CS 6214-001: Randomized Algorithms

Lecture 1. Introduction to Randomness

September 2, 2021

Administrativia

- Lecture is 13:50-16:20 in room ECES 114.

Administrativia

- Lecture is 13:50-16:20 in room ECES 114.
- Instructor: Alexandra Kolla, 122 ECEE. Office hours by appointment.

Administrativia

- Lecture is 13:50-16:20 in room ECES 114.
- Instructor: Alexandra Kolla, 122 ECEE. Office hours by appointment.
- Class webpage: https://home.cs.colorado.edu/ alko5368/indexCSCI6214.html

Prerequisites/Grading

- Advanced undergraduate algorithms or equivalent.

Prerequisites/Grading

- Advanced undergraduate algorithms or equivalent.
- Probability theory and statistics or equivalent.

Prerequisites/Grading

- Advanced undergraduate algorithms or equivalent.
- Probability theory and statistics or equivalent.
- For example, if you have never heard what is a probability distribution, expectation, variance, Bernoulli distribution, Binomial distribution, Gaussian distribution, Union Bound (as a small sample) then probably taking the class won't be a good idea for you.

Prerequisites/Grading

- Advanced undergraduate algorithms or equivalent.
- Probability theory and statistics or equivalent.
- For example, if you have never heard what is a probability distribution, expectation, variance, Bernoulli distribution, Binomial distribution, Gaussian distribution, Union Bound (as a small sample) then probably taking the class won't be a good idea for you.
- Talk to the instructor at the end of class today if you think you don't meet the requirements.

Prerequisites/Grading

- Advanced undergraduate algorithms or equivalent.
- Probability theory and statistics or equivalent.
- For example, if you have never heard what is a probability distribution, expectation, variance, Bernoulli distribution, Binomial distribution, Gaussian distribution, Union Bound (as a small sample) then probably taking the class won't be a good idea for you.
- Talk to the instructor at the end of class today if you think you don't meet the requirements.
- By the end of second week of classes (September 8), you must have filled all possible prerequisite gaps. You will be tested on (some of) those skills in the first homework.

Prerequisites/Grading

- Advanced undergraduate algorithms or equivalent.
- Probability theory and statistics or equivalent.
- For example, if you have never heard what is a probability distribution, expectation, variance, Bernoulli distribution, Binomial distribution, Gaussian distribution, Union Bound (as a small sample) then probably taking the class won't be a good idea for you.
- Talk to the instructor at the end of class today if you think you don't meet the requirements.
- By the end of second week of classes (September 8), you must have filled all possible prerequisite gaps. You will be tested on (some of) those skills in the first homework.

Prerequisites/Grading

- Grades are 70% homeworks/in class exercises (weekly) and 30% final exam (take-home).

Prerequisites/Grading

- Grades are 70% homeworks/in class exercises (weekly) and 30% final exam (take-home).
- New homework will be assigned by the end of the week, except the first lecture. You will have one week to complete each homework.

Prerequisites/Grading

- Grades are 70% homeworks/in class exercises (weekly) and 30% final exam (take-home).
- New homework will be assigned by the end of the week, except the first lecture. You will have one week to complete each homework.

Tentative Syllabus

Weeks 1-6, Discrete Probability: First and Second Moment method, coupon collector problem, Probabilistic Method, Chernoff Bound and applications, Martingales and Azuma. Lovasz Local Lemma, Method of Conditional Probabilities

Tentative Syllabus

Weeks 7-9, High-dimensional probability: Bourgain's embedding, Curse of Dimensionality, Dimension Reduction, Matrix Concentration (Golden-Thompson, Bernstein), Random Graph eigenvalues via matrix concentration, Spectral Graph Sparsification via Sampling.

Tentative Syllabus

Weeks 10-12, Random Walk topics: Random Walks: hitting times, cover times etc, Markov Chains and Mixing, Eigenvalues, Expanders and Mixing.

Remaining time, Special Topics: Including but not limited to Lifts and expansion, Algorithms for Stochastic Block Models, Random Graph Spectra.

Class Format

- Class slides will be provided for some lectures (but not all) but they are meant to give only the lecture **skeleton**. Most of the material will be covered on the board, so please take notes.

Class Format

- Class slides will be provided for some lectures (but not all) but they are meant to give only the lecture **skeleton**. Most of the material will be covered on the board, so please take notes.
- I will post the sildes (if applicable) and supplementary reading material for each lecture on the webpage.

Class Format

- Class slides will be provided for some lectures (but not all) but they are meant to give only the lecture **skeleton**. Most of the material will be covered on the board, so please take notes.
- I will post the sildes (if applicable) and supplementary reading material for each lecture on the webpage.
- There is no fixed textbook, but most of the material we will cover can be found in " Probability and Computing: Randomized Algorithms and Probabilistic Analysis" by Mitzenmacher and Upfal.

Class Format

- Class slides will be provided for some lectures (but not all) but they are meant to give only the lecture **skeleton**. Most of the material will be covered on the board, so please take notes.
- I will post the sildes (if applicable) and supplementary reading material for each lecture on the webpage.
- There is no fixed textbook, but most of the material we will cover can be found in " Probability and Computing: Randomized Algorithms and Probabilistic Analysis" by Mitzenmacher and Upfal.
- We will have class assignments in almost every class, where you will work in groups and solve a question relevant to the lecture topic.

Why Randomness?

- Nature is random (quantum physics)!

Why Randomness?

- Nature is random (quantum physics)!
- Flip a coin 1000 times, comes out heads about 500 ± 35 times.

Why Randomness?

- Nature is random (quantum physics)!
- Flip a coin 1000 times, comes out heads about 500 ± 35 times.
- For n coin tosses, about $n/2 \pm \sqrt{n}$. Converges to $1/2$ quickly.

Why Randomness?

- Nature is random (quantum physics)!
- Flip a coin 1000 times, comes out heads about 500 ± 35 times.
- For n coin tosses, about $n/2 \pm \sqrt{n}$. Converges to 1/2 quickly.
- Random Cooking works well: Randomly cook one side of the onion each time. Expect half the time on each side.

Why Randomness?

- Nature is random (quantum physics)!
- Flip a coin 1000 times, comes out heads about 500 ± 35 times.
- For n coin tosses, about $n/2 \pm \sqrt{n}$. Converges to 1/2 quickly.
- Random Cooking works well: Randomly cook one side of the onion each time. Expect half the time on each side.
- Polling for elections. Huge population. Sample size independent of population.

Why Randomness?

- Nature is random (quantum physics)!
- Flip a coin 1000 times, comes out heads about 500 ± 35 times.
- For n coin tosses, about $n/2 \pm \sqrt{n}$. Converges to $1/2$ quickly.
- Random Cooking works well: Randomly cook one side of the onion each time. Expect half the time on each side.
- Polling for elections. Huge population. Sample size independent of population.
- Final exams: large size of material, choose problems independently.

Randomness in Computer Science

- Random algorithms make random choices during their execution.

Randomness in Computer Science

- Random algorithms make random choices during their execution.
- Can be much faster than deterministic counterparts.

Randomness in Computer Science

- Random algorithms make random choices during their execution.
- Can be much faster than deterministic counterparts.
- Simpler and easier to program.

Randomness in Computer Science

- Random algorithms make random choices during their execution.
- Can be much faster than deterministic counterparts.
- Simpler and easier to program.
- Uses in cryptography (eg. primality testing), NP-hard problems, average case complexity...

Randomness in Computer Science

- Random algorithms make random choices during their execution.
- Can be much faster than deterministic counterparts.
- Simpler and easier to program.
- Uses in cryptography (eg. primality testing), NP-hard problems, average case complexity...
- They come with a price (running time, error).

Randomness in Computer Science

- Random algorithms make random choices during their execution.
- Can be much faster than deterministic counterparts.
- Simpler and easier to program.
- Uses in cryptography (eg. primality testing), NP-hard problems, average case complexity...
- They come with a price (running time, error).
- This class: analyze running time, complexity, techniques...

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.
- He wants to check if he has the same file as Alice, but obviously they don't want to send back another 4 GB file.

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.
- He wants to check if he has the same file as Alice, but obviously they don't want to send back another 4 GB file.
- The solution is hashing!

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.
- He wants to check if he has the same file as Alice, but obviously they don't want to send back another 4 GB file.
- The solution is hashing!
- For two n-bit strings A and B, Alice picks random prime number $p \in \{2, 3, \cdots, T\}$ (where T t.b.d later) and sends Bob the hash $H_p(A) = A \mod p$.

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.
- He wants to check if he has the same file as Alice, but obviously they don't want to send back another 4 GB file.
- The solution is hashing!
- For two n-bit strings A and B, Alice picks random prime number $p \in \{2, 3, \cdots, T\}$ (where T t.b.d later) and sends Bob the hash $H_p(A) = A \mod p$.
- Bob accepts that they have the same string if $H_p(B) = H_p(A)$. Length of message is $\log T$ bits.

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.
- He wants to check if he has the same file as Alice, but obviously they don't want to send back another 4 GB file.
- The solution is hashing!
- For two n-bit strings A and B, Alice picks random prime number $p \in \{2, 3, \cdots, T\}$ (where T t.b.d later) and sends Bob the hash $H_p(A) = A \mod p$.
- Bob accepts that they have the same string if $H_p(B) = H_p(A)$. Length of message is $\log T$ bits.
- Otherwise, Bob REJECTS and is really annoyed cause a main character is dead and doesn't even know it!

A little probability of error goes a long way

- Suppose Bob just spent hours downloading the season finale of Game of Thrones (approx 4GB file) from Alice's database.
- He wants to check if he has the same file as Alice, but obviously they don't want to send back another 4 GB file.
- The solution is hashing!
- For two n-bit strings A and B, Alice picks random prime number $p \in \{2, 3, \cdots, T\}$ (where T t.b.d later) and sends Bob the hash $H_p(A) = A \mod p$.
- Bob accepts that they have the same string if $H_p(B) = H_p(A)$. Length of message is $\log T$ bits.
- Otherwise, Bob REJECTS and is really annoyed cause a main character is dead and doesn't even know it!
- One-sided error.

Probabillity Technique: Counting

We hope that if $A \neq B$ then Bob almost always rejects!

Probabillity Technique: Counting

We hope that if $A \neq B$ then Bob almost always rejects!

Claim

If $A \neq B$ then

$$Pr[H_p(A) = H_p(B)] \leq 1.26 \frac{n \ln T}{T \ln n}$$

Probabillity Technique: Counting

We hope that if $A \neq B$ then Bob almost always rejects!

Claim

If $A \neq B$ then

$$Pr[H_p(A) = H_p(B)] \leq 1.26 \frac{n \ln T}{T \ln n}$$

- Example: setting $T = n^2$, we get an $O(\log n)$ bit message with error probability $O(1/n)$.

Probabillity Technique: Counting

We hope that if $A \neq B$ then Bob almost always rejects!

Claim

If $A \neq B$ then

$$Pr[H_p(A) = H_p(B)] \leq 1.26 \frac{n \ln T}{T \ln n}$$

- Example: setting $T = n^2$, we get an $O(\log n)$ bit message with error probability $O(1/n)$.
- How to choose a uniformly random prime?

Probabillity Technique: Counting

We hope that if $A \neq B$ then Bob almost always rejects!

Claim

If $A \neq B$ then

$$Pr[H_p(A) = H_p(B)] \leq 1.26 \frac{n \ln T}{T \ln n}$$

- Example: setting $T = n^2$, we get an $O(\log n)$ bit message with error probability $O(1/n)$.
- How to choose a uniformly random prime?
- Use randomized primality testing algorithm (Miller-Rabin)!

Pattern Matching

- Given two strings $X = x_1 x_2 \cdots x_n$ and $Y = y_1 y_2 \cdots y_m$, with $m < n$ we want to check whether $Y = X(j)$ for some $j \in \{1, \cdots, n-m+1\}$. Here $X(j) = x_j x_{j+1} \cdots x_{j+m-1}$.

Pattern Matching

- Given two strings $X = x_1 x_2 \cdots x_n$ and $Y = y_1 y_2 \cdots y_m$, with $m < n$ we want to check whether $Y = X(j)$ for some $j \in \{1, \cdots, n - m + 1\}$. Here $X(j) = x_j x_{j+1} \cdots x_{j+m-1}$.
- Naive algorithm too slow, there are $O(n + m)$ deterministic algorithms but we will see a simple random one (Karp-Rabin).

Karp-Rabin Algorithm

Choose a prime $p \in \{1, 3, \cdots, T\}$ at random.

Karp-Rabin Algorithm

Choose a prime $p \in \{1, 3, \cdots, T\}$ at random.
Compute $H_p(Y) = Y \mod p$

Karp-Rabin Algorithm

Choose a prime $p \in \{1, 3, \cdots, T\}$ at random.
Compute $H_p(Y) = Y \mod p$
For $j = 1, \cdots, n - m + 1$ compute $H_p(X(j))$ and check if $H_p(X(j)) = H_p(Y)$. If yes, output "matched at position j"

Karp-Rabin Algorithm

Choose a prime $p \in \{1, 3, \cdots, T\}$ at random.
Compute $H_p(Y) = Y \mod p$
For $j = 1, \cdots, n-m+1$ compute $H_p(X(j))$ and check if $H_p(X(j)) = H_p(Y)$. If yes, output "matched at position j"
Output "no match".

Karp-Rabin Algorithm

Choose a prime $p \in \{1, 3, \cdots, T\}$ at random.
Compute $H_p(Y) = Y \mod p$
For $j = 1, \cdots, n - m + 1$ compute $H_p(X(j))$ and check if $H_p(X(j)) = H_p(Y)$. If yes, output "matched at position j"
Output "no match".

- One sided error again, can be converted to zero error if it checks the match.

Karp-Rabin Algorithm

Choose a prime $p \in \{1, 3, \cdots, T\}$ at random.
Compute $H_p(Y) = Y \mod p$
For $j = 1, \cdots, n - m + 1$ compute $H_p(X(j))$ and check if $H_p(X(j)) = H_p(Y)$. If yes, output "matched at position j"
Output "no match".

- One sided error again, can be converted to zero error if it checks the match.
- Still $O(nm)$ running time. Can we do something more clever?

When the Answer is Everywhere

- Consider a task known as program checking.

When the Answer is Everywhere

- Consider a task known as program checking.
- Given $n \times n$ real matrices A, B, C we want to check if $A = BC$.

When the Answer is Everywhere

- Consider a task known as program checking.
- Given $n \times n$ real matrices A, B, C we want to check if $A = BC$.
- Expensive to multiply matrices.

When the Answer is Everywhere

- Consider a task known as program checking.
- Given $n \times n$ real matrices A, B, C we want to check if $A = BC$.
- Expensive to multiply matrices.
- Choose a random vector instead $v \in \{-1, 1\}^n$ and check if $Av = (BC)v$.

When the Answer is Everywhere

- Consider a task known as program checking.
- Given $n \times n$ real matrices A, B, C we want to check if $A = BC$.
- Expensive to multiply matrices.
- Choose a random vector instead $v \in \{-1, 1\}^n$ and check if $Av = (BC)v$.

Claim

If $A \neq BC$ *then* $Pr[Av = BCv] \leq 1/2$

When the Answer is Everywhere

- Consider a task known as program checking.
- Given $n \times n$ real matrices A, B, C we want to check if $A = BC$.
- Expensive to multiply matrices.
- Choose a random vector instead $v \in \{-1, 1\}^n$ and check if $Av = (BC)v$.

Claim

If $A \neq BC$ then $Pr[Av = BCv] \leq 1/2$

- Principle of deferred decisions

When the Answer is Everywhere

- Consider a task known as program checking.
- Given $n \times n$ real matrices A, B, C we want to check if $A = BC$.
- Expensive to multiply matrices.
- Choose a random vector instead $v \in \{-1, 1\}^n$ and check if $Av = (BC)v$.

Claim

If $A \neq BC$ *then* $Pr[Av = BCv] \leq 1/2$

- Principle of deferred decisions
- **Probability Technique: Error Amplification**. Can choose k independent vectors...