Randomized Algorithms (CS 574)

algorithms that can make random choices

Why randomization?
- faster or simpler alg’s in countless appl’n’s
- fundamental to theoretical CS
- derandomization

Possible topics

Example 1: quicksort & quickselect

quicksort \((a_1, \ldots, a_n)\):

1. if \(n = 1\) return
2. pick “pivot” \(X\) \(\text{AT RANDOM from } \{a_i: a_i \leq X\}\)
3. partition into \(L = \{a_i: a_i \leq X\}\)
   \(R = \{a_i: a_i > X\}\)
4. return quicksort\((L) \ast\) quicksort\((R)\)

\(O(n)\) time

Original version: \(X = a_1\)
usually fast for \(\text{rand. input}\)
but bad input \(n, n-1, \ldots, 1\)
\(\Rightarrow\)  \(T(n) = T(n-1) + T(1) + O(n)\)
\(\Rightarrow\)  \(O(n^2)\).

Idea: randomize!
no bad input
expected runtime  \(T(n) = O(n \log n)\)
\(\text{(pf later...)}\)
expected runtime \[ T(n) = O\left(\frac{3n}{4}\right) \]

quickselect \( (a, \ldots, an, k) \):

1. // find the \( k \)-th smallest in \( \{a, \ldots, an\} \)
2. \{ Same \}
3.
4. if \( k \leq \frac{n}{4} \), return quickselect \( (L, k) \)
else quickselect \( (R, k - \frac{n}{4}) \)

Analysis of randomized quickselect:

\[ T(n) \leq \max \{ T(\frac{n}{4}), T(n - \frac{n}{4}) \} + O(n) \]

\( \frac{n}{4} = \text{rank of } x \)

\( \frac{n}{2} \) \( \Rightarrow \) \( T(n) = T(\frac{n}{2}) + O(n) \)

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

\[ \Pr( x \text{ is good} ) = \frac{\# \text{ good elements}}{n} = \frac{\frac{3n}{4} - \frac{n}{4}}{n} = \frac{1}{2} \]

\( \Rightarrow \) expected number of iterations till we see a pivot

\( \leq 2 \) ("geometric distribution")

\( \Rightarrow \) expected runtime

\[ T(n) \leq T\left(\frac{3n}{4}\right) + 2 \times O(n) \]

by linearity of expectation

\[ \mathbb{E}(X + Y) = \mathbb{E}(X) + \mathbb{E}(Y) \]
by linearity of expectation
\[ E(X+Y) = E(X) + E(Y) \]

\[ T(n) = \Theta(n) \]

( 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 \( \leq 2 \).

\[ T(n) \leq \sum_{i=1}^{n} T(n_i) + 2 \times O(n) \]

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

\[ T(n) = O(n \log n) \]

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

Note - input not rand., algo makes rand. choices
- 2 types of rand. algms:
  - "Las Vegas": always correct runtime depends on rand. choices
  - "Monte Carlo": correctness depends on rand. choices
    - analyze prob. of error

Example 2: Primality Testing

Given large number \( N \) with \( n \) bits
Is \( N \) prime or composite?

Given \( N \), check \( 2 \) to \( \sqrt{N} \).
Trivial alg/m: for $a = 2$ to $\sqrt{N} - 1$, if $N$ is div of $a$, return composite

time $O(\sqrt{N \cdot \phi(n)}) = O(2^{\frac{3}{2}} \cdot n^2)$
still exponential!!

Wilson's Thm (1700s)
$N$ prime $\iff$ $(N-1)! \equiv -1 \pmod{N}$
$\quad$ too slow!

Fermat's little Thm (1640)
$N$ prime $\Rightarrow$ $\forall a \in \{1, \ldots, N-1\}$,
$a^{N-1} \equiv 1 \pmod{N}$.

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

Can be tested in $O((\log N) \text{ mult. of } \frac{m \cdot \text{bit mit.}}{O(n^2)}$ by repeated squaring

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

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

$\begin{cases} a^{N-1} \not\equiv 1 \pmod{N} \\ \text{or for some } k = (N-1)/2^i, \ a^{2^k} \equiv 1 \text{ but } a^{2^{k+1}} \not\equiv 1 \pmod{N} \end{cases}$

then $N$ is composite
(and $a$ is called "witness")

Miller's Thm ('76)
If $N$ is composite, $-a, a^2 \equiv 2N$
If \( N \) is composite,
then \( \exists \) witness \( a \leq 2 \log^2 N \) ...

\[ \Rightarrow \] polynomial time det. alg'm!!

Assuming extended Riemann hypothesis

&

Still wide open!!

Rabin's Thm ('76)

If \( N \) is composite,
then \# witnesses \( a \Rightarrow \frac{3}{4} N \).

PICK a random \( a \).