COT 6936: Topics in Algorithms

Giri Narasimhan

ECS 254A / EC 2474; Phone x3748; Email: giri@cs.fiu.edu
HOMEPAGE: http://www.cs.fiu.edu/~giri
https://moodle.cis.fiu.edu/v2.1/course/view.php?id=612

Jan 30 & Feb 4, 2014

Presentation Outline

What is a Randomized Algorithm?

- It is an algorithm that has random steps,

What is a Randomized Algorithm?

- It is an algorithm that has random steps, i.e., actions that depend on the result of a coin toss

What is a Randomized Algorithm?

- It is an algorithm that has random steps, i.e., actions that depend on the result of a coin toss or a *random number generator*

What is a Randomized Algorithm?

- It is an algorithm that has random steps, i.e., actions that depend on the result of a coin toss or a *random number generator*
- Applications
  - Protocol in Ethernet Cards to decide when it should (re)try access to shared medium

What is a Randomized Algorithm?

- It is an algorithm that has random steps, i.e., actions that depend on the result of a coin toss or a *random number generator*
- Applications
  - Protocol in Ethernet Cards to decide when it should (re)try access to shared medium
  - Primality testing and crytpography

What is a Randomized Algorithm?

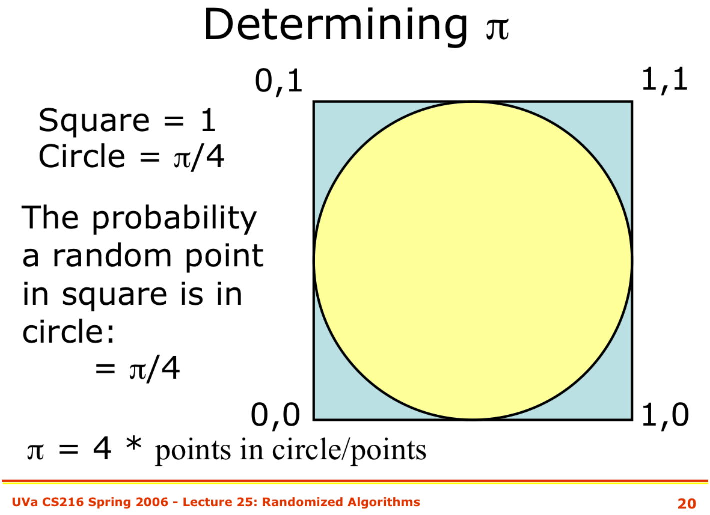
- It is an algorithm that has random steps, i.e., actions that depend on the result of a coin toss or a *random number generator*
- Applications
  - Protocol in Ethernet Cards to decide when it should (re)try access to shared medium
  - Primality testing and crytpography
  - Monte Carlo simulations
  - ...

What is a Randomized Algorithm?

- It is an algorithm that has random steps, i.e., actions that depend on the result of a coin toss or a *random number generator*
- Applications
  - Protocol in Ethernet Cards to decide when it should (re)try access to shared medium
  - Primality testing and crytpography
  - Monte Carlo simulations
  - ...
- Advantages: Often easier to implement and more efficient

Example: Monte Carlo Simulations

Example: Monte Carlo Simulations



Slide by David Evans

Presentation Outline

QuickSort vs Randomized QuickSort

- QuickSort
  - Pick a fixed pivot
  - Partition input based on pivot into two sets
  - Recursively sort the two partitions

QuickSort vs Randomized QuickSort

- QuickSort
  - Pick a fixed pivot
  - Partition input based on pivot into two sets
  - Recursively sort the two partitions
- Randomized QuickSort

QuickSort vs Randomized QuickSort

- QuickSort
  - Pick a fixed pivot
  - Partition input based on pivot into two sets
  - Recursively sort the two partitions
- Randomized QuickSort
  - Pick a random pivot

QuickSort vs Randomized QuickSort

- QuickSort
  - Pick a fixed pivot
  - Partition input based on pivot into two sets
  - Recursively sort the two partitions
- Randomized QuickSort
  - Pick a random pivot
  - Partition input based on pivot into two sets
  - Recursively sort the two partitions

QuickSort: Probabilistic Analysis

QuickSort: Probabilistic Analysis

Worst-case $= O(n^2)$

QuickSort: Probabilistic Analysis

Worst-case $= O(n^2)$
To analyze average case, we need to know input distribution

QuickSort: Probabilistic Analysis

Worst-case $= O(n^2)$

To analyze average case, we need to know input distribution

- Expected rank of pivot $= n/2$.

QuickSort: Probabilistic Analysis

Worst-case $= O(n^2)$

To analyze average case, we need to know input distribution

- Expected rank of pivot $= n/2$. (Why?)

QuickSort: Probabilistic Analysis

Worst-case = $O(n^2)$
To analyze average case, we need to know input distribution

- Expected rank of pivot = $n/2$. (Why?)
- Expected size of sublists after partition = $n/2$

QuickSort: Probabilistic Analysis

Worst-case $= O(n^2)$
To analyze average case, we need to know input distribution

- Expected rank of pivot $= n/2$. (Why?)
- Expected size of sublists after partition $= n/2$
- Thus recurrence relation is

$$T(n) = 2T(n/2) + O(n)$$

QuickSort: Probabilistic Analysis

Worst-case $= O(n^2)$
To analyze average case, we need to know input distribution

- Expected rank of pivot $= n/2$. (Why?)
- Expected size of sublists after partition $= n/2$
- Thus recurrence relation is

$$T(n) = 2T(n/2) + O(n)$$

- Average Time Complexity $=$

$$T(n) = O(n \log n)$$

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.
- For $i < j$, let X_{ij} be a random variable that takes on value 1 if y_i is compared to y_j and 0 otherwise.

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.
- For $i < j$, let X_{ij} be a random variable that takes on value 1 if y_i is compared to y_j and 0 otherwise.

$$\text{Total number of comparisons,} \quad X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.
- For $i < j$, let X_{ij} be a random variable that takes on value 1 if y_i is compared to y_j and 0 otherwise.

$$\text{Total number of comparisons,} \quad X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

- By linearity of expectation, we have

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.
- For $i < j$, let X_{ij} be a random variable that takes on value 1 if y_i is compared to y_j and 0 otherwise.

$$\text{Total number of comparisons,} \quad X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

- By linearity of expectation, we have

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

- y_i and y_j are compared iff one of them is the first to be picked as a pivot among items $y_i, y_{i+1}, \ldots, y_j$.

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.
- For $i < j$, let X_{ij} be a random variable that takes on value 1 if y_i is compared to y_j and 0 otherwise.

$$\text{Total number of comparisons,} \quad X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

- By linearity of expectation, we have

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

- y_i and y_j are compared iff one of them is the first to be picked as a pivot among items $y_i, y_{i+1}, \ldots, y_j$. Thus,

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

Randomized QuickSort: Randomized Analysis

- Let $y_1, y_2, \ldots, y_n$ be the input set in sorted order.
- For $i < j$, let X_{ij} be a random variable that takes on value 1 if y_i is compared to y_j and 0 otherwise.

$$\text{Total number of comparisons,} \quad X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

- By linearity of expectation, we have

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

- y_i and y_j are compared iff one of them is the first to be picked as a pivot among items $y_i, y_{i+1}, \ldots, y_j$. Thus,

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

Randomized QuickSort: Randomized Analysis

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Randomized QuickSort: Randomized Analysis

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} = \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k}
\end{aligned}
$$

Randomized QuickSort: Randomized Analysis

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} = \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k) \frac{2}{k}
\end{aligned}
$$

Randomized QuickSort: Randomized Analysis

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} = \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k) \frac{2}{k} \\
&= \left((n+1) \sum_{k=2}^{n} \frac{2}{k} \right) - 2(n-1)
\end{aligned}
$$

Randomized QuickSort: Randomized Analysis

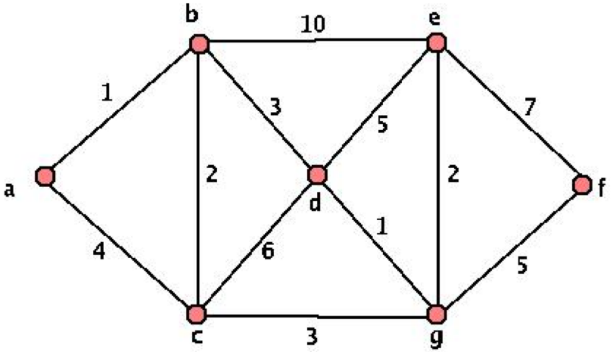
$$\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} = \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k) \frac{2}{k} \\
&= \left((n+1) \sum_{k=2}^{n} \frac{2}{k} \right) - 2(n-1) \\
&= (2n+2) \sum_{k=1}^{n} \frac{1}{k} - 4n
\end{aligned}$$

Randomized QuickSort: Randomized Analysis

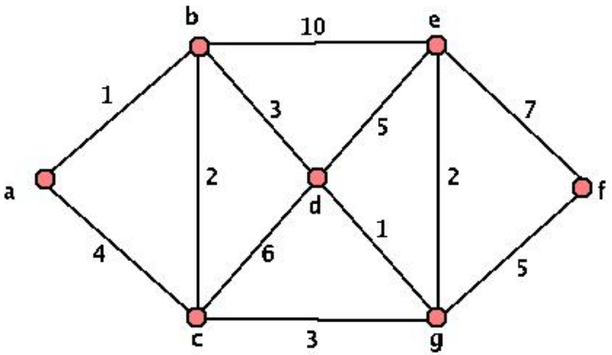
$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} = \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k) \frac{2}{k} \\
&= \left((n+1) \sum_{k=2}^{n} \frac{2}{k} \right) - 2(n-1) \\
&= (2n+2) \sum_{k=1}^{n} \frac{1}{k} - 4n \\
&= 2n \ln n + \Theta(n)
\end{aligned}
$$

Presentation Outline

Cut-Sets and Min-Cuts

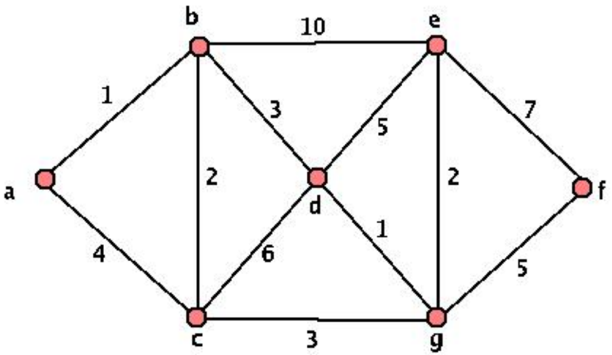


Cut-Sets and Min-Cuts



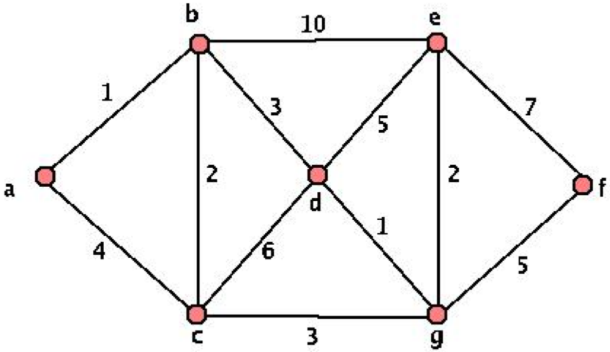
- Cut-set 1: $(\{a, b, c, d\}, \{e, f, g\})$ Weight $= 19$

Cut-Sets and Min-Cuts



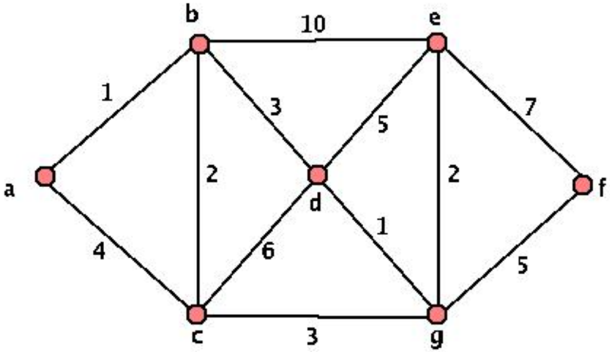
- Cut-set 1: $(\{a, b, c, d\}, \{e, f, g\})$ Weight $= 19$
- Cut-set 2: $(\{a, b, g\}, \{c, d, e, f\})$ Weight $= 30$

Cut-Sets and Min-Cuts



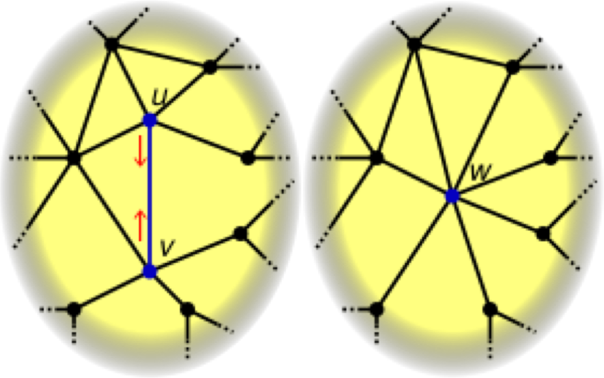
- Cut-set 1: $(\{a, b, c, d\}, \{e, f, g\})$ Weight $= 19$
- Cut-set 2: $(\{a, b, g\}, \{c, d, e, f\})$ Weight $= 30$
- Cut-set 3: $(\{a\}, \{b, c, d, e, f, g\})$ Weight $= 5$

Cut-Sets and Min-Cuts



- Cut-set 1: $(\{a, b, c, d\}, \{e, f, g\})$ Weight $= 19$
- Cut-set 2: $(\{a, b, g\}, \{c, d, e, f\})$ Weight $= 30$
- Cut-set 3: $(\{a\}, \{b, c, d, e, f, g\})$ Weight $= 5$

Edge Contraction



http://en.wikipedia.org/wiki/Edge_contraction

Edge Contractions and Min-Cuts

- Lemma: If you are not contracting an edge from the cut-set, edge contractions do not affect the size of min-cuts.

Edge Contractions and Min-Cuts

- Lemma: If you are not contracting an edge from the cut-set, edge contractions do not affect the size of min-cuts.
- Observation: Most edges are not part of the min-cut.

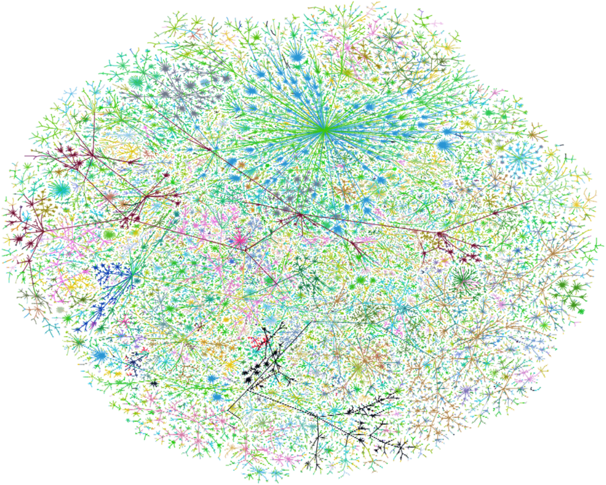
Edge Contractions and Min-Cuts

- Lemma: If you are not contracting an edge from the cut-set, edge contractions do not affect the size of min-cuts.
- Observation: Most edges are not part of the min-cut.
- Idea: Use randomization

Edge Contractions and Min-Cuts

- Lemma: If you are not contracting an edge from the cut-set, edge contractions do not affect the size of min-cuts.
- Observation: Most edges are not part of the min-cut.
- Idea: Use randomization

Min-Cuts in the Internet Graph



June 1999 Internet graph, Bill Cheswick `http://research.lumeta.com/ches/map/gallery/index.html`

- Algorithm

Randomized Algorithms: Unweighted Min-Cut

- Algorithm
  - Pick a random edge and contract it until only 2 vertices are remaining

Randomized Algorithms: Unweighted Min-Cut

- Algorithm
  - Pick a random edge and contract it until only 2 vertices are remaining
  - Report edges connecting the 2 remaining vertices as the min-cut

Randomized Algorithms: Unweighted Min-Cut

- Algorithm
  - Pick a random edge and contract it until only 2 vertices are remaining
  - Report edges connecting the 2 remaining vertices as the min-cut
- Steps of the Analysis

Randomized Algorithms: Unweighted Min-Cut

- Algorithm
  - Pick a random edge and contract it until only 2 vertices are remaining
  - Report edges connecting the 2 remaining vertices as the min-cut
- Steps of the Analysis
  - Assume that Unweighted Min-cut has k edges

Randomized Algorithms: Unweighted Min-Cut

- Algorithm
  - Pick a random edge and contract it until only 2 vertices are remaining
  - Report edges connecting the 2 remaining vertices as the min-cut
- Steps of the Analysis
  - Assume that Unweighted Min-cut has k edges
  - Prob $\{$edge is not in Min-cut$\} \geq 1 - 2/n$

Randomized Algorithms: Unweighted Min-Cut

- Algorithm
  - Pick a random edge and contract it until only 2 vertices are remaining
  - Report edges connecting the 2 remaining vertices as the min-cut
- Steps of the Analysis
  - Assume that Unweighted Min-cut has k edges
  - Prob $\{$edge is not in Min-cut$\} \geq 1 - 2/n$
  - Prob $\{$Min-cut is output$\} \geq 2/n(n-1)$

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Observation:

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Observation:
  - If Min-Cut has k edges, then minimum degree of every vertex is k. (Why?)

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Observation:
  - If Min-Cut has k edges, then minimum degree of every vertex is k. (Why?)
- At start, number of edges in graph $\geq kn/2$

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Observation:
  - If Min-Cut has k edges, then minimum degree of every vertex is k. (Why?)
- At start, number of edges in graph $\geq kn/2$
- Probability that an edge from Min-Cut is picked in iteration 1 is $\leq k/(kn/2) \leq 2/n$

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Observation:
  - If Min-Cut has k edges, then minimum degree of every vertex is k. (Why?)
- At start, number of edges in graph $\geq kn/2$
- Probability that an edge from Min-Cut is picked in iteration 1 is $\leq k/(kn/2) \leq 2/n$
- Probability that no edge from Min-Cut is picked in iteration 1 is $\geq 1 - 2/n$

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Iteration i?

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Iteration i?
- E_i = Event that no edge from Min-Cut is picked in iteration i

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Iteration i?
- E_i = Event that no edge from Min-Cut is picked in iteration i
- F_i = Event that no edge from Min-Cut is picked in iteration 1 through i

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Iteration i?
- E_i = Event that no edge from Min-Cut is picked in iteration i
- F_i = Event that no edge from Min-Cut is picked in iteration 1 through i

$$Pr(E_i|F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}$$

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Iteration i?
- E_i = Event that no edge from Min-Cut is picked in iteration i
- F_i = Event that no edge from Min-Cut is picked in iteration 1 through i

$$Pr(E_i|F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}$$

- We need F_{n-2}.

Analysis: Unweighted Min-Cut Algorithm (Contd)

$$
\begin{aligned}
Pr(F_{n-2}) &= Pr(E_{n-2} \cap F_{n-3}) = Pr(E_{n-2}|F_{n-3})Pr(F_{n-3}) \\
&= Pr(E_{n-2}|F_{n-3}) \cdot Pr(E_{n-3}|F_{n-4}) \dots Pr(E_2|F_1)Pr(F_1) \\
&\geq \Pi_{i=1}^{n-2}\left(1 - \frac{2}{n-i+1}\right) = \Pi_{i=1}^{n-2}\frac{n-i-1}{n-I+1} \\
&= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\dots\frac{4}{6}\frac{3}{5}\frac{2}{4}\frac{1}{3} \\
&= \frac{2}{n(n-1)}.
\end{aligned}
$$

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut $\geq 2/n(n-1)$
  - Rather low! Also, dependent on n.

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut $\geq 2/n(n-1)$
  - Rather low! Also, dependent on n.
- To boost success probability, repeat algorithm.

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut $\geq 2/n(n-1)$
  - Rather low! Also, dependent on n.
- To boost success probability, repeat algorithm.
  - How many times?

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut $\geq 2/n(n-1)$
  - Rather low! Also, dependent on n.
- To boost success probability, repeat algorithm.
  - How many times?
  - Goal: repeat until prob of error is very small

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut $\geq 2/n(n-1)$
  - Rather low! Also, dependent on n.
- To boost success probability, repeat algorithm.
  - How many times?
  - Goal: repeat until prob of error is very small
  - Use the following fact: $(1 - 1/h)^h \leq e$.

Analysis: Unweighted Min-Cut Algorithm (Contd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut $\geq 2/n(n-1)$
  - Rather low! Also, dependent on n.
- To boost success probability, repeat algorithm.
  - How many times?
  - Goal: repeat until prob of error is very small
  - Use the following fact: $(1-1/h)^h \leq e$. Thus,

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

Presentation Outline

Monte Carlo vs Las Vegas

Monte Carlo vs Las Vegas

- Monte Carlo algorithms: Always fast. Often correct, but with bounded probability

Monte Carlo vs Las Vegas

- Monte Carlo algorithms: Always fast. Often correct, but with bounded probability
  - One-sided vs Two-sided errors

Monte Carlo vs Las Vegas

- Monte Carlo algorithms: Always fast. Often correct, but with bounded probability
  - One-sided vs Two-sided errors
- Las Vegas algorithms: Always correct, Often fast

Presentation Outline

Balls and Bins Model

Balls and Bins Model

- Throw m balls into n bins

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? ?

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table
  - How many bins are empty?

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table
  - How many bins are empty? $e^{m/n}$

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table
  - How many bins are empty? $e^{m/n}$
  - How many balls in the fullest bin?

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table
  - How many bins are empty? $e^{m/n}$
  - How many balls in the fullest bin? $\Theta(\ln n/ \ln \ln n)$

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table
  - How many bins are empty? $e^{m/n}$
  - How many balls in the fullest bin? $\Theta(\ln n / \ln \ln n)$ Hashing worst-case time
  - If $m = n$, how many bins are expected to have > 1 balls?

Balls and Bins Model

- Throw m balls into n bins
- Location of each ball chosen independently and uniformly at random
- Questions to ask?
  - How many balls in a bin on the average? 2 Average size of a chain in a hash table
  - How many bins are empty? $e^{m/n}$
  - How many balls in the fullest bin? $\Theta(\ln n / \ln \ln n)$ Hashing worst-case time
  - If $m = n$, how many bins are expected to have > 1 balls? Birthday Paradox

Balls and Bins: Applications

- Chain Hashing

Balls and Bins: Applications

- Chain Hashing
- Bucket Sort

Balls and Bins: Applications

- Chain Hashing
- Bucket Sort
- Hash Tables for passwords

Balls and Bins: Applications

- Chain Hashing
- Bucket Sort
- Hash Tables for passwords
  - If entry is not free then password rejected

Balls and Bins: Applications

- Chain Hashing
- Bucket Sort
- Hash Tables for passwords
  - If entry is not free then password rejected
- Bloom Filters (generalize hash tables)

Balls and Bins: Applications

- Chain Hashing
- Bucket Sort
- Hash Tables for passwords
  - If entry is not free then password rejected
- Bloom Filters (generalize hash tables)
  - See later slides

Presentation Outline

Birthday Paradox

- Probability that m balls are put into distinct bins is:

$$\leq \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{m-1}{n}\right) = \prod_{j=1}^{m-1}\left(1 - \frac{j}{n}\right)$$

- To achieve probability at least $1/2$, we need:

Birthday Paradox

- Probability that m balls are put into distinct bins is:

$$\leq \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{m-1}{n}\right) = \prod_{j=1}^{m-1}\left(1 - \frac{j}{n}\right)$$

- To achieve probability at least 1/2, we need:
  - $m^2/2n \geq \ln 2$

Birthday Paradox

- Probability that m balls are put into distinct bins is:

$$\leq \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{m-1}{n}\right) = \prod_{j=1}^{m-1} \left(1 - \frac{j}{n}\right)$$

- To achieve probability at least 1/2, we need:
  - $m^2/2n \geq \ln 2$
  - $m \geq \sqrt{2n \ln 2}$

Birthday Paradox

- Probability that m balls are put into distinct bins is:

$$\leq \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{m-1}{n}\right) = \prod_{j=1}^{m-1}\left(1 - \frac{j}{n}\right)$$

- To achieve probability at least 1/2, we need:
  - $m^2/2n \geq \ln 2$
  - $m \geq \sqrt{2n \ln 2}$
- In a room with at least 23 people, the probability that at least two people have the same birthday is more than 1/2.

Presentation Outline

Average Search Time for Hashing

- We want *Average Length of Chain in Hash Table*
- Let N be number of possible hash values
- Let k be number of items in hash table

Average Search Time for Hashing

- We want *Average Length of Chain in Hash Table*
- Let N be number of possible hash values
- Let k be number of items in hash table
- Prob that exactly i out of k items hash to same value:

Average Search Time for Hashing

- We want *Average Length of Chain in Hash Table*
- Let N be number of possible hash values
- Let k be number of items in hash table
- Prob that exactly i out of k items hash to same value:

$$p_i = \binom{k}{i} (N-1)^{k-i} N^{-k}$$

Average Search Time for Hashing

- We want *Average Length of Chain in Hash Table*
- Let N be number of possible hash values
- Let k be number of items in hash table
- Prob that exactly i out of k items hash to same value:

$$p_i = \binom{k}{i} (N-1)^{k-i} N^{-k}$$

- Time for unsuccessful search =

Average Search Time for Hashing

- We want *Average Length of Chain in Hash Table*
- Let N be number of possible hash values
- Let k be number of items in hash table
- Prob that exactly i out of k items hash to same value:

$$p_i = \binom{k}{i} (N-1)^{k-i} N^{-k}$$

- Time for unsuccessful search = length of chain + 1

Average Search Time for Hashing

- We want *Average Length of Chain in Hash Table*
- Let N be number of possible hash values
- Let k be number of items in hash table
- Prob that exactly i out of k items hash to same value:

$$p_i = \binom{k}{i} (N-1)^{k-i} N^{-k}$$

- Time for unsuccessful search = length of chain + 1
- Average time for unsuccessful search:

$$A = \sum_i (i+1) p_i$$

Average (Unsuccessful) Search Time

$$
\begin{aligned}
A &= \sum_i (i+1)p_i = \sum_i \binom{k}{i}(i+1)(N-1)^{k-i}N^{-k} \\
&= \sum_i \binom{k}{i} i(N-1)^{k-i}N^{-k} + \sum_i \binom{k}{i}(N-1)^{k-i}N^{-k} \\
&= \sum_i k\binom{k-1}{i-1}(N-1)^{k-i}N^{-k} + 1 \\
&= kN^{-k}\sum_i k\binom{k-1}{i}(N-1)^{k-i-1} + 1 \\
&= kN^{-k}N^{k-1} + 1 = 1 + k/N
\end{aligned}
$$

Average (Successful) Search Time

$$A' = \sum_{i,j} j q_{ij} = 1 + \frac{k-1}{2N}$$

Maximum Load

- Prob that a bin has at least j items is

$$\binom{n}{j}\left(\frac{1}{n}\right)^j \le \frac{1}{j!} \le \left(\frac{e}{j}\right)^j$$

- Prob that a bin has $\ge j = 3\ln n/\ln\ln n$ items is:

$$\begin{aligned}
n\left(\frac{e}{j}\right)^j &\le n\left(\frac{e\ln\ln n}{3\ln n}\right)^{3\ln n/\ln\ln n} \\
&\le n\left(\frac{\ln\ln n}{3\ln n}\right)^{3\ln n/\ln\ln n} \\
&= e^{\ln n}\left(e^{\ln\ln\ln n - \ln\ln n}\right)^{3\ln n/\ln\ln n} \\
&= e^{-2\ln n + 3(\ln n)(\ln\ln\ln n)/\ln\ln n} \\
&\le 1/n
\end{aligned}$$

Presentation Outline

Randomized Algorithm for MAX-3SAT

Randomized Algorithm for MAX-3SAT

- Randomly assign 0/1 to all variables

Randomized Algorithm for MAX-3SAT

- Randomly assign 0/1 to all variables
- Each clause is satisfied with prob $7/8$

Randomized Algorithm for MAX-3SAT

- Randomly assign $0/1$ to all variables
- Each clause is satisfied with prob $7/8$
- Expected number of clauses satisfied $= 7/8$

Randomized Algorithm for MAX-3SAT

- Randomly assign $0/1$ to all variables
- Each clause is satisfied with prob $7/8$
- Expected number of clauses satisfied $= 7/8$

Lemma: There exists a truth assignment that satisfies at least $7/8$-th of the clauses.

How to find such a truth assignment? Derandomization

Presentation Outline

Contention Resolution

- N processes $P_1, \ldots, P_N$ each competing for access to a single resource (shared database, shared communication channel, etc.)

Contention Resolution

- N processes $P_1, \ldots, P_N$ each competing for access to a single resource (shared database, shared communication channel, etc.)
- Time is divided into rounds

Contention Resolution

- N processes $P_1, \ldots, P_N$ each competing for access to a single resource (shared database, shared communication channel, etc.)
- Time is divided into rounds
- If more than one process attempts to access resource, then all processes are locked out

Contention Resolution

- N processes $P_1, \ldots, P_N$ each competing for access to a single resource (shared database, shared communication channel, etc.)
- Time is divided into rounds
- If more than one process attempts to access resource, then all processes are locked out
- No communication between processes

Contention Resolution

- N processes $P_1, \ldots, P_N$ each competing for access to a single resource (shared database, shared communication channel, etc.)
- Time is divided into rounds
- If more than one process attempts to access resource, then all processes are locked out
- No communication between processes
- Need fair algorithm for large N

Contention Resolution

- N processes $P_1, \ldots, P_N$ each competing for access to a single resource (shared database, shared communication channel, etc.)
- Time is divided into rounds
- If more than one process attempts to access resource, then all processes are locked out
- No communication between processes
- Need fair algorithm for large N
- Use randomization to break symmetry

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t.

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t. Not Scalable!

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t. Not Scalable!
- If N is large, then each process attempts to access the resource in round t with probability p.
  - To maximize probability of success, set $p =$

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t. Not Scalable!
- If N is large, then each process attempts to access the resource in round t with probability p.
  - To maximize probability of success, set $p = 1/n$.

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t. Not Scalable!
- If N is large, then each process attempts to access the resource in round t with probability p.
  - To maximize probability of success, set $p = 1/n$. Not surprising!
  - Prob of failure after $e \cdot n$ rounds is bounded by a constant.

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t. Not Scalable!
- If N is large, then each process attempts to access the resource in round t with probability p.
  - To maximize probability of success, set $p = 1/n$. Not surprising!
  - Prob of failure after $e \cdot n$ rounds is bounded by a constant. Fair!
  - W.h.p. all N processes can access the resource in $t = 2e \cdot n \ln n$ rounds.

Breaking Symmetry

- If N is small, then assign round $t \mod N$ to process t. Not Scalable!
- If N is large, then each process attempts to access the resource in round t with probability p.
  - To maximize probability of success, set $p = 1/n$. Not surprising!
  - Prob of failure after $e \cdot n$ rounds is bounded by a constant. Fair!
  - W.h.p. all N processes can access the resource in $t = 2e \cdot n \ln n$ rounds. Scalable!

Bloom Filters

- Bloom Filters: Used to test set membership by using bit arrays to indicate which positions have been hashed to. For every element k hash function are used instead of 1.

Bloom Filters

- Bloom Filters: Used to test set membership by using bit arrays to indicate which positions have been hashed to. For every element k hash function are used instead of 1. How to pick k?

Bloom Filters

- Bloom Filters: Used to test set membership by using bit arrays to indicate which positions have been hashed to. For every element k hash function are used instead of 1. How to pick k?

Presentation Outline

Power of Two Choices

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller “chain”

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller "chain"
  - Dramatically reduces the expected size of the largest bin

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller "chain"
  - Dramatically reduces the expected size of the largest bin while doubling the average search cost.

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller "chain"
  - Dramatically reduces the expected size of the largest bin while doubling the average search cost.
- Dynamic Resource Allocation: When multiple identical resources to choose from:

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller "chain"
  - Dramatically reduces the expected size of the largest bin while doubling the average search cost.
- Dynamic Resource Allocation: When multiple identical resources to choose from:
  - Deterministic Choice: Find load of each one and pick least loaded

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller "chain"
  - Dramatically reduces the expected size of the largest bin while doubling the average search cost.
- Dynamic Resource Allocation: When multiple identical resources to choose from:
  - Deterministic Choice: Find load of each one and pick least loaded
  - One Random Choice: Pick random resource

Power of Two Choices

- Hashing with two hash functions
  - Among two hash values, pick value with smaller "chain"
  - Dramatically reduces the expected size of the largest bin while doubling the average search cost.
- Dynamic Resource Allocation: When multiple identical resources to choose from:
  - Deterministic Choice: Find load of each one and pick least loaded
  - One Random Choice: Pick random resource
  - Two Random Choices: Sample 2 random resources and pick less loaded one

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins.

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins. Assume labels are chosen independently at random.
- Ball is placed in the least full bin among the d choices.

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins. Assume labels are chosen independently at random.
- Ball is placed in the least full bin among the d choices. Ties broken arbitrarily.
- (Amazingly) we have W.h.p.:

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins. Assume labels are chosen independently at random.
- Ball is placed in the least full bin among the d choices. Ties broken arbitrarily.
- (Amazingly) we have W.h.p.:
  - MAX LOAD $= \ln \ln n / \ln 2 + O(1)$

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins. Assume labels are chosen independently at random.
- Ball is placed in the least full bin among the d choices. Ties broken arbitrarily.
- (Amazingly) we have W.h.p.:
  - MAX LOAD $= \ln \ln n / \ln 2 + O(1)$
  - Down from $\Theta(\ln n / \ln \ln n)$ for $d = 1$

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins. Assume labels are chosen independently at random.
- Ball is placed in the least full bin among the d choices. Ties broken arbitrarily.
- (Amazingly) we have W.h.p.:
  - MAX LOAD $= \ln\ln n / \ln 2 + O(1)$
  - Down from $\Theta(\ln n / \ln\ln n)$ for $d = 1$
  - In general, when d 2,

Power of Two Choices

- Each ball comes with $d = 2$ labels, and can be placed in one of d possible bins. Assume labels are chosen independently at random.
- Ball is placed in the least full bin among the d choices. Ties broken arbitrarily.
- (Amazingly) we have W.h.p.:
  - MAX LOAD $= \ln \ln n / \ln 2 + O(1)$
  - Down from $\Theta(\ln n / \ln \ln n)$ for $d = 1$
  - In general, when d 2, MAX LOAD $= \ln \ln n / \ln d + \Theta(1)$