CS 473 ✧ Fall 2024

# Practice Final Exam

**December 10, 2024**

| | |
|---|---|
| Name: | Jeff Erickson |
| NetID: | jeffe |

- ***Don't panic!***

- You have 180 minutes to answer six questions. The questions are described in more detail in a separate handout.

- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.

- A page of potentially useful definitions and formulas appears on the back of this cover sheet.

- Please clearly print your name and your NetID in the boxes above.

- Please also print your name at the top of every page of the answer booklet, except this cover page. We want to make sure that if a staple falls out, we can reassemble your answer booklet. (It doesn't happen often, but it does happen.)

- **Do not write outside the black boxes on each page**. These indicate the area of the page that our scanners will actually scan. If the scanners can't see your work, we can't grade it.

- The answer booklet includes several blank pages for overflow/scratch work. If you run out of space for an answer, please use the overflow pages at the back of the answer booklet, but **please clearly indicate where we should look**. If we can't find your work, we can't grade it.

- **Only work that is written into the stapled answer booklet will be graded.** We will provide scratch paper to anyone who asks, but we will not grade anything written on that scratch paper. Similarly, we will not grade any overflow pages that you separate from the answer booklet.

- Please return ***all*** paper with your answer booklet: your question sheet, your cheat sheet, and all scratch paper. **Please put all loose paper *inside* your answer booklet.**

- **The World's Most Useful Inequality:** $1 + x \le e^x$ for all $x$
- **The World's Most Useful Limit:** $\lim_{n\to\infty} \left(1+\frac{1}{n}\right)^n = e$

## Some Useful Probability Bounds

Suppose $X$ is the sum of random indicator variables $X_1, X_2, \ldots, X_n$.
For each index $i$, let $p_i = \Pr[X_i = 1] = \mathrm{E}[X_i]$, and let $\underline{\mu = \sum_i p_i = \mathrm{E}[X]}$.

- **Markov's Inequality:**

$$\boxed{\Pr[X \ge x] \le \frac{\mu}{x}} \qquad \text{for all } x > 0\text{, and therefore...}$$

$$\Pr[X \ge (1+\delta)\mu] \le \frac{1}{1+\delta} \qquad \text{for all } \delta > 0$$

For example: $P[X \ge 3\mu] \le \frac{1}{3}$

- **Chebyshev's Inequality:** If the variables $X_i$ are pairwise independent, then...

$$\Pr[(X-\mu)^2 \ge z] < \frac{\mu}{z} \qquad \text{for all } z > 0\text{, and therefore...}$$

$$\Pr[X \ge (1+\delta)\mu] < \frac{1}{\delta^2\mu} \qquad \text{for all } \delta > 0$$

For example: $P[X \ge 3\mu] \le \frac{1}{4\mu}$

- **Chernoff's Inequality:** If the variables $X_i$ are fully independent, then...

$$\Pr[X \ge x] \le e^{x-\mu}\left(\frac{\mu}{x}\right)^x \qquad \text{for all } x \ge \mu\text{, and therefore...}$$

$$\Pr[X \ge (1+\delta)\mu] \le e^{-\delta\mu/2} \qquad \text{for all } \delta \ge 2$$

For example: $P[X \ge 3\mu] \le e^{-\mu}$

## Hashing Properties

$\mathcal{H}$ is a set of functions from some universe $\mathcal{U}$ to $[m] = \{0, 1, 2, \ldots, m-1\}$.

- **Universal:** $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \le \frac{1}{m}$ for all distinct items $x \ne y$
- **2-uniform:** $\Pr_{h\in\mathcal{H}}[h(x) = i \text{ and } h(y) = j] = \frac{1}{m^2}$ for all distinct $x \ne y$ and all $i$ and $j$
- **Strongly universal:** Same as 2-uniform.
- **Ideal Random:** Useful fiction.

## Flow Algorithms

- Given a flow network with integer capacities, Ford-Fulkerson computes an integer maximum flow in $O(E \cdot |f^*|)$ time, where $|f^*|$ is the maximum flow value.
- Given a flow network with real capacities, Orlin's algorithm computes a maximum flow in $O(VE)$ time. If all edge capacities are integers, the algorithm computes an integer maximum flow.
- Any flow can be decomposed into a weighted sum of paths and cycles in $O(VE)$ time.

1. Suppose we are given a set of $n$ rectangular boxes, each specified by their height, width, and depth in centimeters. All three dimensions of each box lie strictly between 10cm and 20cm, and all $3n$ dimensions are distinct. As you might expect, one box can be nested inside another if the first box can be rotated so that is is smaller in every dimension than the second box. Boxes can be nested recursively, but two boxes cannot be nested side-by-side inside a third box. A box is *visible* if it is not nested inside another box.

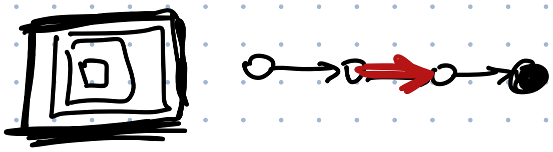
   Describe and analyze an algorithm to nest the boxes, so that the number of visible boxes is as small as possible.

graph DP/Flow?

Define $G=(V,E)$ — dag!

$n \rightarrow V =$ boxes

$O(n^2) \rightarrow E = \{u \rightarrow v \mid$ box $u$ fits inside box $v\}$



Path in $G \Longleftrightarrow$ nested seq. of boxes

Visible box $\Longleftrightarrow$ last vertex in path

Find paths in $G$

- every node in 1 path → disjoint!
- #paths is as small as possible

Assign next-outer boxes to as many boxes as possible

Match

Build $G' = (L \cup R, E')$ bipartite

2n verts → $L =$ boxes

$R =$ boxes

$O(n^2)$ edges — $E' = \{l_i r_j \mid$ box $i$ fits in box $j\}$

Maximum matching in $G' \Longleftrightarrow$ optimal box nesting

↓

$O(VE)$ time [textbook]

$= \boxed{O(n^3)}$

2. Consider the following randomized version of mergesort. The input is an unsorted array $A[1..n]$ of distinct numbers. The MERGE subroutine takes two sorted arrays as input and returns a single sorted array, containing the elements of both input arrays, in linear time.

```
RANDOMIZEDMERGESORT($A[1..n]$):
  if $n \le 1$
      return $A$
  $\ell \leftarrow 0$;  $r \leftarrow 0$
  for $i \leftarrow 1$ to $n$
      with probability 1/2
          $\ell \leftarrow \ell + 1$
          $L[\ell] \leftarrow A[i]$
      else
          $r \leftarrow r + 1$
          $R[r] \leftarrow A[i]$
  $L \leftarrow$ RANDOMIZEDMERGESORT($L[1..\ell]$)
  $R \leftarrow$ RANDOMIZEDMERGESORT($R[1..r]$)
  return MERGE($L, R$)
```

(a) What is the probability that two distinct items $A[i]$ and $A[j]$ are both copied into $L$?

(b) What is the probability that two distinct items $A[i]$ and $A[j]$ appear in the same subproblem for more than $k$ levels of recursion?

(c) What is the expected number of pairs of items that appear in the same subproblem for more than $k$ levels of recursion?

(d) Upper-bound the probability that ~~no pairs~~ at least one pair of items appear in the same subproblem for more than $k$ levels of recursion. Equivalently, upper bound the probability that the recursion tree of RANDOMIZEDMERGESORT has depth at most $k$.

(e) Prove that RANDOMIZEDMERGESORT runs in $O(n \log n)$ time with high probability.

(a) $1/4$

(b) $1/2^k$

(c) $E[\underline{\#\text{pairs} \geq k \text{ levels}}]$

$= \sum_{(i,j)} \Pr[i,j \geq k \text{ levels}]$

$= \binom{n}{2}/2^k$

See p. 9

(d) $\Pr[\underline{\#\text{pairs} \geq k \text{ levels}} \geq 1]$

$\leq \binom{n}{2}/2^k$ by Markov

(e) We need to show depth of recursion tree in $O(\log n)$ whp.

↳ $\Pr \geq 1 - 1/n$

If we set $k = c \cdot \log n$

$\Pr(\text{Fail}] \leq \binom{n}{2}/2^{c \log n} = \binom{n}{2}/n^c$

$= O(n^2)/n^3 = 1/n$

$c=3$

Prob 2 cont'd

Let $k = 3 \lg n$

$\Pr[\text{depth of rec tree} > k] \leq \binom{n}{2} / 2^k$ by (c)

$= \binom{n}{2} / n^3$ by def. k

$\leq 1/2n$ math

So $\Pr[\text{depth} \leq k] \geq 1 - 1/2n$

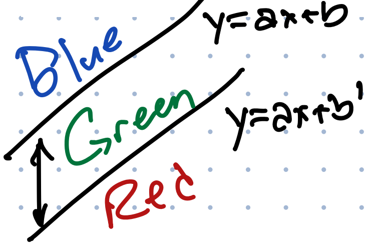
So whp rec tree has depth $\leq k = 3 \lg n = O(\lg n)$ □

3. Suppose we are given a set $R$ of $n$ *red* points, a set $G$ of $n$ *green* points, and a set $B$ of $n$ *blue* points; each point is given as a pair $(x, y)$ of real numbers. We call these sets *separable* if there is a pair of parallel lines $y = ax + b$ and $y = ax + b'$ such that (1) all red points are below both lines, (2) all blue points are above both lines, and (3) all green points are between the lines.

   (a) ✓ Describe a linear program that is feasible if and only if the point sets $G, B, R$ are separable.

   (b) Describe a linear program whose solution describes a pair of parallel lines that separates $G, B, R$ whose *vertical* distance is as small as possible. (Here you can assume that $G, B, R$ are separable.) ↳ $\boxed{b-b'}$

*[Hint: Don't try to solve these problems; just describe the linear programs.]*



$(x_i, y_i)$ = coords of $i$th point

ⓐ variables: $a, b, b'$

ⓑ ~~maximize whatever~~ $\boxed{\min\ b-b'}$

s.t. ~~$y_i \le ax_i + b$~~
$y_i \le ax_i + b'$ } for all red $(x_i, y_i)$

$y_i \ge ax_i + b$
~~$y_i \ge ax_i + b'$~~ } for all blue $(x_i, y_i)$

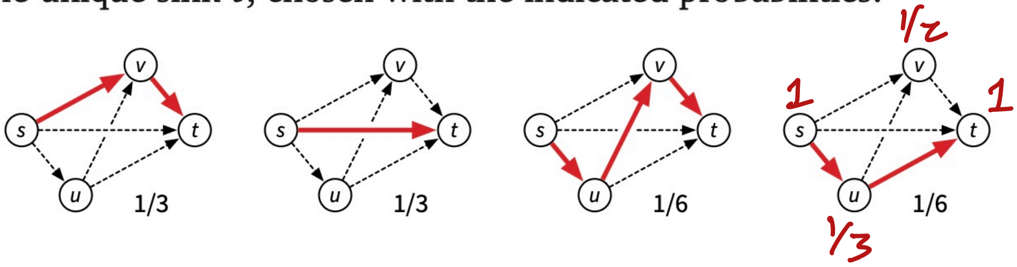
$y_i \le ax_i + b$
$y_i \ge ax_i + b'$ } for all green $(x_i, y_i)$

$b' \le b \Longleftarrow$

top sort! DP / ~~Flow~~?

4. Let $G = (V, E)$ be an arbitrary dag with a unique source $s$ and a unique sink $t$. Suppose we compute a random walk from $s$ to $t$, where at each node $v$ (except $t$), we choose an outgoing edge $v \to w$ uniformly at random to determine the successor of $v$.

For example, in the following four-node graph, there are four walks from the unique source $s$ to the unique sink $t$, chosen with the indicated probabilities:



(a) Describe and analyze an algorithm to compute, for every vertex $v$, the probability that the random walk visits $v$. For example, in the graph shown above, a random walk visits the source $s$ with probability 1, the bottom vertex $u$ with probability $1/3$, the top vertex $v$ with probability $1/2$, and the sink $t$ with probability 1.

(b) Describe and analyze an algorithm to compute the expected number of edges in the random walk. For example, given the graph shown above, your algorithm should return the number $1 \cdot 1/3 + 2 \cdot (1/2 + 1/6) + 3 \cdot 1/6 = 11/6$.

Assume all relevant arithmetic operations can be performed exactly in $O(1)$ time.

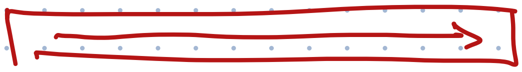
(a) Let Prob(v) = probability that random walk visits v

$$\text{Prob}(v) = \begin{cases} 1 & v = s \\ \sum_{u \to v} \text{Prob}(u) \frac{1}{\text{outdeg}(u)} & \text{for all } v \neq s \end{cases}$$

Memoize into G itself
Store Prob(v) at v.prob

eval in top. order

Index nodes in top. order
Store Prob(v) at Prob[v.index]



Time = $\sum_v O(\text{indeg}(v)) = \boxed{O(E)}$

(b) $E[\#\text{edges in walk}] = E[\#\text{verts in walk}] - 1$
$= \sum_v \Pr[v \text{ in walk}] - 1 = \sum_v \text{Prob}(v) - 1$

Part (a)
+ O(V) time
= O(E) time

5. You are managing a company with a large number of project teams. Each project team contains exactly three people. A single employee could belong to any number of project teams, but all of the project teams are different.

You need to organize a meeting to make a major announcement. Because this announcement will affect every project team, you need to ensure that at least one member of every project team attends the meeting. On the other hand, there isn't enough space to invite every employee. Moreover, you expect the announcement to be unpopular, so you want to avoid inviting employees who are likely to cause trouble.

For each employee $x$, you know a positive real number $\mathit{trouble}(x)$, which roughly estimates the probability that $x$ will cause trouble. Your task is to choose a subset $X$ of the employees that includes at least one member of each project team, such that $\sum_{x \in X} \mathit{trouble}(x)$ is as small as possible.

(a) Write a linear programming relaxation for this problem.

(b) Describe an efficient **3-*approximation*** algorithm for this problem. *[Hint: Round the solution to your linear program from part (a).]*

A 3-approximation algorithm that is correct when $\mathit{trouble}(x) = 1$ for every employee $x$ is worth half credit.

Variable $x_i$ = "employee $i$ is invited" → 1

0 otherwise

ILP: $x_i \in \{0,1\}$

relax!

(a)

given

$$\boxed{\begin{aligned} &\min \sum_{i=1}^{n} x_i \cdot \mathit{trouble}(i) \\ &x_i + x_j + x_k \geq 1 \quad \text{for every team } \{i,j,k\} \\ &0 \leq x_i \leq 1 \quad \text{for all } i \end{aligned}}$$

(b) Let $x^*$ be fractional solution to LP

$$OPT^* = \sum_i x_i^* \cdot \mathit{trouble}(i)$$

Let OPT be actual integer optimum score

Then $OPT^* \leq OPT$

Define
$$x_i' = \begin{cases} 0 & \text{if } x_i^* < 1/3 \\ 1 & \text{if } x_i^* \geq 1/3 \end{cases}$$

1 round

See page 8

$\boxed{\text{Problem 5 continued}}$

for each team $\{i,j,k\}$

$$x_i^* + x_j^* + x_k^* \geq 1 \quad \leftarrow \quad x^* \text{ is feasible}$$

② feasible

$$\Downarrow$$

$$\max\{\underline{\quad}^*\} \geq 1/3$$

$$\Downarrow$$

$$\max\{\underline{\quad}'\} \geq \cancel{1/3}\ 1$$

$$\Downarrow$$

$$x_i' + x_j' + x_k' \geq 1 \quad \longrightarrow \quad x' \text{ is feasible}$$

③ approx ratio

$$x_i' \leq 3\, x_i^*$$

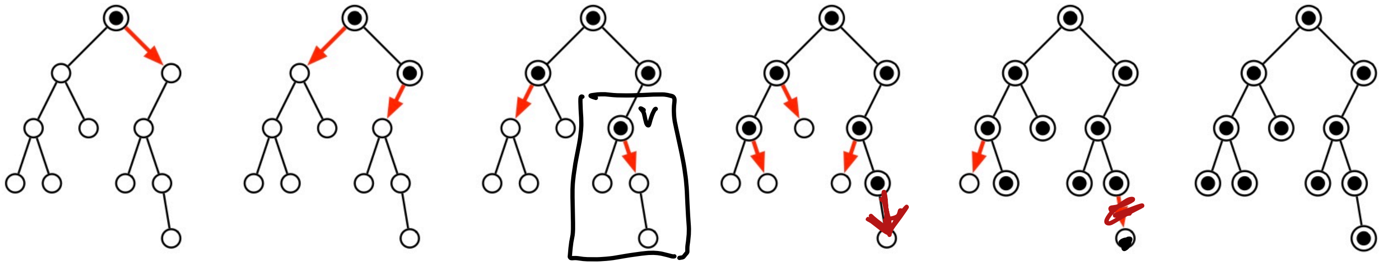
$$\Rightarrow \sum_i x_i' \,\text{trouble}(i) \leq 3 \cdot \sum_i x_i^* \,\text{trouble}(i)$$

$$= 3 \cdot OPT^* \leq 3 \cdot OPT$$

Recursion / DP

6. Suppose we need to distribute a message to all the nodes in a given binary tree. Initially, only the root node knows the message. In a single round, each node that knows the message is allowed (but not required) to forward it to at most one of its children. Describe and analyze an algorithm to compute the minimum number of rounds required for the message to be delivered to all nodes in the tree.

For example, given the tree below as input, your algorithm should return the integer 5.

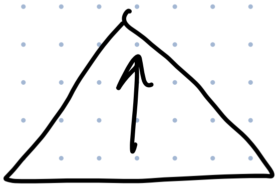


A message being distributed through a binary tree in five rounds.

Define MinRds(v) = min # rounds to distribute message at v to all v's descendants

$$\text{MinRds}(v) = \begin{cases} 0 & \text{v is leaf} \\ 1 + \text{MinRds}(v.\text{left}) & \text{only left child} \\ 1 + \text{MinRds}(v.\text{right}) & \text{only rt child} \\ \min \begin{cases} \max\{1 + \text{MinRds}(v.\text{left}), 2 + \text{MinRds}(v.\text{right})\} \\ \max\{2 + \text{MinRds}(v.\text{left}), 1 + \text{MinRds}(v.\text{right})\} \end{cases} \end{cases}$$

Memoize MinRds(v) into v.MinRds



Eval in postorder

in $\boxed{O(n) \text{ time}}$

$O(V)$