M, 11/25/19

Fall'19 CSCE 629

# Analysis of Algorithms


Fang Song

Texas A&M U


## Lecture 34

- Randomized algorithms
- Final review

# Randomized quicksort

- Pick the pivot randomly

```
Rand-QuickSort(A):
 if (array A has zero or one element)
    Return
 Pick pivot p ∈ A uniformly at random
 (L, M, R) ← PARTITION − 3 − WAY(A, p)
 Rand-QuickSort(L)
 Rand-QuickSort(R)
```

$\longrightarrow O(n)$

$\longrightarrow T(i)$

$\longrightarrow T(n-i-1)$

**Theorem.** The expected number of compares to quicksort an array of $n$ distinct elements is $O(n\log n)$.

# Probability 102

- Random variable $X: \Omega \to \mathbb{N}$
  - Assign each outcome a number
  - "$X = x$" is the event $E := \{\omega \in \Omega: X(\omega) = x\}$
  - Independent random variables:

$X, Y$ are indep. iff. for all possible $x$ and $y$, events $X = x$ and $Y = y$ are indep.

- Expectation: a weighed average
  - $\mathbb{E}[X] = \sum_{z \in Z} \Pr(X = z) \cdot z$
  - Linearity: $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ (independence NOT needed)
- Ex. $\Omega =$ roll 4 dices independently
  - Let $X$ be the sum of 4 rolls; $X_i$ be value of $i$th roll, $i = 1, \ldots, 4$
  - $\mathbb{E}[X] = \mathbb{E}[X_1 + \cdots + X_4] = 4 \cdot \mathbb{E}[X_1] = 4 \times 3.5 = 14$

# Randomized quicksort: analysis

Theorem. The expected number of compares to quicksort an array of $n$ distinct elements is $O(n \log n)$.

Assume $A = \{z_1, z_2, \dots, z_n\}, z_1, < z_2 < \cdots < z_n$

Observation: any pair $z_i$ & $z_j$ $(i < j)$ is compared at most once

- How many comparisons? $X :=$ total number of comparisons
  - Indicator variable: $X_{ij} := \begin{cases} 1, \text{if } z_i \text{ is compared to } z_j \\ 0, \text{otherwise} \end{cases}$

$$\Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right]$$
$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1]$$

Linearity

# Randomized quicksort: analysis cont'd

**Theorem.** The expected number of compares to quicksort an array of $n$ distinct elements is $O(n \log n)$.

$$\boxed{\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1]} \qquad X_{ij} := \begin{cases} 1, \text{if } z_i \text{ is compared to } z_j \\ 0, \text{otherwise} \end{cases}$$

- When two items are compared?

$$2 \quad 1 \quad 3 \quad \underbrace{4 \quad 6 \quad 5 \qquad \boxed{7} \qquad 9 \quad 8}_{\text{No comparison across these two groups}} \quad 10$$

- Observation: $z_i$ & $z_j$ compared iff. $z_i$ or $z_j$ was the first chosen as a pivot from $Z_{ij} = \{z_i, z_{i+1}, \dots, z_j\}$

# Randomized quicksort: analysis cont'd

- Observation: $z_i$ & $z_j$ compared iff. $z_i$ or $z_j$ was the first chosen as a pivot from $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

$$
\begin{aligned}
&\Pr[X_{ij} = 1] \\
&= \Pr[z_i \text{ \& } z_j \text{ compared}] = \Pr[z_i \text{ or } z_j \text{ is 1st pivot chosen from } Z_{ij}] \\
&= \Pr[z_i \text{ is 1st pivot from } Z_{ij}] + \Pr[z_j \text{ is 1st pivot from } Z_{ij}] \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}
\end{aligned}
$$

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \leq 2 \cdot \sum_{i=1}^{n-1} \boxed{\sum_{k=1}^{n} \frac{1}{k}} = O(n \cdot \log n)$$

Harmonic series

# LP relaxation for set cover

(Set cover ILP $\Pi$) Min $\sum_{i=1}^{m} x_i$
Subject to:
$$\sum_{i:u\in S_i} x_i \geq 1, \ \forall u \in U$$
$$x_i \in \{0,1\}, \qquad \forall i \in \{1, \ldots, m\}$$

(Set cover LP $\Sigma$) Min $\sum_{i=1}^{m} x_i$
Subject to:
$$\sum_{i:u\in S_i} x_i \geq 1, \ \forall u \in U$$
$$0 \leq x_i \leq 1, \forall i \in \{1, \ldots, m\}$$

Let $x^*$ be an optimal soln. for LP $\Sigma$
& optimal value $\mathrm{OPT} = \sum_i x_i^*$

☹ $x_i \coloneqq \lfloor x_i^* \rceil$

- Randomized rounding: set $x_i = 1$ with probability $x_i^*$

$$\mathbb{E}[\sum_{i=1}^{m} x_i] = \sum_{i=1}^{m} \mathbb{E}[x_i] = \sum_{i=1}^{m} x_i{}^*$$

- But is it feasible? [Further analysis on Panigrahi's notes]

Theorem. There is a poly-time randomized algorithm achieving $O(\log n)$ expected approximation ratio, except w. probability $O(1/n)$.

# Final exam

- When & where
  - December 10, 2019, Tuesday 8 am - 10 am @ HRBB 113
- What
  - Comprehensive, slightly more focused on 2nd half
- How
  - Similar format as mid-term: short-answer questions and algorithm designs
  - Closed book
  - 2 pages (letter-size) 2-sided cheat sheet permitted
  - No credit for unintelligible hand writing
  - More on practice exam

**You've accomplished a lot!**

**Be proud of yourselves!**

# Principal questions

0 What problem to solve?

be as precise as possible

1 Is the algorithm correct?

2 How much resource it costs?

(time, space, …)

3 Can we do better?

DESIGN

ANALYSIS

# Major topics

- Basics: asymptotic, graphs (BFS/DFS), data structures
- Algorithmic techniques
  1. Divide-&-Conquer
  2. Dynamic Programming
  3. Greedy
  4. Network flow & linear programming
  5. Randomization
  
  ★ Reduction
- Computational intractability: P, NP, NPC, approximation

1st half

2nd half

# 1. Divide-&-Conquer

- Idea
  - Divide into independent subproblems – recurse - combine
- Examples
  - Merge sort $O(n \log n)$
  - Fast multiplication $O(n^{1.59})$ [Karatsuba60]; $O(n \log n)$ [HarveyHoeven19]
  - Matrix multiplication $O(n^{2.81})$ [Strassen69]; $O(n^{2.376})$ [CoppersmithWinograd90];
  - Exponentiation $O(n \log n)$
  - Quick sort $O(n^2)$ worst-case; Expected $O(n \log n)$ random pivoting
- Analysis.
  - Solving recurrence: $T(n) = aT(n/b) + f(n)$
  - Recursion tree & Master theorem

# 2. Dynamic programming

- Idea
  - Divide into overlapping subproblems – smart recurse by memoization
  - Usually bottom-up iteration (topological order of implicit DAG)
- Examples
  - Fibonacci $O(n)$
  - Longest increasing subsequence $O(n^2)$
  - Weighted interval scheduling $O(n \log n)$
  - Matrix-chain multiplication $O(n^3)$
  - Longest common subsequence (aka Edit Distance) $O(mn)$
  - Shortest path (w. negative lengths) $O(mn)$ [Bellman-Ford]

# 3. Greedy

- Idea
  - Special case of DP: when lucky, lazy choice works
- Examples
  - Shortest path (w. non-negative lengths) $O((m+n)\log n)$ [Dijkstra]
  - Interval scheduling (weight = 1) $O(n\log n)$
  - Interval partitioning $O(n\log n)$
  - Minimum spanning tree $O(m\log n)$ [Kruskal]; $O((m+n)\log n)$ [Prim]
- Warning! 0 credit in exam without correctness proofs

**Detour**

- data structures [Prioirity Queue, Union Find]
- amortized analysis

# 4. Network flow - Linear programming

**Network flow $\leq$ Linear programming**

- Analytical
  - Max-Flow $\equiv$ Min-Cut
- Algorithms
  - Augmenting path: $O(mnC)$ [Ford-Fulkerson]
  - Capacity scaling: $O(m^2 \log C)$
  - In exam: quote $O(mn)$
- Applications
  - Bipartite perfect matching

- Analytical
  - Duality: OPT(Primal) = OPT(Dual)
- Algorithms
  - Simplex [efficient in practice/ but not poly-time worst-case]
  - Ellipsoid [poly-time but not practical]
  - Interior point [poly-time & practical]
- Warning: don't reduce to LP unless stated explicitly

# 5. Randomization

- Idea
  - Make random choices to get correct answers with high probability in (expected) poly-time
- Examples
  - Contention resolution
  - Randomized quicksort
  - Randomized rounding for LP relaxation
- Important probabilistic tools
  - Union bound
  - Linearity of expectation
  - Reducing errors (tail inequalities)

# Computational intractability

- Classify problems by "hardness"
  - **P**: feasible problems (solvable in poly-time)
  - **NP**: ∃ poly-time certifier verifying a solution

**P vs. NP?**

- Reduction: relating hardness ($A \leq B \Rightarrow A$ no harder than $B$)
  - Cook reduction [aka poly-time reduction]
  - Karp reduction [aka poly-time transformation]
- **NP−complete**: 1) $A \in \mathbf{NP}$ & 2) $\forall B \in \mathbf{NP}, B \leq_{Karp,P} A$ [$aka$ **NP**-hard]
  - Circuit−SAT is **NPC** [Cook-Levin]
  - Circuit−SAT ≤ 3−SAT ≤ INDEPENDENT−SET ≤ VERTEX−COVER ≤ SET−COVER ≤ IntegerLP [Karp]
  - 3−SAT ≤ HAM−CYCLE

# Coping with NPC: approximation algorithms

- Greedy
  - Vertex cover & set cover

- LP relaxation
  - Threshold rounding: 2-approx. vertex cover
  - Randomized rounding: $O(\log n)$-approx. vertex cover

★ Know the facts and ideas! Details less important

# FAQs

- How can I come up with the ideas in 2hrs?
  - Hints given sometimes, and/or subproblems to guide your way
- How should I study for it?
  - Review the fundamentals
  - Reproduce the algorithms & analysis for all problems you've seen (lecs, text, hw…)
  - Practice exam: emulate a real exam environment
- More?