Probabilistic analysis of algorithms: What’s it good for?

Conrado Martínez

Univ. Politècnica de Catalunya, Spain

February 2008
The goal

• Given some algorithm \( A \) taking inputs from some set \( I \), we would like to analyze the performance of the algorithm as a function of the input size (and possibly other parameters).
Why should we analyze algorithms?

- To predict the resources (time, space, ...) that the algorithm will consume
- To compare algorithm $A$ with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior
Why should we analyze algorithms?

- To predict the resources (time, space, ...) that the algorithm will consume
- To compare algorithm A with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior
Why should we analyze algorithms?

- To predict the resources (time, space, ...) that the algorithm will consume
- To compare algorithm A with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior
Why should we analyze algorithms?

- To predict the resources (time, space, ...) that the algorithm will consume
- To compare algorithm \(A\) with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior
Concepts & definitions

• The performance $\mu : I \to \mathbb{N}$ depends on each particular instance of the input.

• We have to introduce some notion of size: $|\cdot| : I \to \mathbb{N}$; we may safely assume that each $I_n = \{x \in I \mid |x| = n\}$ is finite.

• Worst-case:

$$\mu^{[\text{worst}]}(n) = \max\{\mu(x) \mid x \in I_n\}$$
To analyze "typical behavior" or the performance of randomized algorithms, we have to assume some probabilistic distribution on the input and/or the algorithm's choices; hence, we consider the performance as a family of random variables \( \{\mu_n\}_{n \geq 0}; \mu_n : I_n \rightarrow \mathbb{N} \)

**Average-case:**

\[
\mu^{[\text{avg}]}(n) = \mathbb{E}[\mu_n] = \sum_{k \geq 0} k \cdot \mathbb{P}[\mu_n = k]
\]
When we assume uniformly distributed inputs

$$P[x] = \frac{1}{\#I_n}, \text{ for all } x \in I_n$$

our problem is one of counting, e.g.,

$$E[\mu_n] = \frac{\sum_{x \in I_n} \mu(x)}{\#I_n}$$
One of the most important tools in the analysis of algorithms are generating functions:

\[ A(z, u) = \sum_{n \geq 0} \sum_{k \geq 0} \mathbb{P}[\mu_n = k] z^n u^k \]

For the uniform distribution

\[ [z^n u^k] A(z, u) = [z^n u^k] \frac{\sum_{n \geq 0} \sum_{k \geq 0} a_{n,k} z^n u^k}{[z^n] \sum_{n \geq 0} a_n z^n} = \frac{[z^n u^k] B(z, u)}{[z^n] B(z, 1)} \]

with \( a_{n,k} = \#\{x \in \mathcal{I} \mid |x| = n \land \mu(x) = k\} \) and \( a_n = \#\mathcal{I}_n \)
The equations before can be expressed symbolically

\[ B(z, u) = \sum_{x \in \mathcal{I}} z^{|x|} u^{\mu(x)} \]

The ratio of the \( n \)-th coefficients of \( B(z, u) \) and \( B(z, 1) \) is the probability generating function of \( \mu_n \)

\[ p_n(u) = \sum_{k \geq 0} \mathbb{P}[\mu_n = k] u^k = \frac{[z^n] B(z, u)}{[z^n] B(z, 1)} \]
Taking derivatives w.r.t. \( u \) and setting \( u = 1 \) we get the expected value, second factorial moment, ...

\[
A^{(r)}(z) = \frac{\partial^r A(z, u)}{\partial u^r} \bigg|_{u=1} \\
= \sum_{n \geq 0} \mathbb{E}[\mu_n^r] z^n
\]

For example,

\[
\mathbb{V}[\mu_n] = \mathbb{E}[\mu_n^2] + \mathbb{E}[\mu_n] - \mathbb{E}[\mu_n]^2 = [z^n] A^{(2)}(z) + [z^n] A(z) - ([z^n] A(z))^2
\]
The symbolic method

The symbolic method translates combinatorial constructions to functional equations over generating functions.

Example: Consider the counting generating function of a combinatorial class $A$:

$$A(z) = \sum_{n \geq 0} a_n z^n = \sum_{\alpha \in A} z^{\lvert \alpha \rvert}$$

If $A = B \times C$ then

$$A(z) = \sum_{\alpha \in A} z^{\lvert \alpha \rvert} = \sum_{(\beta, \gamma) \in B \times C} z^{\lvert \beta \rvert + \lvert \gamma \rvert} = \left( \sum_{\beta \in B} z^{\lvert \beta \rvert} \right) \left( \sum_{\gamma \in C} z^{\lvert \gamma \rvert} \right)$$

$$= B(z) \cdot C(z)$$
A dictionary of (labelled) combinatorial constructions and G.F.’s

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( { \epsilon } )</td>
<td>1</td>
</tr>
<tr>
<td>( { Z } )</td>
<td>( z )</td>
</tr>
<tr>
<td>( A + B )</td>
<td>( A + B )</td>
</tr>
<tr>
<td>( A \times B )</td>
<td>( A \cdot B )</td>
</tr>
<tr>
<td>Seq(( A ))</td>
<td>( \frac{1}{1 - A} )</td>
</tr>
<tr>
<td>Set(( A ))</td>
<td>exp(( A ))</td>
</tr>
<tr>
<td>Cycle(( A ))</td>
<td>( \log \frac{1}{1 - A} )</td>
</tr>
</tbody>
</table>
Complex analysis

Another important set of techniques comes from complex variable analysis. Under suitable technical conditions, if \( F(z) \) is analytic in a disk \( D = \{z \in \mathbb{C} | |z| < 1\} \) and has a single dominant singularity at \( z = 1 \) then

\[
F(z) \sim G(z) \implies [z^n]F(z) \sim [z^n]G(z)
\]

This is one of the useful transfer lemmas of Flajolet and Odlyzko (1990). Many other similar results are extremely useful when computing asymptotic estimates for the \( n \)-th coefficient of a generating function.
For example, if

\[ F(z) \sim G(z) \cdot \left(1 - \frac{z}{\rho}\right)^{-\alpha} + H(z) \]

as \( z \to \rho \), the dominant singularity of \( F(z) \), for some analytic \( G(z) \) and \( H(z) \) and \( \alpha \not\in \{-1, -2, \ldots\} \) then

\[ [z^n]F(z) \sim G(\rho)\rho^{-n} \frac{n^{\alpha-1}}{\Gamma(\alpha)} \left(1 + O(n^{-1})\right) \]
Complex analysis

In recent years, complex analysis techniques and perturbation theory have been used to prove powerful results such as Hwang’s quasi-power theorem, which allows one to prove the convergence in law to a Gaussian distribution of many combinatorial parameters in strings, permutations, trees, etc., as well as local limits and the speed of convergence.
A trivial example: Counting binary trees

A binary tree is either an empty tree (leaf) or a root together with two binary (sub)trees:

\[ B = \{\epsilon\} + \{Z\} \times B \times B \]

Hence the counting GF of binary trees is

\[ B(z) = 1 + zB^2(z) \]
Solving the equation before for $B(z)$ and since $B(0) = b_0 = 1$,

$$B(z) = \begin{cases} \frac{1 - \sqrt{1 - 4z}}{2z} & z \neq 0, \\ 1 & z = 0. \end{cases}$$

Extracting the $n$-th coefficient of $B(z)$ we find

$$[z^n] B(z) = \frac{\binom{2n}{n}}{n + 1} \sim 4^n \frac{n^{-3/2}}{\sqrt{\pi}}$$
2nd example: The average cost of building BSTs

A binary search tree $T$ for a set of elements $X$ contains some $y \in X$ at its root; its subtrees $L$ and $R$ are binary search trees recursively constructed for the sets $X^- = \{x \in X \mid x < y\}$ and $X^+ = \{z \in X \mid y < z\}$. Binary search trees (BSTs) are useful for fast lookup, and support both dynamic insertions and deletions.
To insert some new item $w$ into a BST, we compare $w$ to the element $y$ at the root of $T$. If $w < y$ then we insert $w$ recursively in the left subtree of $T$. If $y < w$ we insert recursively in the right subtree. If at any point we have to insert the element into an empty tree, we simply make $w$ the root of the new tree.
How much does it cost to build a BST of size $n$? Let $d(x, T)$ denote the depth (= edges from the root) of element $x$ in $T$. It is the number of elements with which $x$ was compared when we inserted it at some previous step. Hence, the cost $I(T)$ to build $T$ is

$$I(T) = \sum_{x \in T} d(x, T)$$

$I(T)$ is also called the internal path length of $T$. 

Probabilistic analysis of algorithms: What's it good for?
Let $n = |T|$. 

- If $T$ is a linked list (at most one non-empty subtree per node) then $I(T) = n(n - 1)/2$.
- If $T$ is perfectly balanced then $I(T) = \Theta(n \log n)$. 
In a random BST any element has the same probability of being its root; hence the probability that $|L| = k$ is $1/n$ for all $k$, $0 \leq k < n$.

$$\mathbb{P}[T] = \begin{cases} 
1 & \text{if } T \text{ is empty,} \\
\frac{\mathbb{P}[L] \cdot \mathbb{P}[R]}{|T|} & \text{otherwise.}
\end{cases}$$
\[ I(T) = \begin{cases} 
0 & \text{if } T \text{ is empty,} \\
I(L) + I(R) + |T| - 1 & \text{otherwise.} 
\end{cases} \]
\[ \mathbb{E}[I_n] = [z^n] I(z) \]

\[ I(z) = \sum_{T \in \mathcal{B}} \mathbb{P}[T] I(T) z^{|T|} \]

\[ = \sum_{(L,R) \in \mathcal{B} \times \mathcal{B}} \frac{\mathbb{P}[L] \mathbb{P}[R]}{|L| + |R| + 1} (I(L) + I(R) + |L| + |R|) z^{|L|+|R|+1} \]
\[
\frac{dI}{dz} = \sum_{(L,R) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[L] \mathbb{P}[R] (I(L) + I(R) + |L| + |R|) z^{|L|+|R|} \\
= \sum_{(L,R) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[L] \mathbb{P}[R] I(L) z^{|L|+|R|} \\
+ \sum_{(L,R) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[L] \mathbb{P}[R] I(R) z^{|L|+|R|} \\
+ \sum_{(L,R) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[L] \mathbb{P}[R] |L| z^{|L|+|R|} \\
+ \sum_{(L,R) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[L] \mathbb{P}[R] |R| z^{|L|+|R|}
\]
\[
\frac{dI}{dz} = 2 \sum_{(T_1, T_2) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[T_1] I(T_1) z^{T_1} \mathbb{P}[T_2] z^{T_2} \\
+ 2 \sum_{(T_1, T_2) \in \mathcal{B} \times \mathcal{B}} \mathbb{P}[T_1] |T_1| z^{T_1} \mathbb{P}[T_2] z^{T_2} \\
= 2 \left( \sum_{T_1 \in \mathcal{B}} \mathbb{P}[T_1] I(T_1) z^{T_1} \right) \cdot \left( \sum_{T_2 \in \mathcal{B}} \mathbb{P}[T_2] z^{T_2} \right) \\
+ 2 \left( \sum_{T_1 \in \mathcal{B}} \mathbb{P}[T_1] |T_1| z^{T_1} \right) \cdot \left( \sum_{T_2 \in \mathcal{B}} \mathbb{P}[T_2] z^{T_2} \right)
\]
\[ \sum_{T \in \mathcal{B}} \mathbb{P}[T] \cdot z^{|T|} = \sum_{n \geq 0} z^n = \frac{1}{1 - z} \]

\[ \sum_{T \in \mathcal{B}} \mathbb{P}[T] \cdot |T| \cdot z^{|T|} = \sum_{n \geq 0} n z^n = z \frac{d}{dz} \frac{1}{1 - z} = \frac{z}{(1 - z)^2} \]
\[
\frac{dI}{dz} = 2 \frac{I(z)}{1 - z} + \frac{2z}{(1 - z)^3}
\]

\[I(0) = 0\]

\[\Downarrow\]

\[I(z) = 2 \frac{\ln \left( \frac{1}{1-z} \right)}{(1 - z)^2} - \frac{2z}{(1 - z)^2}\]
\[ \mathbb{E}[I_n] = [z^n]I(z) = 2(n - 1)H_n \]
\[ \sim 2n \ln n + 2n\gamma + O(\log n) \]

\[ H_n = \sum_{1 \leq k \leq n} \frac{1}{k} \sim \ln n + \gamma + O(n^{-1}) \]
3rd example: Insertion in $K$-d trees
3rd example: Insertion in $K$-d trees
3rd example: Insertion in $K$-d trees
3rd example: Insertion in $K$-d trees
3rd example: Insertion in $K$-d trees
3rd example: Insertion in $K$-d trees
**Insertion in relaxed $K$-d trees**

```cpp
rkdtt insert(rkdtt t, const Elem& x) {
    int n = size(t);
    int u = random(0,n);
    if (u == n)
        return insert_at_root(t, x);
    else { // t cannot be empty
        int i = t -> descr;
        if (x[i] < t -> key[i])
            t -> left = insert(t -> left, x);
        else
            t -> right = insert(t -> right, x);
        return t;
    }
}
```
Deletion in relaxed $K$-d trees

```c
rkdt delete(rkdt t, const Elem& x) {
    if (t == NULL) return NULL;
    if (t -> key == x)
        return join(t -> left, t -> right);

    int i = t -> descr;
    if (x -> key[i] < t -> key[i])
        t -> left = delete(t -> left, x);
    else
        t -> right = delete(t -> right, x);
    return t;
}
```
Split: Case #1
Split: Case #1

Probabilistic analysis of algorithms: What’s it good for?

Univ. von Stellenbosch
Split: Case #2
Split: Case #2
Split: Case #2

Probabilistic analysis of algorithms: What’s it good for?

Univ. von Stellenbosch
Split: Case #2
Analysis of split/join

- $s_n = \text{avg. number of visited nodes in a split}$
- $m_n = \text{avg. number of visited nodes in a join}$

$$s_n = 1 + \frac{2}{nK} \sum_{0 \leq j < n} \frac{j + 1}{n + 1} s_j + \frac{2(K - 1)}{nK} \sum_{0 \leq j < n} s_j$$

$$+ \frac{K - 1}{K} \sum_{0 \leq j < n} \pi_{n,j} m_j,$$

where $\pi_{n,j}$ is probability of joining two trees with total size $j$. 
Analysis of split/join

- $s_n =$ avg. number of visited nodes in a split
- $m_n =$ avg. number of visited nodes in a join

\[
s_n = 1 + \frac{2}{nK} \sum_{0 \leq j < n} \frac{j + 1}{n + 1} s_j + \frac{2(K - 1)}{nK} \sum_{0 \leq j < n} s_j
+ \frac{K - 1}{K} \sum_{0 \leq j < n} \pi_{n,j} m_j,
\]

where $\pi_{n,j}$ is probability of joining two trees with total size $j$. 
Analysis of split/join

- $s_n = \text{avg. number of visited nodes in a split}$
- $m_n = \text{avg. number of visited nodes in a join}$

\[
s_n = 1 + \frac{2}{nK} \sum_{0 \leq j < n} \frac{j + 1}{n + 1} s_j + \frac{2(K - 1)}{nK} \sum_{0 \leq j < n} s_j + \frac{K - 1}{K} \sum_{0 \leq j < n} \pi_{n,j} m_j,
\]

where $\pi_{n,j}$ is probability of joining two trees with total size $j$. 
The recurrence for $s_n$ is

$$s_n = 1 + \frac{2}{nK} \sum_{0 \leq j < n} \frac{j + 1}{n + 1} s_j + \frac{2(K - 1)}{nK} \sum_{0 \leq j < n} s_j$$

$$+ \frac{2(K - 1)}{nK} \sum_{0 \leq j < n} \frac{n - j}{n + 1} m_j,$$

with $s_0 = 0$.

The recurrence for $m_n$ has exactly the same shape with the rôles of $s_n$ and $m_n$ interchanged; it easily follows that $s_n = m_n$. 

Probabilistic analysis of algorithms: What’s it good for?

Univ. von Stellenbosch
Define

\[ S(z) = \sum_{n \geq 0} s_n z^n \]

The recurrence for \( s_n \) translates to

\[
z \frac{d^2 S}{dz^2} + 2 \frac{1 - 2z}{1 - z} \frac{dS}{dz} \]

\[
- 2 \left( \frac{3K - 2}{K} - z \right) \frac{S(z)}{(1 - z)^2} = \frac{2}{(1 - z)^3},
\]

with initial conditions \( S(0) = 0 \) and \( S'(0) = 1 \).
The homogeneous second order linear ODE is of hypergeometric type.

An easy particular solution of the ODE is

$$-\frac{1}{2} \left( \frac{K}{K-1} \right) \frac{1}{1 - z}$$
Theorem

The generating function $S(z)$ of the expected cost of split is, for any $K \geq 2$,

$$S(z) = \frac{1}{2} \frac{1}{1 - \frac{1}{K}} \left[ (1 - z)^{-\alpha} \cdot _2F_1 \left( \begin{array}{c} 1 - \alpha, 2 - \alpha \\ 2 \end{array} \bigg| z \right) - \frac{1}{1 - z} \right],$$

where $\alpha = \alpha(K) = \frac{1}{2} \left( 1 + \sqrt{17 - \frac{16}{K}} \right)$.
Theorem

The expected cost $s_n$ of splitting a relaxed $K$-d tree of size $n$ is

$$s_n = \eta(K) n^{\phi(K)} + o(n),$$

with

$$\eta = \frac{1}{2} \frac{1}{1 - \frac{1}{K}} \frac{\Gamma(2\alpha - 1)}{\alpha \Gamma^3(\alpha)},$$

$$\phi = \alpha - 1 = \frac{1}{2} \left( \sqrt{17 - \frac{16}{K}} - 1 \right).$$
Probabilistic analysis of algorithms: What's it good for?

Plot of $\phi(K)$
Plot of $\eta(K)$