Lecture #22
Randomized Algorithms

- When we can’t be both fast and correct, settle for:
  - Correct, and probably fast (Las Vegas)
  - Fast, and probably correct (Monte Carlo)
Quick Review of Probability (CS70)

• Random variable $X$ takes values $x_1, x_2, \ldots$ with probabilities $P(X = x_i) = p_i \geq 0$, $\sum p_i = 1$

• Roll fair die, $X \in \{1, \ldots, 6\}$ each with $p_i = \frac{1}{6}$

• Expectation of $X$ = "average value of $X$"

  $= \mathbb{E}(X) = \sum x_i p_i$

• Roll fair die, $\mathbb{E}(X) = \frac{1}{6}(1+2+\cdots+6) = 3.5$

• If $X$ and $Y$ are random variables, $\mathbb{E}(aX+bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$

• Markov's Inequality: If $X \geq 0$ then
Randomized Quicksort of $A(1:n)$

- Assume w.l.o.g. that all $A(i)$ distinct
- Else lexicographic: $(A(i), i) < (A(j), j)$ if $A(i) < A(j)$ else if $i < j$

Quicksort $(A(1:n))$:

if $n = 1$ return $A(1)$, else
pick uniformly random pivot $i \in \{1, \ldots, n\}$
$L \leftarrow \{i : A(i) < A(pivot)\}$
$R \leftarrow \{i : A(i) > A(pivot)\}$
return $(\text{Quicksort}(A(L)), A(pivot), \text{Quicksort}(A(R)))$

Worst case:
Best case:
Hope:
Proof that $E T(n) = O(n \log n)$ (1/2)

- $T(n) = \Theta(\# \text{ comparisons})$
- $X_{ij} = 1$ if $i^{th}$ smallest entry compared to $j^{th}$ smallest, else $0$
- Each $X_{ij}$ is a random variable so

- How is $X_{ij}$ determined?
  - Let $a_1 < a_2 < \ldots < a_n$ be sorted array
Proof that $\mathbb{E}[T(n)] = O(n \log n)$ (2/2)

- $T(n) = \Theta(\# \text{comparisons})$
- $X_{ij} = 1$ if $i^{th}$ smallest entry compared to $j^{th}$ smallest, else 0
  - $\# \text{comparisons} = \sum_{i<j} X_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$
- Each $X_{ij}$ is a random variable so
  $$\mathbb{E}(\# \text{comparisons}) =$$

- Markov inequality:
Freivald's Algorithm

• Given nxn matrices A, B, C, test whether \( C = A \cdot B \) faster than multiplying \( A \cdot B \)

• Intuition:

• Thm:

• Cor:
Freivald’s Algorithm (2/2)

- Thm: if $x \in \Sigma^0, 1^*$ chosen with each entry $x_i$ independent, $P(x_i = 0) = \frac{1}{2} = P(x_i = 1)$, and $C \neq A \cdot B$, then $P(Cx \neq ABx) \geq \frac{1}{2}$

- Proof:
Karger's Global Mincut Algorithm (1/6)

- Def: A cut of an undirected graph $G(V,E)$ is a partition $V = S \cup \overline{S}$, $S \cap \overline{S} = \emptyset$, $S \neq \emptyset$, $\overline{S} \neq \emptyset$

- Def: Size of a cut = # edges connecting $S, \overline{S}$
  $$= |E \cap (S \times \overline{S})|$$

- Def: Global Mincut (GMC) = $(S, \overline{S})$ minimizing size
Karger's Global Mincut Algorithm (2/6)

- Def: Given \( G(V,E) \), contract(e), e=(u,v)

  - means 1)
  - 2)
  - 3)

- Karger's Algorithm:
Karger's Global Mincut Algorithm (3/6)

- Karger's Algorithm:
  
  for $i = 1$ to $|V| - 2$
  
  pick random edge $e$, contract $(e)$
  
  return cut determined by last 2 vertices
Karger's Global Mincut Algorithm (4/6)

• Karger's Algorithm:
  
  for i = 1 to |V| - 2 ... n=|V| below
  
    pick random edge e, contract(e)
  
  return cut determined by last 2 vertices

• Fact: Karger returns \(GMC (S, S)\)
  
  \(\iff\) never contracts an edge in GMC

• What is probability that Karger
  
  never contracts an edge in GMC?
Karger's Global Mincut Algorithm (5/6)

- Karger's Algorithm:
  
  for $i = 1$ to $|V| - 2$

  pick random edge $e$, contract $(e)$

  return cut determined by last 2 vertices

- $m_i =$ number of edges, $k =$ size of GMC ($\#$ edges from $S$ to $\bar{S}$)

- 

- 

- 

- 

- 

- 

-
Karger's Global Mincut Algorithm (6/6)

- **Karger's Algorithm:**
  
  for $i = 1$ to $|V| - 2$
  
  pick random edge $e$, contract($e$)
  
  return cut determined by last 2 vertices

- $P(\text{Karger gets right answer}) \geq 1/(2^n)$
One more “hot topic”

• Lots of current research on randomized linear algebra algorithms
  • Least squares problems $\min_x \|Ax-b\|_2$
  • PCA ( Principle Component Analysis )
  • SVD ( Singular Value Decomposition )

• see “References for Randomized Algorithms” at
  people.eecs.berkeley.edu/~demmel/ma221-Fall20

• High level common approach: replace $A$ by $RA$, $R=$ random matrix, solve using $RA$ instead