Introduction to Randomized Algorithms

Srikrishnan Divakaran
DA-IICT

Talk Outline

- Preliminaries and Motivation
- Analysis of
  - Randomized Quick Sort
  - Karger’s Min-cut Algorithm
- Basic Analytical Tools
- Yao’s Lower Bounding Technique
- References

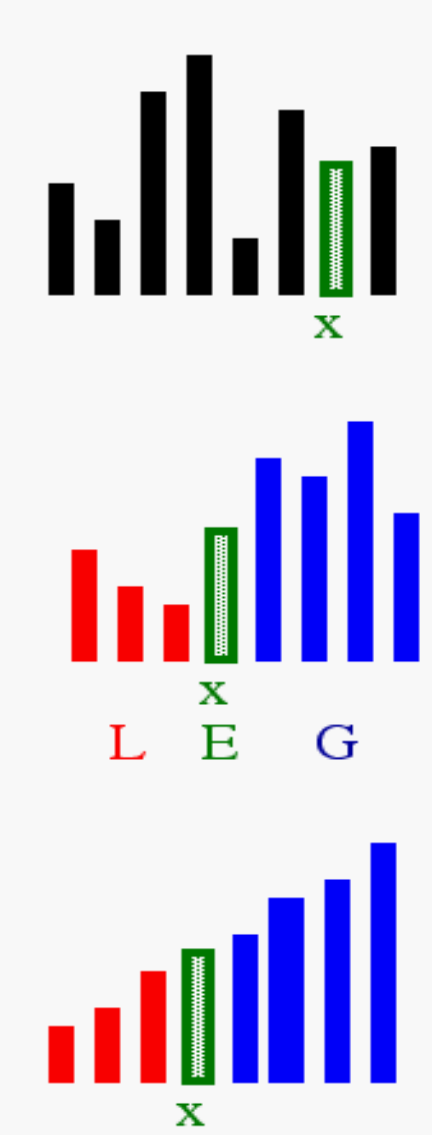
Preliminaries and Motivation

Quick Sort

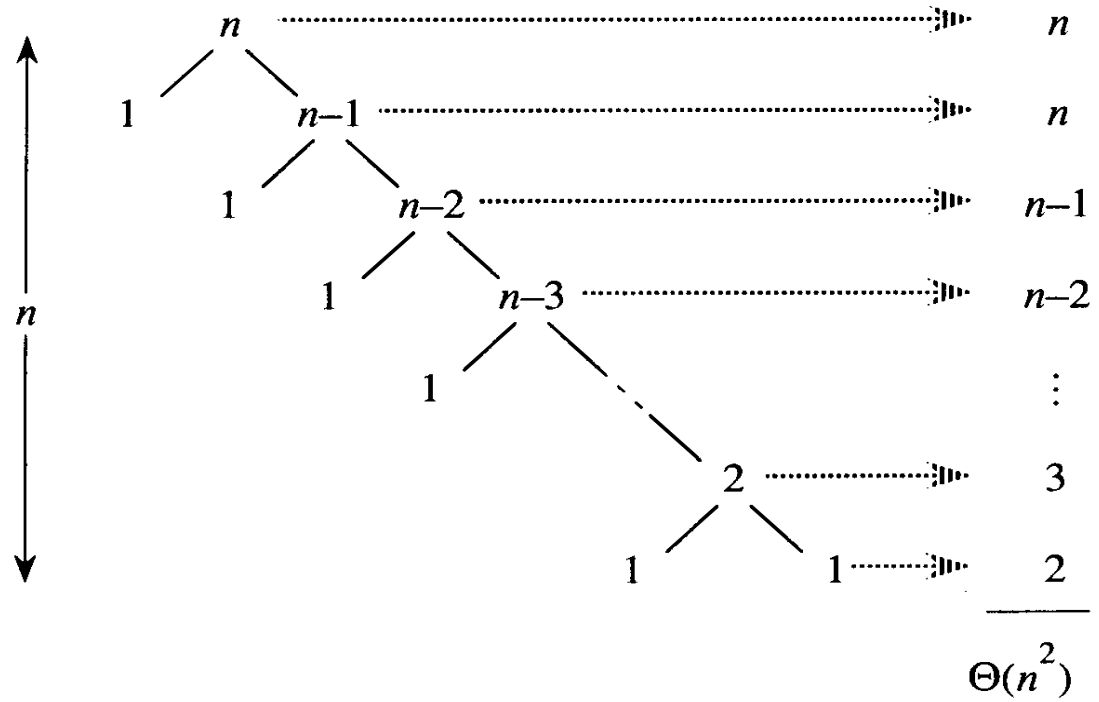
Select: pick an arbitrary element x in S to be the pivot.

Partition: rearrange elements so that elements with value less than x go to List L to the left of x and elements with value greater than x go to the List R to the right of x.

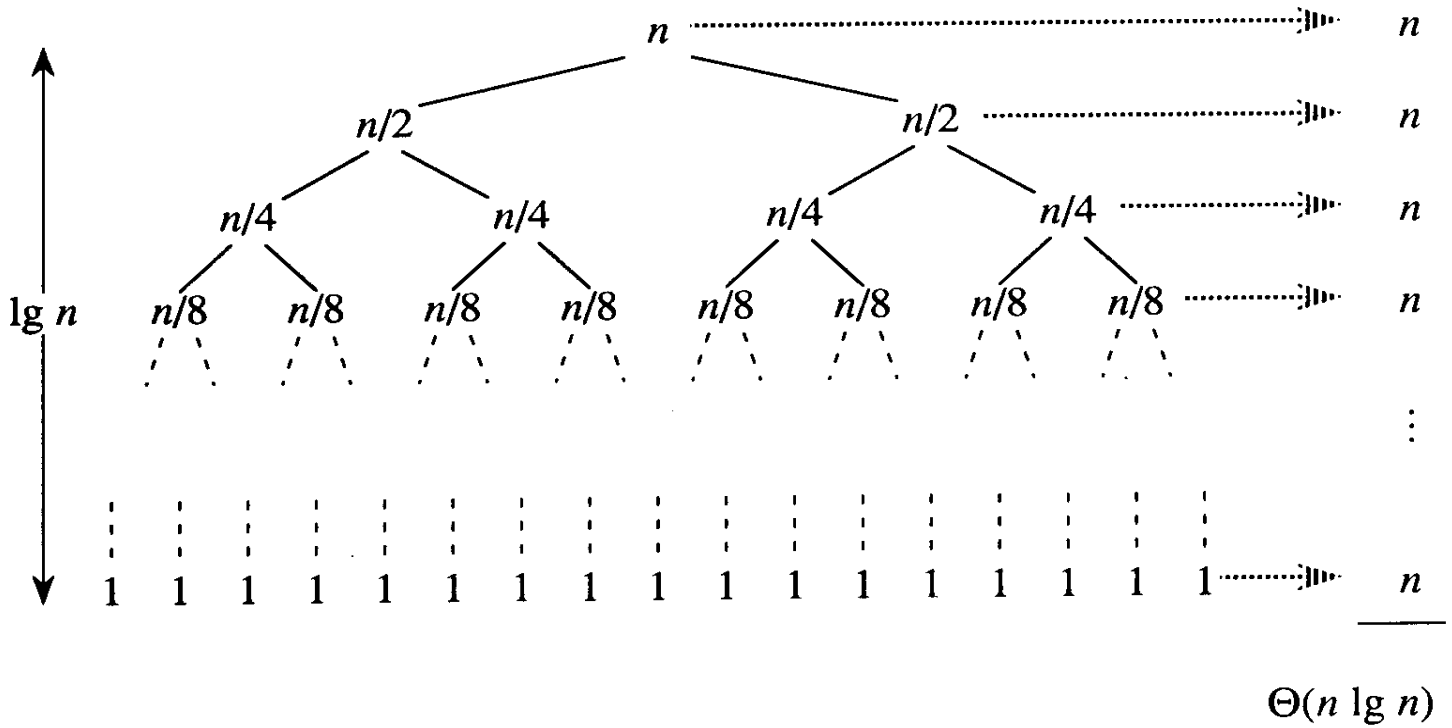
Recursion: recursively sort the lists L and R.



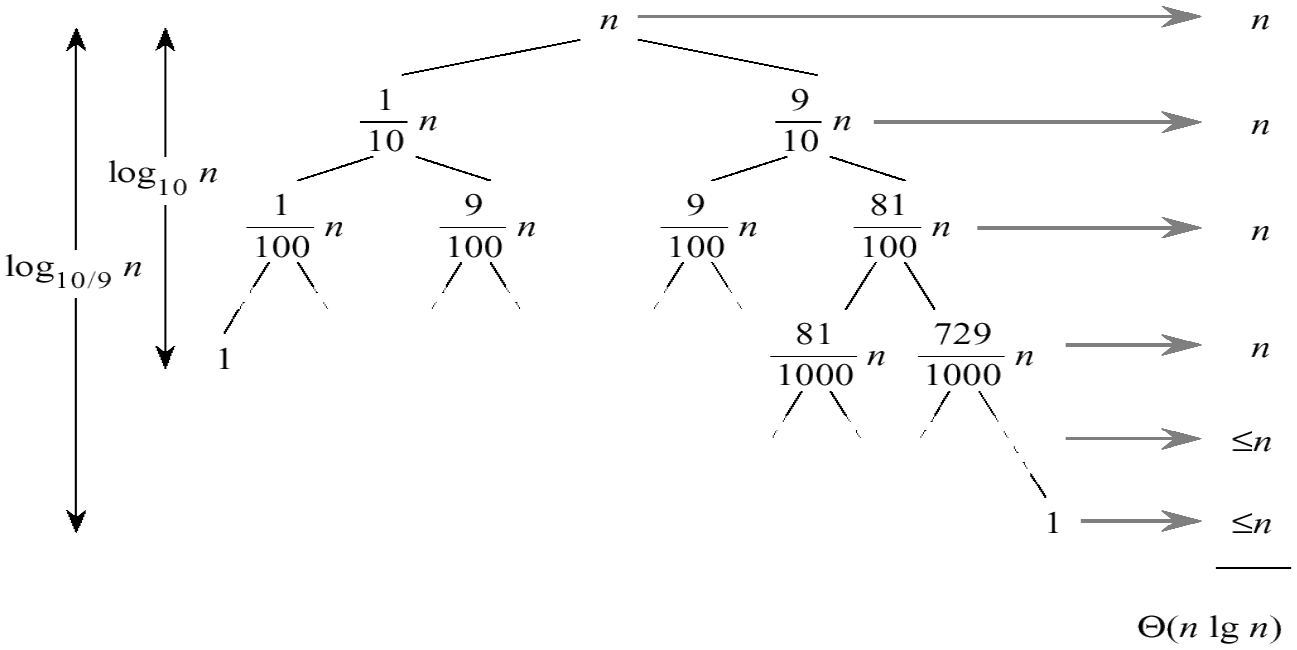
Worst Case Partitioning of Quick Sort



Best Case Partitioning of Quick Sort



Average Case of Quick Sort



Randomized Quick Sort

Randomized-Partition(*A, p, r*)

1. $i \leftarrow Random(p, r)$
2. exchange $A[r] \leftrightarrow A[i]$
3. **return Partition**(A, p, r)

Randomized-Quicksort(*A, p, r*)

1. **if** $p < r$
2. **then** $q \leftarrow$ **Randomized-Partition**(A, p, r)
3. **Randomized-Quicksort**(A, p , q-1)
4. **Randomized-Quicksort**(A, q+1, r)

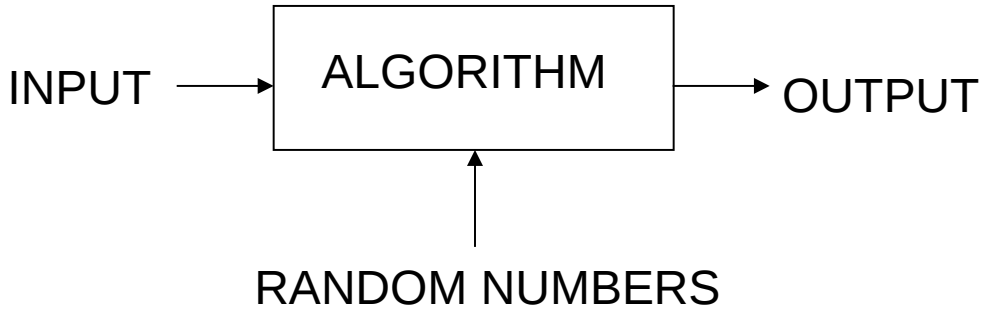
Randomized Quick Sort

- Exchange *A*[*r*] with an element chosen at random from *A*[*p…r*] in **Partition**.
- The pivot element is equally likely to be any of input elements.
- *For any given input, the behavior of Randomized Quick Sort* is determined not only by the input but also by the *random choices of the pivot.*
- We add randomization to Quick Sort to obtain for any input the expected performance of the algorithm to be good.

Deterministic Algorithms

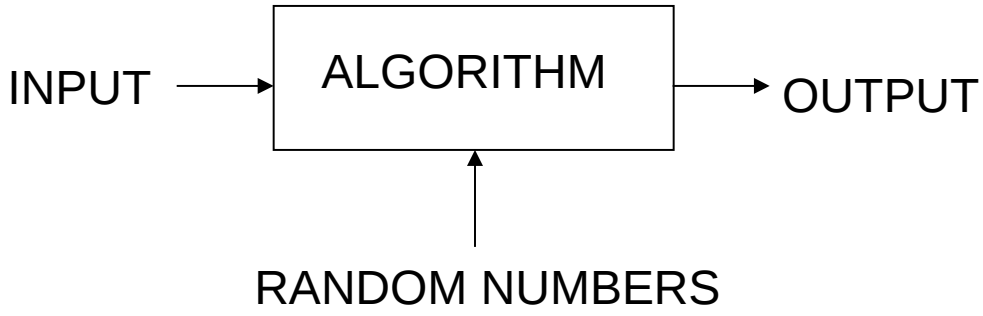
Goal: Prove for all input instances the algorithm solves the problem correctly and the number of steps is bounded by a polynomial in the size of the input.

Randomized Algorithms



- In addition to input, algorithm takes a source of random numbers and makes random choices during execution;
- Behavior can vary even on a fixed input;

Las Vegas Randomized Algorithms



Goal: Prove that for all input instances the algorithm solves the problem correctly and the expected number of steps is bounded by a polynomial in the input size.

Note: The expectation is over the random choices made by the algorithm.

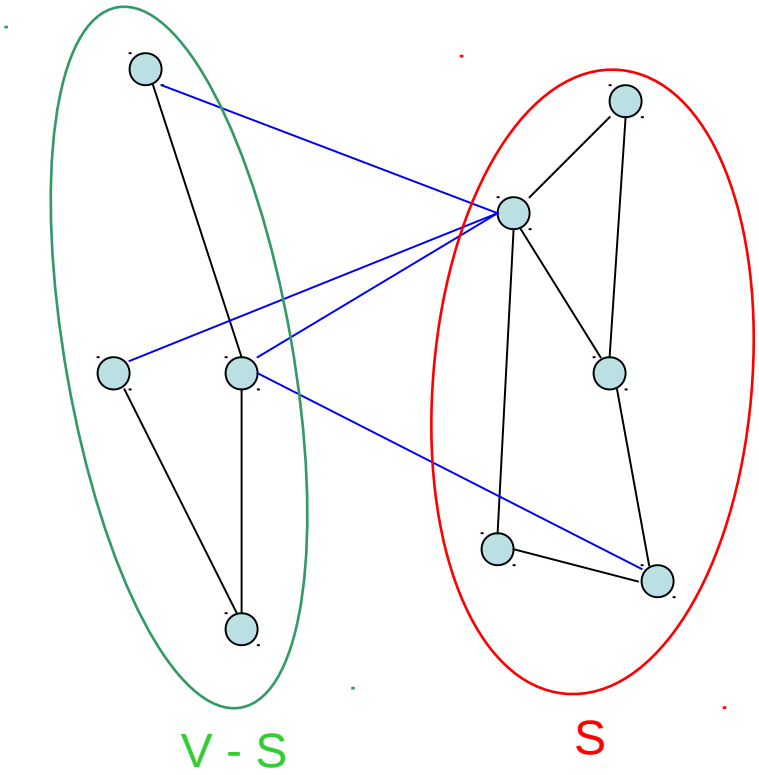
Probabilistic Analysis of Algorithms

Input is assumed to be from a probability distribution.

Goal: Show that for all inputs the algorithm works correctly and for most inputs the number of steps is bounded by a polynomial in the size of the input.

Min-cut for Undirected Graphs

Given an undirected graph, a global *min-cut* is a cut (S,V-S) minimizing the number of *crossing edges*, where a crossing edge is an edge (u,v) s.t. u∈S and v∈ V-S.

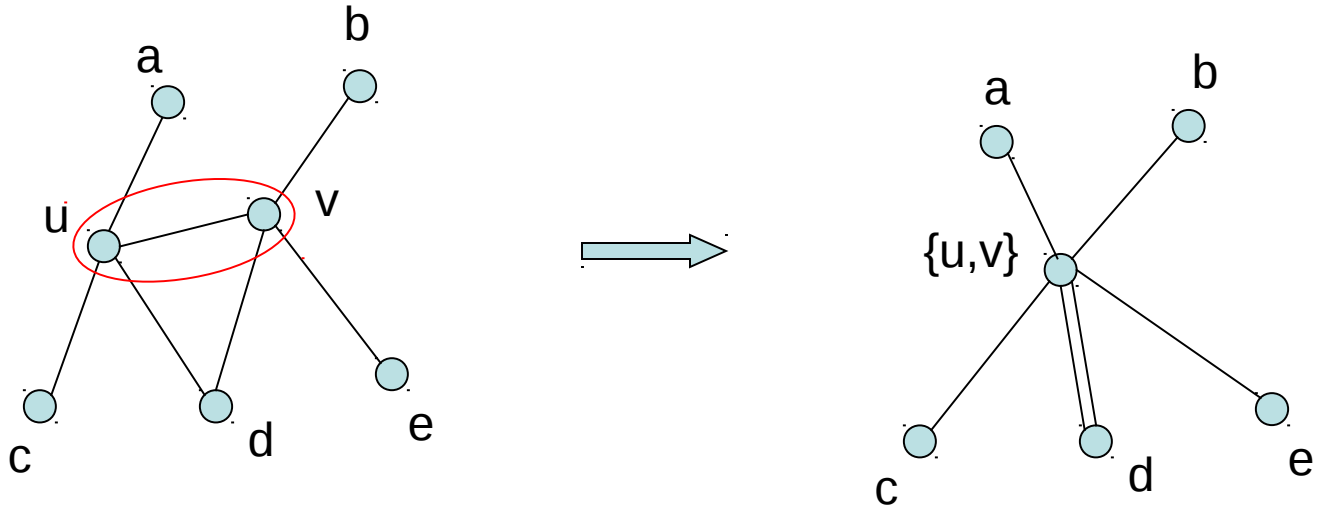


Graph Contraction

For an undirected graph G, we can construct a new graph G' by *contracting* two vertices u, v in G as follows:

- u and v become one vertex {u,v} and the edge (u,v) is removed;
- the other edges incident to u or v in G are now incident on the new vertex {u,v} in G';

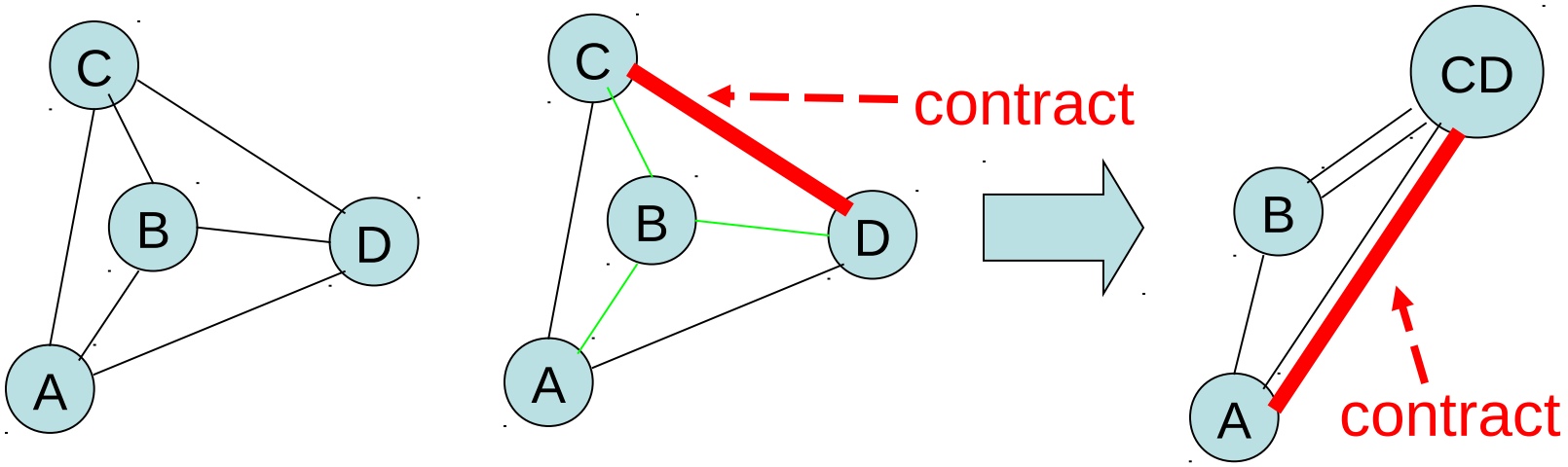
Note: There may be multi-edges between two vertices. We just keep them.



Graph G

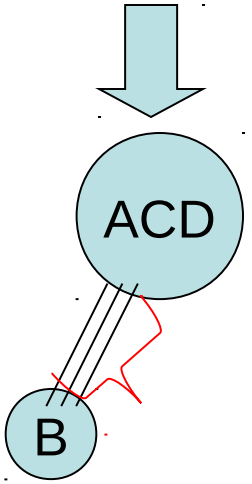
Graph G'

Karger's Min-cut Algorithm



(i) Graph G (ii) Contract nodes C and D (iii) contract nodes A and CD

Note: C is a cut but not necessarily a min-cut.



(Iv) Cut C=*{(A,B), (B,C), (B,D)}*

Karger's Min-cut Algorithm

For i = 1 to $100n^2$

repeat

randomly pick an edge (u,v)

contract u and v

until two vertices are left

$c_i \leftarrow$ the number of edges between them

Output **mini c_i**

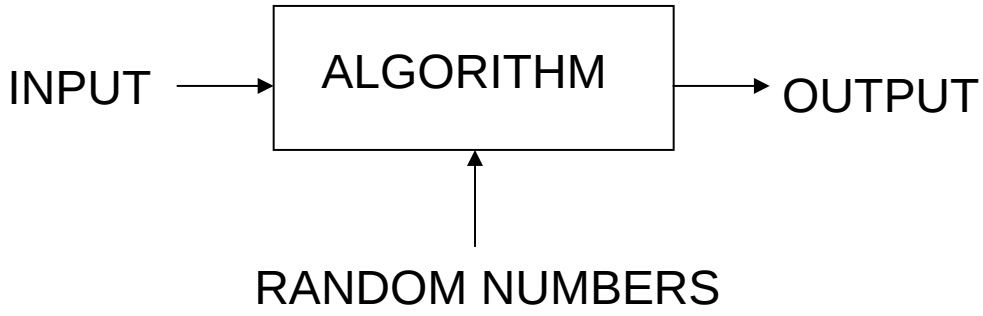
Key Idea

- Let $C^* = \{c_1^*, c_2^*, \ldots, c_k^*\}$ be a min-cut in G and C^i be a cut determined by Karger's algorithm during some iteration i.

- C^i will be a min-cut for G if during iteration "i" none of the edges in C* are contracted.

- If we can show that with *prob. $\Omega(1/n^2)$, where $n = |V|$, C^i will be a min-cut*, then by *repeatedly obtaining min-cuts $O(n^2)$ times* and taking minimum gives the min-cut with high prob.

Monte Carlo Randomized Algorithms



Goal: Prove that the algorithm

- with high probability solves the problem correctly;
- for every input the expected number of steps is bounded by a polynomial in the input size.

Note: The expectation is over the random choices made by the algorithm.

Monte Carlo versus Las Vegas

- A Monte Carlo algorithm runs produces an answer that is correct with non-zero probability, whereas a Las Vegas algorithm always produces the correct answer.

- The running time of both types of randomized algorithms is a random variable whose expectation is bounded say by a polynomial in terms of input size.

- These expectations are only over the random choices made by the algorithm independent of the input. Thus independent repetitions of Monte Carlo algorithms drive down the failure probability exponentially.

Motivation for Randomized Algorithms

- Simplicity;
- Performance;
- Reflects reality better (Online Algorithms);
- For many hard problems helps obtain better complexity bounds when compared to deterministic approaches;

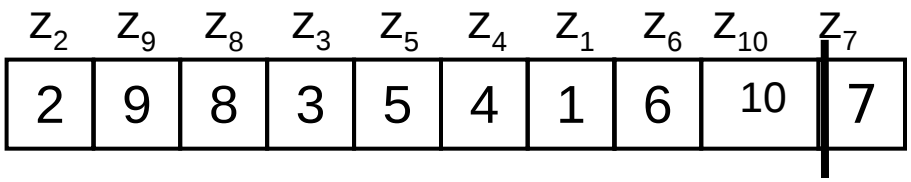
Analysis of Randomized Quick Sort

Linearity of Expectation

If X_1, X_2, …, X_n are random variables, then

$$E\left[\sum_{i=1}^{n} Xi\right] = \sum_{i=1}^{n} E[Xi]$$

Notation



- Rename the elements of A as $z_1, z_2, \ldots, z_n$, with z_i being the i^{th} smallest element (Rank "i").
- Define the set $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$ be the set of elements between z_i and z_j, inclusive.

Expected Number of Total Comparisons in PARTITION

Let $X_{ij} = I\{z_i \text{ is compared to } z_j\}$ ← indicator random variable

Let X be the total number of comparisons performed by the algorithm. Then

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

The expected number of comparisons performed by the algorithm is

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

by linearity of expectation

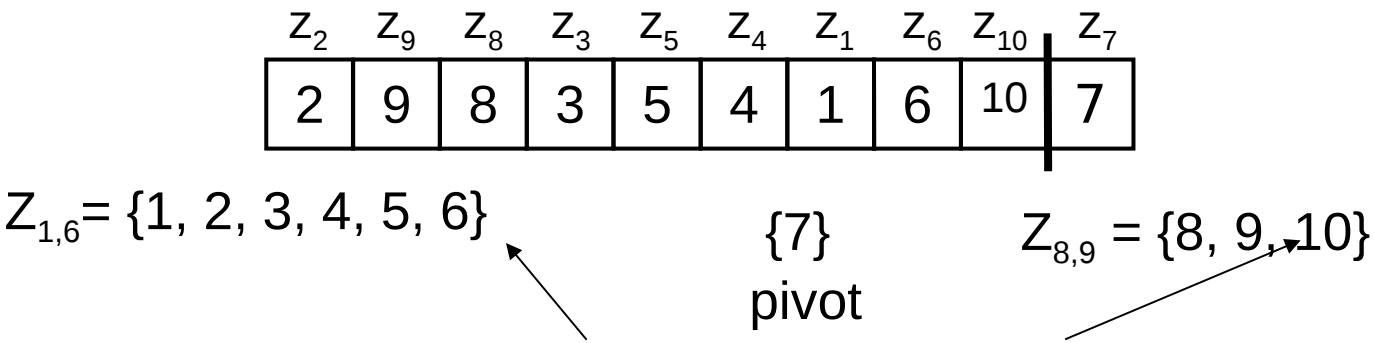
$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ is compared to } z_j\}$$

Comparisons in PARTITION

Observation 1: Each pair of elements is compared **at most once** during the entire execution of the algorithm

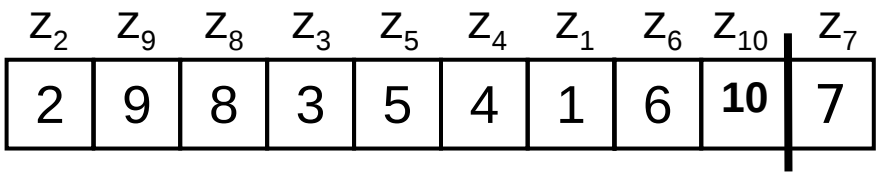
- Elements are compared only to the pivot point!
- Pivot point is excluded from future calls to PARTITION

Observation 2: Only the pivot is compared with elements in both partitions



Elements between different partitions are never compared

Comparisons in PARTITION



$Z_{1,6}$= {1, 2, 3, 4, 5, 6} {7} $Z_{8,9}$ = {8, 9, 10}

$$\Pr\{z_i \text{ is compared to } z_j\}?$$

Case 1: pivot chosen such as: $z_i < x < z_j$

- z_i and z_j will never be compared

Case 2: z_i or z_j is the pivot

- z_i and z_j will be compared
- only if one of them is chosen as pivot before any other element in range z_i to z_j

Expected Number of Comparisons in PARTITION

Pr {Z_i is compared with Z_j}

= Pr{Z_i or Z_j is chosen as pivot before other elements in $Z_{i,j}$} = 2 / (j-i+1)

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ is compared to } z_j\}$$

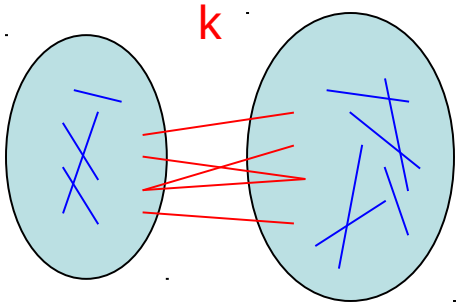
$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n-1} O(\lg n)$$

$$= O(n \lg n)$$

Analysis of Karger's Min-Cut Algorithm

Analysis of Karger's Algorithm

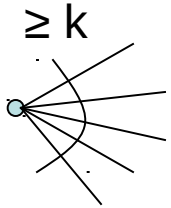
Let k be the number of edges of min cut (S, V-S).



If we never picked a crossing edge in the algorithm, then the number of edges between two last vertices is the correct answer.

The probability that in step 1 of an iteration a crossing edge is not picked = (|E|-k)/|E|.

By def of min cut, we know that each vertex v has degree at least k, Otherwise the cut ({v}, V-{v}) is lighter.



Thus |E| ≥ nk/2 and (|E|-k)/|E| = 1 - k/|E| ≥ 1-2/n.

Analysis of Karger's Algorithm

- In step 1, Pr [no crossing edge picked] >= 1 – 2/n
- Similarly, in step 2, Pr [no crossing edge picked] ≥ 1-2/(n-1)
- In general, in step j, Pr [no crossing edge picked] ≥ 1-2/(n-j+1)
- Pr {the n-2 contractions never contract a crossing edge}
  - = Pr [first step good]
    * Pr [second step good after surviving first step]
    * Pr [third step good after surviving first two steps]
    * …
    * Pr [(n-2)-th step good after surviving first n-3 steps]
    ≥ (1-2/n) (1-2/(n-1)) … (1-2/3)
    = [(n-2)/n] [(n-3)(n-1)] … [1/3] = 2/[n(n-1)] = $\Omega(1/n^2)$

Basic Analytical Tools

Tail Bounds

- In the analysis of randomized algorithms, we need to know how much does an algorithms run-time/cost deviate from its expected run-time/cost.
- That is we need to find an upper bound on Pr[X deviates from E[X] a lot]. This we refer to as the tail bound on X.

Markov and Chebyshev's Inequality

Markov's Inequality If $X \geq 0$, then

$$\Pr[X \geq a] \leq E[X]/a.$$

Proof. Suppose $\Pr[X \geq a] > E[X]/a$. Then

$$E[X] \geq a \cdot \Pr[X \geq a] > a \cdot E[X]/a = E[X].$$

Chebyshev's Inequality: $\Pr[\ |X-E[X]| \geq a\ ] \leq \mathrm{Var}[X] / a^2$.

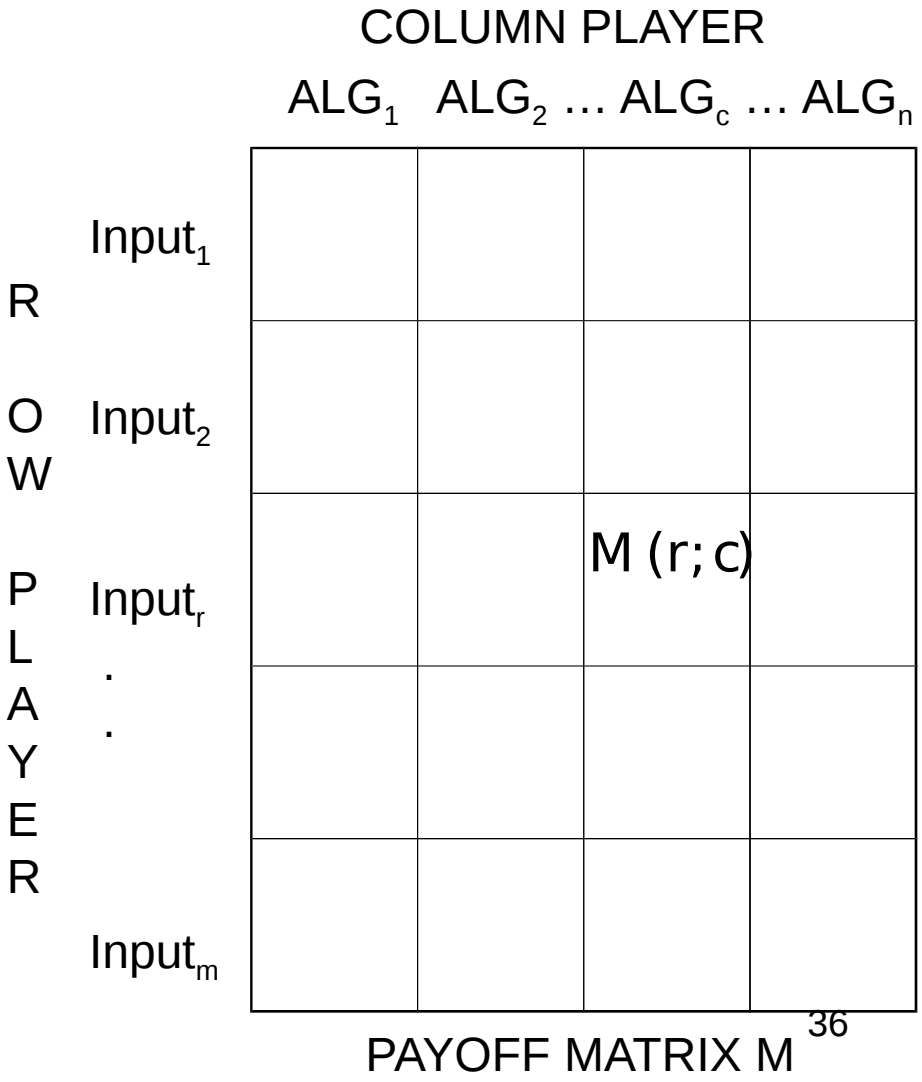
Proof.

$$\begin{aligned}
&\Pr[\ |X-E[X]| \geq a\ ] \\
=\ &\Pr[\ |X-E[X]|^2 \geq a^2\ ] \\
=\ &\Pr[\ (X-E[X])^2 \geq a^2\ ] \\
\leq\ &E[(X-E[X])^2] / a^2 \qquad \text{// Markov on } (X-E[X])^2 \\
=\ &\mathrm{Var}[X] / a^2
\end{aligned}$$

Yao’s Inequality for Establishing Lower Bounds

Algorithm Analysis in terms of Two Player Zero Sum Games

- the sum of payoffs of two players is zero in each cell of the table;
- The row player (one who maximizes cost) is the adversary responsible for designing malicious inputs;
- The column player (one who minimizes cost) is the algorithm designer responsible for designing efficient algorithms;



Pure Strategy

- For the column player (minimizer)
  - Each pure strategy corresponds to a deterministic algorithm.
- For the row player (maximizer)
  - Each pure strategy corresponds to a particular input instance.

Mixed strategies

- For the column player (minimizer)
  - Each mixed strategy corresponds to a Las Vegas randomized algorithm.
- For the row player (maximizer)
  - Each mixed strategy corresponds to a probability distribution over all the input instances.

Von Neumann's Minimax Theorem

For any 2-person 0-sum game, we have

$$\max_{p} \min_{q} p^T M\ q = \min_{q} \max_{p} p^T M\ q:$$

That is, each 2-person zero-sum game has a saddle point with respect to mixed strategies.

Loomis' Theorem

For any 2-person 0-sum game, we have

$$\max_{p} \min_{c} p^T M e_c = \min_{q} \max_{r} e_r^T M q;$$

where e_i means running the i-th strategy with probability 1.

To see the theorem, just observe that when p is known, the column player has an optimal strategy that is a pure strategy. A similar observation holds for the row player, too.

Yao's interpretation for Loomis' Theorem

Let $T(I;A)$ denote the time required for algorithm A to run on input I. Then by Loomis Theorem, we have

$$\max_{p} \min_{\text{deterministic algorithm } A} E[T(I_p;A)] = \min_{q} \max_{\text{input } I} E[T(I;A_q)].$$

Therefore, the following inequality holds for any probability distribution p and q:

$$\min_{\text{deterministic algorithm } A} E[T(I_p;A)] \cdot \max_{\text{input } I} E[T(I;A_q)].$$

How to use Yao's Inequality?

- Task 1:
  - Design a probability distribution *p* for the input instance.

- Task 2:
  - Obtain a lower bound on the expected running for any deterministic algorithm running on I_p.

Application of Yao's Inequality

Find bill problem: There are n boxes and exactly one box contains a dollar bill, and the rest of the boxes are empty. A probe is defined as opening a box to see if it contains the dollar bill. The objective is to locate the box containing the dollar bill while minimizing the number of probes performed.

Randomized Find

1. select x ε {H, T} uniformly at random
2. if x = H then
   (a) probe boxes in order from 1 through n and stop if bill is located
3. else
   (a) probe boxes in order from n through 1 and stop if bill is located

The expected number of probes made by the algorithm is (n+1)/2. Since, if the dollar bill is in the i^{th} box, then with probability 0:5, i probes are made and with probability 0:5, (n - i + 1) probes are needed.

Application of Yao's Lemma

Lemma: A lower bound on the expected number of probes required by any randomized algorithm to solve the Find-bill problem is (n + 1)/2.

Proof: We assume that the bill is located in any one of the n boxes uniformly at random. We only consider deterministic algorithms that does not probe the same box twice. By symmetry we can assume that the probe order for the deterministic algorithm is 1 through n.
B Yao's in-equality, we have

$$\text{Min}_{A \acute{\epsilon} A} \; E[C(A; I_p)] = \sum i/n = (n+1)/2 <= \max_{I \acute{\epsilon} I} E[C(I;A_q)]$$

Therefore any randomized algorithm A_q requires at least (n+1)/2 probes.

References

1. Amihood Amir, Karger's Min-cut Algorithm, Bar-Ilan University, 2009.
2. George Bebis, Randomizing Quick Sort, Lecture Notes of CS 477/677: Analysis of Algorithms, University of Nevada.
3. Avrim Blum and Amit Gupta, Lecture Notes on Randomized Algorithms, CMU, 2011.
4. Hsueh-I Lu, Yao's Theorem, Lecture Notes on Randomized Algorithms, National Taiwan University, 2010.
5. Rada Mihalcea, Quick Sort, Lecture Notes of CSCE3110: Data Structures, University of North Texas, http://www.cs.unt.edu/~rada/CSCE3110.
6. Rajeev Motwani and Prabhakar Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
7. Prabhakar Raghavan, AMS talk on Randomized Algorithms, Stanford University.

1000 Whats, What Nuts
&
Wall (Face Book) Nuts?