke}\ p(v); p(n-1-v);\}\}; \\ 0.5 : \{\texttt{sample}\ v \leftarrow \texttt{muniform}(n)\ \texttt{in}\ \{\texttt{pre}(n);\ \texttt{invoke}\ p(v);\}\} \end{cases}$$

□

C The Detailed Semantics for PRRs

Consider a PRR generated from the grammar in Figure 1 with procedure name p, a *configuration* σ is a pair $\sigma = (comm, \widehat{n})$ where $comm$ is a statement that represents the current statement to be executed and $\widehat{n} \geq c_p$ is a positive integer that is the current value for the variable n. A *PRR state* μ is a triple $\langle \sigma, C, \mathbf{K} \rangle$ for which:

- σ is either the current configuration or halt that represents the termination of the whole PRR.
- $C \geq 0$ records the cumulative preprocessing time so far.
- $\mathbf{K}$ is a stack of configurations that remain to be executed.

We use emp to denote an empty stack, and say that a PRR state $\langle \sigma, C, \mathbf{K} \rangle$ is *final* if $\mathbf{K} = \mathsf{emp}$ and $\sigma = \mathsf{halt}$. Note that in a final configuration $\langle \mathsf{halt}, C, \mathsf{emp} \rangle$, the value C represents the total execution runtime of the PRR.

The semantics of the PRR is defined as a discrete-time Markov chain whose state space is the set of all PRR states and whose transition function $\mathbf{P}$ is defined in the way that the probability $\mathbf{P}(\mu, \mu')$ that the next PRR state is μ' given the current PRR state $\mu = ((comm, \widehat{n}), C, \mathbf{K})$ is determined by the following cases. Below we abbreviate $e[\widehat{n}/n, \widehat{v}/v], e_i[\widehat{n}/n, \widehat{v}/v], (s-v)[\widehat{n}/n, \widehat{v}/v]$ as $\widehat{e}, \widehat{e_i}, \widehat{s} - \widehat{v}$ respectively, and denote by $\mathsf{top}(\mathbf{K}), \mathsf{pop}(\mathbf{K})$ the top element and resp. the stack after popping the top element in a stack $\mathbf{K}$ respectively.

- In the case $comm = \mathsf{halt}$ and $\mathbf{K} = \mathsf{emp}$, we define $\mathbf{P}(\mu, \mu) := 1$ and $\mathbf{P}(\mu, \mu') := 0$ for other μ'. This means that the PRR stays at termination once it terminates.

– In the case $comm = \texttt{sample}\ v \leftarrow dist\ \texttt{in}\ \{\texttt{pre}(e); \texttt{invoke}\ p(v); p(s-v)\}$ with distribution $dist$, pseudo-polynomial expression e and size expression s, we have that for every $\widehat{v} \in \mathrm{supp}(dist)$, $\mathbf{P}(\mu, \mu_{\widehat{v}}) := dist(\widehat{v})$ for which

$$\mu_{\widehat{v}} := \begin{cases} ((\mathsf{func}(p), \widehat{v}), C + \widehat{e}, (\mathsf{func}(p), \widehat{s} - \widehat{v}) \cdot \mathbf{K}) & \text{if } \widehat{v}, \widehat{s} - \widehat{v} \geq c_p \\ ((\mathsf{func}(p), \widehat{v}), C + \widehat{e}, \mathbf{K}) & \text{if } \widehat{v} \geq c_p \ \&\ \widehat{s} - \widehat{v} < c_p \\ ((\mathsf{func}(p), \widehat{s} - \widehat{v}), C + \widehat{e}, \mathbf{K}) & \text{if } \widehat{v} < c_p \ \&\ \widehat{s} - \widehat{v} \geq c_p \\ (\mathsf{top}(\mathbf{K}), C + \widehat{e}, \mathsf{pop}(\mathbf{K})) & \text{if } \widehat{v}, \widehat{s} - \widehat{v} < c_p \ \&\ \mathbf{K} \neq \mathsf{emp} \\ (\mathsf{halt}, C + \widehat{e}, \mathsf{emp}) & \text{if } \widehat{v}, \widehat{s} - \widehat{v} < c_p \ \&\ \mathbf{K} = \mathsf{emp} \end{cases}$$

– In the case $comm = \texttt{sample}\ v \leftarrow dist\ \texttt{in}\ \{\texttt{pre}(e); \texttt{invoke}\ p(v)\}$ with distribution $dist$ and pseudo-polynomial expression e. we have that for every $\widehat{v} \in \mathrm{supp}(dist)$, we define $\mathbf{P}(\mu, \mu_{\widehat{v}}) := dist(\widehat{v})$ for which

$$\mu'_{\widehat{v}} := \begin{cases} ((\mathsf{func}(p), \widehat{v}), C + \widehat{e}, \mathbf{K}) & \text{if } \widehat{v} \geq c_p \\ (\mathsf{top}(\mathbf{K}), C + \widehat{e}, \mathsf{pop}(\mathbf{K})) & \text{if } \widehat{v} < c_p \ \&\ \mathbf{K} \neq \mathsf{emp} \\ (\mathsf{halt}, C + \widehat{e}, \mathsf{emp}) & \text{if } \widehat{v} < c_p \ \&\ \mathbf{K} = \mathsf{emp} \end{cases}$$

  The case of $comm = \texttt{sample}\ v \leftarrow dist\ \texttt{in}\ \{\texttt{pre}(e); \texttt{invoke}\ p(n-v)\}$ can be obtained analogously.
– In the case $comm = \bigoplus_{i=1}^{k} e_i : comm_i$, we have that $\mathbf{P}(\mu, \mu_i) = \widehat{e_i}$ for each $1 \leq i \leq k$ for which we have $\mu_i := ((comm_i, \widehat{n}), C, \mathbf{K})$.

With an initial PRR state $((\mathsf{func}(p), n^*), 0, \mathsf{emp})$ where $n^* \geq c_p$ is the input size, the Markov chain induces a probability space where the sample space is the set of all infinite sequences of PRR states, the σ-algebra is generated by all sets of infinite sequences of PRR states that share a common prefix (called *cylinder sets*), and the probability measure is uniquely determined by the probability transition function $\mathbf{P}$. We refer to [4, Chapter 10] for details. In the following, we denote the probability measure by $\Pr_{n^*}$ where $n^* \geq c_p$ is the input size.

D Details in Section 3

D.1 Details of the translation

The definition of $\mathsf{Tf}(\hat{n}, Prog)$ is defined in Figure 2. For recursive bodies, the translation is straightforward. For sample, we first draw $\hat{v}$ from the distribution dist under the concrete input $\hat{n}$, and we substitute concrete values $\hat{n}, \hat{v}$ with n, v in the translation result for its recursive body. For probabilistic choice, we use a discrete distribution, where for each choice $1 \leq i \leq k$, there is probability expr_i and returns the translation result for the sub-command comm_i. The following theorem formalizes its correctness.

Theorem 6. *Given a procedure $p(n; c_p)$, then for every $\hat{n} \geq c_p$, $\mathsf{Tf}(\hat{n}, \mathsf{func}(p))$ outputs the joint distribution of the triple of random variables $(S(\hat{n}), \mathsf{size}_1(\hat{n}), \mathsf{size}_2(\hat{n}), r)$ in (2).*

Proof. The proof is straightforward by structural induction. The induction cases have been illustrated in the above text.

$$\textbf{(Command)}\ \mathsf{Tf}(\hat{n}, \texttt{sample}\ v \leftarrow \mathsf{dist}\ \texttt{in}\ \{\mathsf{body}\}) = [\hat{n}/n, \hat{v}/v]\mathsf{Tf}(\hat{n}, \mathsf{body})$$
$$\textit{where } \hat{v} \textit{ is a meta variable drawn from the distribution } [\hat{n}/n]\texttt{dist}$$
$$\mathsf{Tf}(\hat{n}, \bigoplus_{i=1}^{k} c_i : \mathsf{comm}_i) = \mathsf{discrete}\{c_1 : \mathsf{Tf}(\hat{n}, \mathsf{comm}_1), \ldots, c_k : \mathsf{Tf}(\hat{n}, \mathsf{comm}_k)\}$$
$$\textbf{(Rec Body)}\ \mathsf{Tf}(\hat{n}, \mathtt{pre}(\mathsf{expr}); \mathtt{invoke}\ p(\mathsf{size}_1); p(\mathsf{size}_2)) = ([\hat{n}/n]\mathsf{expr}, [\hat{n}/n]\mathsf{size}_1, [\hat{n}/n]\mathsf{size}_2, 2)$$
$$\mathsf{Tf}(\hat{n}, \mathtt{pre}(\mathsf{expr}); \mathtt{invoke}\ p(\mathsf{size}_1)) = ([\hat{n}/n]\mathsf{expr}, [\hat{n}/n]\mathsf{size}_1, 0, 1)$$

Fig. 2. Transformation Function Tf

D.2 Full Proof of Theorem 2

Below we present the full proof of Theorem 2. To prove Theorem 2, we need the extension of conditional expectation to non-negative random variables that are not necessarily integrable. We adopt the method in the previous work [2], where the conditional expectation is extended for non-negative random variables. A random variable X is *non-negative* if $X(\omega) \geq 0$ for all elements ω in the sample space.

Theorem 7 ([2, Proposition 3.1]). *Let X be any non-negative random variable from a probability space $(\Omega, \mathcal{F}, \Pr)$ and $\mathcal{G}$ be a sub-σ-algebra of $\mathcal{F}$ (i.e., $\mathcal{G} \subseteq \mathcal{F}$). Then there exists a random variable $\mathbb{E}[X \mid \mathcal{G}]$ from $(\Omega, \mathcal{F}, \Pr)$ (called a conditional expectation of X w.r.t $\mathcal{G}$) that fulfills the two conditions (E1) and (E2) below:*

(E1) $\mathbb{E}[X \mid \mathcal{G}]$ is $\mathcal{G}$-measurable, and
(E2) for all $A \in \mathcal{G}$, we have $\int_A \mathbb{E}[X \mid \mathcal{G}]\ \mathrm{dPr} = \int_A X\ \mathrm{dPr}$.

Moreover, the conditional expectation is almost surely unique, i.e., for any random variables Y, Z from $(\Omega, \mathcal{F}, \Pr)$ that both fulfill (E1) and (E2), we have that $\Pr(Y = Z) = 1$.

Below we show that some basic property we utilize in our proof is still valid upon the extension. Below we fix a probability space $(\Omega, \mathcal{F}, \Pr)$ and a sub-σ-algebra $\mathcal{G} \subseteq \mathcal{F}$. The following properties hold for *non-negative* random variables X, Y, Z.

(E3) ('**Taking out what is known**') If Z is $\mathcal{G}$-measurable, then $\mathbb{E}[Z \cdot X \mid \mathcal{G}] = Z \cdot \mathbb{E}[X \mid \mathcal{G}]$ a.s. The proof is exactly the same as under 'Proof of (j)' on [37, Page 90] for the non-negative case.

Our proof relies on the celebrated optional stopping theorem, which is stated below:

Theorem 8 (Optional stopping theorem [37, Theorem 10.10]). *For any discrete-time supermartingale $\Gamma = \{X_n\}_{n \in \mathbb{N}}$ adapted to a filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}}$ where every random variable X_n is non-negative, and any stopping time ρ w.r.t the filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}}$ that is almost surely finite (i.e. $\Pr(\rho < \infty) = 1$), we have $\mathbb{E}[X_\rho] \leq \mathbb{E}[X_0]$.*

Now we are ready to prove Theorem 2.

Proof (Proof of Theorem 2). We fix some sufficiently large α and n^*. In general, we apply the martingale theory to prove this theorem. To construct a martingale, we need to make two preparations.

First, since $t_*(\alpha, n^*) \leq t(\alpha, n)$ for every $c_p \leq n \leq n^*$, substituting $t(\alpha, n)$ with $t_*(\alpha, n^*)$ in (4) does not affect the validity of (4) by the convexity of $\exp(\cdot)$, i.e., $\forall c_p \leq n \leq n^*$, we have that:

$$\mathbb{E}[\exp(t_*(\alpha, n^*) \cdot \mathsf{Ex}(n \mid f))] \leq t_*(\alpha, n^*) \cdot f(\alpha, n) \tag{11}$$

Second, given an infinite sequence of the PRR states $\rho = \mu_0, \mu_1, \ldots$ in the sample space, we construct its entry subsequence $\rho' = \mu'_0, \mu'_1, \ldots$ as follows. We keep all states $((comm, \hat{n}), C, \mathbf{K}) \in \rho$ at the entry point of the procedure p, i.e., $comm = \mathsf{func}(p)$, we also keep all final states, and remove all other states. We further define $\tau' := \inf\{t : \mu'_t \text{ is final}\}$. Note that $C'_{\tau'} = C_\tau$. Furthermore, for every $i \geq 0$, if the state μ'_i is not a final state, then μ'_{i+1} represents the recursive calls of the state μ'_i. Recall that by theorem 6, we have that the function $\mathsf{Tf}(\hat{n}, \mathsf{func}(p))$ outputs the joint distribution of the quadruple of random variables $(S(\hat{n}), \mathsf{size}_1(\hat{n}), \mathsf{size}_2(\hat{n}), r)$ after executing $\mathsf{func}(p)$. Hence, if some state $\mu'_i = ((comm'_i, \hat{n}'_i), C'_i, \mathbf{K}'_i) \in \rho'$ is not final, μ'_{i+1} observes as the following distribution.

- Sample $(\Delta, \mathsf{s}_1, \mathsf{s}_2, r)$ from $\mathsf{Tf}(\hat{n}'_i, \mathsf{func}(p))$.
- If $r = 1$,we have that:

$$\mu'_{i+1} = \begin{cases} ((\mathsf{func}(p), \mathsf{s}_1), C'_i + \Delta, \mathbf{K}'_i) & \mathsf{s}_1 \geq c_p \\ (\mathsf{top}(\mathbf{K}'_i), C'_i + \Delta, \mathsf{pop}(\mathbf{K}'_i)) & \mathsf{s}_1 < c_p \wedge \mathbf{K}'_i \neq \mathsf{emp} \\ (\mathsf{halt}, C'_i + \Delta, \mathsf{emp}) & \mathsf{s}_1 < c_p \wedge \mathbf{K}'_i = \mathsf{emp} \end{cases}$$

- If $r = 2$,we have that:

$$\mu'_{i+1} = \begin{cases} ((\mathsf{func}(p), \mathsf{s}_1), C'_i + \Delta, (\mathsf{func}(p), \mathsf{s}_2) \cdot \mathbf{K}'_i) & \mathsf{s}_1 \geq c_p \wedge \mathsf{s}_2 \geq c_p \\ ((\mathsf{func}(p), \mathsf{s}_1), C'_i + \Delta, \mathbf{K}'_i) & \mathsf{s}_1 \geq c_p \wedge \mathsf{s}_2 < c_p \\ ((\mathsf{func}(p), \mathsf{s}_2), C'_i + \Delta, \mathbf{K}'_i) & \mathsf{s}_1 < c_p \wedge \mathsf{s}_2 \geq c_p \\ (\mathsf{top}(\mathbf{K}'_i), C'_i + \Delta, \mathsf{pop}(\mathbf{K}'_i)) & \mathsf{s}_1 < c_p \wedge \mathsf{s}_2 < c_p \wedge \mathbf{K}'_i \neq \mathsf{emp} \\ (\mathsf{halt}, C'_i + \Delta, \mathsf{emp}) & \mathsf{s}_1 < c_p \wedge \mathsf{s}_2 < c_p \wedge \mathbf{K}'_i = \mathsf{emp} \end{cases}$$

Now we are ready to define the martingale. Given an entry subsequence $\rho' = \mu'_0, \mu'_1, \ldots$ in the sample space, for each $i \geq 0$, suppose $\mu'_i = ((comm'_i, \hat{n}'_i), C'_i, \mathbf{K}'_i)$ and $\mathbf{K}'_i = (\mathsf{func}(p), \mathsf{s}_{i,1}) \cdots (\mathsf{func}(p), \mathsf{s}_{i,q_i})$, where q_i represent the length of the stack. We define

$$y_i := \exp\left(C'_i + f(\alpha, \hat{n}'_i) + \textstyle\sum_{1 \leq j \leq q_i} f(\alpha, \mathsf{s}_{i,j})\right)$$

Note that if μ'_i is not final, we have that:

$$\begin{aligned} \mathbb{E}[y_{i+1} \mid y_i] &\leq y_i \cdot \mathbb{E}\left[\exp\left(S(\hat{n}'_i) + \textstyle\sum_{j=1}^{r} f(\alpha, \mathsf{size}_j(\hat{n}'_i)) - f(\alpha, \hat{n}'_i)\right)\right] \\ &\leq y_i \end{aligned}$$

where the first inequality is obtained by comparing μ'_i and μ'_{i+1} from the explicit formula of μ'_{i+1}, and (E3) of the extended conditional expectation. The second inequality is from (11). Otherwise, if μ_i is final, then it is straightforward that $y_{i+1} = y_i = 1$, thus we also have $\mathbb{E}[y_{i+1} \mid y_i] \leq y_i$ in this case.

Thus, $y_0, y_1, \ldots$ is a supermartingale, thus by the optional stopping theorem (Theorem 8), we have that

$$\begin{aligned}\mathbb{E}\left[\exp\left(t_*(\alpha, n^*) \cdot C_\tau\right)\right] &= \mathbb{E}\left[\exp\left(t_*(\alpha, n^*) \cdot C'_{\tau'}\right)\right] \\ &= \mathbb{E}\left[y_{\tau'}\right] \leq \mathbb{E}[y_0] = \exp\left(t_*(\alpha, n^*) \cdot f(\alpha, n^*)\right)\end{aligned}$$

□

D.3 Proof of Theorem 3

Proof (Proof of Theorem 3). The proof relies on Hoeffiding's Lemma as follows.

Theorem 9 (Hoeffidng's Lemma,[20]). *For every bounded random variable $a \leq X \leq b$, we have that* $\mathbb{E}[\exp(X)] \leq \exp(\mathbb{E}[X]) \cdot \exp\left(\frac{(b-a)^2}{8}\right)$.

We first rephrase the constraint (4) as

$$\mathbb{E}\left[\exp\left(t(\alpha, n) \cdot (S(n) + \textstyle\sum_{i=1}^{r} f(\alpha, \mathsf{size}_i(n)) - f(\alpha, n))\right)\right] \leq 1$$

Define $X(\alpha, n)$ as the exponent in the $\exp(\cdot)$, $X(\alpha, n) := t(\alpha, n) \cdot (S(n) + \sum_{i=1}^{r} f(\alpha, \mathsf{size}_i(n)) - f(\alpha, n))$, since we choose $f(\alpha, n)$ as $w(\alpha) \cdot \mathbb{E}[p(n)]$, and the fact that $\mathbb{E}[p(n)] = \mathbb{E}[S(n)] + \sum_{i=1}^{r} \mathbb{E}[p(\mathsf{size}_i)]$, we have that

$$\begin{aligned}\mathbb{E}[X(\alpha, n)] &= t(\alpha, n) \cdot (\mathbb{E}[S(n)] + w(\alpha) \cdot \textstyle\sum_{i=1}^{r} \mathbb{E}[p(\mathsf{size}_i)] - w(\alpha) \cdot \mathbb{E}[p(n)]) \\ &= t(\alpha, n) \cdot (\mathbb{E}[S(n)] - w(\alpha) \cdot \mathbb{E}[S(n)]) \\ &= \tfrac{\lambda(\alpha)}{\mathbb{E}[S(n)]} \cdot (1 - w(\alpha)) \cdot \mathbb{E}[S(n)] \\ &= -\lambda(\alpha) \cdot (w(\alpha) - 1)\end{aligned}$$

Furthermore, by (A1), we have that $X(\alpha, n)$ ranges from $\lambda(\alpha) \cdot (1 - w(\alpha) + w(\alpha) \cdot M_1)$ to $\lambda(\alpha) \cdot (1 - w(\alpha) + w(\alpha) \cdot M_2)$. Hence, by the theorem above, we have that:

$$\mathbb{E}[\exp(X(\alpha, n))] \leq \exp\left(-\lambda(\alpha) \cdot (w(\alpha) - 1) + \tfrac{\lambda(\alpha)^2 \cdot w(\alpha)^2 \cdot (M_2 - M_1)^2}{8}\right) \leq 1$$

where the second inequality is due to our choice of $\lambda(\alpha)$. Hence the theorem follows. □

E Details in Our Algorithmic Approach

E.1 Proof of Theorem 4

Suppose c_t, c_f are valid and c'_t, c'_f are also valid, then due to the convexity, for any $0 < \lambda < 1$, $\lambda \cdot c_t + (1-\lambda) \cdot c'_t, \lambda \cdot c_f + (1-\lambda) \cdot c'_f$ is also valid. Thus the second item holds since $0, c_f$ is always valid. The first items holds from the inequality $\sum_{i=1}^{r} f(\alpha, \mathsf{size}_i(n)) \leq f(\alpha, n)$ if $\sum_{i=1}^{r} \mathsf{size}_i \leq n$.

E.2 An overview of The Decision Procedure for the Canonical Constraint (7)

• *First*, we can safely bound the range of n into constants from the monotonicity of pseudo-polynomials. In detail, for each $1 \leq i \leq k$, we compute the limit M_i of $g_i(n)$, and make the decision as follows.

- If $M_i = +\infty$, then $\exp(g_i(n) + f_i(\alpha))$ could be arbitrarily large when $n \to \infty$. As a result, we can safely conclude that $Q(\alpha, n)$ does not hold.
- Otherwise, by (C2), either $g_i(n)$ is a constant function, or $M_i = -\infty$. In both cases, we can compute $L_i \in \mathbb{N}$ such that $g_i(n)$ is non-increasing when $n \geq L_i$.

By taking L as the maximum of L_i's and c_p, we conclude that the LHS of $Q(\alpha, n)$ is non-increasing as long as $n \geq L$. Hence, it suffices to consider $c_p \leq n \leq L$.

• *Second*, for every integer $c_p \leq \overline{n} \leq L$, we substitute n with $\overline{n}$ to eliminate n in the exponent of the canonical form (7). Then, each exponent $f_i(\alpha) + g_i(\overline{n})$ becomes a pseudo-polynomial solely over α. Since we only concern sufficiently large α, we can compute the limit $R_{\overline{n}}$ for LHS of $Q(\alpha, \overline{n})$ when $\alpha \to \infty$. We make our decision based on the limit $R_{\overline{n}}$ as follows.

- If $R_{\overline{n}} < 1$ for every $c_p \leq \overline{n} \leq L$, then we conclude that $Q(\alpha, n)$ holds for every sufficiently large α and every $n \geq c_p$.
- If $R_{\overline{n}} > 1$ for some $c_p \leq \overline{n} \leq L$, then we conclude that $Q(\alpha, n)$ does not hold.
- If $R_{\overline{n}} = 1$ for some $c_p \leq \overline{n} \leq L$, then the LHS of $Q(\alpha, \overline{n})$ fluctuates around 1, to make sure that we never accept invalid $Q(\alpha, n)$, we still decide $Q(\alpha, n)$ does not hold in this case.

E.3 The Algorithm *Decide*

We first present some computer algebra backgrounds.

Monotonicity of pseudo-monomials. We need a complete depiction of the monotonicity of pseudo monomials, which could be straightforwardly proved by computing the derivative.

the monotonicity of a pseudo-monomial $n^a \cdot \ln^b n$ is presented in Table 2.

Table 2. Monotonicity of pseudo-monomials, where we use $\uparrow (a, b)$ to represent the pseudo-monomial is non-decreasing when $n \in [a, b]$, use $\downarrow (a, b)$ to represent that the function is non-increasing when $n \in [a, b]$.

	$b < 0$	$b = 0$	$b > 0$
$a < 0$	$\downarrow (c_p, \infty)$		$\uparrow (c_p, \exp\left(-\frac{b}{a}\right))$ and $\downarrow (\exp\left(-\frac{b}{a}\right), \infty)$
$a = 0$	$\downarrow (0, \infty)$	constant	$\uparrow (c_p, \infty)$
$a > 0$	$\downarrow (c_p, \exp\left(-\frac{b}{a}\right))$ and $\uparrow (\exp\left(-\frac{b}{a}\right), \infty)$	$\uparrow (c_p, \infty)$	

The function $\mathsf{NegativeLB}(P)$**.** Our decision procedure for canonical forms relies on a function $\mathsf{NegativeLB}(P)$, it takes on input as a pseudo-polynomial $P(n)$

whose sub-monomial term with highest magnitude is negative, and outputs an integer L^* such that $P(n) < 0$ for every $n \geq L^*$. It is achieved by three steps below.

1. Our algorithm firstly finds the term with largest order. In detail, our algorithm will scan each term in the polynomial and finds the term with highest degree of n, if there are several terms with the same highest degree of n, our algorithm chooses the term among them with the highest degree of $\ln n$. Formally, our algorithm finds the term $n^{a^*} \cdot \ln^{b^*} n$ with $c_{a^*,b^*} \neq 0$ and highest a^*, if there are more than one terms such that their degree of n are all equal to a^*, we choose the largest b^* among them. Then, our algorithm divides each term in $P(n)$ by $n^{a^*} \cdot \ln^{b^*} n$, and derives the pseudo-polynomial $P_1(n)$ below:

   $$P_1(n) = c_{a^*,b^*} + \sum_{a,b \neq a^*,b^*} c_{a,b} \cdot n^{a-a^*} \cdot \ln^{b-b^*} n$$

   Note that by our choice of a^*, b^*, the degree of n and $\ln n$ in each non-constant term in $P_1(n)$, i.e., $a - a^*, b - b^*$ either satisfies $a - a^* \leq -1$, or satisfies $a - a^* = 0$ and $b - b^* \leq -1$. Furthermore, we have that $P(n) < 0$ if and only of $P_1(n) < 0$ when $n \geq c_p$. Hence, the transformation from $P(n)$ to $P_1(n)$ does not affect the answer. Below, we consider to compute $\mathsf{NegativeLB}(P_1)$.
2. Next, we consider an over-approximation $P_2(n)$ the function $P_1(n)$ as follows.

   $$P_2(n) := c_{a^*,b^*} + \sum_{a,b \neq a^*,b^*} \max\{0, c_{a,b}\} \cdot n^{a-a^*} \cdot \ln^{b-b^*} n$$

   It removes all non-constant terms with negative coefficients. We prove that there exists a threshold n_e such that $P_2(n)$ will monotonically decreasing for every $n \geq n_e$. Formally, we have that:

   Proposition 3. *Let*

   $$n_e := \lceil \max\{\exp(-(b-b^*)/(a-a^*)) \mid (a,b) \neq (a^*,b^*) \text{ and } c_{a,b} \neq 0\} \rceil$$

   be a constant. Then we have that $P_2(n)$ monotonically decreases for every $n \geq n_e$.

   Proof. Note that every coefficient for non-constant terms in $P_2(n)$ is non-negative. Thus the proposition directly follows from the monotonicity of the function $n^a \cdot \ln^b n$ (Table 2).

   Since $P_1(n) \leq P_2(n)$, we have that $\mathsf{NegativeLB}(P_2)$ also serves as an valid answer to $\mathsf{NegativeLB}(P_1)$. Blow, we consider to compute $\mathsf{NegativeLB}(P_2)$.
3. To compute $\mathsf{NegativeLB}(P_2)$, we simply enumerate every $n \geq n_e$ until we find some n^* such that $P_2(n^*) < 0$. Since $P_2(n)$ will monotonically decreasing for every $n \geq n_e$, for every $n \geq n^*$, $P_1(n) \leq P_2(n) \leq P_2(n^*) < 0$. Thus, n^* is a valid answer to $\mathsf{NegativeLB}(P_1)$ and $\mathsf{NegativeLB}(P)$.

Since sub-monomial term in $P(n)$ with highest magnitude has negative coefficient, we have that $c_{a^*,b^*} < 0$. Hence, the function $\mathsf{NegativeLB}(P)$ always terminates since as long as n is sufficiently large, the $P_2(n)$ will be negative.

The decision procedure. Our goal is to decide if $Q(\alpha, n)$ holds for every sufficiently large α, n^* and every $c_p \leq n \leq n^*$. Since $Q(\alpha, n)$ is irrelevant to n^*, it suffices to check whether the constraint holds over every sufficiently large α and every $n \geq c_p$. For each $1 \leq i \leq k$, we use S_i^α to represent the sub-monomial term $f_i(\alpha)$ with highest magnitude, i.e., the sub-monomial term that firstly maximizes the degree of α, and then maximizes the degree of $\ln \alpha$. Similarly, we use S_i^n to represent the sub-monomial term $g_i(n)$ with highest magnitude. The decision procedure consists of two steps. The pseudo-code is presented in Algorithm 2.

Algorithm 2: The Decision procedure for canonical forms

Input : A constraint $Q(\alpha, n)$ in the canonical form (7)
Output: Decide whether $Q(\alpha, n)$ holds over every sufficiently large α and every $n \geq c_p$.

```
1  Procedure Decide(Q(α, n)):
2    //The first step
3    L := c_p
4    for i := 1, 2, ..., k do
5      S_i^n := the sub-monomial term in g_i(n) with highest magnitude.
6      if S_i^n > 0 then
7        Return False
8      else
9        g_i'(n) := the derivative of g_i(n)
10       L := max{L, NegativeLB(g_i'(n))}
11   //The second step
12   for n̄ := c_p, ..., L do
13     Q(α, n̄) := substitute n with n̄ in Q(α, n)
14     R := 0
15     for i := 1, 2, ..., k do
16       S_i^α := the sub-monomial term in f_i(α) + g_i(n̄) with highest magnitude.
17       Calculate Δ as the limit of f_i(α) + g_i(n̄) when α → ∞ by Proposition 4.
18       if Δ = ∞ then
19         Return False
20       else
21         R := R + γ_i · Δ
22     if R ≥ 1 then Return False;
23   Return True
```

• *First*, in Lines 3–10, for each $1 \leq i \leq k$, we inspect the term S_i^n, if S_i^n has positive coefficient, then clearly $f_i(n) \to \infty$ when $n \to \infty$. Thus, the term $\exp(g_i(n) + f_i(\alpha))$ could be arbitrarily large. As a result, we can safely conclude that $Q(\alpha, n)$ does not hold (Lines 6–7). Otherwise, by computing $\mathsf{NegativeLB}(g_i'(n))$, we have that conclude that $\exp(g_i(n) + f_i(\alpha))$ will be non-increasing as long as $n \geq \mathsf{NegativeLB}(g_i'(n))$. By taking the maximum over all $1 \leq i \leq k$ (the maximum is stored in L) (Line 10), we have that the LHS of $Q(\alpha, n)$ will be non-increasing as long as $n \geq L$. Hence, it suffices to check every $c_p \leq n \leq L$.

• *Second*, for every integer $c_p \leq \overline{n} \leq L$, we substitute n with $\overline{n}$ to eliminate n in the exponent of the canonical form (7). Then, each exponent $f_i(\alpha) + g_i(\overline{n})$ becomes a pseudo-polynomial solely over α. Since we only concern sufficiently large α, we can compute the limit Δ of $\exp(f_i(\alpha) + g_i(\overline{n}))$ when $\alpha \to \infty$ by the proposition (Proposition 4) below, which could be proved straightforwardly.

Proposition 4. *The limit Δ of $\exp(f_i(\alpha) + g_i(\overline{n}))$ when $\alpha \to \infty$ is as follows.*
- *If S_i^α is superconsant and $S_i^\alpha > 0$, then $\Delta = \infty$.*
- *If S_i^α is superconsant and $S_i^\alpha < 0$, then $\Delta = 0$.*
- *If S_i^α is constant C, then $\Delta = \exp(C)$.*
- *If S_i^α is subconstant, then $\Delta = 1$.*

By sum up the Δ for every $1 \leq i \leq k$ as R, we compute the limit of the LHS of $Q(\alpha, \overline{n})$ when $\alpha \to \infty$. We conclude that $Q(\alpha, n)$ if and only if $R < 1$ for every $c_p \leq \overline{n} \leq L$. The property of our algorithm is as follows.

Theorem 10. *The algorithm (Algorithm 2) has the following property.*
- *If there exists two constants $\varepsilon, M > 0$, such that for every $\alpha \geq M$ and every $n \geq c_p$, the LHS of $Q(\alpha, n) \leq 1 - \varepsilon$, then our algorithm will conclude that $Q(\alpha, n)$ holds.*
- *If $Q(\alpha, n)$ does not hold for sufficiently large α and every $n \geq c_p$, then our algorithm will conclude that $Q(\alpha, n)$ does not hold.*

Proof. Since R is the limit of $Q(\alpha, n)$, the two items follows by the definition of limit. □

F Detailed Synthesis Result

Table 3. Detailed Synthesis Result

Benchmark	$\kappa(n^*)$	Function	Tail bound $u(\alpha, n^*)$	Time(s)
QuickSelect	n^*	$\overline{f}(\alpha,n) = \frac{2\cdot\alpha}{\ln\alpha}\cdot n$ $\overline{t}(\alpha,n) = \ln\alpha\cdot n^{-1}$	$\exp(2\cdot\alpha - \alpha\cdot\ln\alpha)$	2.90
QuickSort	$n^*\cdot\ln n^*$	$\overline{f}(\alpha,n) = 4\cdot n\cdot\ln n$ $\overline{t}(\alpha,n) = n^{-1}$	$\exp((4-\alpha)\cdot\ln n)$	17.38
L1Diameter	n^*	$\overline{f}(\alpha,n) = \frac{\alpha}{\ln\alpha}\cdot n$ $\overline{t}(\alpha,n) = \ln\alpha\cdot n^{-1}$	$\exp(\alpha - \alpha\cdot\ln\alpha)$	2.81
L2Diameter	$n^*\cdot\ln n^*$	$\overline{f}(\alpha,n) = \frac{\alpha}{\ln\alpha}\cdot n\cdot\ln n$ $\overline{t}(\alpha,n) = \ln\alpha\cdot n^{-1}\cdot\ln^{-1} n$	$\exp(\alpha - \alpha\cdot\ln\alpha)$	2.98
RandSearch	$\ln n^*$	$\overline{f}(\alpha,n) = \frac{2\cdot\alpha}{\ln\alpha}\cdot\ln n$ $\overline{t}(\alpha,n) = \ln\alpha$	$\exp((2\cdot\alpha - \alpha\cdot\ln\alpha)\cdot\ln n^*)$	2.94
Channel	n^*	$\overline{f}(\alpha,n) = 2\cdot n$ $\overline{t}(\alpha,n) = \frac{1}{2}$	$\exp(\frac{1}{2}\cdot(2-\alpha)\cdot n^*)$	17.47
Rdwalk	n^*	$\overline{f}(\alpha,n) = n$ $\overline{t}(\alpha,n) = \frac{1}{4}$	$\exp(\frac{1}{4}\cdot(1-\alpha)\cdot n^*)$	17.18
Rdadder	n^*	$\overline{f}(\alpha,n) = 4\cdot n$ $\overline{t}(\alpha,n) = \frac{1}{4}$	$\exp(\frac{1}{4}\cdot(1-\alpha)\cdot n^*)$	17.76
MC1	$\ln n^*$	$\overline{f}(\alpha,n) = \frac{\alpha}{\ln\alpha}\cdot\ln n$ $\overline{t}(\alpha,n) = \ln\alpha$	$\exp(\alpha - \alpha\cdot\ln\alpha)$	3.06
MC2	$\ln^2 n^*$	$\overline{f}(\alpha,n) = \frac{\alpha}{\ln\alpha}\cdot\ln^2 n$ $\overline{t}(\alpha,n) = \frac{\ln\alpha}{\ln n}$	$\exp((\alpha - \alpha\ln\alpha)\cdot\ln n^*)$	2.87
MC3	$n^*\cdot\ln^2 n^*$	$\overline{f}(\alpha,n) = \frac{2\cdot\alpha}{\ln\alpha}\cdot n\cdot\ln^2 n$ $\overline{t}(\alpha,n) = \frac{\ln\alpha}{n\cdot\ln^2 n}$	$\exp(2\cdot\alpha - \alpha\ln\alpha)$	3.00
MC4	n^*	$\overline{f}(\alpha,n) = \frac{2\cdot\alpha}{\ln\alpha}\cdot n$ $\overline{t}(\alpha,n) = \ln\alpha\cdot n^{-1}$	$\exp((\alpha - \alpha\ln\alpha)\cdot\ln n^*)$	2.98

G The Tail Bound by Our Completeness Theorem 3

Table 4. Tail Bound By the Completeness Theorem

Benchmark	$\kappa(n^*)$	Tail bound $u(\alpha, n^*)$
QuickSelect	$4n^*$	$\exp(-2 \cdot \frac{(\alpha-1)^2}{\alpha})$
QuickSort	$2n^* \cdot \ln n^*$	$\exp(-0.7 \cdot \frac{(\alpha-1)^2}{\alpha} \cdot \ln n^*)$
L1Diameter	$2n^*$	$\exp(-\frac{(\alpha-1)^2}{\alpha})$
L2Diameter	$2n^* \cdot \ln n^*$	$\exp(-\frac{(\alpha-1)^2}{\alpha})$
RandSearch	$\ln n^* / \ln \frac{4}{3}$	$\exp(-1.19 \cdot \frac{(\alpha-1)^2}{\alpha} \cdot \ln n^*)$
Channel	en^*	$\exp(-0.74 \cdot \frac{(\alpha-1)^2}{\alpha} \cdot n^*)$
Rdwalk	$0.5n^*$	$\exp(-\frac{1}{3} \cdot \frac{(\alpha-1)^2}{\alpha} \cdot n^*)$
Rdadder	$3n^*$	$\exp(-\frac{(\alpha-1)^2}{\alpha} \cdot n^*)$
MC1	$\ln n^* / \ln 2$	Not applicable
MC2	$\ln^2 n^* / \ln 2$	Not applicable
MC3	$2n^* \cdot \ln^2 n^*$	$\exp(-\frac{(\alpha-1)^2}{\alpha})$
MC4	$2.5n^*$	$\exp(-\frac{(\alpha-1)^2}{\alpha})$

H Missing Details in Related Work 6

H.1 Coupon Collector

The example above shows that our approach can obtain bounds that are considerably tighter than Karp's method. Moreover, we will now extend the results to systems of PRRs consisting of several recursive equations (modeling recursive algorithms with several procedures). This is also beyond the reach of Karp's method and all previous methods. However, our approach is unable to handle a program, called

Problem 1. CouponCollector[29, Chapter 3.6] as follows.

Example 26. Consider the Coupon-Collector problem with n different types of coupons($n \in \mathbb{N}$). The randomized process proceeds in rounds: at each round, a coupon is collected uniformly at random from the coupon types the rounds continue until all the n types of coupons are collected. We model the rounds as a recurrence relation with two variables n, m, where n represents the total number of coupon types and m represents the remaining number of uncollected coupon types. The recurrence relation is as follows:

$$p(n, m; 1) = \bigoplus \begin{cases} m/n : \texttt{pre}(1); \texttt{invoke}\ p(n, m-1); \\ 1 - m/n : \texttt{pre}(1); \texttt{invoke}\ p(n, m); \end{cases}$$

Our approach fails in case since it is hard to apply Markov's inequality to derive exponentially decreasing tail bounds. However, Karp's method could produce the following:

$$\Pr[p(n, n) \geq n \ln n + \alpha \cdot n] \leq \exp(-\alpha).$$

□

H.2 Non-Prefixed-Point of Karp's Bound on Quick-Select

Consider the PRR for Quick-Select:

$$\texttt{def}\ p(n; 2) = \{\texttt{sample}\ v \leftarrow \texttt{muniform}(n)\ \texttt{in}\ \{\texttt{pre}(n);\ \texttt{invoke}\ p(v);\}\}$$

where $h(n)$ is the muniform distribution. Karp's method gives the following tail bound:

$$\Pr(p(n) > 4 \cdot n + k \cdot n) \leq \left(\frac{3}{4}\right)^k.$$

Although Karp's result is derived (indirectly) from fixed point theory, we show that the derived tail bound $\left(\frac{3}{4}\right)^k$ is not a prefixed point when k is allowed to be any non-negative real number. First, recall that the prefixed point condition for the situation $k \geq 2$ and an even number n states that

$$\left(\frac{3}{4}\right)^k \geq 2 \cdot \frac{1}{n} \sum_{i=\frac{n}{2}+1}^{n} \left(\frac{3}{4}\right)^{(3+k)-4\cdot\frac{i}{n}}.$$

But we have

$$\begin{aligned}
2 \cdot \frac{1}{n} \cdot \sum_{i=\frac{n}{2}}^{n-1} \left(\frac{3}{4}\right)^{(3+k)-4\cdot\frac{i}{n}} &= 2 \cdot \frac{1}{n} \cdot \left(\frac{3}{4}\right)^{k+1} \cdot \frac{1-\left[\left(\frac{3}{4}\right)^{-\frac{4}{n}}\right]^{\frac{n}{2}}}{1-\left(\frac{3}{4}\right)^{-\frac{4}{n}}} \\
&= 2 \cdot \frac{1}{n} \cdot \left(\frac{3}{4}\right)^{k+1} \cdot \frac{-\frac{7}{9}}{1-e^{\frac{4}{n}\cdot\ln\frac{4}{3}}} \\
&\approx 2 \cdot \frac{1}{n} \cdot \left(\frac{3}{4}\right)^{k+1} \cdot \frac{\frac{7}{9}}{\frac{4}{n}\cdot\ln\frac{4}{3}} \\
&\approx 2 \cdot \frac{1}{n} \cdot n \cdot \left(\frac{3}{4}\right)^{k+1} \cdot 0.676 \\
&= (2 \cdot 0.75 \cdot 0.676) \cdot \left(\frac{3}{4}\right)^{k} \geq 1.01 \cdot \left(\frac{3}{4}\right)^{k}
\end{aligned}$$

for which the first $\approx$ are with the ratio 1 when n tends to infinity. Hence, we have

$$\left(\frac{3}{4}\right)^{k} < 2 \cdot \frac{1}{n} \sum_{i=\frac{n}{2}+1}^{n} \left(\frac{3}{4}\right)^{(3+k)-4\cdot\frac{i}{n}}.$$

As a result, the prefixed point condition is violated for every sufficiently large even number n and $k \geq 2$.