Applied Algorithms Lec 8: Probability

Sam McCauley (and Shikha Singh!)

September 28, 2025

Williams College

Probability

Probability Takeaways

What I want you to know by the end of this section of the course:

1. Definition of probability/basic calculations

2. Determine if two events are independent

3. Calculate expectation

4. Linearity of expectation

5. Difference between “concentration bounds” vs expected performance

Definition of Probability

- Defined over a set of possible *outcomes* (often called the *sample space*)

Definition of Probability

- Defined over a set of possible *outcomes* (often called the *sample space*)
- An *event* is a subset of the outcomes

Definition of Probability

- Defined over a set of possible *outcomes* (often called the *sample space*)
- An *event* is a subset of the outcomes
- $$\Pr[\text{Event } E] = \frac{\text{\# outcomes in the event}}{\text{Total \# of outcomes}}$$

Definition of Probability

- Defined over a set of possible *outcomes* (often called the *sample space*)
- An *event* is a subset of the outcomes
- $$\Pr[\text{Event } E] = \frac{\text{\# outcomes in the event}}{\text{Total \# of outcomes}}$$
- Formal definition probability generally applies weights to the events (in which case the definition of probability is the weight of outcomes in the event, divided by total weight of all events). We will usually have equal-weight events.

Probability Calculation Examples

- Let's say I roll a 20-sided die. What is the probability that an even number comes up?

Probability Calculation Examples

- Let's say I roll a 20-sided die. What is the probability that an even number comes up?
- Answer: $10/20 = 1/2$.

Probability Calculation Examples

- Let's say I roll a 20-sided die. What is the probability that an even number comes up?
- Answer: $10/20 = 1/2$.
- Let's say I flip a coin 10 times. What is the probability of getting exactly 5 heads? (Let's do this on the board)

Probability Calculation Examples

- Let's say I roll a 20-sided die. What is the probability that an even number comes up?
- Answer: $10/20 = 1/2$.
- Let's say I flip a coin 10 times. What is the probability of getting exactly 5 heads? (Let's do this on the board)
- $\binom{10}{5}/2^{10}$ Which is $\frac{10!}{5!5!2^{10}} \approx .246$

Probability Calculation Examples

- Let's say I roll a 20-sided die. What is the probability that an even number comes up?
- Answer: $10/20 = 1/2$.
- Let's say I flip a coin 10 times. What is the probability of getting exactly 5 heads? (Let's do this on the board)
- $\binom{10}{5}/2^{10}$ Which is $\frac{10!}{5!5!2^{10}} \approx .246$
- (In a couple lectures we'll see tools to estimate probabilities without lots of large numbers all over the place. For now, use wolfram alpha or whatever)

Probability Calculation: Final Example

- What is the probability of getting a straight flush in poker?
- Simple version: you are dealt 5 cards. You want to know the probability that they all have the same suit, and they are increasing numbers, like 3-4-5-6-7. (Ace is high only)
- 52 cards: 4 suits; 13 cards of each suit
- Let's do the anlaysis out on the board
- Reminder:

$$\Pr[\text{Event } E] = \frac{\text{\# outcomes in the event}}{\text{Total \# of outcomes}}$$

  - Number of possible hands is $\binom{52}{5}$
  - Number of straights is $4 * 9$ (4 suits; straight starts with 2-9)
  - Probability is $36/\binom{52}{5} \approx 0.0000138517$

Conditional Probability

- Sometimes (especially in algorithms) we want to calculate the probability of an event, when we already have some *partial information* about the outcome
  - Who has seen conditional probability before?

Conditional Probability

- Sometimes (especially in algorithms) we want to calculate the probability of an event, when we already have some *partial information* about the outcome
  - Who has seen conditional probability before?
- Specifically: want to calculate the probability of event E_1, already knowing that the outcome is in E_2. Denoted $\Pr[E_1|E_2]$.

Conditional Probability

- Sometimes (especially in algorithms) we want to calculate the probability of an event, when we already have some *partial information* about the outcome
  - Who has seen conditional probability before?
- Specifically: want to calculate the probability of event E_1, already knowing that the outcome is in E_2. Denoted $\Pr[E_1|E_2]$.
- Example: let's say I'm playing cards with a 52-card deck. I have already drawn three cards; all three were clubs. What is the probability that the fourth card is a club?

Conditional Probability

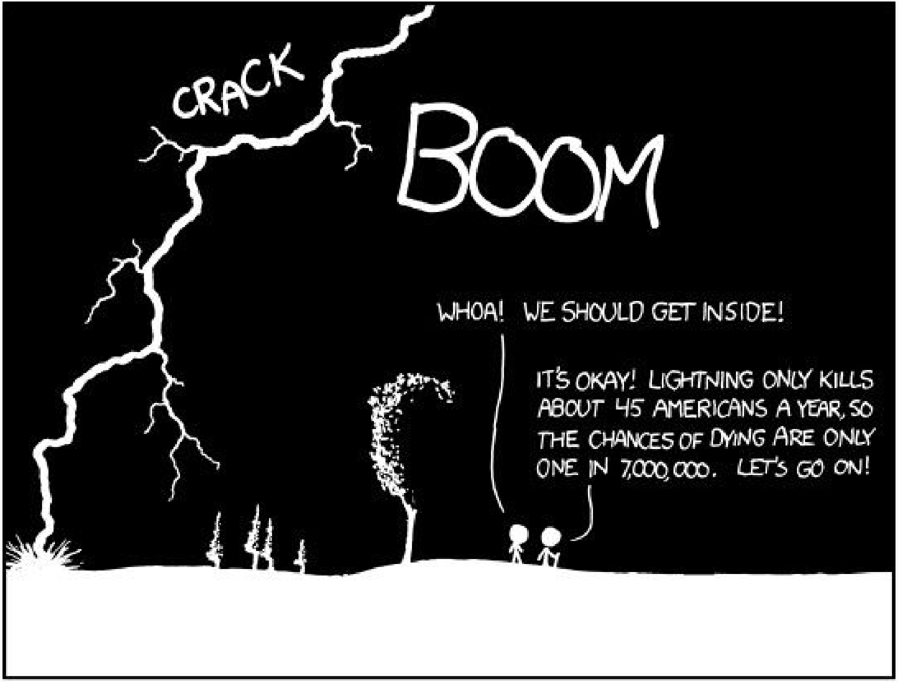
- Sometimes (especially in algorithms) we want to calculate the probability of an event, when we already have some *partial information* about the outcome
  - Who has seen conditional probability before?
- Specifically: want to calculate the probability of event E_1, already knowing that the outcome is in E_2. Denoted $\Pr[E_1|E_2]$.
- Example: let's say I'm playing cards with a 52-card deck. I have already drawn three cards; all three were clubs. What is the probability that the fourth card is a club?
- $\Pr[\text{draw 4 clubs} \mid \text{first three cards were a club}]$

Conditional Probability

- Sometimes (especially in algorithms) we want to calculate the probability of an event, when we already have some *partial information* about the outcome
  - Who has seen conditional probability before?
- Specifically: want to calculate the probability of event E_1, already knowing that the outcome is in E_2. Denoted $\Pr[E_1|E_2]$.
- Example: let's say I'm playing cards with a 52-card deck. I have already drawn three cards; all three were clubs. What is the probability that the fourth card is a club?
- $\Pr[\text{draw 4 clubs} \mid \text{first three cards were a club}]$
- How many outcomes are there for the fourth card? How many of them are a club?

Conditional Probability

- Sometimes (especially in algorithms) we want to calculate the probability of an event, when we already have some *partial information* about the outcome
    - Who has seen conditional probability before?
- Specifically: want to calculate the probability of event E_1, already knowing that the outcome is in E_2. Denoted $\Pr[E_1|E_2]$.
- Example: let's say I'm playing cards with a 52-card deck. I have already drawn three cards; all three were clubs. What is the probability that the fourth card is a club?
- $\Pr[\text{draw 4 clubs} \mid \text{first three cards were a club}]$
- How many outcomes are there for the fourth card? How many of them are a club?
- 49 outcomes. 10 of them are clubs. Probability: $10/49$.



THE ANNUAL DEATH RATE AMONG PEOPLE WHO KNOW THAT *STATISTIC* IS ONE IN SIX.

Conditional Probability Second Example

- Let’s say that I have one child who is a boy (in a very simplified model where each child is a “boy” or “girl” according to an independent coin flip). What is the probability that when I have a second child, it winds up being a boy?

Conditional Probability Second Example

- Let’s say that I have one child who is a boy (in a very simplified model where each child is a “boy” or “girl” according to an independent coin flip). What is the probability that when I have a second child, it winds up being a boy?

- What is your answer intuitively?

Conditional Probability Second Example

- Let’s say that I have one child who is a boy (in a very simplified model where each child is a “boy” or “girl” according to an independent coin flip). What is the probability that when I have a second child, it winds up being a boy?
- What is your answer intuitively?
- Outcomes for my children: BB   BG   GB   GG

Conditional Probability Second Example

- Let’s say that I have one child who is a boy (in a very simplified model where each child is a “boy” or “girl” according to an independent coin flip). What is the probability that when I have a second child, it winds up being a boy?
- What is your answer intuitively?
- Outcomes for my children: BB BG GB GG
- Outcomes consistent with "the second one is a boy": BB GB

Conditional Probability Second Example

- Let's say that I have one child who is a boy (in a very simplified model where each child is a "boy" or "girl" according to an independent coin flip). What is the probability that when I have a second child, it winds up being a boy?
- What is your answer intuitively?
- Outcomes for my children: BB BG GB GG
- Outcomes consistent with "the second one is a boy": BB GB
- Probability $1/2$

Conditional Probability Third Example

Conditional probability can be a bit unintuitive at times! Break it down to be sure you're getting the right answer. (This is why we go over the formal definitions)

- Let's say that I have two children. One of them is a boy. What is the probability that both of them are boys?

Conditional Probability Third Example

Conditional probability can be a bit unintuitive at times! Break it down to be sure you’re getting the right answer. (This is why we go over the formal definitions)

- Let’s say that I have two children. One of them is a boy. What is the probability that both of them are boys?
- Outcomes for my children: BB BG GB GG

Conditional Probability Third Example

Conditional probability can be a bit unintuitive at times! Break it down to be sure you’re getting the right answer. (This is why we go over the formal definitions)

- Let’s say that I have two children. One of them is a boy. What is the probability that both of them are boys?
- Outcomes for my children: BB BG GB GG
- Outcomes consistent with "one of them is a boy": BB BG GB

Conditional Probability Third Example

Conditional probability can be a bit unintuitive at times! Break it down to be sure you’re getting the right answer. (This is why we go over the formal definitions)

- Let’s say that I have two children. One of them is a boy. What is the probability that both of them are boys?
- Outcomes for my children: BB BG GB GG
- Outcomes consistent with "one of them is a boy": BB BG GB
- Probability that both of them are boys: $1/3$

Conditional Probability Third Example

Conditional probability can be a bit unintuitive at times! Break it down to be sure you're getting the right answer. (This is why we go over the formal definitions)

- Let's say that I have two children. One of them is a boy. What is the probability that both of them are boys?
- Outcomes for my children: BB BG GB GG
- Outcomes consistent with "one of them is a boy": BB BG GB
- Probability that both of them are boys: $1/3$
- Rephrasing the question: Let's say I have two children. They are not both girls. What is the probability that they are both boys?

Conditional Probability Third Example

Conditional probability can be a bit unintuitive at times! Break it down to be sure you're getting the right answer. (This is why we go over the formal definitions)

- Let's say that I have two children. One of them is a boy. What is the probability that both of them are boys?
- Outcomes for my children: BB   BG   GB   GG
- Outcomes consistent with "one of them is a boy": BB   BG   GB
- Probability that both of them are boys: $1/3$
- Rephrasing the question: Let's say I have two children. They are not both girls. What is the probability that they are both boys?
- Yes, this kind of a wording trick. But similar issues can arise when analyzing algorithms.

Independence

- Idea: two events are independent if one does not have any impact on the other

Independence

- Idea: two events are independent if one does not have any impact on the other
- Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.

Independence

- Idea: two events are independent if one does not have any impact on the other
- Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.
- Formal definition: E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.

Independence

- Idea: two events are independent if one does not have any impact on the other
- Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.
- Formal definition: E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.
    - Knowing the outcome of one does not affect the probability of the other!

Independence

- Idea: two events are independent if one does not have any impact on the other
- Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.
- Formal definition: E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.
    - Knowing the outcome of one does not affect the probability of the other!
- I'll generally ask you if things are independent *intuitively* rather than asking for a proof. But, let's look at a couple classic examples of independence and how this definition works with them.

Proving Independence

Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.

Definition

E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.

All possible outcomes: HH HT TH TT

Proving Independence

Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.

Definition

E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.

All possible outcomes: HH HT TH TT

$$\Pr[E_2] = 1/2$$

Proving Independence

Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.

Definition

E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.

All possible outcomes: $\boxed{HH}$ $\boxed{HT}$ TH TT

$$\Pr[E_2 \mid E_1]$$

Proving Independence

Example: let's say I flip a (fair) coin twice. Let E_1 be the event that the first flip is heads, and E_2 be the event that the second flip is heads. E_1 and E_2 are independent.

Definition

E_1 and E_2 are independent if $\Pr[E_1 \mid E_2] = \Pr[E_1]$ and $\Pr[E_2 \mid E_1] = \Pr[E_2]$.

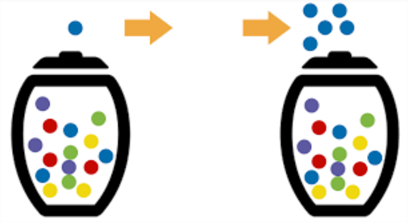
All possible outcomes: $\boxed{HH}$ $\boxed{HT}$ TH TT

$$\Pr[E_2 \mid E_1] = 1/2$$

Again: we'll normally be looking at this intuitively.

Independence: Examples

- Let’s say I have a bag of balls, half of which are black, and half of which are white. I take a ball out of the bag and look at what color it is. Then I take another ball out of the bag and look at what color it is.
  - Event 1: The first ball is white.
  - Event 2: The second ball is black
  - Are these events independent?

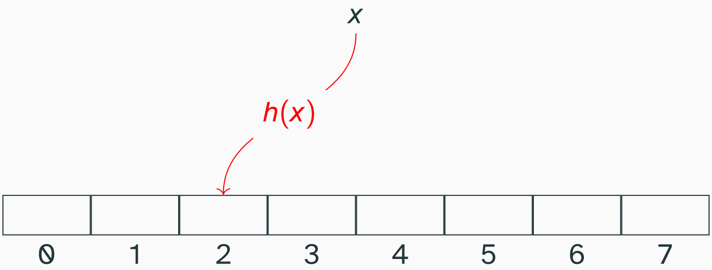


Independence: Examples

- Let's say I have a bag of balls, half of which are black, and half of which are white. I take a ball out of the bag and look at what color it is. Then I take another ball out of the bag and look at what color it is.
  - Event 1: The first ball is white.
  - Event 2: The second ball is black
  - Are these events independent?

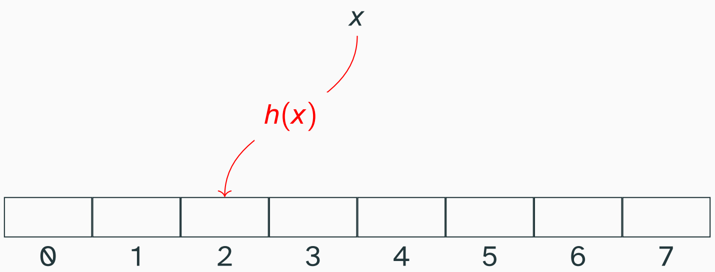
- No! If the first ball is white, there will be more black balls then white balls remaining in the bag for the next draw.

Hash Functions



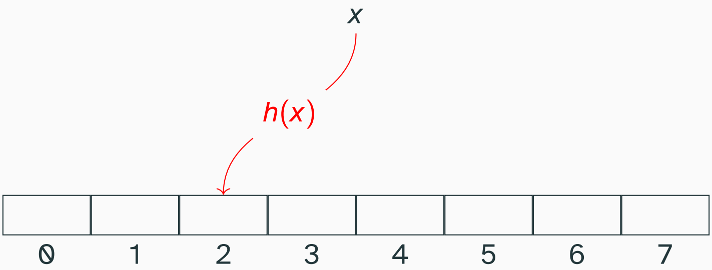
- Let’s do an example using hash functions

Hash Functions



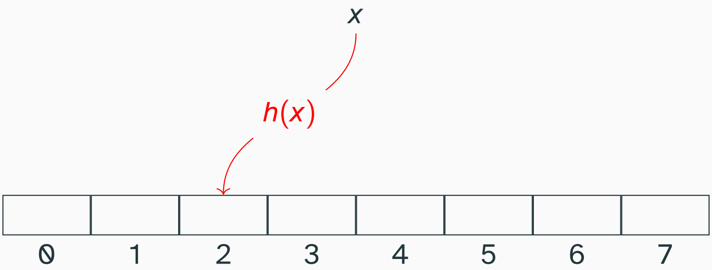
- Let’s do an example using hash functions
- Recall: A hash function is used to index into a hash table

Hash Functions



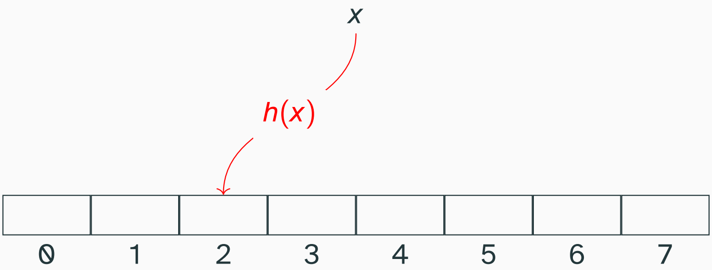
- Let's do an example using hash functions
- Recall: A hash function is used to index into a hash table
- Idea: if the hash function is random, our items will (usually) be distributed evenly throughout the hash table

Hash Functions



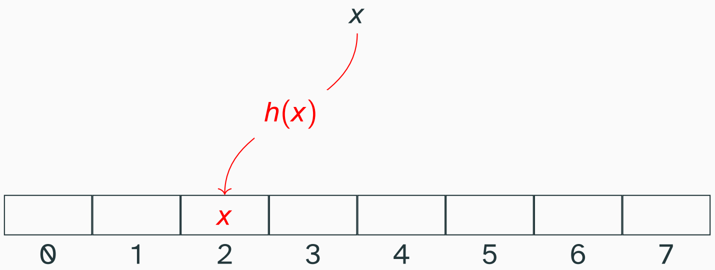
- Let's do an example using hash functions
- Recall: A hash function is used to index into a hash table
- Idea: if the hash function is random, our items will (usually) be distributed evenly throughout the hash table

Hash Functions



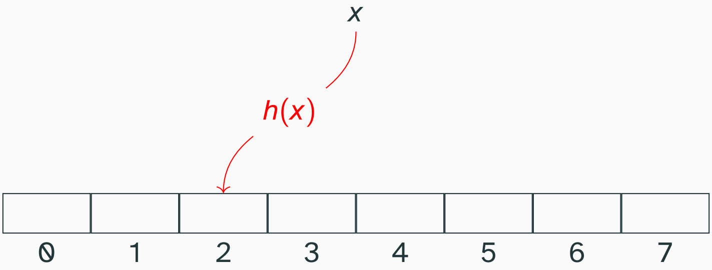
- Let's do an example using hash functions
- Recall: A hash function is used to index into a hash table
- Idea: if the hash function is random, our items will (usually) be distributed evenly throughout the hash table

Hash Functions



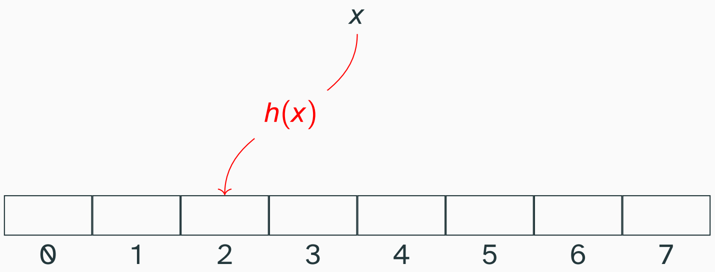
- Let’s do an example using hash functions
- Recall: A hash function is used to index into a hash table
- Idea: if the hash function is random, our items will (usually) be distributed evenly throughout the hash table

Hash Functions in this Class



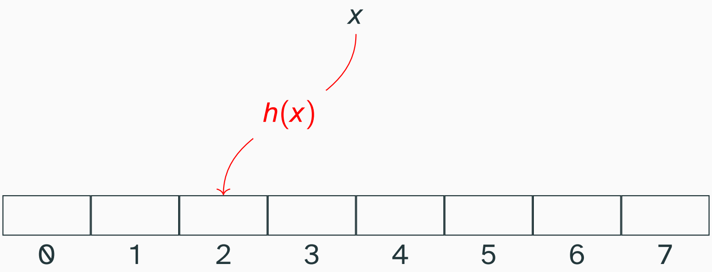
- In this class, for analysis, we will assume that all hash functions are *uniform random*

Hash Functions in this Class



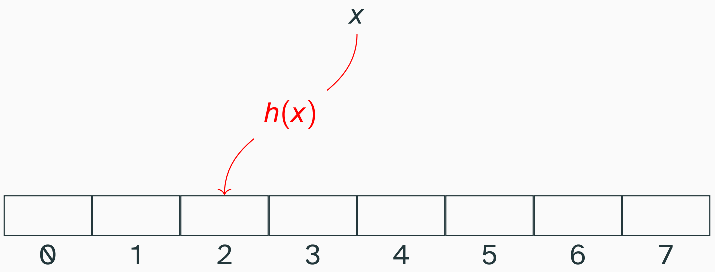
- In this class, for analysis, we will assume that all hash functions are *uniform random*
- The hash function maps any item x to a random slot from 0 to $m-1$; each with probability $1/m$

Hash Functions in this Class



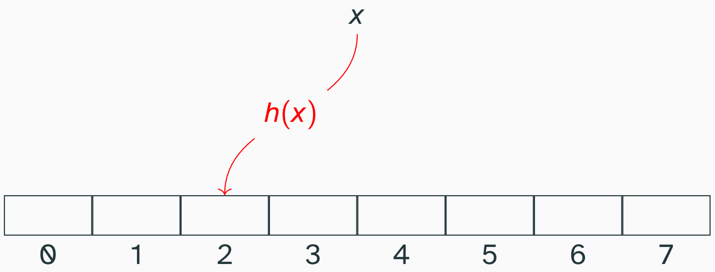
- In this class, for analysis, we will assume that all hash functions are *uniform random*
- The hash function maps any item x to a random slot from 0 to $m-1$; each with probability $1/m$
- $h(x)$ is independent of $h(y)$ for any $y \neq x$
  - In fact, we'll assume that $h(x)$ is independent of the hash of any *set* of other items—in short, $h(x)$ is like rolling an m-sided die

Hash Functions in this Class



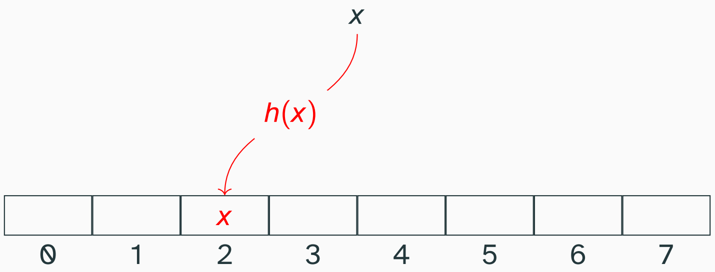
- In this class, for analysis, we will assume that all hash functions are *uniform random*
- The hash function maps any item x to a random slot from 0 to $m-1$; each with probability $1/m$
- $h(x)$ is independent of $h(y)$ for any $y \neq x$
  - In fact, we'll assume that $h(x)$ is independent of the hash of any *set* of other items—in short, $h(x)$ is like rolling an m-sided die

Hash Functions in this Class



- In this class, for analysis, we will assume that all hash functions are *uniform random*
- The hash function maps any item x to a random slot from 0 to $m-1$; each with probability $1/m$
- $h(x)$ is independent of $h(y)$ for any $y \neq x$
  - In fact, we'll assume that $h(x)$ is independent of the hash of any *set* of other items—in short, $h(x)$ is like rolling an m-sided die

Hash Functions in this Class



- In this class, for analysis, we will assume that all hash functions are *uniform random*
- The hash function maps any item x to a random slot from 0 to $m-1$; each with probability $1/m$
- $h(x)$ is independent of $h(y)$ for any $y \neq x$
    - In fact, we'll assume that $h(x)$ is independent of the hash of any *set* of other items—in short, $h(x)$ is like rolling an m-sided die

Probability example for hash tables

- Let's say I use a (uniform random) hash function h to store a set of elements in a hash table

Probability example for hash tables

- Let's say I use a (uniform random) hash function h to store a set of elements in a hash table

- Are the following two events independent?
    - Some element x hashes to bucket 1
    - Some other element y hashes to bucket 1

Probability example for hash tables

- Let's say I use a (uniform random) hash function h to store a set of elements in a hash table

- Are the following two events independent?
    - Some element x hashes to bucket 1
    - Some other element y hashes to bucket 1

- Yes! We assumed that the hashes of distinct elements are independent. Each event happens with probability $1/m$, independently

Independence: Examples

- Let's say I use a *random* hash function h to store a set of n elements in a hash table

Independence: Examples

- Let's say I use a *random* hash function h to store a set of n elements in a hash table
- Are the following two events independent?
  - The first bucket of the hash table contains no elements
  - The second bucket of the hash table contains no elements

Independence: Examples

- Let's say I use a *random* hash function h to store a set of n elements in a hash table
- Are the following two events independent?
  - The first bucket of the hash table contains no elements
  - The second bucket of the hash table contains no elements
- No! Since the first bucket contains no elements, the remaining elements are *slightly* more likely to hash to the second bucket

Independence: Examples

- Let's say I use a *random* hash function h to store a set of n elements in a hash table
- Are the following two events independent?
  - The first bucket of the hash table contains no elements
  - The second bucket of the hash table contains no elements
- No! Since the first bucket contains no elements, the remaining elements are *slightly* more likely to hash to the second bucket
- For intuition: let's say there are only two slots in the hash table. Then these are not independent at all—in fact, if the first occurs, the second *cannot* occur

Why independence is useful

- If A and B are independent, then $\Pr[A \text{ and } B] = \Pr(A) \cdot \Pr(B)$.

Why independence is useful

- If A and B are independent, then $\Pr[A \text{ and } B] = \Pr(A) \cdot \Pr(B)$.
- What is the probability of flipping 10 heads in a row when flipping a fair coin?

Why independence is useful

- If A and B are independent, then $\Pr[A \text{ and } B] = \Pr(A) \cdot \Pr(B)$.
- What is the probability of flipping 10 heads in a row when flipping a fair coin?
  - All 10 are independent, so $1/2^{10}$.

Why independence is useful

- Let's say you're in a class of n students. Every day the professor asks a student to explain the previous night's reading (the student is chosen by rolling an n-sided die). What is the probability that you *won't* be chosen after all k lectures in the course?

Why independence is useful

- Let's say you're in a class of n students. Every day the professor asks a student to explain the previous night's reading (the student is chosen by rolling an n-sided die). What is the probability that you *won't* be chosen after all k lectures in the course?

- Probability (not being chosen on one day) is $(1 - 1/n)$

Why independence is useful

- Let's say you're in a class of n students. Every day the professor asks a student to explain the previous night's reading (the student is chosen by rolling an n-sided die). What is the probability that you *won't* be chosen after all k lectures in the course?

- Probability (not being chosen on one day) is $(1 - 1/n)$

- Probability (not being chosen after k days) is $(1 - 1/n)^k$

Why independence is useful

- Let's say you're in a class of n students. Every day the professor asks a student to explain the previous night's reading (the student is chosen by rolling an n-sided die). What is the probability that you *won't* be chosen after all k lectures in the course?

- Probability (not being chosen on one day) is $(1 - 1/n)$

- Probability (not being chosen after k days) is $(1 - 1/n)^k$

- OK, that's a formula for the probability, but it doesn't tell me much. Is this large or small? How does it change?

Useful formulas for probability ($e = 2.71\ldots$)

Two useful approximations for simplifying exponents (presented as inequalities, but really quite tight even for moderate n):

Lemma

$$(1 + 1/n)^n \leq e \qquad\qquad (1 - 1/n)^n \leq 1/e$$

Example: $(1.1)^{10} = 2.593\ldots$

Useful formulas for probability ($e = 2.71\ldots$)

Two useful approximations for simplifying exponents (presented as inequalities, but really quite tight even for moderate n):

Lemma

$$(1 + 1/n)^n \leq e \qquad\qquad (1 - 1/n)^n \leq 1/e$$

Example: $(1.1)^{10} = 2.593\ldots$

With probability we often use choose (a.k.a. binomial) notation, but it's similarly inscrutible. These inequalities can help approximate it:

Lemma

$$\left(\frac{x}{y}\right)^y \leq \binom{x}{y} \leq \left(\frac{ex}{y}\right)^y$$

Example: $\binom{n}{10} = \Theta(n^{10})$

Why independence is useful (simplifying from before)

- Let's say you're in a class of n students. Every day the professor asks a student to explain the previous night's reading (the student is chosen by rolling an n-sided die). What is the probability that you *won't* be chosen after all k lectures in the course?
- Probability (not being chosen on one day) is $(1 - 1/n)$
- Probability (not being chosen after k days) is $(1 - 1/n)^k$
- Can simplify (assuming n is large):

$$(1 - 1/n)^k = ((1 - 1/n)^n)^{k/n} \approx e^{k/n}.$$

Where are we

- Why we we doing all of this probability stuff again?

Where are we

- Why we we doing all of this probability stuff again?
- We want to talk about hash table performance!

Where are we

- Why we we doing all of this probability stuff again?
- We want to talk about hash table performance!
- Goal: we said last time that a hash table query *could* be $\Theta(n)$. But usually it won't be

Where are we

- Why we we doing all of this probability stuff again?
- We want to talk about hash table performance!
- Goal: we said last time that a hash table query *could* be $\Theta(n)$. But usually it won't be
- How can we talk about "usually" in a formal sense?

Where are we

- Why we we doing all of this probability stuff again?
- We want to talk about hash table performance!
- Goal: we said last time that a hash table query *could* be $\Theta(n)$. But usually it won't be
- How can we talk about "usually" in a formal sense?
- (So: one more probability definition, then we can talk about algorithmic performance)

Random Variable and Expectation

Random Variable

- A variable whose values depend on the outcome of a random process

Random Variable

- A variable whose values depend on the outcome of a random process
- We’re using mostly for the sake of notation

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?
    - S is at least 0 and at most 4

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?
  - S is at least 0 and at most 4
  - What is the probability that S is 0?

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?
    - S is at least 0 and at most 4
    - What is the probability that S is 0?
    - $13/52 \cdot 13/51 \cdot 13/50 \cdot 13/49 \approx .0043$

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?
  - S is at least 0 and at most 4
  - What is the probability that S is 0?
  - $13/52 \cdot 13/51 \cdot 13/50 \cdot 13/49 \approx .0043$
  - What is the probability that S is 4?

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?
  - S is at least 0 and at most 4
  - What is the probability that S is 0?
  - $13/52 \cdot 13/51 \cdot 13/50 \cdot 13/49 \approx .0043$
  - What is the probability that S is 4?
  - $13/52 \cdot 12/51 \cdot 11/50 \cdot 10/49 \approx .00264$

Random Variable

- A variable whose values depend on the outcome of a random process
- We're using mostly for the sake of notation
- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.
- What can we say about S?
  - S is at least 0 and at most 4
  - What is the probability that S is 0?
  - $13/52 \cdot 13/51 \cdot 13/50 \cdot 13/49 \approx .0043$
  - What is the probability that S is 4?
  - $13/52 \cdot 12/51 \cdot 11/50 \cdot 10/49 \approx .00264$
- S is a *random variable.* Like normal variables, it is used to represent a value. Here, the value is the outcome of a random process.

Random Variable

- Let's say I draw four cards from a deck of cards. Let S be a random variable indicating the number of clubs I draw.

- One more comment about S. Since each card is a club with probability (about) $1/4$, and we draw 4 cards, it seems like S should generally be around 1. Can we formalize this intuition?

Expectation

- When we make random decisions, we often care about the *average* outcome

Expectation

- When we make random decisions, we often care about the *average* outcome
- Example: let's say I flip a fair coin until I get a heads. How long will it take me on average?

Expectation

- When we make random decisions, we often care about the *average* outcome
- Example: let's say I flip a fair coin until I get a heads. How long will it take me on average?
- 2 flips

Expectation

- When we make random decisions, we often care about the *average* outcome
- Example: let's say I flip a fair coin until I get a heads. How long will it take me on average?
- 2 flips
- Another example: quicksort is $O(n^2)$ in the worst case, but if you pick pivots randomly, it is $O(n \log n)$ in expectation

Definition of Expectation

- Let's say a random variable X takes values $\{1, \ldots k\}$

Definition of Expectation

- Let's say a random variable X takes values $\{1, \ldots k\}$
- Then the expectation of X is

$$\mathrm{E}[X] = \sum_{i=1}^{k} i \cdot \Pr[X = i].$$

Definition of Expectation

- Let's say a random variable X takes values $\{1, \ldots k\}$
- Then the expectation of X is

$$\mathrm{E}[X] = \sum_{i=1}^{k} i \cdot \Pr[X = i].$$

- It is a weighted average of the outcomes: the outcome is i, and it is weighted by the probability that i occurs.

Expectation example

Let's say I roll a 20-sided die, and I give you money equal to the number that shows up on top. I charge $10 to play this game. Should you play it?

Split into pairs and try to calculate the expected outcome.

Expectation example

Let's say I roll a 20-sided die, and I give you money equal to the number that shows up on top. I charge $10 to play this game. Should you play it?

Let's look at what you win on average

- Random variable X to represent how much you win

Expectation example

Let’s say I roll a 20-sided die, and I give you money equal to the number that shows up on top. I charge \$10 to play this game. Should you play it?

Let’s look at what you win on average

- Random variable X to represent how much you win
- $\mathrm{E}[X] = \sum_{i=1}^{20} i/20$

Expectation example

Let's say I roll a 20-sided die, and I give you money equal to the number that shows up on top. I charge \$10 to play this game. Should you play it?

Let's look at what you win on average

- Random variable X to represent how much you win
- $\mathrm{E}[X] = \sum_{i=1}^{20} i/20$
- $\mathrm{E}[X] = \frac{20 \cdot 21}{2 \cdot 20} = 10.5$

Expectation example

Let's say I roll a 20-sided die, and I give you money equal to the number that shows up on top. I charge $10 to play this game. Should you play it?

Let's look at what you win on average

- Random variable X to represent how much you win
- $\mathrm{E}[X] = \sum_{i=1}^{20} i/20$
- $\mathrm{E}[X] = \frac{20 \cdot 21}{2 \cdot 20} = 10.5$
- So you'll win $.50 on average; you should probably play the game

Linearity of Expectation

- Consider a random variable that can be represented as the sum of other random variables (we'll see an example in a moment). Linearity of expectation means that if I sum the expectations of the parts, I get the expectation of the whole

Theorem (Linearity of Expectation)

For any random variable $X = X_1 + X_2 + \ldots + X_n$,

$$E[X] = E[X_1] + E[X_2] + \ldots + E[X_n]$$

True even if the X_i are not independent!!!

Using Linearity of Expectation

- Let's say I flip a coin 100 times. How many heads will I see on average?

Using Linearity of Expectation

- Let’s say I flip a coin 100 times. How many heads will I see on average?
- Let’s figure this out on the board using linearity of expectation

Using Linearity of Expectation

- Let's say I flip a coin 100 times. How many heads will I see on average?
- Let's figure this out on the board using linearity of expectation
- X = number of heads I see in 100 flips.
- $$X_i = \begin{cases} 1 & \text{if the } i\text{th flip is heads} \\ 0 & \text{otherwise} \end{cases}$$
- So then $X = X_1 + X_2 + \ldots X_{100}$
- We can see that $\mathrm{E}[X_i] = 1/2$.
- $\mathrm{E}[X] = 50$ by linearity of expectation. This would have been very difficult to calculate directly using the definition!

Linearity of Expectation

- If we can break a variable into the sum of different parts—even if the parts are *not independent*—we can calculate the expectation

Linearity of Expectation

- If we can break a variable into the sum of different parts—even if the parts are *not independent*—we can calculate the expectation

- Makes analysis of expected values massively easier

Linearity of Expectation

- If we can break a variable into the sum of different parts—even if the parts are *not independent*—we can calculate the expectation

- Makes analysis of expected values massively easier

- We will use a lot over the next few days!

Linearity of Expectation

- If we can break a variable into the sum of different parts—even if the parts are *not independent*—we can calculate the expectation

- Makes analysis of expected values massively easier

- We will use a lot over the next few days!

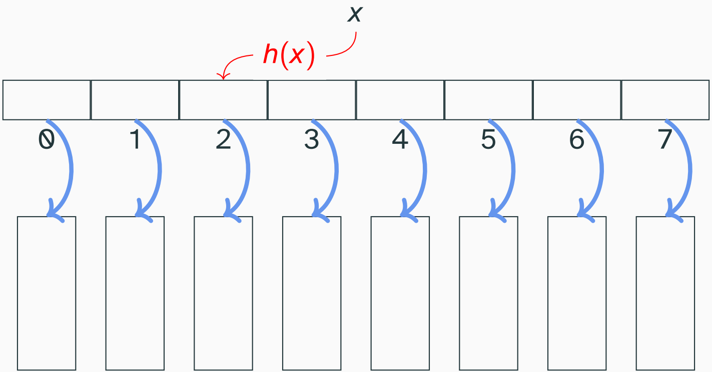
- Downside: expectation only discusses average (we'll come back to this)

Classic Hash Tables

Collisions

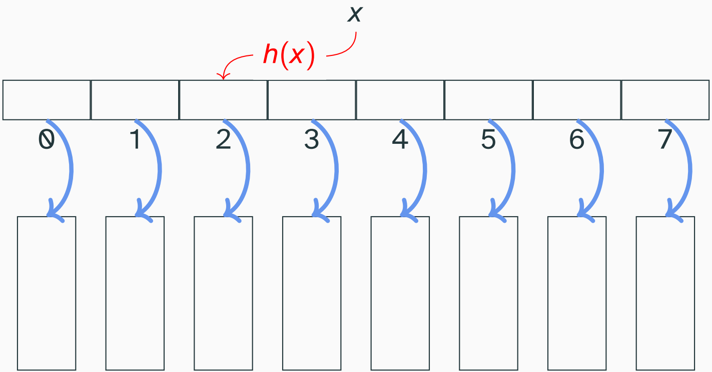
- (We discussed last class)
- Several items might hash to the same location. How can we resolve this?
  - Chaining
  - Linear Probing

Chaining



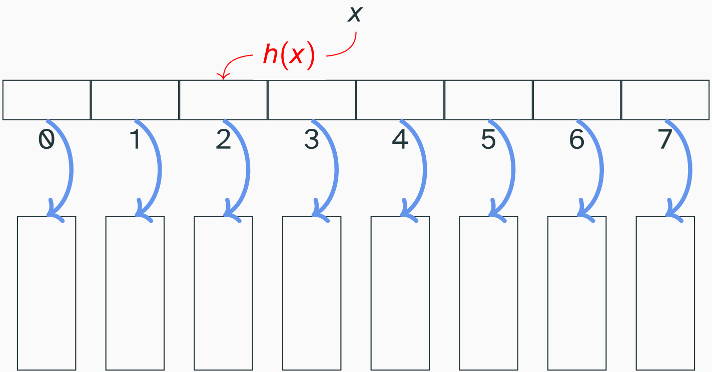
- Each entry in our array A is the head of a new data structure

Chaining



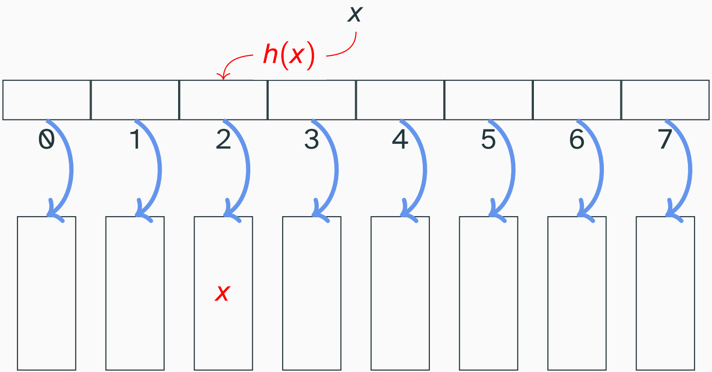
- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)

Chaining



- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- A has size cn; can insert at most n items (double if we ever go over the size)

Chaining



- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list; Query: find item in linked list

Chaining

- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- Assume A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list
- Query: find item in linked list
- Advantages?

Chaining

- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- Assume A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list
- Query: find item in linked list
- Advantages?
    - Space-efficient (just need pointers for linked list)

Chaining

- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- Assume A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list
- Query: find item in linked list
- Advantages?
  - Space-efficient (just need pointers for linked list)
  - Simple.

Chaining

- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- Assume A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list
- Query: find item in linked list
- Advantages?
  - Space-efficient (just need pointers for linked list)
  - Simple.
  - Good *worst-case* insert time of $O(1)$; we'll analyze query time in a moment

Chaining

- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- Assume A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list
- Query: find item in linked list
- Advantages?
    - Space-efficient (just need pointers for linked list)
    - Simple.
    - Good *worst-case* insert time of $O(1)$; we'll analyze query time in a moment
- Disadvantages?

Chaining

- Each entry in our array A is the head of a new data structure
- Often implemented as a singly-linked list (let's draw this on the board)
- Assume A has size cn; can insert at most n items (double if we ever go over the size)
- Insert: add item to linked list
- Query: find item in linked list
- Advantages?
  - Space-efficient (just need pointers for linked list)
  - Simple.
  - Good *worst-case* insert time of $O(1)$; we'll analyze query time in a moment
- Disadvantages?
  - Cache inefficient

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?
- $X^j =$ number of non-query items in chain j

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?
- $X^j =$ number of non-query items in chain j
- $X_i^j = 1$ if the ith item hashes to slot j

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?
- X^j = number of non-query items in chain j
- $X_i^j = 1$ if the ith item hashes to slot j
- $\mathrm{E}[X_i^j] = 1/cn$

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?
- $X^j =$ number of non-query items in chain j
- $X_i^j = 1$ if the ith item hashes to slot j
- $\mathrm{E}[X_i^j] = 1/cn$
- $\mathrm{E}[X^j] = \sum_{i=1}^{n} \mathrm{E}[X_i^j] = 1/c$

Linearity of Expectation Example 2: Chaining

- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?
- X^j = number of non-query items in chain j
- $X_i^j = 1$ if the ith item hashes to slot j
- $\mathrm{E}[X_i^j] = 1/cn$
- $\mathrm{E}[X^j] = \sum_{i=1}^{n} \mathrm{E}[X_i^j] = 1/c$
- So the expected length of the chain is $O(1 + 1/c) = O(1)$

Linearity of Expectation Example 2: Chaining

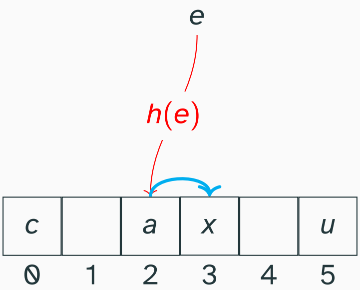
- Chaining running time: in $O(1)$ time we compute the hash; then we need to compare the query to each item in the chain
- What's the expected number of *non-query elements* in a given chain?
- In pairs: can you use expectation to calculate this number?
- $X^j =$ number of non-query items in chain j
- $X_i^j = 1$ if the ith item hashes to slot j
- $\mathrm{E}[X_i^j] = 1/cn$
- $\mathrm{E}[X^j] = \sum_{i=1}^{n} \mathrm{E}[X_i^j] = 1/c$
- So the expected length of the chain is $O(1 + 1/c) = O(1)$
- Chaining has $O(1)$ expected query time!

Linear Probing

- Set $c > 1$ (often have $1.5n$ or $2n$ slots in practice)

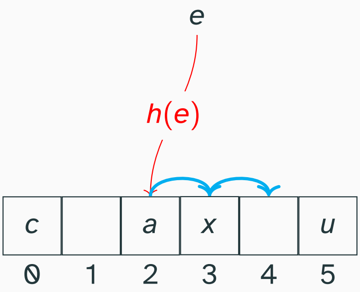
Linear Probing

- Set $c > 1$ (often have $1.5n$ or $2n$ slots in practice)
- Insert: attempt to insert x into $h(x)$. If slot is full, move to the next slot of the table. Keep going until finding an empty slot.



Linear Probing

- Set $c > 1$ (often have $1.5n$ or $2n$ slots in practice)
- Insert: attempt to insert x into $h(x)$. If slot is full, move to the next slot of the table. Keep going until finding an empty slot.



Linear Probing

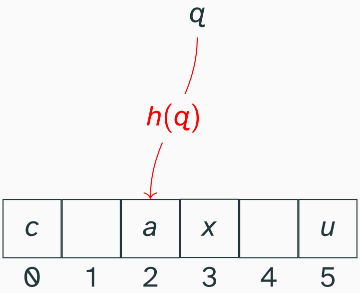
- Set $c > 1$ (often have $1.5n$ or $2n$ slots in practice)
- Insert: attempt to insert x into $h(x)$. If slot is full, move to the next slot of the table. Keep going until finding an empty slot.

Linear Probing

- Set $c > 1$ (often have $1.5n$ or $2n$ slots in practice)
- Insert: attempt to insert x into $h(x)$. If slot is full, move to the next slot of the table. Keep going until finding an empty slot.
- Any ideas how we can query for an item q?

Linear Probing

- Set $c > 1$ (often have $1.5n$ or $2n$ slots in practice)
- Insert: attempt to insert x into $h(x)$. If slot is full, move to the next slot of the table. Keep going until finding an empty slot.
- Any ideas how we can query for an item q?
- Answer: look in slot $h(q)$. If it has q we are done; if it is empty we are done. Otherwise, continue to slot $h(q) + 1$



Linear Probing

- Advantages?

Linear Probing

- Advantages?
    - Somewhat space-efficient

Linear Probing

- Advantages?
  - Somewhat space-efficient
  - Insert is $O(1 + \frac{1}{1-1/c}) = O(1)$ and query is $O(1 + \frac{1}{(1-1/c)}) = O(1)$ in expectation (Classic result of Donald Knuth; nontrivial)

Linear Probing

- Advantages?
    - Somewhat space-efficient
    - Insert is $O(1 + \frac{1}{1 - 1/c}) = O(1)$ and query is $O(1 + \frac{1}{(1-1/c)}) = O(1)$ in expectation (Classic result of Donald Knuth; nontrivial)
    - *Cache-efficient!*

Linear Probing

- Advantages?
  - Somewhat space-efficient
  - Insert is $O(1 + \frac{1}{1-1/c}) = O(1)$ and query is $O(1 + \frac{1}{(1-1/c)}) = O(1)$ in expectation (Classic result of Donald Knuth; nontrivial)
  - *Cache-efficient!*
- Disadvantages?

Linear Probing

- Advantages?
  - Somewhat space-efficient
  - Insert is $O(1 + \frac{1}{1-1/c}) = O(1)$ and query is $O(1 + \frac{1}{(1-1/c)}) = O(1)$ in expectation (Classic result of Donald Knuth; nontrivial)
  - *Cache-efficient!*
- Disadvantages?
  - Performance is terrible if c is close to 1—that is to say, if A is nearly full

Linear Probing

- Advantages?
  - Somewhat space-efficient
  - Insert is $O(1+\frac{1}{1-1/c}) = O(1)$ and query is $O(1+\frac{1}{(1-1/c)}) = O(1)$ in expectation (Classic result of Donald Knuth; nontrivial)
  - *Cache-efficient!*
- Disadvantages?
  - Performance is terrible if c is close to 1—that is to say, if A is nearly full
- This is probably the most common choice in practice: it's simple and cache-efficient

Cuckoo Hashing

Our Results so Far

- Linear probing has $O(1)$ expected insert and query

- Chaining has $O(1)$ insert (worst-case), $O(1)$ expected query

Our Results so Far

- Linear probing has $O(1)$ expected insert and query
- Chaining has $O(1)$ insert (worst-case), $O(1)$ expected query
- But most hash tables spend *more* time inserting then query

Our Results so Far

- Linear probing has $O(1)$ expected insert and query
- Chaining has $O(1)$ insert (worst-case), $O(1)$ expected query
- But most hash tables spend *more* time inserting then query
- It would be really nice to get the reverse: $O(1)$ expected insert time, $O(1)$ worst-case query time

Cuckoo Hashing [Pagh, Rodler 2005]

- A third method of resolving collisions

Cuckoo Hashing [Pagh, Rodler 2005]

- A third method of resolving collisions
- Queries are $O(1)$ *worst case*

Cuckoo Hashing [Pagh, Rodler 2005]

- A third method of resolving collisions
- Queries are $O(1)$ *worst case*
- Insert will still be $O(1)$ in expectation

Cuckoo Hashing [Pagh, Rodler 2005]

- A third method of resolving collisions
- Queries are $O(1)$ *worst case*
- Insert will still be $O(1)$ in expectation
- Comparison to linear probing and chaining?

Cuckoo Hashing [Pagh, Rodler 2005]

- A third method of resolving collisions
- Queries are $O(1)$ *worst case*
- Insert will still be $O(1)$ in expectation
- Comparison to linear probing and chaining?
  - Chaining: $O(1)$ worst case inserts; $O(1)$ expected queries. Not as good for query-heavy workloads!

Cuckoo Hashing [Pagh, Rodler 2005]

- A third method of resolving collisions
- Queries are $O(1)$ *worst case*
- Insert will still be $O(1)$ in expectation
- Comparison to linear probing and chaining?
  - Chaining: $O(1)$ worst case inserts; $O(1)$ expected queries. Not as good for query-heavy workloads!
  - Linear probing: more cache-efficient, but both inserts and queries are only $O(1)$ on average

Cuckoo Hashing Invariant

- Table of size cn with $c = 2$ (for now)
- Have two (uniform random) hash functions h_1, h_2; outputs a number from `0` to $cn - 1$

Cuckoo Hashing Invariant

- Table of size cn with $c = 2$ (for now)
- Have two (uniform random) hash functions h_1, h_2; outputs a number from 0 to $cn - 1$
- Invariant: item x is either stored at $h_1(x)$, or at $h_2(x)$.

Cuckoo Hashing Invariant

- Table of size cn with $c = 2$ (for now)
- Have two (uniform random) hash functions h_1, h_2; outputs a number from 0 to $cn - 1$
- Invariant: item x is either stored at $h_1(x)$, or at $h_2(x)$.
- We'll come back to inserts. But how can we query? How much time does a query take?

Cuckoo Hashing Invariant

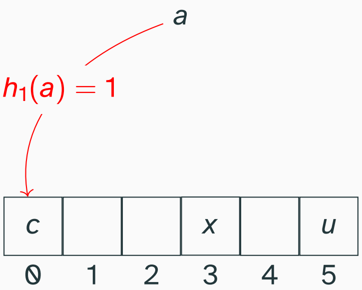
- Table of size cn with $c = 2$ (for now)
- Have two (uniform random) hash functions h_1, h_2; outputs a number from 0 to $cn - 1$
- Invariant: item x is either stored at $h_1(x)$, or at $h_2(x)$.
- We'll come back to inserts. But how can we query? How much time does a query take?
  - Check both hash slots. Immediately get $O(1)$ time

Cuckoo Hashing Inserts

c			x		u
0	1	2	3	4	5

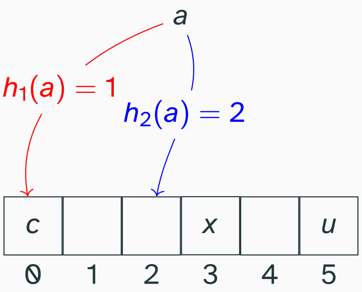
- Let’s say we want to insert a new item a. How can we do that?

Cuckoo Hashing Inserts



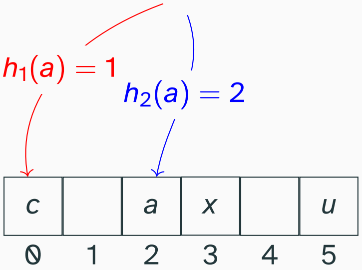
- Let's say we want to insert a new item a. How can we do that?

Cuckoo Hashing Inserts



- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.

Cuckoo Hashing Inserts



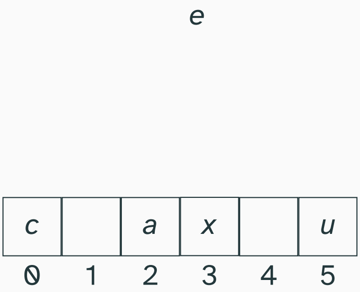
- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.

Cuckoo Hashing Inserts

c		a	x		u
0	1	2	3	4	5

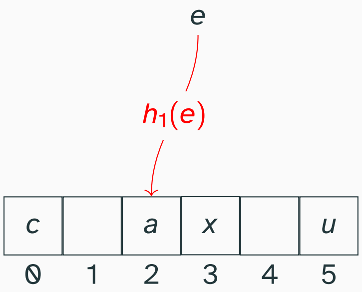
- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.

Cuckoo Hashing Inserts



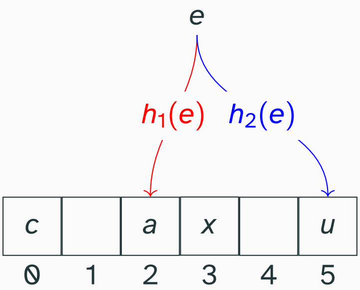
- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.
- What do we do if both are full?

Cuckoo Hashing Inserts



- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.
- What do we do if both are full?

Cuckoo Hashing Inserts

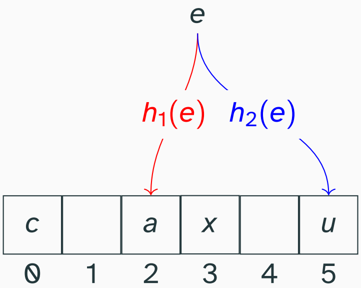


- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.
- What do we do if both are full?

Cuckooing!

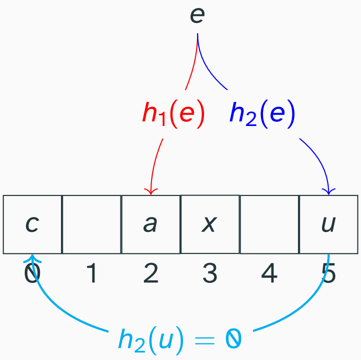
Cuckoos kick other birds' eggs out of the nest, replacing them with their own.

Cuckoo Hashing Inserts



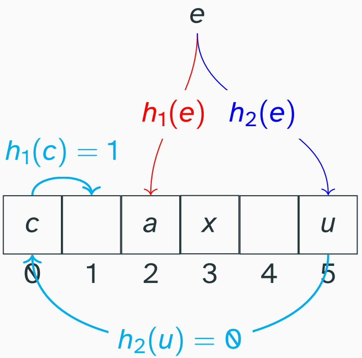
- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.
- What do we do if both are full?
- Move one of the items in the way to its other slot!
- If there's an item THERE, recurse

Cuckoo Hashing Inserts



- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.
- What do we do if both are full?
- Move one of the items in the way to its other slot!
- If there's an item THERE, recurse

Cuckoo Hashing Inserts



- Let's say we want to insert a new item a. How can we do that?
- Easy case: if $h_1(a)$ or $h_2(a)$ is free, can just store a immediately.
- What do we do if both are full?
- Move one of the items in the way to its other slot!
- If there's an item THERE, recurse

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$

Does this *always* work?

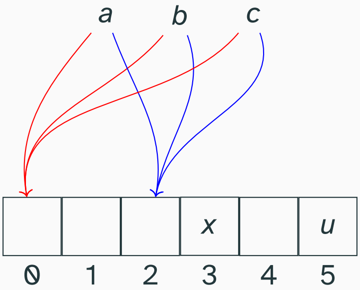
- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One potential problem: three items x, y, and z all have the same two hashes. Can't maintain the invariant!

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One potential problem: three items x, y, and z all have the same two hashes. Can't maintain the invariant!
- If this occurs, our insert algorithm (so far) loops infinitely



Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- In pairs: What is the probability that this exact scenario happens if we store n items in $2n$ slots?

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- In pairs: What is the probability that this exact scenario happens if we store n items in $2n$ slots?
    - There exist slots s_1 and s_2 such that all of x, y, and z all hash to one of these two slots

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- In pairs: What is the probability that this exact scenario happens if we store n items in $2n$ slots?
    - There exist slots s_1 and s_2 such that all of x, y, and z all hash to one of these two slots
    - For a given x, y, z, s_1, and s_2, how often does $h_1(x) = h_1(y) = h_1(z) = s_1$ and $h_2(x) = h_2(y) = h_2(z) = s_2$?

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- In pairs: What is the probability that this exact scenario happens if we store n items in $2n$ slots?
  - There exist slots s_1 and s_2 such that all of x, y, and z all hash to one of these two slots
  - For a given x, y, z, s_1, and s_2, how often does $h_1(x) = h_1(y) = h_1(z) = s_1$ and $h_2(x) = h_2(y) = h_2(z) = s_2$?
  - $(1/2n)^6$

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- In pairs: What is the probability that this exact scenario happens if we store n items in $2n$ slots?
    - There exist slots s_1 and s_2 such that all of x, y, and z all hash to one of these two slots
    - For a given x, y, z, s_1, and s_2, how often does $h_1(x) = h_1(y) = h_1(z) = s_1$ and $h_2(x) = h_2(y) = h_2(z) = s_2$?
    - $(1/2n)^6$
    - There are $\binom{n}{3}\binom{2n}{2}$ choices of x, y, z, s_1, and s_2

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- In pairs: What is the probability that this exact scenario happens if we store n items in $2n$ slots?
    - There exist slots s_1 and s_2 such that all of x, y, and z all hash to one of these two slots
    - For a given x, y, z, s_1, and s_2, how often does $h_1(x) = h_1(y) = h_1(z) = s_1$ and $h_2(x) = h_2(y) = h_2(z) = s_2$?
    - $(1/2n)^6$
    - There are $\binom{n}{3}\binom{2n}{2}$ choices of x, y, z, s_1, and s_2
    - So this happens with probability $\Theta(n^3 n^2/n^6) = \Theta(1/n)$.

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- This occurs with probability $O(1/n)$
- With more work: probability of an insert looping infinitely is $O(1/n)$ (proof is outside the scope of the course)

Does this *always* work?

- Recall our invariant: every item x is stored at $h_1(x)$ or $h_2(x)$
- Is there a simple example where this is impossible?
- One option: three items x, y, and z all have the same two hashes
- This occurs with probability $O(1/n)$
- With more work: probability of an insert looping infinitely is $O(1/n)$ (proof is outside the scope of the course)
- Inserts loop very rarely if n is large (you probably will not see this happen on Assignment 3). Usually put in a maximum number of iterations, after which the insert fails, to prevent looping infinitely

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
  - Two cache misses per query. Is that good?

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
  - Two cache misses per query. Is that good?
  - Kind of! Probably better than chaining. But linear probing has only $\approx$one cache miss on any query, so long as $\log n$ items fit in a cache line

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
  - Two cache misses per query. Is that good?
  - Kind of! Probably better than chaining. But linear probing has only $\approx$one cache miss on any query, so long as $\log n$ items fit in a cache line
- What is the Insert performance?

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
  - Two cache misses per query. Is that good?
  - Kind of! Probably better than chaining. But linear probing has only $\approx$one cache miss on any query, so long as $\log n$ items fit in a cache line
- What is the Insert performance?
  - Result: Inserts are $O(1)$ in expectation

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
  - Two cache misses per query. Is that good?
  - Kind of! Probably better than chaining. But linear probing has only $\approx$one cache miss on any query, so long as $\log n$ items fit in a cache line
- What is the Insert performance?
  - Result: Inserts are $O(1)$ in expectation
  - Idea: half the slots are empty, so each time we go to a new slot, we should have a $\approx 1/2$ probability of being done

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
  - Two cache misses per query. Is that good?
  - Kind of! Probably better than chaining. But linear probing has only $\approx$one cache miss on any query, so long as $\log n$ items fit in a cache line
- What is the Insert performance?
  - Result: Inserts are $O(1)$ in expectation
  - Idea: half the slots are empty, so each time we go to a new slot, we should have a $\approx 1/2$ probability of being done
  - (Analysis is nontrivial since these are not truly independent, and we need to carefully avoid cases with an infinite loop)

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
- Insert cache performance?

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
- Insert cache performance?
  - One cache miss per “cuckoo”–OK but not great

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
- Insert cache performance?
  - One cache miss per "cuckoo"–OK but not great
  - In practice, inserts are really pretty bad for cuckoo hashing due to poor constants

Cuckoo Hashing Performance

- Queries: $O(1)$ worst case
- Cache performance?
- Insert cache performance?
  - One cache miss per “cuckoo”–OK but not great
  - In practice, inserts are really pretty bad for cuckoo hashing due to poor constants
- Idea: cuckoo hashing does great on queries (though with potentially worse cache efficiency than linear probing), but pays for it with expensive inserts

Limits of Expectation

Limits of Expectation

- Let's say I charge you \$1000 to play a game. With probability 1 in 1 million, I give you \$10 billion. Otherwise, I give you \$0.

Limits of Expectation

- Let's say I charge you $1000 to play a game. With probability 1 in 1 million, I give you $10 billion. Otherwise, I give you $0.

- Would you play this game? (Like in real life, right now.)

Limits of Expectation

- Let's say I charge you $1000 to play a game. With probability 1 in 1 million, I give you $10 billion. Otherwise, I give you $0.

- Would you play this game? (Like in real life, right now.)

- Answer: some of you might, but I'm guessing many of you would not. You're just going to lose $1000.

- But expectation is good! You expect to win $9000.

Concentration bounds

- Rather than giving the average performance, bound the probability of performance.

Concentration bounds

- Rather than giving the average performance, bound the probability of 

performance.

- Let's say I flip a coin k times. On average, I see $k/2$ heads. But what is the probability I *never* see a heads?

Concentration bounds

- Rather than giving the average performance, bound the probability of performance.

- Let's say I flip a coin k times. On average, I see $k/2$ heads. But what is the probability I *never* see a heads?

- Answer: $1/2^k$

Concentration bounds

- Rather than giving the average performance, bound the probability of performance.
- Let's say I flip a coin k times. On average, I see $k/2$ heads. But what is the probability I *never* see a heads?
- Answer: $1/2^k$
- Quicksort has expected runtime $O(n \log n)$. What is the probability that the running time is more than $O(n \log n)$?

Concentration bounds

- Rather than giving the average performance, bound the probability of [illegible] performance.
- Let's say I flip a coin k times. On average, I see $k/2$ heads. But what is the probability I *never* see a heads?
- Answer: $1/2^k$
- Quicksort has expected runtime $O(n \log n)$. What is the probability that the running time is more than $O(n \log n)$?
- Answer: $O(1/n)$ (this is why quicksort is not worse than merge sort even though it can be $\Theta(n^2)$: you'll never see the worst case if n is at all large)

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
    - Quicksort is $O(n \log n)$ with high probability

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
  - Quicksort is $O(n \log n)$ with high probability
  - Cuckoo hashing inserts finish without looping with high probability

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
  - Quicksort is $O(n \log n)$ with high probability
  - Cuckoo hashing inserts finish without looping with high probability
- Some new results (each is $O(1)$ in expectation):

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
  - Quicksort is $O(n \log n)$ with high probability
  - Cuckoo hashing inserts finish without looping with high probability
- Some new results (each is $O(1)$ in expectation):
  - Cuckoo hashing inserts require $O(\log n)$ swaps with high probability

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
  - Quicksort is $O(n \log n)$ with high probability
  - Cuckoo hashing inserts finish without looping with high probability
- Some new results (each is $O(1)$ in expectation):
  - Cuckoo hashing inserts require $O(\log n)$ swaps with high probability
  - Linear probing queries require $O(\log n)$ time with high probability.

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
  - Quicksort is $O(n \log n)$ with high probability
  - Cuckoo hashing inserts finish without looping with high probability
- Some new results (each is $O(1)$ in expectation):
  - Cuckoo hashing inserts require $O(\log n)$ swaps with high probability
  - Linear probing queries require $O(\log n)$ time with high probability.
  - Chaining queries require $O(\frac{\log n}{\log \log n})$ time with high probability

With High Probability

- An event happens *with high probability* (with respect to n) if it happens with probability $1 - O(1/n)$
- We've seen:
  - Quicksort is $O(n \log n)$ with high probability
  - Cuckoo hashing inserts finish without looping with high probability
- Some new results (each is $O(1)$ in expectation):
  - Cuckoo hashing inserts require $O(\log n)$ swaps with high probability
  - Linear probing queries require $O(\log n)$ time with high probability.
  - Chaining queries require $O(\frac{\log n}{\log \log n})$ time with high probability
- With high probability is always with respect to a variable. Assume that it's with respect to n unless stated otherwise.

WHP example

- How many coins do I need to flip before I see a heads with high probability? (With respect to some variable n)

WHP example

- How many coins do I need to flip before I see a heads with high probability? (With respect to some variable n)

- If I flip k times, I see a heads with probability $1 - 1/2^k$.

WHP example

- How many coins do I need to flip before I see a heads with high probability? (With respect to some variable n)
- If I flip k times, I see a heads with probability $1 - 1/2^k$.
- So I need $1/2^k = O(1/n)$. Solving, $k = \Theta(\log n)$.

WHP example

- How many coins do I need to flip before I see a heads with high probability? (With respect to some variable n)

- If I flip k times, I see a heads with probability $1 - 1/2^k$.

- So I need $1/2^k = O(1/n)$. Solving, $k = \Theta(\log n)$.

- This is (a simplified version of) the analysis leading to the $O(\log n)$ worst case bounds on the last slide

Expectation vs Concentration (WHP)

- We'll usually use "with high probability" for concentration bounds

Expectation vs Concentration (WHP)

- We'll usually use "with high probability" for concentration bounds
- Expectation states how well the algorithm does on average. Could be much better or worse sometimes!

Expectation vs Concentration (WHP)

- We'll usually use "with high probability" for concentration bounds

- Expectation states how well the algorithm does on average. Could be much better or worse sometimes!

- "With high probability" gives a guarantee that will almost always be met: if n is large it becomes vanishingly unlikely that the bound will be violated.

Expectation vs Concentration (WHP)

- We'll usually use "with high probability" for concentration bounds
- Expectation states how well the algorithm does on average. Could be much better or worse sometimes!
- "With high probability" gives a guarantee that will almost always be met: if n is large it becomes vanishingly unlikely that the bound will be violated.
- Largely fulfills the promise of classic worst-case algorithm analysis, but applied to randomized algorithms