

Randomized Algs (CS 574)

algorithms that can make random choices

Why randomization?

- faster or simpler algs in countless appl'ns
- fundamental to theoretical CS
- derandomization

⋮

possible topics

Example 1: quicksort & quickselect

quicksort(a_1, \dots, a_n):

1. if $n \leq 1$ return
2. pick "pivot" x AT RANDOM from $\{a_1, \dots, a_n\}$
3. partition into $L = \{a_i : a_i \leq x\}$
 $R = \{a_i : a_i > x\}$
4. return quicksort(L) * quicksort(R)

$O(n)$
time

original version: $x = a_1$

usually fast for rand. input

but bad input $n, n-1, \dots, 1$ ←

$$\Rightarrow T(n) = T(n-1) + \cancel{T(1)} + O(n)$$

$$\Rightarrow O(n^2)$$

idea: randomize!

no bad input

expected runtime $T(n) = O(n \log n)$
(pf later...)

expected runtime $T(n) = O(n \log n)$
(pf later...)

quickselect (a_1, \dots, a_n, k):

// find the k^{th} smallest in $\{a_1, \dots, a_n\}$

⚡ there is complicated $O(n)$ -time
det. algm by Blum-Floyd-Pratt-Rivest-Tarjan '75

1. }
2. } Same
3. }

4, if $k \leq |L|$, return quickselect(L, k)
else quickselect($R, k - |L|$)

Analysis of randomized quickselect:

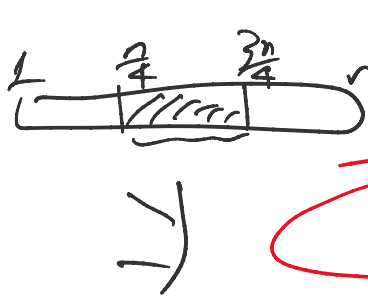
$$T(n) \leq \max \{T(|L|), T(n - |L|)\} + O(n).$$

$|L| = \text{rank of } x$

(ideally, $|L| = \frac{n}{2} \Rightarrow T(n) = T(\frac{n}{2}) + O(n)$
 $\Rightarrow O(n \log n)$)

Call x good if x has rank $\in [\frac{n}{4}, \frac{3n}{4}]$.

$\Pr[x \text{ is good}] = \frac{\# \text{ good elems}}{n}$

\Rightarrow  $= \frac{\frac{3n}{4} - \frac{n}{4}}{n} = \frac{1}{2}$.

\Rightarrow expected # iterations till we see a pivot
 ≤ 2 ("geometric distribution")

\Rightarrow expected runtime

$$T(n) \leq T(\frac{3}{4}n) + 2 \times O(n)$$

by linearity of expectation
 $E[X+Y] = E[X] + E[Y]$

by linearity of E
 $E[X+Y] = E[X] + E[Y]$

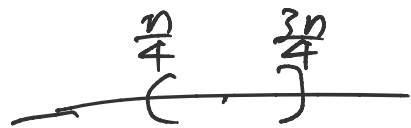
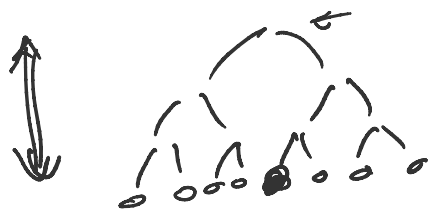
$$\Rightarrow T(n) = \boxed{O(n)}$$

(more careful analysis: $\leq (2+2\ln 2)n$ comps.)

$$T(n) = \frac{1}{n} \sum_{i=1}^n T(i) + O(n) \approx 3.4n$$

Analysis of randomized quicksort:

expected # levels of recursion till we set a good pivot is ≤ 2 .



$$\Rightarrow \text{expected runtime } T(n) \leq \sum T(n_i) + 2 \times O(n)$$

where $n_i \leq \frac{3n}{4}$, $\sum n_i = n$.

$$\Rightarrow T(n) = \boxed{O(n \log n)}$$

(more careful analysis: $\leq 2n \ln n$ ^{expected} comps)

Note - input not rand, algn makes rand. choices

- 2 types of rand. algn's:

- "Las Vegas" - always correct runtime depends on rand choices ^{analyze expected runtime}
- "Monte Carlo" - correctness depends on rand. choices
 \hookrightarrow analyze prob. of error

Example 2: Primality Testing

Given large number N with n bits
 is N prime or composite?

10101011010101
 $(N \sim 2^n)$

... to $\sqrt{N-1}$.

Trivial alg'm: for $a = 2$ to $\sqrt{N}-1$,
if N is div of a , return composite

time $O(\sqrt{N} \cdot \frac{n^2}{p}) = O(2^{n/2} \cdot n^2)$
still exponential!!

Wilson's Thm (1700s)

N prime $\iff (N-1)! \equiv -1 \pmod{N}$
too slow!

Fermat's Little Thm (1640)

N prime $\implies \forall a \in \{1, \dots, N-1\}$,
 $a^{N-1} \equiv 1 \pmod{N}$.

$a^N = (a^{N/2})^2$

Can be tested in $O(\log N)$ mult. of n -bit #'s
 $O(n)$ $\xrightarrow{\text{time}} O(n^2)$
 $= O(n^3)$ time
by repeated squaring

but there are bad inputs ("Carmichael numbers")
that are composite but most a 's don't work...

Modified Fermat's Thm Given $a \in \{1, \dots, N-1\}$,

If $\left\{ \begin{array}{l} a^{N-1} \not\equiv 1 \pmod{N} \\ \text{or for some } k = (N-1)/2^i \\ a^{2^k} \equiv 1 \text{ but } a^{2^{k-1}} \not\equiv \pm 1 \pmod{N} \end{array} \right.$,
then N is composite (and a is called "witness")
can also be tested in $O(n^3)$ time
 $\left(\begin{array}{l} \text{if } N \text{ prime,} \\ x^2 \equiv 1 \pmod{N} \\ \implies x \equiv \pm 1 \pmod{N} \end{array} \right.)$

Miller's Thm ('76)

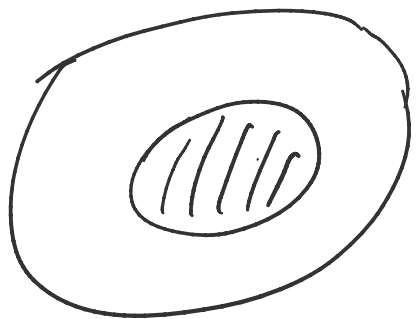
If N is composite, $n, 2, n-1 = 2n^2$

If N is composite,
then \exists witness $a \leq \underline{2 \log^2 N}$...
 \Rightarrow polytime det. alg'm!!

assuming Extended Riemann Hypothesis
& still wide open!!

Rabin's Thm ('76)

If N is composite,
then # witnesses $a \geq \frac{3}{4}N$.



PICK a random a

⋮