Advanced Algorithmics

Strategies for Tackling Hard Problems

Sebastian Wild

Markus Nebel

Lecture 12

2017-05-29

Error Bounds Matter

Remark 4.17 (Success Probability)

From the point of view of complexities, the success probability bounds are flexible:

- $\mathcal{BPP}$ only requires success probability $\frac{1}{2} + \varepsilon$, but using *Majority Voting*, we can also obtain any fixed success probability $\delta \in (\frac{1}{2}, 1)$, so we could also define $\mathcal{BPP}$ to require, say, $\Pr\big[A(x) = [x \in L]\big] \geq \frac{2}{3}$.
- Similarly for $\mathcal{ZPP}$, we can use probability amplification on Las Vegas algorithms to obtain any success probability $\delta \in (\frac{1}{2}, 1)$.

But recall: this is *not* true for unbounded errors and class $\mathcal{PP}$.
In fact, we have the following result.

Theorem 4.18 (PP can simulate nondeterminism)

$\mathcal{NP} \cup \text{co-}\mathcal{NP} \subseteq \mathcal{PP}$. ◀

$\rightsquigarrow$ Useful algorithms must avoid unbounded errors.

Proof: $\mathcal{PP}$ allows for poly-time overhead

↳ use any poly-time reduction as preprocessing before we start with RA

SAT $\mathcal{NP}$-complete ⇝ $\underline{\text{SAT} \in \mathcal{PP}}$ ⟹ $\mathcal{NP} \subseteq \mathcal{PP}$

$L \in \mathcal{NP}$ ⇝ $f(x) \in \text{SAT} \Leftrightarrow x \in L$

TAUT $\in \mathcal{PP}$ TAUT co-$\mathcal{NP}$-complete ⟹ co-$\mathcal{NP} \subseteq \mathcal{PP}$

↖ similar

SAT: Given φ of length n with k variables

A (UE-MC poly-time)

(1) Generate a random assignment $\alpha \in \{0,1\}^k$ uniformly at random } k random bits

(2) If α satisfies φ ⇝ accept | time linear n (poly-time)

(3) Otherwise accept iff $B(p) = 1$

$$p = \frac{1}{2} - \frac{1}{2^{k+1}} = \frac{2^k - 1}{2^{k+1}} < \frac{1}{2}$$

$k+1$ random bits

all rb indep.

Time: linear in n and k ✓

Correctness: $\underset{\text{OE-MC}}{L(A)} = SAT$

- $\varphi \in SAT \leadsto \Pr[\alpha(\varphi) = 1] \geq \frac{1}{2^k}$ (at least one sat. assign.)

$\overset{!}{<} \frac{1}{2}$

$$\Pr[A(\varphi)=0] = \Pr[\alpha(\varphi)=0] \cdot \Pr[B(p)=0]^{= 1-p}$$

$$\leq \left(1 - \frac{1}{2^k}\right) \cdot \left(\frac{1}{2} + \frac{1}{2^{k+1}}\right)$$

$$= \frac{1}{2} - \frac{1}{2^{2k+1}} < \frac{1}{2}$$

- $\varphi \notin SAT \leadsto \Pr[\alpha(\varphi) = 1] = 0$

$$\Pr[A(\varphi)=0] = 1\cdot(1-p) = \frac{1}{2}+\frac{1}{2^{k+1}} > \frac{1}{2}$$
$$\overset{!}{>}\frac{1}{2}$$

$\Rightarrow$ A is an UE-MC for SAT, runs in polytime

$\Rightarrow$ SAT $\in$ ZP $\square$

One-sided errors

In many cases, errors of MC algorithm are only *one-sided*.

Example: (simplistic) randomized algorithm for SAT
Guess assignment, output $[\phi \text{ satisfied}]$.
(NB: This is not a MC algorithm, since we cannot give a fixed error bound!)

Observation: No false positives; unsatisfiable ϕ always yield 0.
... does this help?

Definition 4.19 (One-sided error Monte Carlo algorithms)

A randomized algorithm A for language L (i.e., for $f(x) = [x \in L]$) is a one-sided-error Monte-Carlo (OSE-MC) algorithm if we have

1. $\Pr[A(x) = 1] \geq \frac{1}{2}$ for all $x \in L$, and
2. $\Pr[A(x) = 0] = 1$ for all $x \notin L$.

equivalent: $A(x) = 1$ always correct
$A(x) = 0$ might be wrong

Definition 4.20 (RP, co-RP)

The classes $\mathcal{RP}$ and co-$\mathcal{RP}$ are the sets of all languages L with a poly-time OSE-MC algorithm for L resp. $\overline{L}$.

Theorem 4.21 (Complementation feasible → errors avoidable)

$\mathcal{RP} \cap \text{co-}\mathcal{RP} = \mathcal{ZPP}$.

yes-certificate   No-certificate

Note the similarly to the open problem $\mathcal{NP} \cap \text{co-}\mathcal{NP} \stackrel{?}{=} \mathcal{P}$;
… a first hint that randomization might not help too much?

PRIMES = {n prime}

PRIMES ∈ co-$\mathcal{NP}$ (divisor)

PRIMES ∈ $\mathcal{NP}$ (complicated)

PRIMES ∈ $\mathcal{P}$ (rel. recent)

Proof (Thm 4.21).

"$\supseteq$" trivial since A poly-time LV for L

$$B(x) = \begin{cases} A(x) & A(x) \in \{0,1\} \\ 0 & A(x) = ? \end{cases} \qquad B \text{ OSE-MC } L$$

$$\overline{B}(x) = \begin{cases} 1 - A(x) & A(x) \in \{0,1\} \\ 1 & A(x) = ? \end{cases} \qquad \overline{B} \text{ OSE-MC } \overline{L}$$

$\Rightarrow L \in RP \cap \text{co-}RP$

"$\subseteq$" B poly-time OSE-MC for L

$\overline{B}$ " " $\overline{L}$

A: Run $B(x)$ and $\overline{B}(x)$

- $\underline{B(x) = 1}$, $\overline{B}(x) = 0$ $\rightarrow$ accept
- $B(x) = 0$, $\underline{\overline{B}(x) = 1}$ $\rightarrow$ reject
- $B(x) = 0 = \overline{B}(x)$ $\rightarrow$? prob $\leq \frac{1}{2}$

($B(x) = 1 = \overline{B}(x)$ $\lightning$ cannot happen)

$\Rightarrow$ $\exists$ poly-time LV for L $\rightsquigarrow$ $L \in \mathcal{ZPP}$ $\square$

Derandomization

Trivial observation: If $Random_A(n) \leq \underbrace{c \operatorname{ld} n}_{O(\log n)}$, there are only $2^{Random_A(n)} = n^c$ different computations.

$\rightsquigarrow$ We can simply execute all of them sequentially in poly-time!

often use limited independence $\rightsquigarrow$ reduce the number real random bits

We can extend this to more randomized bits using *pseudorandom generators*, i.e., algorithms that use a limited amount of real randomness and compute from this a much longer sequence of bits that look random (pseudorandom) to *every* efficient algorithm.

It is not proven that such a method exists, but under widely believed assumptions on circuit complexity lower bounds, there is such a pseudorandom generator that allows to derandomize $\mathcal{BPP}$.

$\rightsquigarrow$ Current belief is $\boxed{\mathcal{BPP} = \mathcal{P}}$ ... and hence $\mathcal{BPP} = \mathcal{RP} = \text{co-}\mathcal{RP} = \mathcal{ZPP} = \mathcal{P}$ (!)

For solving hard problems in theory, randomization does not help at all!

(or: no sufficiently strong lower bound techniques known!)

$\rightsquigarrow$ NP-hard $\rightsquigarrow$ probably not solvable poly-time BE-MC

4.5 Examples of Randomized Algorithms

⇝ Focus on practical benefits of randomization

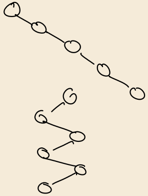
Randomized approaches can be grouped into categories:

1. Coping with adversarial inputs (algorithmic complexity attacks)
   Randomized Quicksort, randomized BSTs, Treaps, skip lists
2. Abundance of Witnesses ⇝ OSE-MC guess a random candidate for certificate & check
   Solovay-Strassen primality test
3. Fingerprinting reduce universe and accept collisions
   universal hashing
4. Random Sampling know "good" structure exists, can draw one, succeeds >0 prob.
   Perfect hashing, 3SAT also by Schöning, Karger's Min-Cut also
5. LP Relaxation & Randomized Rounding
   Set-Cover Approximation (next chapter)

① Coping with worst-case inputs

Naturally Grown BSTs

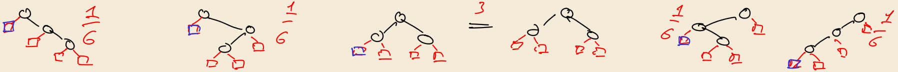
First: Ignore adversaries.



Naturally grown / Random BST: all $n!$ insertion orders *equally likely*.

Example $n=3$

123   132   213   231   312   321



1   1   2   2   3

$$\frac{1+1+2\cdot 2+2+3}{6} = \frac{11}{6} < 2$$

Lemma 4.22 (Random insertion yields random BST)

Let $n \geq 0$ be arbitrary and let T_n be a random BST over n keys. Inserting an element equally likely in one of the $n+1$ *gaps* in T_n (external leaves) results in a new BST T_{n+1} that has the same shape as a random BST of $n+1$ keys.

insertion order in T_{n+1} $\cong$ permutation of $[n+1]$

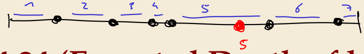
$\cong$ permutation of $[n]$ and the "last value" in $[n+1]$

$1\ 2\ 4\ 7\ 6\ 5 \mid 3 \ \rightsquigarrow\ (1\ 2\ 3\ 6\ 5\ 4,\ 3)$

Corollary 4.23

independent! — any continuous distr. would do

A BST built by inserting n i.i.d. $\mathcal{U}(0,1)$ r.v. has the shape of a random BST.



Theorem 4.24 (Expected Depth of leftmost leaf)

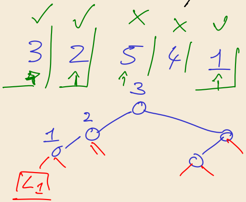
The *expected depth* (number of edges from root) of the leftmost external leaf (leaf for $-\infty$) in a random BST on $n \geq 1$ nodes is H_n.

$$= \sum_{i=1}^{n} \frac{1}{i} \sim \ln(n)$$

$$H_3 = 1 + \frac{1}{2} + \frac{1}{3} = \frac{6+3+2}{6} = \frac{11}{6}$$

Proof: depth(L_1) = # left-to-right minima in insertion sequence

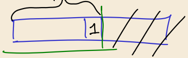
labels on the path to L_1 = left-to-right minima



$L2R_n$ = #l2r min in rp of $[n]$

$$= \sum_{i=1}^{n} X_i \qquad X_i = [\text{pos } i \text{ is a l2r min}]$$

$\cong$ rp of $[i]$



$$\Pr[X_i = 1] = \frac{1}{i}$$

$$\mathbb{E}[L2R_n] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} = H_n$$