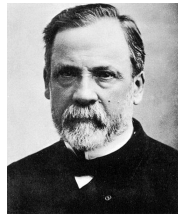


Randomized Algorithms

*Chance favors only the prepared mind.
Those unprepared cannot see the hand
stretched out to them by fortune*

Louis Pasteur (1822-1895)



1

A Game of Chance

- You are invited to play the following TV game:
 - You pay \$1000 to play and get 10 tokens to pay for “mistakes”
 - There are 100 closed boxes containing \$1,...,\$100, respectively, arranged in unknown order
 - Host suggests which box to open next but you decide which one to open
 - When the box you open contains the largest amount seen so far you pay a token and forfeit the amount in that box
 - If you have no tokens left and need to pay, you lose everything; otherwise, if you manage to open all the boxes, you keep your winnings (but not the \$1000 you paid to play)
- Would you play?
 - What should be your strategy?
 - What should be the host’s strategy?

2

Deterministic Algorithms



Goal: prove that the algorithm *always* solves the problem correctly using a specific # of steps (e.g., a polynomial in the input size n). Neither the output nor the number of steps change with repeated runs on the same input

Analysis:

worst-case, best-case, *average*-case

3

Randomness in Computation

Two different philosophical outlooks characterize the uses of randomness in computation:

1. The world behaves randomly
 - Algorithm is deterministic, input is random
 - Behavior of algorithm is averaged over probability distribution of inputs
2. The algorithm behaves randomly
 - Input is given, algorithm makes random choices
 - Randomization is internal to algorithm and its use does not require assumptions about the input

4

Average Case Analysis

- Use probability theory to analyze some aspect of an algorithm, such as running time or memory used
- Must make assumptions about distribution of inputs

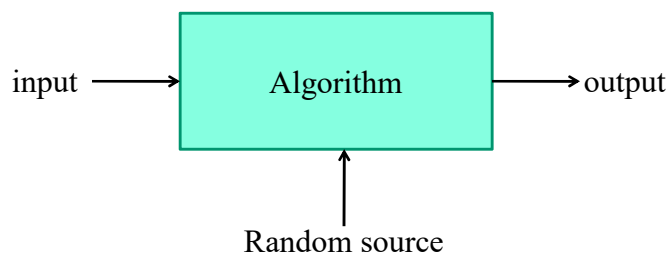
Goal: show algorithm “runs well” on average

Example: what is the expected running time of Quicksort under the assumption that all $n!$ input permutations are equally likely



5

Randomized Algorithms



- In addition to *deterministic input*, algorithm makes decisions based on values obtained from *random stream*
 - Behavior varies even when run again on same input
- Goal:** design the algorithm so that its behavior is likely to be good *for all inputs*

6

Example

- Given an array A of n distinct integers, determine the number of times that line 4 is executed

```

MAX( $A, n$ )
1   $max \leftarrow -\infty$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      do if  $A[i] > max$ 
→ 4          then  $max \leftarrow A[i]$ 
5  return  $max$ 

```

- Algorithm is deterministic with worst case in $\Theta(n)$
- Average case requires assumptions about input distribution, e.g., all permutations equally likely

7

The Flaw of Average Case Analysis

- Every deterministic algorithm has an input that elicits its worst-case behavior
- A malicious adversary may always provide us with this worst-case input or the input distribution may change over time

\Rightarrow

The input distribution we assumed in analysis is not valid.

8

Randomized Algorithms (beating the adversary)

- Use randomness as an algorithm design tool
- If don't know input distribution then impose one that makes analysis possible and meaningful
 - No more questionable assumptions
 - No input is guaranteed to elicit a worst-case behavior
 - Useful when most of the inputs can be processed efficiently and worst case caused by few of them
- Examples
 - *RandomMax*: choose the next array element at random
 - *Quicksort*: choose pivot at random

9

Example Revisited

- Find the number of times that line 5 executed

```

RANDOMIZEDMAX( $A, n$ )
1   $max \leftarrow -\infty$ 
2  Permute  $A$  uniformly at random
3  for  $i \leftarrow 1$  to  $n$ 
4      do if  $A[i] > max$ 
→ 5          then  $max \leftarrow A[i]$ 
6  return  $max$ 

```

- Algorithm is *randomized* with worst case in $\Theta(n)$
- Average case does not require assumptions about input distribution because a distribution is forced
- How do you permute randomly and uniformly?

10

Permute Algorithm 1

PERMUTE-BY-SORTING(A)

```

1   $n \leftarrow \text{length}[A]$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      do  $P[i] = \text{RANDOM}(1, n^3)$ 
4  sort  $A$ , using  $P$  as sort keys
5  return  $A$ 

```

a	b	c	d	e	f	g	h	i	j
---	---	---	---	---	---	---	---	---	---

i	b	d	j	e	g	c	a	h	f
---	---	---	---	---	---	---	---	---	---

340	23	223	55	99	722	182	500	15	88
-----	----	-----	----	----	-----	-----	-----	----	----

15	23	55	88	99	182	223	340	500	722
----	----	----	----	----	-----	-----	-----	-----	-----

Runs in $O(n \log n)$ time

11

Permute Algorithm 2

RANDOMIZE-IN-PLACE(A)

```

1   $n \leftarrow \text{length}[A]$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      do swap  $A[i] \leftrightarrow A[\text{RANDOM}(i, n)]$ 

```

Runs in $O(n)$ time

12

Average Case vs. Randomized

- **Average-case analysis** studies the average running time of deterministic algorithms.
 - Assume a random distribution of the input.
- **Randomized algorithms** make choices based on output of random values.
 - Impose a random distribution on some property of the input.
- Random choices are easier to make than provably good deterministic choices.
- Works well if the number of good choices is big.
- Randomized algorithms have **no worst-case inputs**.

13

Probability Theory: Review

- Random experiment
- *Sample space* is set S of possible outcomes
 - Example.* 2 dice yield pair (a, b) , $1 \leq a, b \leq 6$
- An *event* is a subset A of S
 - Example.* rolling 6 = $\{(1,5),(2,4),(3,3),(4,2),(5,1)\}$
 - The individual outcomes are called *elementary events*
- A probability distribution **Pr** is a mapping from events of S to \mathbb{R} (the reals) that satisfies the following axioms:
 1. $\Pr(A) \geq 0$
 2. $\Pr(S) = 1$
 3. $\Pr(A \cup B) = \Pr(A) + \Pr(B)$ if $A \cap B = \emptyset$
 - Example.* With fair dice, $\Pr(A) = |A| / 36$
- **Theorem.** $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$
 - Exercise.* What is the probability to roll 4 or doubles?

14

Discrete Probability Distribution

- A probability distribution is *discrete* if it is defined over a finite or countably infinite sample space
Examples. Number on die, # of comparisons of quicksort on a permutation of $1..n$, rank of pivot, etc.
- What is the probability of an event A in terms of the probability of elementary outcomes?

$$\Pr(A) = \sum_{s \in A} \Pr(s)$$

- To fully characterize the probability distribution it suffices to specify it for the elementary events (i.e., for the set of all possible outcomes)

15

Random Variables

- A *random variable* X is a function $S \rightarrow \mathbb{R}$
- The event $\{X = x\} = \{s \in S, X(s) = x\}$
- Expected value of X is $E(X) = \sum_x x \Pr(X = x)$
Example. With the 2 dice, let $X = a + b$
 $E(X) = 2(1/36) + 3(2/36) + 4(3/36) + \dots + 12(1/36) = 7$
- *Linearity.* $E(aX + Y) = aE(X) + E(Y)$, constant a
Example. X_1 = number on die 1, X_2 = number on die 2
 $E(X) = E(X_1 + X_2) = E(X_1) + E(X_2) = 2(1+2+\dots+6)/6$

16

Exercise

1. Consider the random experiment \mathcal{E} of flipping a fair coin three times
 - What is the sample space S of \mathcal{E} ?
 - What is the probability distribution of \mathcal{E} ?
2. Consider now the random variable $X: S \rightarrow \mathbb{N}$ defined as the number of heads observed
 - What is the sample space of X ?
 - What is the probability distribution of X ? What is $E[X]$?
3. Consider the random experiment of flipping a fair coin until heads appears for the first time. What is S ?
4. Consider the random variable defined by the number of flips required in (3). What is $E[Y]$?

17

Independence

- Two events A and B are *independent* iff $\Pr[A \cap B] = \Pr[A] \Pr[B]$
Exercise. Prove that $\Pr[A|B] = \Pr[A]$ iff A and B are independent
- Two random variables X and Y are *independent* iff $\Pr[X = x, Y = y] = \Pr[X = x] \Pr[Y = y], \forall x, y$
- If X and Y are independent then $E(XY) = E(X)E(Y)$

Example. Returning to the two dice example

$$\Pr[X = 12] = \Pr[X_1 = 6, X_2 = 6] = \Pr[X_1 = 6] \cdot \Pr[X_2 = 6] = (1/6)(1/6) = 1/36$$

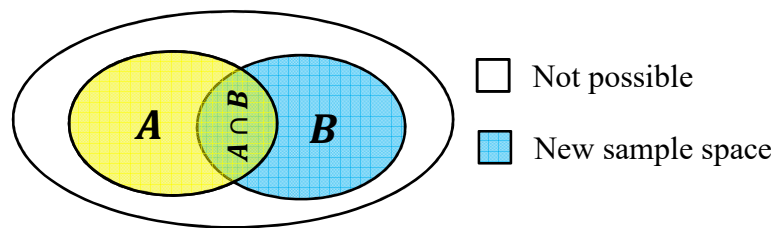
18

Conditional Probability

- The *conditional probability* of A given B is

$$\Pr[A|B] = \Pr[A \cap B] / \Pr[B]$$
- Corresponds to the probability that A happens if we know that B happened.

Example: $\Pr(X \geq 7 \mid X_2 = 4) = ?$



19

Exercise

- Consider the event of flipping two fair coins. Let A denote the event that at least one head occurs, and B the event that both heads occur
 - What are $\Pr[A|B]$ and $\Pr[B|A]$?
- Prove that $\Pr[A|B] = \Pr[B|A] \Pr[A] / \Pr[B]$
- You have a 4-sided die and a bunch of 6-sided dice. You roll the 4-sided die and, whatever number k comes up, you compute the sum S of the numbers obtained after rolling k 6-sided dice. Compute
 - $\Pr[S = 4]$
 - $\Pr[S = 4 | k = 1]$
 - $\Pr[k = 1 | S = 4]$

20

Indicator Variables

- An *indicator variable* is a random variable with sample space $\{0,1\}$
- *Notation.* For event A , define

$$I_A = I(A) = \begin{cases} 1 & \text{if } A \text{ occurs} \\ 0 & \text{if } A \text{ does not occur} \end{cases}$$

- What is the expected value of an indicator variable? $E[I_A] = \Pr[A]$
- What is the expected value of a sum of indicator variables? $E[I_A + I_B] = E[I_A] + E[I_B]$

21

Max Example Analysis

```

RANDOMIZEDMAX( $A, n$ )
1   $max \leftarrow -\infty$ 
2  Permute  $A$  uniformly at random
3  for  $i \leftarrow 1$  to  $n$ 
4      do if  $A[i] > max$ 
5          then  $max \leftarrow A[i]$ 
6  return  $max$ 

```

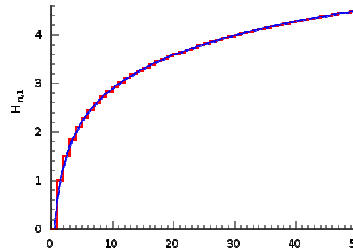
- Without indicator variables
Let $X = \#$ times max is updated $E[X] = \sum_{i=1}^n i \cdot \Pr[X = i]$
- With indicator variables:
Let $X_i = I[\text{max is updated in iteration } i]$
Then $X = X_1 + X_2 + \dots + X_n$
What is $E[X_i]$? $E[X_i] = 1/i$

22

Harmonic Numbers

- The n -th harmonic number is defined as:

$$H_n = \sum_{i=1}^n \frac{1}{i}$$

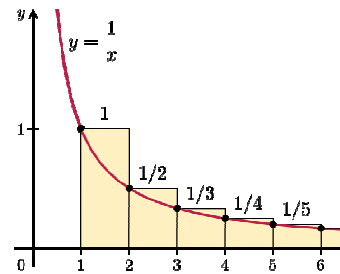


- How does H_n grow?

$$\ln(n+1) < H_n < 1 + \ln n$$

$$\lim_{n \rightarrow \infty} H_n - \ln n = \gamma$$

$$\gamma = 0.5772156649\dots$$



23

TV Game Revisited (with indicator variables)

Let $X_i = I [i\text{-th box was a mistake}]$

Then $m = X = X_1 + X_2 + \dots + X_n$

What is $E[X_i]$? $\leq 1/i$

What is $E[X]$? $\leq H_n = \sum_{i=1}^n \frac{1}{i}$

HarmonicNumber[100] = 5.18738

24

Exercise

- Suppose you have n incoming students that need to be assigned to d dorms. You are given a list of m incompatible pairs (two students in one such pair should not be assigned to the same dorm). Your goal is to maximize the number X of satisfied constraints (i.e., pairs in which the two students are assigned to different dorms). If you assign the dorms at random, what is the expected value of X ?

25

Markov's Inequality

Theorem. If X is a non-negative random variable then $P(X \geq t) \leq E[X]/t$

Proof. Define an indicator variable $Y = I(X \geq t)$. Then, $P(X \geq t) = E(Y)$. Since $Y \leq X/t$ for all t , then $P(X \geq t) = E(Y) \leq E(X/t) = E[X]/t$

Example. In the context of the TV game, find an upper bound for the probability of losing.

26

Types of Randomized Algorithms

- *Montecarlo* (e.g., randomized primality test)
 - Probably correct, provably fast
- *Las Vegas* (e.g., randomized quicksort)
 - Probably fast, provably correct
- Transformations
 - Convert Las Vegas B to Montecarlo B' by truncating
 - Stop B if it is taking too long. Since B runs fast with high probability then B' is correct with high probability
 - Convert Montecarlo A to Las Vegas A' by iterating
 - Repeatedly run A until a correct answer is found
 - Requires certification; otherwise risky!

27

Design Paradigms

- *Input randomization*
 - Permute the input to get rid of undesirable patterns
 - Randomized sorting, BSTs, convex hull
- *Control randomization*
 - Select algorithm from a parameterized set of algorithms
 - Universal hashing, string matching
- *Fingerprinting*
 - Determine equivalence of objects by comparing “fingerprints”
 - Pattern matching, bipartite graph matching
- *Abundance of witnesses*
 - If many elements of a set satisfy a desirable property, find one of them by *random sampling*
 - Primality testing, perfect hashing, polynomial equivalence

28

Tools for Analysis

- Basic identities and inequalities
 - *Success amplification.* If the probability of success of a single run is p then the probability of success in k independent runs is $1 - (1 - p)^k$
 - If p is the probability of success of a single run, then expected number of runs to success is $p + 2p(1 - p) + 3(1 - p)^2p + \dots + i(1 - p)^{i-1}p + \dots = 1/p$
 - *Markov's inequality.* $\Pr[X \geq t] \leq E[X]/t$
- Indicator variables
- Backwards analysis

29

Randomized Quicksort

- A Las Vegas algorithm

RANDOMIZED-QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2      then  $q \leftarrow \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3          RANDOMIZED-QUICKSORT( $A, p, q - 1$ )
4          RANDOMIZED-QUICKSORT( $A, q + 1, r$ )

```

RANDOMIZED-PARTITION(A, p, r)

```

1   $i \leftarrow \text{RANDOM}(p, r)$ 
2  exchange  $A[r] \leftrightarrow A[i]$ 
3  return PARTITION( $A, p, r$ )

```

30

Analysis

- Complexity dominated by time spent partitioning
- One call to partition requires $\Theta(1 + Z)$ time, where $Z = \#$ comparisons in line 4 of Partition
- Total time is $\Theta(\rho + X)$, where
 - $\rho = \#$ calls to partition
 - $X =$ total # executions of line 4 (throughout entire run)

```

PARTITION(A, p, r)
1  x ← A[r]
2  i ← p - 1
3  for j ← p to r - 1
4      do if A[j] ≤ x
5          then i ← i + 1
6              exchange A[i] ↔ A[j]
7  exchange A[i + 1] ↔ A[r]
8  return i + 1

```

31

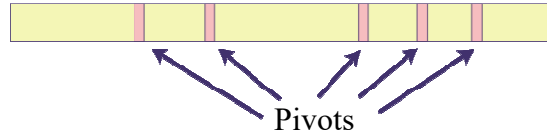
Notation

- $z_i =$ element of A whose rank is i
 - Note: $z_1 \leq z_2 \leq \dots \leq z_n$ and $\forall i: z_i = A[j]$, for some j
- $Z_{ij} = \{z_i, \dots, z_j\}$
- $X_{ij} = \#$ times z_i is compared to z_j
 - This is an indicator variable! Why?
- Then

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij}, \text{ where } X_{ij} = I[z_i \text{ compared to } z_j]$$

32

Probability of Comparing 2 Elements

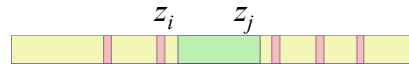


- Consider the set of pivots chosen so far.
- Every element has been compared to some of these pivots to produce the current partition into chunks.
- No two elements in *the same* chunk have been compared to each other.
- Two elements in different chunks will never be compared to each other.

33

Probability of Comparing 2 Elements

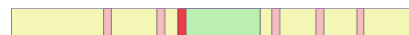
- Now consider the first pivot x chosen from Z_{ij}



- If $x \notin \{z_i, z_j\}$, then z_i and z_j will never be compared



- If $x \in \{z_i, z_j\}$, then z_i and z_j are compared when partitioning around x the chunk containing Z_{ij}



$$Pr(z_i \text{ and } z_j \text{ are compared}) = Pr(z_i \text{ or } z_j \text{ is the first pivot chosen from } Z_{ij}) = \frac{2}{j - i + 1}$$

34

Analysis

Since

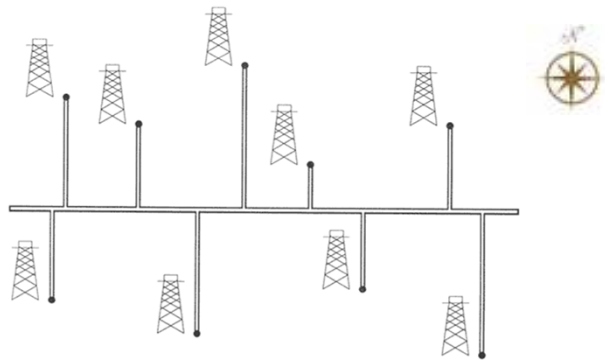
$$E[X_{ij}] = \frac{2}{j-i+1}$$

we have

$$\begin{aligned} E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j-i+1} = \\ &\sum_{i=1}^{n-1} \sum_{j=2}^{n-i+1} \frac{2}{j} < \sum_{i=1}^{n-1} \sum_{j=1}^n \frac{2}{j} < \sum_{i=1}^{n-1} 2(1 + \ln n) = O(n \log n) \end{aligned}$$

35

Minimizing Pipe Length



- What is the optimal location of the West-East pipeline that minimizes total pipe cost

Exercise. Prove that the median of the y 's is optimal

36

Order Statistics

- The i -th order statistic of a set of n elements is the i -th smallest element of the set, i.e., element of rank i

Example: $S = \{8, 40, 30, 17, 31, 4, 50, 42, 41, 27, 19\}$

$i = 3 \Rightarrow 3^{\text{rd}}$ smallest element is 17

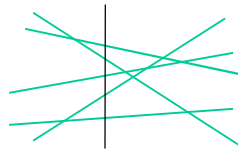
$i = 1 \Rightarrow$ smallest element (minimum) is 4

$i = n \Rightarrow$ largest element (maximum) is 50

$i = \lfloor (n+1)/2 \rfloor$?

What is the median for an even number of elements?

- The elements need not be numbers



37

1st Order Statistic

MINIMUM(A)

```

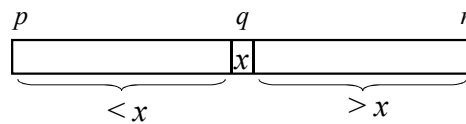
1   $min \leftarrow A[1]$ 
2  for  $i \leftarrow 2$  to  $length[A]$ 
3      do if  $min > A[i]$ 
4          then  $min \leftarrow A[i]$ 
5  return  $min$ 
```

- Cost = number of comparisons (line 3)
- Can we do better?

38

i -th Order Statistic

- Can *partition* help?



- What is the relation between i , p , and q ?
- Simple recursive algorithm

$$T(n) = T(k) + \Theta(n), \text{ for some } k < n$$

39

A Las Vegas Algorithm

RANDOMIZED-SELECT(A, p, r, i)

```

1  if  $p = r$ 
2    then return  $A[p]$ 
3   $q \leftarrow \text{RANDOMIZED-PARTITION}(A, p, r)$ 
4   $k \leftarrow q - p + 1$ 
5  if  $i = k$             $\triangleright$  the pivot value is the answer
6    then return  $A[q]$ 
7  elseif  $i < k$ 
8    then return RANDOMIZED-SELECT( $A, p, q - 1, i$ )
9  else return RANDOMIZED-SELECT( $A, q + 1, r, i - k$ )
```

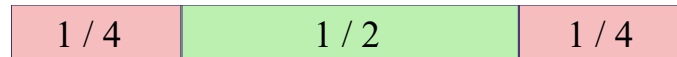
$T(n)$ = time for randomized select with n elements

- A random variable
- $E[T(n)]$?
- Worst case?

40

Informal Analysis

- Choose the pivot at random.
- Consider the sorted array



- At least half of the elements produce a 25%:75% split or better.
- Roughly every other iteration the array reduces by at least 25%

$$n \rightarrow \frac{3n}{4} \rightarrow \frac{9n}{16} \rightarrow \dots \rightarrow \left(\frac{3}{4}\right)^k n$$

41

A Las Vegas Algorithm

RANDOMIZED-SELECT(A, p, r, i)

```

1  if  $p = r$ 
2    then return  $A[p]$ 
3   $q \leftarrow \text{RANDOMIZED-PARTITION}(A, p, r)$ 
4   $k \leftarrow q - p + 1$ 
5  if  $i = k$            ▷ the pivot value is the answer
6    then return  $A[q]$ 
7  elseif  $i < k$ 
8    then return RANDOMIZED-SELECT( $A, p, q - 1, i$ )
9  else return RANDOMIZED-SELECT( $A, q + 1, r, i - k$ )
```

$T(n)$ = time for randomized select with n elements

- A random variable
- $E[T(n)]$?
- Worst case?

42

Formal Analysis

- $T(n)$ is our random variable whose expected value we wish to find.
- Element z_k of rank k , chosen with probability $1/n$
- Since $E(T(n))$ is monotonically increasing:

$$E[T(n)] \leq \frac{1}{n} \sum_{k=1}^n E[T(\max(k-1, n-k)) + \Theta(n)]$$

$$\max(k-1, n-k) = \begin{cases} k-1 & \text{if } k > \lceil n/2 \rceil \\ n-k & \text{if } k \leq \lceil n/2 \rceil \end{cases}$$

$$E[T(n)] \leq \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} E[T(k)] + \Theta(n)$$

43

Claim: $E[T(n)] \leq cn$

- Base case?
 - Let k satisfy $T(n) \leq cn$, if $n \leq d$ (will find d later)
- Inductive step

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} E[T(k)] + an \\ &\leq \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} ck + an \\ &= \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k \right) + an \end{aligned}$$

44

$$\begin{aligned}
E[T(n)] &\leq \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k \right) + an \\
&= \frac{2c}{n} \left(\frac{n(n-1)}{2} - \frac{\lfloor n/2 \rfloor (\lfloor n/2 \rfloor - 1)}{2} \right) + an \\
&\leq \frac{2c}{n} \left(\frac{n(n-1)}{2} - \frac{(n/2 - 1)(n/2 - 2)}{2} \right) + an \\
&= \frac{c}{n} \left(\frac{3n^2}{4} + \frac{n}{2} - 2 \right) + an \\
&= c \left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n} \right) + an \\
&\leq \frac{3cn}{4} + \frac{c}{2} + an \\
&= cn - \left(\frac{cn}{4} - \frac{c}{2} - an \right) \leq cn \text{ if } c > 4a, n \geq \frac{2c}{c-4a}
\end{aligned}$$

45

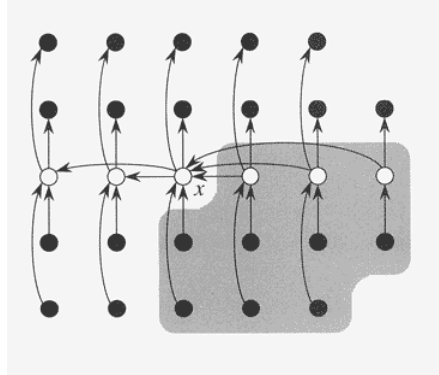
A Deterministic Linear Time Algorithm

- Worst case is $O(n)$ time
- Idea: use as pivot a reasonable estimate of the true median
 1. Divide the input into $\lceil n/5 \rceil$ groups of 5 elements
 2. Insertion-sort each group of 5, independently, and record its median (third smallest value)
 3. Recursively find the median x of the medians
 4. Partition A using x
 5. Select recursively on one of the two sub-arrays

Ex: $A = \{9, 6, 14, 16, 87, 21, 11, 42, 31, 56, 62, 21, 71, 90, 81, 10, 2, 5, 22, 26, 13, 52, 74, 56, 6, 9, 14, 16, 87, 11, 21, 31, 42, 56, 21, 62, 71, 81, 90, 2, 5, 10, 22, 26, 13, 52, 56, 74\}$

46

A Reasonably Balanced Partition



$$n - k \geq 3 \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2 \right) \geq \frac{3n}{10} - 6 \Rightarrow k \leq \frac{7n}{10} + 6$$

47

Analysis

$$T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + an$$

- $T(n) \in O(1)$ when $n < 140$ (or some other constant)
- Claim: $T(n) \leq cn$ for all n

$$\begin{aligned} T(n) &\leq c \lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\leq c(n/5 + 1) + 7cn/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn - (cn/10 - 7c - an) \\ &\leq cn, \text{ for } n \geq 140 \text{ if } c \geq 20a \end{aligned}$$

48

Closest Pair

Input: set S of points in 2D

Output: distance between two closest points in S

- Problem has $\Omega(n \log n)$ lower bound
- DAC solution takes $\Theta(n \log n)$
- Can do better with randomized algorithm
 - Las Vegas incremental, $O(n)$ expected time
 - $d(p, q)$: Euclidean distance between p and q
 - δ^* : actual CP distance
 - δ : (upper bound) estimate on CP distance

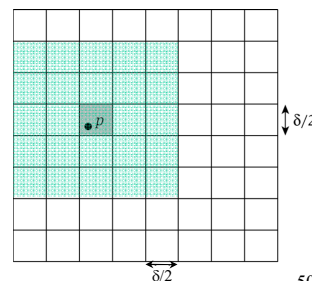
49

General Idea

- Randomly permute the points: p_1, p_2, \dots, p_n
(Let $\delta_i :=$ closest pair distance of p_1, p_2, \dots, p_i)
- Initially, $\delta := d(p_1, p_2) = \delta_2$
- For $i = 3$ to n , update $\delta := \delta_i$

Key Problem: How do we tell if p_i updates δ ?

- Keep a partition of the plane into $\delta/2 \times \delta/2$ cells
- Neighborhood of p is 5×5 sub-grid centered at p 's cell



50

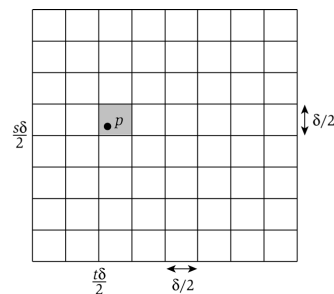
Properties

- No two points lie in same cell
If p and q lie in the same cell then $d(p,q) < \delta$
- If $d(p,q) < \delta$ then q is in p 's neighborhood
- Enough to check neighborhood of p_i
 1. Neighborhood is empty $\Rightarrow \delta_i = \delta_{i-1}$
 2. Neighborhood not empty \Rightarrow for each point p_j ($j < i$) in the neighborhood, check $d(p_i, p_j)$, update δ , and regrid with new δ (if estimate improved)

51

Managing the Grid

- Operations: create, insert, lookup
- Use a dictionary, e.g., a hash table H
- H stores non-empty grid cells
- Cells have integer coordinates (s,t)



$$S_{st} = \{(x, y) : s\delta/2 \leq x < (s+1)\delta/2; t\delta/2 \leq y < (t+1)\delta/2\}$$

$$p = (x, y) \rightarrow \underbrace{\left(\left\lfloor \frac{x}{\delta/2} \right\rfloor, \left\lfloor \frac{y}{\delta/2} \right\rfloor \right)}_{\text{key for } (x,y)}$$

52

A Las Vegas Algorithm

```

Order the points in a random sequence  $p_1, p_2, \dots, p_n$ 
Let  $\delta$  denote the minimum distance found so far
Initialize  $\delta = d(p_1, p_2)$ 
Invoke MakeDictionary for storing subsquares of side length  $\delta/2$ 
For  $i = 1, 2, \dots, n$ :
    Determine the subsquare  $S_{st}$  containing  $p_i$ 
    Look up the 25 subsquares close to  $p_i$ 
    Compute the distance from  $p_i$  to any points found in these subsquares
    If there is a point  $p_j$  ( $j < i$ ) such that  $\delta' = d(p_j, p_i) < \delta$  then
        Delete the current dictionary
        Invoke MakeDictionary for storing subsquares of side length  $\delta'/2$ 
        For each of the points  $p_1, p_2, \dots, p_i$ :
            Determine the subsquare of side length  $\delta'/2$  that contains it
            Insert this subsquare into the new dictionary
        Endfor
    Else
        Insert  $p_i$  into the current dictionary
    Endif
Endfor

```

53

Analysis

- What is the cost of the i -th iteration?
 - $O(1)$ if δ does not change
 - $O(i)$ if δ changes

$$X_i = I\{p_i \text{ changes } \delta\} = \begin{cases} 0 & \text{if } \delta \text{ does not change} \\ 1 & \text{if } \delta \text{ changes} \end{cases}$$

$$E[X_i] = \Pr\{p_i \text{ changes } \delta\} \leq 2/i$$

$$E[T(n)] = n + \sum_{i=1}^n E(X_i)O(i) = O(n)$$

54

A Monte Carlo Sorting Algorithm

- Run $k/2$ independent instances of Las Vegas quicksort and halt after a combined total of $k \cdot E(T)$ steps, if not done
- The probability of success is at least $1 - (1/2)^{k/2}$ as each run of $2E(T)$ steps has probability of failure at most $1/2$ and all runs fail with probability at most $(1/2)^{k/2}$
- Runs in $O(n \log n)$ time

Verifying a Matrix Product

- Given $k \times k$ matrices A, B, C , determine if $C = A \times B$
- We can solve deterministically by carrying out the product in $O(k^{\log 7})$ time and comparing two matrices in $O(k^2)$ additional time
- Can we do better with a randomized algorithm?
 1. Choose a random vector r uniformly from $\{0,1\}^k$
 2. Compute Cr and $(A(Br))$
 3. If $Cr = (A(Br))$ report $C = A \times B$ else $C \neq A \times B$
- Randomized algorithm takes $O(k^2)$ time but can make one-sided errors (when $C \neq AB$)

Correctness

Claim. Let A, B, C be $k \times k$ matrices such that $C \neq AB$. Then for r chosen uniformly at random from $\{0,1\}^k$, we have $\Pr[Cr = ABr] \leq 1/2$

Proof. Let $D = C - AB \neq \mathbf{0}$. Without loss of generality let's assume that $d_{11} \neq 0$. We want to upper bound $\Pr[Dr = 0]$. For $Dr = 0$, we must have

$$\sum_{j=1}^k d_{1j}r_j = 0 \quad \text{i.e., } r_1 = -\frac{\sum_{j=2}^k d_{1j}r_j}{d_{11}} \quad (*)$$

For any fixed but random setting of r_2, \dots, r_k , there is at most one of the two possible values of r_1 satisfies (*).

Therefore, $\Pr[Dr = 0] \leq 1/2$ ■

57

Reducing the Error

- Success amplification: run the algorithm t times with independent and uniform vectors r (success amplification)
 - Let E_i = event that run i returns wrong answer

$$\Pr[E_1 \cap E_2 \cap \dots \cap E_t] = \prod_{i=1}^t \Pr[E_i] \leq \left(\frac{1}{2}\right)^t$$

$$\Pr[\text{success}] \geq 1 - (0.5)^t$$

Exercise

- What is the probability of error by the Monte Carlo algorithm for matrix product verification if we choose r uniformly at random from $\{0,1,2\}^k$

59

Fingerprinting

- The matrix product verification algorithm is an instance of a more general *fingerprinting* technique used to decide the equality of two elements x and y drawn from a large universe U
- Under any reasonable model of computation determining equality requires $\Omega(\log|U|)$ operations
- Instead, pick a random mapping γ from U into a much smaller universe V such that the probability is high that $x = y$ iff $\gamma(x) = \gamma(y)$
- Thus, after computing γ , equality can be determined in $O(\log|V|)$ time by comparing fingerprints

Testing Polynomial Identities

Is $(2x + 1)(x - 2)(4x + 3)(x + 5)(3x - 4) = 120 + 274x - 85x^2 - 277x^3 + 70x^4 + 24x^5$?

- Given polynomial G and binomials F_1, F_2, \dots, F_n determine if $G(x) = F_1(x) \cdot F_2(x) \cdots F_n(x)$
- Can multiply the F_i 's and compare the coefficients with G
- How long does this take?
 - Takes $\Theta(2^n)$ time where n is the number of binomials as well as the max degree of G
 - Can we do better?


A Monte Carlo Algorithm

- Choose a random value r in $\Omega = \{1, \dots, 100n\}$, evaluate all polynomials at r , and check if $F_1(r)F_2(r) \cdots F_n(r) = G(r)$
- Takes $O(n)$ time only
- Error is possible *but one-sided*:
 - If G has degree $> n$ or $F_1(r) \cdots F_n(r) \neq G(r)$
 - \Rightarrow conclude *identity false*
 - correct conclusion
 - If $F_1(r) \cdots F_n(r) = G(r) \Rightarrow$ conclude *identity true*
 - may be correct or incorrect
- Since $Q(x) = F_1(x) \cdots F_n(x) - G(x)$ has $\leq n$ roots $\Pr[\text{wrong answer}] \leq n/100n = 1\%$

Reducing the Probability of Error

- Our algorithm is correct $\geq 99\%$ of the time
- How can we improve confidence?
 1. Increase the size of the sample space
 - With $\Omega = \{1, \dots, 1000n\}$, $\Pr[\text{error}] \leq 0.1\%$
 2. Success amplification
 - Run the algorithm multiple times
 - What is the probability of error if we run the algorithm twice? k times?

$$\Pr(\text{success}) \geq 1 - (0.01)^k$$


 prob. error in 1 run

Generalization

- Given polynomials $Q(x_1, \dots, x_n)$ and $R(x_1, \dots, x_n)$ on $n \geq 1$ variables, determine whether $Q = R$
 - Equivalently, determine if

$$P(x_1, \dots, x_n) = Q(x_1, \dots, x_n) - R(x_1, \dots, x_n) = 0$$
- Brute force: expand Q and R as sum of monomials and compare coefficients
 - May take exponential time

Schwartz-Zippel Theorem

Definition. Let $P(x_1, \dots, x_n)$ be a multivariate polynomial. The degree of a term in P is the sum of the exponents of the variables involved. The *total degree* of P is the maximum of the degrees of its terms.

Example: $2x^3 - 3x^2y^2 + 5xy^2 + 7y - 2$

Claim. Let $P(x_1, \dots, x_n)$ be a **non-zero** multivariate polynomial of total degree d . Let r_1, \dots, r_n be chosen at random from any fixed set S of size m . Then

$$\Pr[P(r_1, \dots, r_n) = 0] \leq \frac{d}{m}$$

Proof

- By induction on n (# of variables)
- *Base case.* If $n = 1$, then $P(x_1)$ is a polynomial of degree d on one variable with $\leq d$ roots. Hence, $\Pr[P(r_1) = 0] \leq d/m$.
- *Inductive step.* Let $n > 1$ and k the max degree of x_1 in P . Then $P(x_1, \dots, x_n) = Q(x_2, \dots, x_n)x^k + R(x_1, \dots, x_n)$ where Q has degree $\leq d - k$ and the degree of x_1 in R is $< k$.

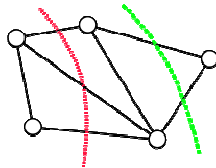
Let E denote the event $Q(r_2, \dots, r_n) = 0$. By the inductive hypothesis, $\Pr[E] \leq (d - k)/m$. Furthermore, if $\neg E$, after $x_2 \dots x_n$ have been set, $P' = P(x_1, r_2, \dots, r_n)$ is a non-zero polynomial of degree k in one variable. Hence, $\Pr[P'(r_1) = 0 | \neg E] \leq k/m$.

By the *law of total probability* on E and $\neg E$:

$$\begin{aligned} \Pr[P(r_1, \dots, r_n) = 0] &= \overbrace{\Pr[P(r_1, \dots, r_n) = 0 | E]}^{\leq 1} \cdot \Pr[E] + \\ &\Pr[P(r_1, \dots, r_n) = 0 | \neg E] \cdot \Pr[\neg E] \leq (d - k)/m + k/m = d/m \end{aligned}$$

The Min-Cut Problem

- A *cut* in a connected, undirected, multigraph $G(V,E)$ is a set of edges whose removal results in G being broken into two or more components
- Applications: clustering, identifying network bottlenecks, community detection in social networks
- Formally, a cut is a partition of V into two non-empty sets X and Y . The *crossing edges* of the cut $C = (X,Y)$ are those with one endpoint in X and one in Y
- *Goal*: find a cut of minimum cardinality

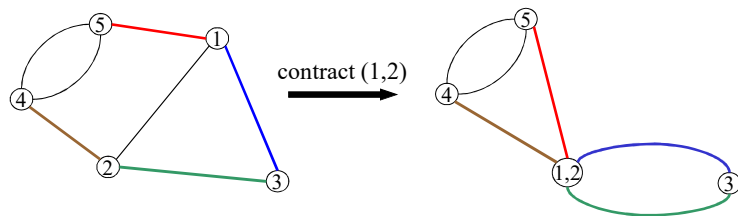


A Brute Force Algorithm

- Try all possible cuts (X,Y) . For each, find all edges in $X \times Y$ and report one with smallest number of crossing edges
- How many cuts are there?
 - Can encode a cut as a binary number $1 \leq c \leq 2^n - 2$
- How many edges are there in a min-cut?
- Since the size of the cut is its number of crossing edges, we may not be able to stop early and you may end up examining an exponential number of cuts
- Will present a Monte Carlo algorithm (Karger 1993)

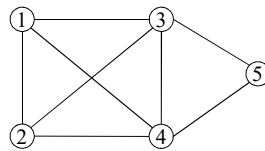
Karger's Contraction Algorithm

1. **while** there are more than 2 vertices
 - a. pick a remaining edge $e = (u, v)$ uniformly at random
 - b. contract e by fusing u and v into a single vertex
 - c. remove any resulting self-loops
 2. **return** the cut between the final two vertices
- The **contraction** of $e = (u, v)$ is done by eliminating all edges between u and v and retaining all other edges

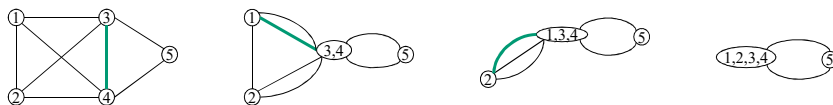


69

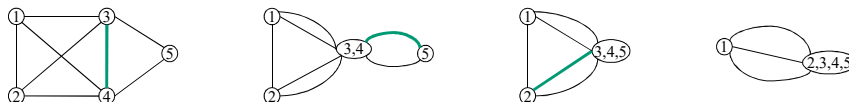
Example: 2 Randomized Runs



1. Contract (3,4), (1,4), (1,2)



2. Contract (3,4), (4,5), (2,3)

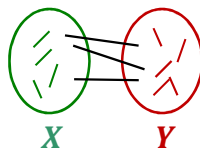


Properties

- Our algorithm outputs a valid cut but not necessarily a minimum cut
- Any cut of an *intermediate* graph is a cut of the original graph
- Not every cut of the original graph is a cut of an intermediate graph
- A contracted edge is not part of the cut we output
- If there is a min cut none of whose edges have been contracted then a min cut of an intermediate graph is also a min cut of the original graph
- A contraction does not reduce the min cut size but may increase it

Analysis

- Fix a particular target cut $C = (X, Y)$ of $G(V, E)$. What is the probability that we output C ?
- When does the algorithm fail to find C ?
 - If an edge of C is contracted $\Rightarrow C$ will not be output
 - If only edges inside X or Y are contracted $\Rightarrow C$ is output



- Therefore,

$$\Pr[\text{output } C] = \Pr[\text{never contract an edge of } C]$$

Theorem. The contraction algorithm outputs a valid cut with probability at least $2/(n(n-1))$

Proof

- Let C be a particular min cut of size k
 - $n_i = \#$ of vertices before i^{th} iteration, $n_i = n - i + 1$
 - $m_i = \#$ of edges before i^{th} iteration
 - $E_i =$ event that algorithm **does not** pick an edge of C in i^{th} iteration

$$\Pr[E_1 \cap E_2 \cap \dots \cap E_{n-2}] =$$

$$\Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1 \cap E_2] \cdots \Pr[E_{n-2}|E_1 \cap \dots \cap E_{n-3}]$$

- What is $\Pr[E_1]$? $\Pr[E_1] = 1 - \frac{k}{m}$ don't know m_2 !
- How about $\Pr[E_2|E_1]$? $\Pr[E_2|E_1] = 1 - \frac{k}{m_2}$ ↖

Proof...

Lemma. At all times, $m_i \geq kn_i/2$

Proof. For every vertex v , in any iteration, $\deg v \geq k$. The handshaking lemma implies $\sum \deg v = 2m_i \geq kn_i$

- Thus, $\Pr[E_1] = 1 - \frac{k}{m} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$
- $\Pr[E_2|E_1] = 1 - \frac{k}{m_2} \geq 1 - \frac{k}{k(n-1)/2} = 1 - \frac{2}{n-1}$
- And more generally,

$$\Pr[E_i|E_1 \cap \dots \cap E_{i-1}] = 1 - \frac{k}{m_i} \geq 1 - \frac{2}{n-i+1}$$

Proof...

- Since $\Pr[E_i | E_1 \cap \dots \cap E_{i-1}] = 1 - \frac{k}{m_i} \geq 1 - \frac{2}{n-i+1}$

- Then,

$$\Pr[E_1 \cap E_2 \cap \dots \cap E_{n-2}] \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1}$$

$$= \frac{\cancel{n-2}}{n} \times \frac{\cancel{n-3}}{n-1} \times \frac{\cancel{n-4}}{\cancel{n-2}} \times \dots \times \frac{\cancel{3}}{\cancel{5}} \times \frac{\cancel{2}}{\cancel{4}} \times \frac{\cancel{1}}{\cancel{3}} = \frac{2}{n(n-1)}$$

as claimed

Success Amplification

- Run the algorithm N times and report smallest cut found
- The output is not our min cut with Prob $\leq \left(1 - \frac{2}{n(n-1)}\right)^N$
- Setting $N = \binom{n}{2}$, the probability of failure is no more than $1/e \approx 37\%$
- If we run the algorithm $n(n-1) \ln n$ times, the probability that the output is not our min cut is

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

79