

Lecture 0 Midterm 1 Review: 9.30.08

Lecturer: Satish Rao

Scribe: Rao

0.1 Chapter 0.

- Asymptotic notation: big O , Θ , big Ω , little o , little ω .

0.2 Chapter 1.

- Modular arithmetic. Can use equivalents in calculations for plus and times. Prove this for times: $a = x \bmod n$, $b = y \bmod n$. That is $a = x + in$ and $b = y + jn$, thus $ab = xy + xjn + yin + injn^2 = xy + n(xj + yin + injn)$, or $ab = xy \bmod n$.
- Modular exponentiation: $\text{modexp}(a, b) \dots x = \text{modexp}(a, b/2)$.
Return $x^2 * x^{b\%2}$.
- Modular inverses from Euclid's theorem: for (a, b) there exist integer i, j such that $\text{gcd}(a, b) = ax + by$.
Runtime for Euclid's? (Basic idea: $\text{gcd}(a, b) = \text{gcd}(a, a \bmod b)$. Use formula recursively.)
- Show that every a not divisible by p has an inverse mod prime p .
- Fermat's little theorem. Can use equivalents in exponents. For a prime, $a^{p-1} \bmod p = 1$. Proof: $ai \neq aj \bmod p$ unless $j = i$. Thus, $a^{p-1}(p-1)! = (p-1)! \bmod p$.
- Prime number theorem: random number with b bits has probability $\Omega(1/b)$ of being prime. Probability that x is prime is approximately $1/\ln x$.
- RSA: Private-key: $(N = p, q, e)$. Public-key $(N, d = e^{-1} \bmod (p-1)(q-1))$. Encode m : $m^e \bmod N$. Decode x : $x^d \bmod N$.
- RSA decoding gives back the original message since $a^{(p-1)(q-1)} = 1 \bmod N$ for a not divisible by p or q . Euler's theorem.
- Hashing: universal hash function $h() \rightarrow [1, n]$: for all x, y , $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] = 1/n$. Example.

0.3 Chapter 2

- Karatsuba's algorithm:

$$(a_{\text{top}}2^{n/2} + a_{\text{bot}})(b_{\text{top}}2^{n/2} + b_{\text{bot}}) = 2^n a_{\text{top}}b_{\text{top}} + (a_{\text{top}}b_{\text{bot}} + a_{\text{bot}}b_{\text{top}})2^{n/2} + a_{\text{bot}}b_{\text{bot}}.$$

Four multiplies of $n/2$ bit words. $T(n) = 4T(n/2) + n = O(n^2)$.

- But, $(a_{top} + a_{bot})(b_{bot} + b_{top}) = a_{top}b_{top} + b_{bot}a_{bot} + (a_{top}b_{bot} + a_{bot}b_{top})$. The parenthesized quantity is the middle quantity. Thus, the recurrence is $T(n) = 3T(n/2) + n = O(n^{\log_2 3})$.
- Master's theorem: $T(n) = aT(n/b) + n^c$. If $c > \log_b a$, $T(n) = O(n^c)$, if $c = \log_b a$, $T(n) = O(n^c \log n)$, and if $c < \log_b a$ then $T(n) = n^{\log_b a}$.
- Recursion tree. Number of leaves is $O(n^{\log_b a})$. Depth is $\log_b n$. Cost at root is n^c . Cases are root dominates, all equal, leaves dominate.
- Mergesort. Sorting requires $\Omega(n \log n)$ time.
- Matrix multiply. Can do this with 7 submatrix multiplies. Thus, recursion is $T(n) = 7T(n/2) + O(n^2)$. This is $O(n^{\log_2 7})$. Less than $O(n^3)$.
- FFT. What are the n roots of unity. Definition of the FFT: $FFT(a_1, \dots, a_n)$ is $A(\omega^i)$. For multiplying polynomials: $FFT^{-1}(FFT(a) \cdot FFT(b))$.
- Algorithm: Take FFT of odd and even. Then combine. Each subproblem used for two outputs. Recurse. FFT procedure. $T(n) = 2T(n/2) + O(n)$.

0.4 Chapter 3

- Matrix representation: fast lookup? $O(n^2)$ space. Graph. Adjacency list representation: are (i, j) connected? Space? Get a list of neighbors?
- Depth first search.
- Undirected graphs. Connected components. Properties: no cross edge. Only tree edges and backward edges.
- Directed graphs. Pre, post ordering. (u, v) is back edge $[pre[u], post[u]]$ contained in $[pre[v], post[v]]$. The other way, it is forward or tree edge. If non-intersecting: cross edge, and $post[v]$ is before $pre[u]$.
- Topological sort of dag. Two algorithms: Repeatedly find source. or Reverse post ordering number: since for edge (u, v) , $post[u] \geq post[v]$. Why does this work?
- Strongly connected. Path from i to j and j to i . Partitions graph: if i in different components, then the union is strongly connected. DAG metagraph of components.
- Algorithm: dfs of reverse graph. Dfs ordering by reverse post ordering number: Each explore finds current sink component.