Final next Tuesday Dec 13 in this room 7-10pm

6 questions 3 hours HW0-9

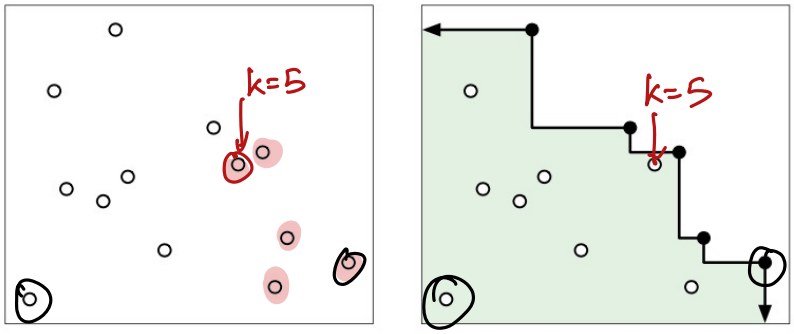
no question 3

Two cheat sheets — HAND WRITTEN

ICES forms due this Thursday, Dec 8.

— TA forms!!

1. ***[Spring 2020]*** Let S be an arbitrary set of n points in the plane with distinct x- and y-coordinates. A point p in S is ***Pareto-optimal*** if no other point in S is both above and to the right of p. The ***staircase*** of S is the set of all points in the plane (not just in S) that have at least one point in S both above and to the right. All Pareto-optimal points lie on the boundary of the staircase.



A set of points in the plane and its staircase (shaded), with Pareto-optimal points in black.

(a) Describe and analyze an algorithm that computes the number of Pareto-optimal points in S in $O(n \log n)$ time. For example, given the points on the left in the figure above, your algorithm should return the number 5.

(b) Suppose each point in S is chosen independently and uniformly at random from the unit square $[0,1] \times [0,1]$. What is the ***exact*** expected number of Pareto-optimal points in S? *[Hint: What is the probability that the leftmost point in S is Pareto-optimal?]*

(a) Sort points by x-coordinate

scan ←

record largest y-coord so far

```
Sort(S, x)
ymax ← -∞    count ← 0
for i ← n down to 1
    if S[i].y > ymax
        count ← count + 1
        ymax ← S[i].y
return count
```

$O(n \log n)$

(b) Hint: $\Pr[\text{leftmost is P.O.}] = 1/n$

$\Pr[k^{th} \text{ pt from right}] = 1/k$

$$E[\#PO] = \sum_{k=1}^{n} \Pr[k^{th} \text{ from right is PO}] = \sum_{k=1}^{n} 1/k = H_n$$

2. ***[Spring 2016]*** Your eight-year-old cousin Elmo decides to teach his favorite new card game to his baby sister Daisy. At the beginning of the game, n cards are dealt face up in a long row. Each card is worth some number of points, which may be positive, negative, or zero. Then Elmo and Daisy take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, each player can decide which of the two cards to take. When the game ends, the player that has collected the most points wins.

   Daisy isn't old enough to get this whole "strategy" thing; she's just happy to play with her big brother. When it's her turn, she takes the either leftmost card or the rightmost card, each with probability 1/2.

   Elmo, on the other hand, *really* wants to win. Having never taken an algorithms class, he follows the obvious greedy strategy—when it's his turn, Elmo *always* takes the card with the higher point value.

   Describe and analyze an algorithm to determine Elmo's expected score, given the initial sequence of n cards as input. Assume Elmo moves first, and that no two cards have the same value.

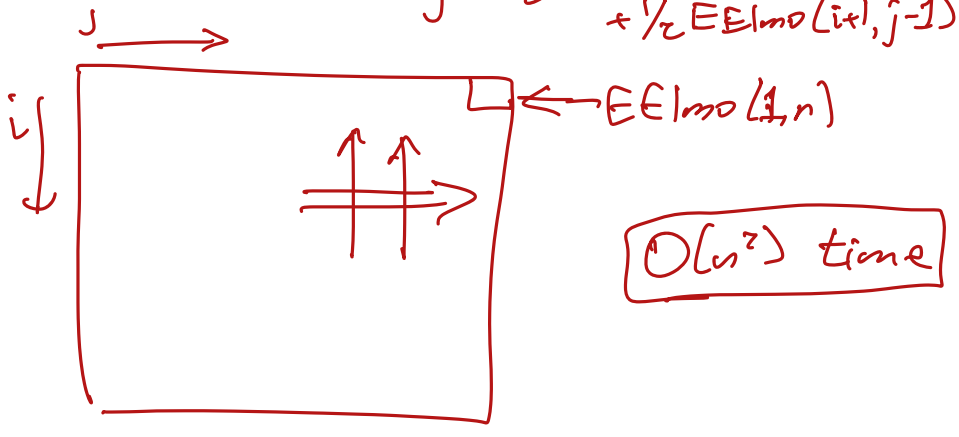
   For example, suppose the initial cards have values $1, 4, 8, 2$. Elmo takes the 2, because it's larger than 1. Then Daisy takes either 1 or 8 with equal probability. If Daisy takes the 1, then Elmo takes the 8; if Daisy takes the 8, then Elmo takes the 4. Thus, Elmo's expected score is $2 + (8 + 4)/2 = 8$.

DP

A[1..n] cards input

EElmo(i,j) = Elmo's expected score from A[i..j]

$$EElmo(i,j) = \begin{cases} 0 & \text{if } i > j \\ A[i] & \text{if } i = j \\ A[i] + \frac{1}{2}EElmo(i+2,j) + \frac{1}{2}EElmo(i+1,j-1) & \text{if } A[i] > A[j] \\ A[j] + \frac{1}{2}EElmo(i,j-2) + \frac{1}{2}EElmo(i+1,j-1) & \text{if } A[i] < A[j] \end{cases}$$



O(n²) time

3. ***[Spring 2016]*** 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.

Define graph $G=(V,E)$ $V=$ boxes

$u \to v$ means u fits inside v.

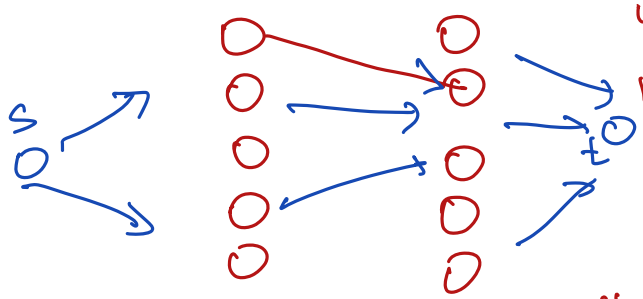
build in $O(n^2)$ time

nested boxes = path

This is a dag!

We want min # vertex disjoint paths in G that cover all vertices.

Matching $H=(V \cup V', E')$



$uv \Leftrightarrow u$ inside v

We're trying to assign each box to a larger box

unique

visible box = # unmatched vertices on left

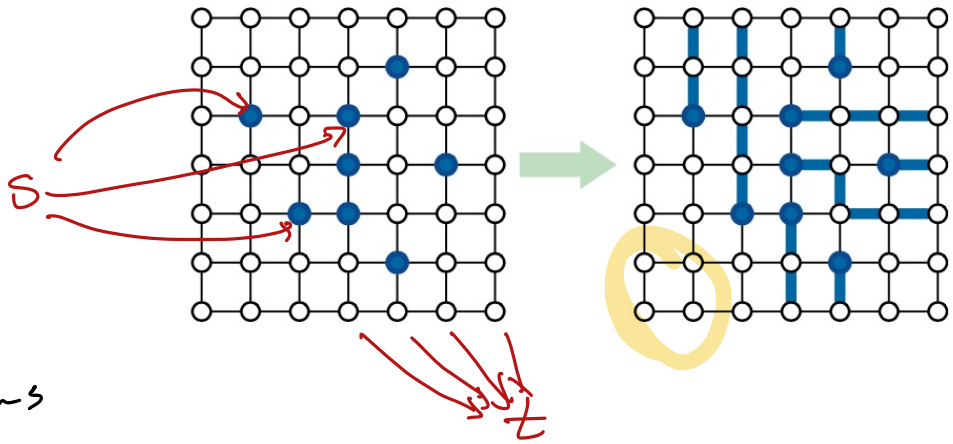
Max matching $O(VE) = O(n^3)$

$2n$ $\leq n^2$

4. ***[Spring 2016, Spring 2020]*** An $n \times n$ grid is an undirected graph with n^2 vertices organized into n rows and n columns. Every vertex is connected to the nearest vertex (if any) above, below, to the right, and to the left.

   Suppose m distinct vertices in the $n \times n$ grid are marked as *terminals*. The ***escape problem*** asks whether there are m vertex-disjoint paths in the grid that connect the terminals to any m distinct boundary vertices. Describe and analyze an efficient algorithm to solve the escape problem.

   For example, given the input on the left below, your algorithm should return TRUE.



Flows

Add source s edges $s \to v$ for every terminal v

target t edges $w \to t$ for every boundary w

replace ○ → ⇆

Give every vertex capacity 1

Max flow in $O(VE) = O(n^4)$ time

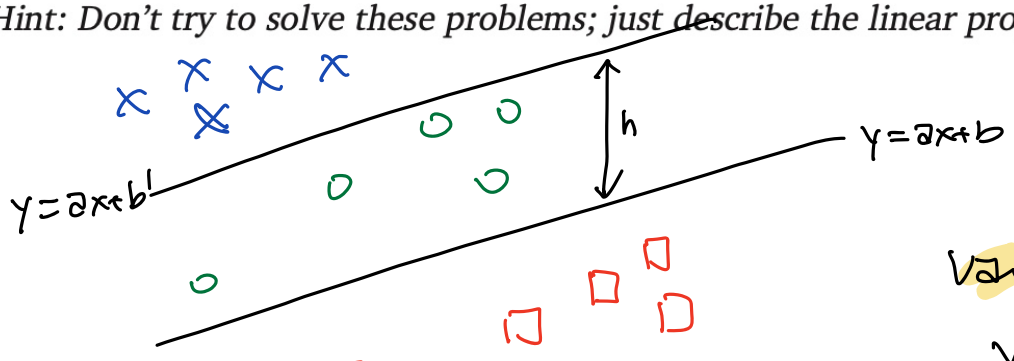
return True iff max flow value = m

5. 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.)

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



vars: a, b, b'

$y = ax + b$
$y = ax + b'$

$\min b' - b$ s.t

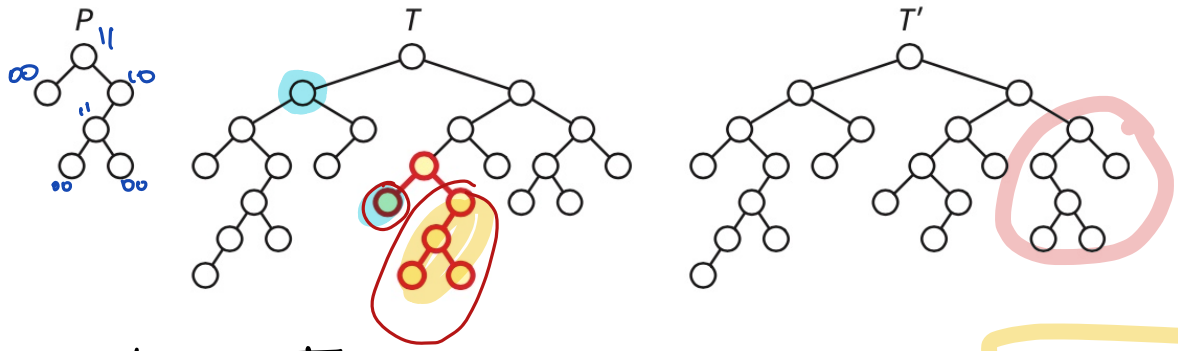
~~$y_i \geq a x_i + b$~~
$y_i \geq a x_i + b'$ } for every blue (x_i, y_i)
$y_i \geq a x_i + b$
$y_i \leq a x_i + b'$ } for every green (x_i, y_i)
$y_i \leq a x_i + b$
~~$y_i \leq a x_i + b'$~~ } for every red (x_i, y_i)

~~$b' \geq b$~~

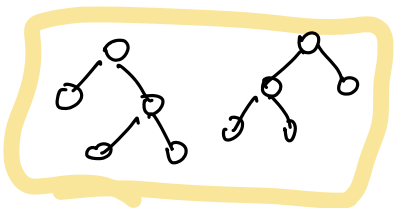
6. Suppose we are given two binary trees—a smaller *pattern* tree P with m vertices, and a larger *text* tree T with n vertices. Describe and analyze an algorithm to determine whether T contains a rooted subtree (a vertex and all of its descendants) that is identical to P.

   There is no actual data stored in the vertices of P and T; these are not binary search trees or heaps. We are only trying to match the *shape* of the trees.

   For example, in the figure below, the middle tree T contains a rooted subtree identical to P, but the right tree T' does not.



For every node v in T
  if subtree@v = P ← O(m) time
    return True
return False



O(mn) time

Transform P and T into strings → ~~BFS~~ DFS?

preorder P → 3 0 2 3 0 0

T → 3 3 3 0 2 3 2 0 0 2 0 3 3 3 0 2 3 0 0 0 3 3 0 0 0

P → (( ) , ((( ) , ( )) ,))

O(n+m)