**CMPSCI 611 Advanced Algorithms** **Fall 2001**

# Lecture 18: November 14

*Lecturer: Micah Adler* *Scribe: Raksit Ashok*

In this lecture:

- Median Finding
- Chernoff Bounds
- NP-completeness

## 18.1 Median Finding

In the last lecture, we presented a randomized algorithm for finding the median in an ordered set S:

- Take a random sample R of S, of size $n^{3/4}$, $n = |S|$
- Sort R, find $a, b \in R$ such that $\text{rank}_R(\text{a}) = \frac{n^{3/4}}{2} - \sqrt{n}$ $\text{rank}_R(\text{b}) = \frac{n^{3/4}}{2} + \sqrt{n}$
- ...

This algorithm works if:

- $a \leq m \leq b$, m is the median
- $|S'|$ is not too large

We will now derive a bound on the probability that this algorithm fails. For this we refer to the Chebyshev's inequality discussed last time:

**Theorem 18.1** *Let $x$ be a random variable with expectation $\text{E}[x] = \mu$ and variance $\text{Var}[x] = \sigma^2$. Then, for any $t \in \Re^+$.*

$$\Pr[|x - \mu| \geq t] \leq \frac{\sigma^2}{t^2}$$

**Theorem 18.2**

$$\Pr[Algorithm fails] \leq \frac{1}{n^{1/4}}$$

**Proof:** Algorithm fails on any of the following:

1. $\text{rank}_S(\text{a}) \geq \frac{n}{2}$ (median sandwiched?)
2. $\text{rank}_S(\text{a}) \leq \frac{n}{2} - 2n^{3/4}$ ($|S'|$ too big?)
3. $\text{rank}_S(\text{b}) \leq \frac{n}{2}$

4. $\text{rank}_S(\text{b}) \geq \frac{n}{2} + 2n^{3/4}$

- (1) only holds if less than $\frac{n^{3/4}}{2} - \sqrt{n}$ elements in R are left of m:
  Let X be the number of elements of R left of m. X can be written as:
  $X = X_1 + X_2 + \ldots + X_{n^{3/4}}$, where $X_i = \begin{cases} 1 & \text{If } i\text{th sample left of m} \\ 0 & \text{Otherwise} \end{cases}$

$$Pr[X_i = 1] = \tfrac{1}{2},\ E[X_i] = \tfrac{1}{2}$$
$$Var(X_i) = E[(X_i - E[X_i])^2] = \tfrac{1}{4} \text{ (because } |X_i - E[X_i]| \text{ is always } \tfrac{1}{2})$$

  Therefore:

  $E[X] = \frac{n^{3/4}}{2}$ (linearity of expectation)

  $Var(x) = \frac{n^{3/4}}{4}$ (variance is linear if variables are independent: they *are* because R picked *with* replacement)

  Using Chebyshev's inequality:
$$Pr[|X - \tfrac{n^{3/4}}{2}| \geq \sqrt{n}] \leq \frac{n^{3/4}/4}{(\sqrt{n})^2} = \frac{n^{3/4}}{4n} = \frac{1}{4n^{1/4}}$$

- Let V be the element such that $\text{rank}_S(\text{V}) = \frac{n}{2} - 2n^{3/4}$
  (2) only holds if more than $\frac{n^{3/4}}{2} - \sqrt{n}$ elements in R are left of V:
  Let Y be the number of elements in R left of V. As before:
  $Y = Y_1 + Y_2 + \ldots + Y_{n^{3/4}}$

$$E[Y_i] = \frac{number\ of\ elements\ in\ S\ left\ of\ V}{number\ of\ elements\ in\ S} = \frac{\frac{n}{2} - 2n^{3/4}}{n} = \frac{1}{2} - \frac{2}{n^{1/4}}$$
$$Var(Y_i) = (\tfrac{1}{2} + \tfrac{2}{n^{1/4}})(\tfrac{1}{2} - \tfrac{2}{n^{1/4}}) \leq \tfrac{1}{4}$$

  Therefore:

  $E[Y] = \frac{n^{3/4}}{2} - 2\sqrt{n}$

  $Var(Y) \leq \frac{n^{3/4}}{4}$

  Using Chebyshev's inequality:
$$Pr[|Y - (\tfrac{n^{3/4}}{2} - 2\sqrt{n})| \geq \sqrt{n}] \leq \tfrac{1}{4n^{1/4}}$$

- Probabilities for (3) and (4) can similarly found to be the same.

Since these 4 events are mutually exclusive, the probability that the algorithm fails is the sum of these 4 probabilities $\leq 4 * \frac{1}{4n^{1/4}} = \frac{1}{n^{1/4}}$ ■

## 18.2 Chernoff Bounds [MR95]

Take the example of tossing a coin. $x_1, x_2, ..., x_n$ are independent coin tossing events such that, for $1 \leq i \leq n$, $\Pr[x_i = head] = p$. Let $X = \sum_{i=1}^{n} x_i$, then, $X$ is said to have the binomial distribution, which is represented as

$$B(n, p).$$

In other words, $B(n,p)$ is the number of heads you expect to see after $n$ flips, with each flip being heads with probability $p$.

For any $\delta > 0$, the Chernoff bounds give:

$$\Pr[B(n,p) \geq (1+\delta)np] \leq \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{np}$$

There are simplified forms of this bound for the special cases where $0 < \delta \leq 1$, which means that the values are not too far away from the expectation $\mu$. These special cases are:

1. $\Pr[B(n,p) \leq (1-\delta)np] \leq e^{-\delta^2 np/2}$
2. $\Pr[B(n,p) \geq (1+\delta)np] \leq e^{-\delta^2 np/3}$

### 18.2.1 Example 1: The Red Sox Have a Winning Season

Assume that the Red Sox win each game independently with probability $p = \frac{1}{4}$. What is the probability Pr[Red Sox have a winning season]?, where winning season means the team wins more than half the games played.

Since $p = \frac{1}{4}$, and $np = \frac{n}{4}$, we know that

$$(1+\delta)np = \frac{n}{2}$$

That is, we have $\delta = 1$. Substituting these values into the special case 2 of the Chernoff bounds, we obtain:

$$\begin{aligned} \Pr[\text{winning season}] &\leq e^{\frac{-np}{3}} \\ &= e^{\frac{-n}{12}} \end{aligned}$$

The Chernoff bounds give us a very tight bound for this probability: exponentially small versus the linearly small ($\frac{3}{n}$) of Chebyshev's inequality.

### 18.2.2 Example 2: The Coupon Collector's Problem

In this problem, there are $n$ types of coupons, and at each trial a coupon is chosen randomly. The random choice of the coupons are mutually independent.

**Question:** How many trials needed till all the coupons are collected?

This problem is related to many randomized algorithms in computer science. Here is an application.

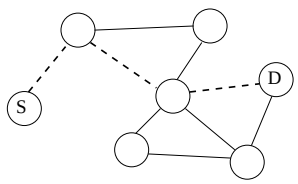


Figure 18.1: An abstracted view of the Internet.

**Application:** Figure 16.1 shows an abstracted view of the Internet. The messages are sent through the network from a source $S$ to a destination $D$. This is shown as the dotted line in the figure. We want to do is trace the source and the path of the message: the easy way is to append to the message the address of every hop along the way to the destination. However, this will significantly increase the length of the message when we have many links in the path. Instead of adding *every* hop's address, we will randomly choose which hop's address to add to the message. So the question becomes, how many times must a message be sent from $S$ to $D$, in order to collect the address of every hop along the path?

**Claim:** After $4n \ln n$ trials, the probability Pr[coupon missing] $\leq \frac{1}{n}$.

**Proof:** $X$ is the random variable defined to be number of coupons of type 1 after $N = 4n \ln n$ trials. We have:

$$X = \sum_{i=1}^{N} x_i$$

Where:

$$x_i = \begin{cases} 1 & \text{If } i\text{th trial is of type 1} \\ 0 & \text{Otherwise} \end{cases}$$

We now compute the probability $\Pr[X = 0]$. Since $p = \Pr[x_i = 1] = \frac{1}{n}$, we get:

$$Np = \frac{N}{n} = 4 \ln n$$

Here we want to use special case 1 of the Chernoff bounds, because we are looking for the probability $\Pr[X = 0]$. Therefore $\delta = 1$. Thus,

$$\begin{aligned} \Pr[\text{no coupons of type 1}] &\leq e^{-4 \ln n/2} \\ &= \frac{1}{n^2} \end{aligned}$$

So multiply this probability by the number of different types of coupons, which is $n$:

$$\begin{aligned} \Pr[\text{any missing coupons}] &\leq \frac{1}{n^2} \times n \\ &= \frac{1}{n} \end{aligned}$$

■

### 18.2.3 Some Different Scenarios

There are other types of applications of Chernoff bounds. We we list a few:

**(a)** Coin flips with different probabilities, namely, the probability Pr[flip $i$ is a head] $= p_i$. Then, the expectation is $\sum_{i=1}^{n} p_i$. To use the Chernoff bounds, we compare this to a similar process which has a fixed value of $p$, but the same value for the expectation. So for $n$ flips,

$$p = \frac{\sum_{i=1}^{n} p_i}{n}$$

**(b)** We also can handle some amount of dependencies between coin flips.

## 18.3 NP-Completeness

This section deals with NP-Completeness, which is an attempt to show that efficient algorithms for certain problems do not exist. In this context, we should define what we mean by an efficient algorithm.

**Definition 16.1** *Efficient Algorithm*: an algorithm which runs in polynomial time.

The motivation behind this definition of efficient algorithms is the observation that most practical problems have either low-degree polynomial time solution or an exponential one. As a benchmark, the highest "useful" polynomial time algorithm is about $O(n^{4.5})$. Further, one can easily come up with exponential time solutions for problems, but finding a polytime solution requires insight into the problem.

### 18.3.1 Examples of Problems

*Min-Cut*

Input: Unweighted, undirected graph $G$.

Problem: Find cut of minimum total weight (i.e., the minimum number of edges).

This is a problem we can solve in polynomial time.

*Max-Cut*

Input: same as *Min-Cut*.

Problem: Find cut of maximum total number of edges.

No polytime solution is known.

*Shortest Path*

Input: weighted, directed graph $G$, nodes $a$ and $b$.

Problem: Find the shortest path from $a$ to $b$.

We know how to solve this in polynomial time.

*Longest Path*

Input: same as *Shortest Path*.

Problem: Find the longest path from $a$ to $b$.

No polytime solution known.

There are a large class of problems which we can solve only in exponential time, so it is supposed that if we can solve one of these in polytime, we can do all of the related problems in polytime too.

### 18.3.2 Decision Problems

In the context of NP-Completeness, it is more convenient to look at *decision problems*, which are problems with a "yes" or "no" answer. Looking at these types of problems does not restrict us in the type of problems we can describe. Algorithms for solving decision problems give algorithms for solving optimization problems, and vice versa.

For example, the *max-cut* problem can be stated as a decision problem:

Input: unweighted, undirected graph $G$, specified integer $k$.

Question: Does $G$ have a cut with size $\geq k$?

The *longest-path* problem:

Input: unweighted, directed graph $G$, vertices a, b, integer $k$.

Question: Does $G$ have a path from a to b of length $\geq k$?

The relation between decision problems and optimization problems can be summarized as follows:

we can find optimal solution $\Longrightarrow$ we can answer decision problem
we can answer decision problem $\Longrightarrow$ we can construct optimal solution

## References

MR95 R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

YJ2000 Yunqing Wang and John Sweeney, *Scribe notes, Lecture 16, Advanced Algorithms Fall 2000*