Introduction to Randomized Algorithms

Arijit Bishnu
(arijit@isical.ac.in)

Advanced Computing and Microelectronics Unit
Indian Statistical Institute
Kolkata 700108, India.

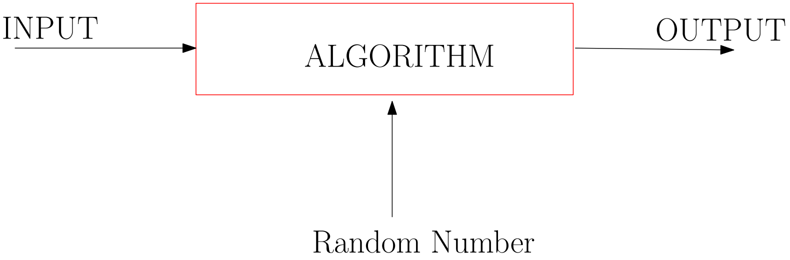
Organization

Introduction

Goal of a Deterministic Algorithm

- The solution produced by the algorithm is correct, and
- the number of computational steps is same for different runs of the algorithm with the same input.

Randomized Algorithm



Randomized Algorithm

- In addition to the input, the algorithm uses a source of pseudo random numbers. During execution, it takes random choices depending on those random numbers.
- The behavior (output) can vary if the algorithm is run multiple times on the same input.

Advantage of Randomized Algorithm

The Paradigm

Instead of making a guaranteed good choice, make a random choice and hope that it is good. This helps because guaranteeing a good choice becomes difficult sometimes.

Advantage of Randomized Algorithm

The Paradigm

Instead of making a guaranteed good choice, make a random choice and hope that it is good. This helps because guaranteeing a good choice becomes difficult sometimes.

Randomized Algorithms

make random choices. The expected running time depends on the random choices, not on any input distribution.

Advantage of Randomized Algorithm

The Paradigm

Instead of making a guaranteed good choice, make a random choice and hope that it is good. This helps because guaranteeing a good choice becomes difficult sometimes.

Randomized Algorithms

make random choices. The expected running time depends on the random choices, not on any input distribution.

Average Case Analysis

analyzes the expected running time of deterministic algorithms assuming a suitable random distribution on the input.

Pros and Cons of Randomized Algorithms

Pros

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.
- An adversary is powerless; randomized algorithms have no worst case inputs.

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.
- An adversary is powerless; randomized algorithms have no worst case inputs.
- Randomized algorithms are often simpler and faster than their deterministic counterparts.

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.
- An adversary is powerless; randomized algorithms have no worst case inputs.
- Randomized algorithms are often simpler and faster than their deterministic counterparts.

Cons

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.
- An adversary is powerless; randomized algorithms have no worst case inputs.
- Randomized algorithms are often simpler and faster than their deterministic counterparts.

Cons

- In the worst case, a randomized algorithm may be very slow.

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.
- An adversary is powerless; randomized algorithms have no worst case inputs.
- Randomized algorithms are often simpler and faster than their deterministic counterparts.

Cons

- In the worst case, a randomized algorithm may be very slow.
- There is a finite probability of getting incorrect answer. However, the probability of getting a wrong answer can be made arbitrarily small by the repeated employment of randomness.

Pros and Cons of Randomized Algorithms

Pros

- Making a random choice is fast.
- An adversary is powerless; randomized algorithms have no worst case inputs.
- Randomized algorithms are often simpler and faster than their deterministic counterparts.

Cons

- In the worst case, a randomized algorithm may be very slow.
- There is a finite probability of getting incorrect answer. However, the probability of getting a wrong answer can be made arbitrarily small by the repeated employment of randomness.
- Getting true random numbers is almost impossible.

Types of Randomized Algorithms

Definition

Las Vegas: a randomized algorithm that always returns a correct result. But the running time may vary between executions.

Example: Randomized QUICKSORT Algorithm

Definition

Monte Carlo: a randomized algorithm that terminates in polynomial time, but might produce erroneous result.

Example: Randomized MINCUT Algorithm

Some basic ideas from Probability

Expectation

Random variable

A function defined on a sample space is called a random variable. Given a random variable X, $Pr[X = j]$ means X's probability of taking the value j.

Expectation – "the average value"

The expectation of a random variable X is defined as:
$E[X] = \sum_{j=0}^{\infty} j \cdot Pr[X = j]$

Waiting for the first success

- Let p be the probability of success and $1 - p$ be the probability of failure of a random experiment.

Waiting for the first success

- Let p be the probability of success and $1 - p$ be the probability of failure of a random experiment.
- If we continue the random experiment till we get success, what is the expected number of experiments we need to perform?

Waiting for the first success

- Let p be the probability of success and $1 - p$ be the probability of failure of a random experiment.
- If we continue the random experiment till we get success, what is the expected number of experiments we need to perform?
- Let X: random variable that equals the number of experiments performed.

Waiting for the first success

- Let p be the probability of success and $1 - p$ be the probability of failure of a random experiment.
- If we continue the random experiment till we get success, what is the expected number of experiments we need to perform?
- Let X: random variable that equals the number of experiments performed.
- For the process to perform exactly j experiments, the first $j - 1$ experiments should be failures and the j-th one should be a success. So, we have $Pr[X = j] = (1 - p)^{(j-1)} \cdot p$.

Waiting for the first success

- Let p be the probability of success and $1 - p$ be the probability of failure of a random experiment.
- If we continue the random experiment till we get success, what is the expected number of experiments we need to perform?
- Let X: random variable that equals the number of experiments performed.
- For the process to perform exactly j experiments, the first $j - 1$ experiments should be failures and the j-th one should be a success. So, we have $Pr[X = j] = (1 - p)^{(j-1)} \cdot p$.
- So, the expectation of X, $E[X] = \sum_{j=0}^{\infty} j \cdot Pr[X = j] = \frac{1}{p}$.

Conditional Probability and Independent Event

Conditional Probability

The conditional probability of X given Y is

$$Pr[X = x \mid Y = y] = \frac{Pr[(X = x) \cap (Y = y)]}{Pr[Y = y]}$$

Conditional Probability and Independent Event

Conditional Probability

The conditional probability of X given Y is

$$Pr[X = x \mid Y = y] = \frac{Pr[(X = x) \cap (Y = y)]}{Pr[Y = y]}$$

An Equivalent Statement

$$Pr[(X = x) \cap (Y = y)] = Pr[X = x \mid Y = y] \cdot Pr[Y = y]$$

Conditional Probability and Independent Event

Conditional Probability

The conditional probability of X given Y is

$$Pr[X = x \mid Y = y] = \frac{Pr[(X = x) \cap (Y = y)]}{Pr[Y = y]}$$

An Equivalent Statement

$$Pr[(X = x) \cap (Y = y)] = Pr[X = x \mid Y = y] \cdot Pr[Y = y]$$

Independent Events

Two events X and Y are independent, if $Pr[(X = x) \cap (Y = y)] = Pr[X = x] \cdot Pr[Y = y]$. In particular, if X and Y are independent, then

$$Pr[X = x \mid Y = y] = Pr[X = x]$$

A Result on Intersection of events

Let $\eta_1, \eta_2, \ldots, \eta_n$ be n events not necessarily independent. Then,

$Pr[\cap_{i=1}^{n} \eta_i] = Pr[\eta_1] \cdot Pr[\eta_2 \mid \eta_1] \cdot Pr[\eta_3 \mid \eta_1 \cap \eta_2] \cdots Pr[\eta_n \mid \eta_1 \cap \ldots \cap \eta_{n-1}]$.

The proof is by induction on n.

Coupon Collection

Coupon Collection

The Problem

A company selling jeans gives a coupon with each jeans. There are n different coupons. Collecting n different coupons would give you a free jeans. How many jeans do you expect to buy before you get a free jeans?

Coupon Collection

The Problem

A company selling jeans gives a coupon with each jeans. There are n different coupons. Collecting n different coupons would give you a free jeans. How many jeans do you expect to buy before you get a free jeans?

- The coupon collection process in phase j when you have already collected j different coupons and are buying to get a new type.

Coupon Collection

The Problem

A company selling jeans gives a coupon with each jeans. There are n different coupons. Collecting n different coupons would give you a free jeans. How many jeans do you expect to buy before you get a free jeans?

- The coupon collection process in phase j when you have already collected j different coupons and are buying to get a new type.
- A new type of coupon ends phase j and you enter phase $j+1$.

Coupon Collection

- Let X_j be the random variable equal to the number of jeans you buy in phase j.

Coupon Collection

- Let X_j be the random variable equal to the number of jeans you buy in phase j.
- Then, $X = \sum_{j=0}^{n-1} X_j$ is the number of jeans bought to have n different coupons.

Coupon Collection

- Let X_j be the random variable equal to the number of jeans you buy in phase j.
- Then, $X = \sum_{j=0}^{n-1} X_j$ is the number of jeans bought to have n different coupons.

Lemma

The expected number of jeans bought in phase j, $E[X_j] = \frac{n}{n-j}$.

Coupon Collection

- Let X_j be the random variable equal to the number of jeans you buy in phase j.
- Then, $X = \sum_{j=0}^{n-1} X_j$ is the number of jeans bought to have n different coupons.

Lemma

The expected number of jeans bought in phase j, $E[X_j] = \frac{n}{n-j}$.

- The success probability, p in the j-th phase is $\frac{n-j}{n}$.

Coupon Collection

- Let X_j be the random variable equal to the number of jeans you buy in phase j.
- Then, $X = \sum_{j=0}^{n-1} X_j$ is the number of jeans bought to have n different coupons.

Lemma

The expected number of jeans bought in phase j, $E[X_j] = \frac{n}{n-j}$.

- The success probability, p in the j-th phase is $\frac{n-j}{n}$.
- By the bound on waiting for success, the expected number of jeans bought $E[X_j]$ is $\frac{1}{p} = \frac{n}{n-j}$.

The expectation

Theorem

The expected number of jeans bought before all n types of coupons are collected is $E[X] = nH_n = \Theta(n \log n)$.

The expectation

Theorem

The expected number of jeans bought before all n types of coupons are collected is $E[X] = nH_n = \Theta(n \log n)$.

Proof

$$E[X] = \sum_{j=0}^{n-1} E[X_j] = n \sum_{j=0}^{n-1} \frac{1}{n-j} = n \sum_{i=1}^{n} \frac{1}{i} = nH_n = \Theta(n \log n)$$

Randomized Quick Sort

Deterministic Quick Sort

The Problem:

Given an array $A[1 \ldots n]$ containing n (comparable) elements, sort them in increasing/decreasing order.

QSORT(A, p, q)

- If $p \geq q$, EXIT.
- Compute $s \leftarrow$ correct position of $A[p]$ in the sorted order of the elements of A from p-th location to q-th location.
- Move the pivot $A[p]$ into position $A[s]$.
- Move the remaining elements of $A[p - q]$ into appropriate sides.
- QSORT(A, p, $s - 1$);
- QSORT(A, $s + 1$, q).

Complexity Results of QSORT

- An INPLACE algorithm
- The worst case time complexity is $O(n^2)$.
- The average case time complexity is $O(n \log n)$.

Randomized Quick Sort

An Useful Concept - The Central Splitter

It is an index s such that the number of elements less (resp. greater) than $A[s]$ is at least $\frac{n}{4}$.

- The algorithm randomly chooses a key, and checks whether it is a central splitter or not.
- If it is a central splitter, then the array is split with that key as was done in the QSORT algorithm.
- It can be shown that the expected number of trials needed to get a central splitter is constant.

Randomized Quick Sort

RandQSORT(A, p, q)

1: If $p \geq q$, then EXIT.

2: While no central splitter has been found, execute the following steps:

  2.1: Choose uniformly at random a number $r \in \{p, p+1, \ldots, q\}$.

  2.2: Compute $s =$ number of elements in A that are less than $A[r]$, and
  $t =$ number of elements in A that are greater than $A[r]$.

  2.3: If $s \geq \frac{q-p}{4}$ and $t \geq \frac{q-p}{4}$, then $A[r]$ is a central splitter.

3: Position $A[r]$ in $A[s+1]$, put the members in A that are smaller than the central splitter in $A[p \ldots s]$ and the members in A that are larger than the central splitter in $A[s+2 \ldots q]$.

4: RandQSORT(A, p, s);

5: RandQSORT(A, $s+2$, q).

Analysis of RandQSORT

Fact: One execution of Step 2 needs $O(q-p)$ time.

Question: How many times Step 2 is executed for finding a central splitter ?

Result:

The probability that the randomly chosen element is a central splitter is $\frac{1}{2}$.

Recall "Waiting for success"

If p be the probability of success of a random experiment, and we continue the random experiment till we get success, the expected number of experiments we need to perform is $\frac{1}{p}$.

Implication in Our Case

- The expected number of times Step 2 needs to be repeated to get a central splitter (success) is 2 as the corresponding success probability is $\frac{1}{2}$.
- Thus, the expected time complexity of Step 2 is $O(n)$

Analysis of RandQSORT

Time Complexity

- The expected running time for the algorithm on a set A, excluding the time spent on recursive calls, is $O(|A|)$.

Analysis of RandQSORT

Time Complexity

- The expected running time for the algorithm on a set A, excluding the time spent on recursive calls, is $O(|A|)$.
- Worst case size of each partition in j-th level of recursion is $n \cdot (\frac{3}{4})^j$, So, the expected time spent excluding recursive calls is $O(n \cdot (\frac{3}{4})^j)$ for each partition.

Analysis of RandQSORT

Time Complexity

- The expected running time for the algorithm on a set A, excluding the time spent on recursive calls, is $O(|A|)$.
- Worst case size of each partition in j-th level of recursion is $n \cdot (\frac{3}{4})^j$, So, the expected time spent excluding recursive calls is $O(n \cdot (\frac{3}{4})^j)$ for each partition.
- The number of partitions of size $n \cdot (\frac{3}{4})^j$ is $O((\frac{4}{3})^j)$.

Analysis of RandQSORT

Time Complexity

- The expected running time for the algorithm on a set A, excluding the time spent on recursive calls, is $O(|A|)$.
- Worst case size of each partition in j-th level of recursion is $n \cdot (\frac{3}{4})^j$, So, the expected time spent excluding recursive calls is $O(n \cdot (\frac{3}{4})^j)$ for each partition.
- The number of partitions of size $n \cdot (\frac{3}{4})^j$ is $O((\frac{4}{3})^j)$.
- By linearity of expectations, the expected time for all partitions of size $n \cdot (\frac{3}{4})^j$ is $O(n)$.

Analysis of RandQSORT

Time Complexity

- The expected running time for the algorithm on a set A, excluding the time spent on recursive calls, is $O(|A|)$.
- Worst case size of each partition in j-th level of recursion is $n \cdot (\frac{3}{4})^j$, So, the expected time spent excluding recursive calls is $O(n \cdot (\frac{3}{4})^j)$ for each partition.
- The number of partitions of size $n \cdot (\frac{3}{4})^j$ is $O((\frac{4}{3})^j)$.
- By linearity of expectations, the expected time for all partitions of size $n \cdot (\frac{3}{4})^j$ is $O(n)$.
- Number of levels of recursion $= \log_{\frac{4}{3}} n = O(\log n)$.

Analysis of RandQSORT

Time Complexity

- The expected running time for the algorithm on a set A, excluding the time spent on recursive calls, is $O(|A|)$.
- Worst case size of each partition in j-th level of recursion is $n \cdot (\frac{3}{4})^j$, So, the expected time spent excluding recursive calls is $O(n \cdot (\frac{3}{4})^j)$ for each partition.
- The number of partitions of size $n \cdot (\frac{3}{4})^j$ is $O((\frac{4}{3})^j)$.
- By linearity of expectations, the expected time for all partitions of size $n \cdot (\frac{3}{4})^j$ is $O(n)$.
- Number of levels of recursion $= \log_{\frac{4}{3}} n = O(\log n)$.
- Thus, the expected running time is $O(n \log n)$.

Finding the k-th largest

Median Finding

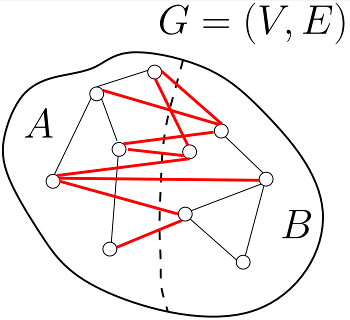
Similar ideas of getting a central splitter and waiting for success bound applies for finding the median in $O(n)$ time.

Global Mincut Problem for an Undirected Graph

Global Mincut Problem

Problem Statement

Given a connected undirected graph $G = (V, E)$, find a cut (A, B) of minimum cardinality.



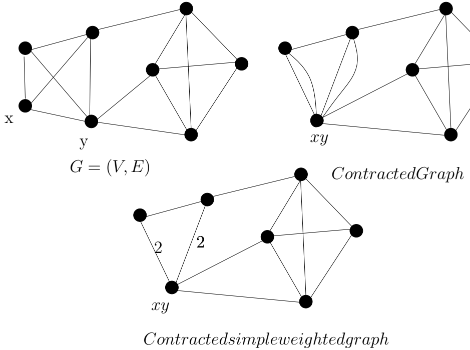
Applications:

- Clustering and partitioning items,
- Network reliability, network design, circuit design, etc.

A Simple Randomized Algorithm

Contraction of an Edge

Contraction of an edge $e = (x, y)$ implies merging the two vertices $x, y \in V$ into a single vertex, and remove the self loop. The contracted graph is denoted by G/xy.



Results on Contraction of Edges

Result - 1

As long as G/xy has at least one edge,

- The size of the minimum cut in the (weighted) graph G/xy is at least as large as the size of the minimum cut in G .

Result - 2

Let $e_1, e_2, \ldots, e_{n-2}$ be a sequence of edges in G, such that

- none of them is in the minimum cut of G, and
- $G' = G/\{e_1, e_2, \ldots, e_{n-2}\}$ is a single multiedge.

Then this multiedge corresponds to the minimum cut in G.

Problem: Which edge sequence is to be chosen for contraction?

Analysis

Algorithm MINCUT(G)

$G_0 \leftarrow G$; $i = 0$
while G_i has more than two vertices **do**
  Pick randomly an edge e_i from the edges in G_i
  $G_{i+1} \leftarrow G_i / e_i$
  $i \leftarrow i + 1$
$(S, V - S)$ is the cut in the original graph
  corresponding to the single edge in G_i.

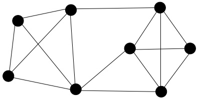
Theorem

Time Complexity: $O(n^2)$

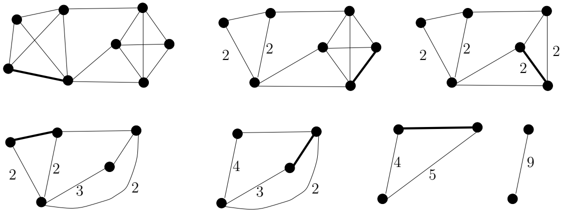
A Trivial Observation: The algorithm outputs a cut whose size is *no smaller than the mincut*.

Demonstration of the Algorithm

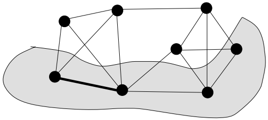
The given graph:



Stages of Contraction:



The corresponding output:



Quality Analysis: How good is the solution?

Result 3: Lower bounding $|E|$

If a graph $G = (V, E)$ has a minimum cut F of size k, and it has n vertices, then $|E| \geq \frac{kn}{2}$.

Quality Analysis: How good is the solution?

Result 3: Lower bounding $|E|$

If a graph $G = (V, E)$ has a minimum cut F of size k, and it has n vertices, then $|E| \geq \frac{kn}{2}$.

Proof

Quality Analysis: How good is the solution?

Result 3: Lower bounding $|E|$

If a graph $G = (V, E)$ has a minimum cut F of size k, and it has n vertices, then $|E| \geq \frac{kn}{2}$.

Proof

- If any node v has degree less than k, then the $\text{cut}(\{v\}, V - \{v\})$ will have size less than k.

Quality Analysis: How good is the solution?

Result 3: Lower bounding $|E|$

If a graph $G = (V, E)$ has a minimum cut F of size k, and it has n vertices, then $|E| \geq \frac{kn}{2}$.

Proof

- If any node v has degree less than k, then the $\text{cut}(\{v\}, V - \{v\})$ will have size less than k.
- This contradicts the fact that (A, B) is a global min-cut.

Quality Analysis: How good is the solution?

Result 3: Lower bounding $|E|$

If a graph $G = (V, E)$ has a minimum cut F of size k, and it has n vertices, then $|E| \geq \frac{kn}{2}$.

Proof

- If any node v has degree less than k, then the $\text{cut}(\{v\}, V - \{v\})$ will have size less than k.
- This contradicts the fact that (A, B) is a global min-cut.
- Thus, every node in G has degree at least k. So, $|E| \geq \frac{1}{2}kn$.

Quality Analysis: How good is the solution?

Result 3: Lower bounding $|E|$

If a graph $G = (V, E)$ has a minimum cut F of size k, and it has n vertices, then $|E| \geq \frac{kn}{2}$.

Proof

- If any node v has degree less than k, then the $\text{cut}(\{v\}, V - \{v\})$ will have size less than k.
- This contradicts the fact that (A, B) is a global min-cut.
- Thus, every node in G has degree at least k. So, $|E| \geq \frac{1}{2}kn$.

So, the probability that an edge in F is contracted is at most
$\frac{k}{(kn)/2} = \frac{2}{n}$
But, we don't know the min cut.

Summing up: Result 4

If we pick a random edge e from the graph G, then the probability of e belonging in the mincut is at most $\frac{2}{n}$.

Summing up: Result 4

If we pick a random edge e from the graph G, then the probability of e belonging in the mincut is at most $\frac{2}{n}$.

Continuing Contraction

Summing up: Result 4

If we pick a random edge e from the graph G, then the probability of e belonging in the mincut is at most $\frac{2}{n}$.

Continuing Contraction

- After i iterations, there are $n - i$ supernodes in the current graph G' and suppose no edge in the cut F has been contracted.

Summing up: Result 4

If we pick a random edge e from the graph G, then the probability of e belonging in the mincut is at most $\frac{2}{n}$.

Continuing Contraction

- After i iterations, there are $n - i$ supernodes in the current graph G' and suppose no edge in the cut F has been contracted.
- Every cut of G' is a cut of G. So, there are at least k edges incident on every supernode of G'.

Summing up: Result 4

If we pick a random edge e from the graph G, then the probability of e belonging in the mincut is at most $\frac{2}{n}$.

Continuing Contraction

- After i iterations, there are $n - i$ supernodes in the current graph G' and suppose no edge in the cut F has been contracted.
- Every cut of G' is a cut of G. So, there are at least k edges incident on every supernode of G'.
- Thus, G' has at least $\frac{1}{2}k(n - i)$ edges.

Summing up: Result 4

If we pick a random edge e from the graph G, then the probability of e belonging in the mincut is at most $\frac{2}{n}$.

Continuing Contraction

- After i iterations, there are $n - i$ supernodes in the current graph G' and suppose no edge in the cut F has been contracted.
- Every cut of G' is a cut of G. So, there are at least k edges incident on every supernode of G'.
- Thus, G' has at least $\frac{1}{2}k(n-i)$ edges.
- So, the probability that an edge in F is contracted in iteration $i+1$ is at most $\frac{k}{\frac{1}{2}k(n-i)} = \frac{2}{n-i}$.

Correctness

Theorem

The procedure MINCUT outputs the mincut with probability $\geq \frac{2}{n(n-1)}$.

Proof:

The correct cut(A, B) will be returned by MINCUT if no edge of F is contracted in any of the iterations $1, 2, \ldots, n-2$.
Let $\eta_i \Rightarrow$ the event that an edge of F is not contracted in the ith iteration.
We have already shown that

- $Pr[\eta_1] \geq 1 - \frac{2}{n}$.
- $Pr[\eta_{i+1} \mid \eta_1 \cap \eta_2 \cap \cdots \cap \eta_i] \geq 1 - \frac{2}{n-i}$

Lower Bounding the Intersection of Events

We want to lower bound $Pr[\eta_1 \cap \cdots \cap \eta_{n-2}]$.
We use the earlier result

$$Pr[\cap_{i=1}^{n} \eta_i] = Pr[\eta_1] \cdot Pr[\eta_2 \mid \eta_1] \cdot Pr[\eta_3 \mid \eta_1 \cap \eta_2] \cdots Pr[\eta_n \mid \eta_1 \cap \ldots \cap \eta_{n-1}].$$

So, we have $Pr[\eta_1] \cdot Pr[\eta_1 \mid \eta_2] \cdots Pr[\eta_{n-2} \mid \eta_1 \cap \eta_2 \cdots \cap \eta_{n-3}]$
$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-i}\right) \cdots \left(1 - \frac{2}{3}\right)$
$= \binom{n}{2}^{-1}$

Bounding the Error Probability

- We know that a single run of the contraction algorithm fails to find a global min-cut with probability at most $1 - \frac{1}{\binom{n}{2}} \approx 1$.

Bounding the Error Probability

- We know that a single run of the contraction algorithm fails to find a global min-cut with probability at most $1 - \frac{1}{\binom{n}{2}} \approx 1$.
- We can amplify our success probability by repeatedly running the algorithm with independent random choices and taking the best cut.

Bounding the Error Probability

- We know that a single run of the contraction algorithm fails to find a global min-cut with probability at most $1 - \frac{1}{\binom{n}{2}} \approx 1$.
- We can amplify our success probability by repeatedly running the algorithm with independent random choices and taking the best cut.
- If we run the algorithm $\binom{n}{2}$ times, then the probability that we fail to find a global min-cut in any run is at most

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}} \leq \frac{1}{e}.$$

Bounding the Error Probability

- We know that a single run of the contraction algorithm fails to find a global min-cut with probability at most $1 - \frac{1}{\binom{n}{2}} \approx 1$.
- We can amplify our success probability by repeatedly running the algorithm with independent random choices and taking the best cut.
- If we run the algorithm $\binom{n}{2}$ times, then the probability that we fail to find a global min-cut in any run is at most

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}} \leq \frac{1}{e}.$$

Result

By spending $O(n^4)$ time, we can reduce the failure probability from $1 - \frac{2}{n^2}$ to a reasonably small constant value $\frac{1}{e}$.

Probability helps in counting

The number of global minimum cuts

Given an undirected graph $G = (V, E)$ with $|V| = n$, what is the maximum number of global minimum cuts?

Probability helps in counting

The number of global minimum cuts

Given an undirected graph $G = (V, E)$ with $|V| = n$, what is the maximum number of global minimum cuts?

- What is your hunch? – exponential in n or polynomial in n?

Probability helps in counting

The number of global minimum cuts

Given an undirected graph $G = (V, E)$ with $|V| = n$, what is the maximum number of global minimum cuts?

- What is your hunch? – exponential in n or polynomial in n?
- Consider C_n, a cycle on n nodes. How many global minimum cuts are possible?

Probability helps in counting

The number of global minimum cuts

Given an undirected graph $G = (V, E)$ with $|V| = n$, what is the maximum number of global minimum cuts?

- What is your hunch? – exponential in n or polynomial in n?
- Consider C_n, a cycle on n nodes. How many global minimum cuts are possible?
- Cut out any two edges to have $\binom{n}{2}$ such cuts.

Probability helps in counting

The number of global minimum cuts

Given an undirected graph $G = (V, E)$ with $|V| = n$, what is the maximum number of global minimum cuts?

- What is your hunch? – exponential in n or polynomial in n?
- Consider C_n, a cycle on n nodes. How many global minimum cuts are possible?
- Cut out any two edges to have $\binom{n}{2}$ such cuts.
- Is this the bound?

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$
- Let $\mathcal{E}_i$ be the event that C_i is returned by the earlier algorithm.

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$
- Let $\mathcal{E}_i$ be the event that C_i is returned by the earlier algorithm.
- $\mathcal{E} = \cup_{i=1}^{r} \mathcal{E}_i$ is the event that the algorithm returns any global minimum cut.

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$
- Let $\mathcal{E}_i$ be the event that C_i is returned by the earlier algorithm.
- $\mathcal{E} = \cup_{i=1}^{r} \mathcal{E}_i$ is the event that the algorithm returns any global minimum cut.
- The earlier algorithm basically shows that $\Pr[\mathcal{E}_i] \geq \frac{1}{\binom{n}{2}}$.

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$
- Let $\mathcal{E}_i$ be the event that C_i is returned by the earlier algorithm.
- $\mathcal{E} = \cup_{i=1}^{r} \mathcal{E}_i$ is the event that the algorithm returns any global minimum cut.
- The earlier algorithm basically shows that $\Pr[\mathcal{E}_i] \geq \frac{1}{\binom{n}{2}}$.
- Each pair of events $\mathcal{E}_i$ and $\mathcal{E}_j$ are disjoint since only one cut is returned by any run of the algorithm.

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$
- Let $\mathcal{E}_i$ be the event that C_i is returned by the earlier algorithm.
- $\mathcal{E} = \cup_{i=1}^{r} \mathcal{E}_i$ is the event that the algorithm returns any global minimum cut.
- The earlier algorithm basically shows that $\Pr[\mathcal{E}_i] \geq \frac{1}{\binom{n}{2}}$.
- Each pair of events $\mathcal{E}_i$ and $\mathcal{E}_j$ are disjoint since only one cut is returned by any run of the algorithm.
- By the union bound for disjoint events, we have $\Pr[\mathcal{E}] = \Pr[\cup_{i=1}^{r} \mathcal{E}_i] = \sum_{i=1}^{r} \Pr[\mathcal{E}_i] \geq \frac{r}{\binom{n}{2}}$.

The proof

- Let there be r such cuts, $C_1, \ldots, C_r$
- Let $\mathcal{E}_i$ be the event that C_i is returned by the earlier algorithm.
- $\mathcal{E} = \cup_{i=1}^{r} \mathcal{E}_i$ is the event that the algorithm returns any global minimum cut.
- The earlier algorithm basically shows that $\Pr[\mathcal{E}_i] \geq \frac{1}{\binom{n}{2}}$.
- Each pair of events $\mathcal{E}_i$ and $\mathcal{E}_j$ are disjoint since only one cut is returned by any run of the algorithm.
- By the union bound for disjoint events, we have $\Pr[\mathcal{E}] = \Pr[\cup_{i=1}^{r} \mathcal{E}_i] = \sum_{i=1}^{r} \Pr[\mathcal{E}_i] \geq \frac{r}{\binom{n}{2}}$.
- Surely, $\Pr[\mathcal{E}] \leq 1$. So, $r \leq \binom{n}{2}$.

Conclusions

- Employing randomness leads to improved simplicity and improved efficiency in solving the problem.
- It assumes the availability of a perfect source of independent and unbiased random bits.
- Access to truly unbiased and independent sequence of random bits is expensive.
  So, it should be considered as an expensive resource like time and space.
- There are ways to reduce the randomness from several algorithms while maintaining the efficiency nearly the same.

Books

- Jon Kleinberg and Éva Tardos, *Algorithm Design*, Pearson Education.
- Rajeev Motwani and Prabhakar Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 2004.
- Michael Mitzenmacher and Eli Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, New York, USA, 2005..