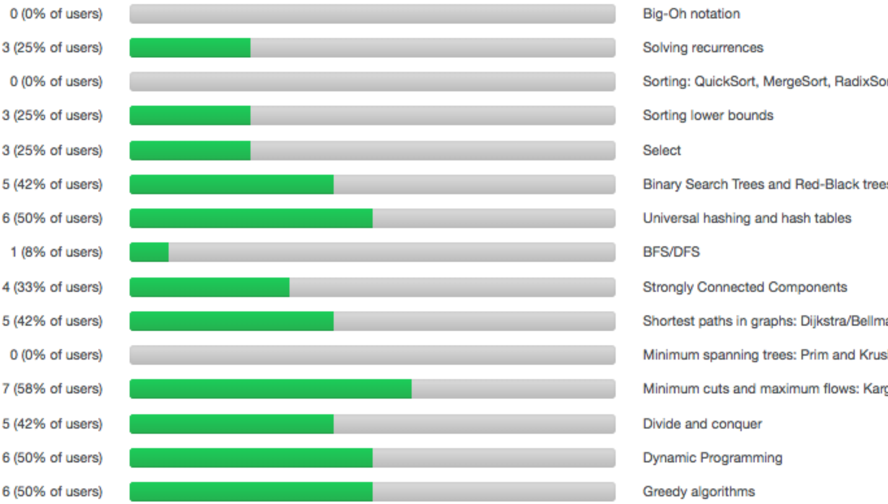
# CS161 Review Session Practice Problems

12/6/2017

# Poll results

As of 1am this morning...



# Agenda

- I have a bunch of practice problems.
- Y'all vote on topics and we'll do them.
- I can also answer particular questions about the material.
- Topics I have problems for:
  - Grab-bag (multiple choice, etc)
  - Hashing
  - Red-Black Trees
  - Ford-Fulkerson
  - Dynamic Programming
  - Greedy algorithms
  - Divide and conquer
  - Randomized algs

# Multiple choice warmup!

For each of the following quantities, **identify all of the options** that correctly describe the quantity.

(a) The function $f(n)$, where $f(n) = n \log(n)$.

(b) $T(n)$ given by $T(n) = T(n/4) + \Theta(n^2)$ with $T(n) = 1$ for all $n \leq 8$.

(c) $T(n)$ which is the running time of the following algorithm:

```
mysteryAlg( n ):
    if n < 3:
        return 1
    return mysteryAlg( n/2 ) + mysteryAlg( (n/2) + 1 )
```

where above all division is integer division (so a/b means $\lfloor a/b \rfloor$).

$(A)$ $O(n^2)$ $(B)$ $\Theta(n^2)$ $(C)$ $\Omega(n)$ $(D)$ $O(n)$ $(E)$ $O(\log^2(n))$.

# Prove or give a counter-example

Let $G = (V, E)$ be an undirected weighted graph, and let $T$ be a minimum spanning tree in $G$. Decide whether the following statements **must be true** or **may be false**, and prove it!

(a) For any pair of distinct vertices $s, t \in V$, there is a unique path from $s$ to $t$ in $T$.

**True** **False**

(b) For any pair of distinct vertices $s, t \in V$, the cost of a path between $s$ and $t$ in $T$ is minimal among all paths from $s$ to $t$ in $G$.

**True** **False**

# Hashing warm-up

Let $\mathcal{U}$ be a universe of size $m$, where $m$ is a prime, and consider the following two hash families which hash $\mathcal{U}$ into $n$ buckets, where $n$ is much smaller than $m$.

- First, consider $\mathcal{H}_1$, which is the set of all functions from $\mathcal{U}$ to $\{1, \ldots, n\}$:

$$\mathcal{H}_1 = \{h \mid h : \mathcal{U} \to \{1, \ldots, n\}\}$$

- Second, let $p = m$ (so $p$ is prime since we assumed $m$ to be prime), and choose $\mathcal{H}_2$ to be

$$\mathcal{H}_2 = \{h_{a,b} \mid a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\}\},$$

  where $h_{a,b}(x) = (ax + b \mod p) \mod n$.
- You want to implement a hash table using one of these two families. Why would you choose $\mathcal{H}_2$ over $\mathcal{H}_1$? **Choose the best answer.**

(A) $\mathcal{H}_1$ isn't a universal hash family.

(B) Storing an element of $\mathcal{H}_1$ takes a lot of space.

(C) Storing all of $\mathcal{H}_1$ takes a lot of space.

# Shortest Paths

- When might you prefer breadth-first search to Dijkstra's algorithm?
- When might you prefer Floyd-Warshall to Bellman-Ford?
- When might you prefer Bellman-Ford to Dijkstra's algorithm?

# Randomized algorithms

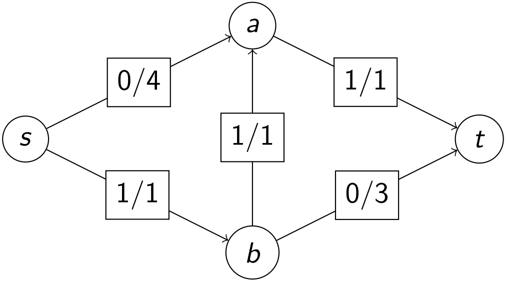
Suppose that $b_1, \ldots, b_n$ are $n$ distinct integers in a **uniformly random order**. Consider the following algorithm:

```
findMax(b_1,...,b_n):
    currentMax = -Infinity
    for i = 1,...,n:
        if b_i > currentMax:
            currentMax = b_i
    return currentMax
```

What is the expected number of times that `currentMax` is updated? (Asymptotic notation is fine).

# Min-cut/Max-flow

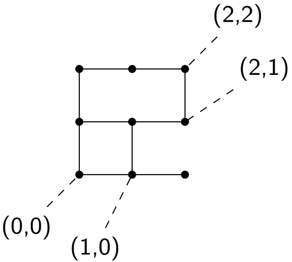
Consider the following flow on a graph. The notation $x/y$ means that an edge has flow $x$ out of capacity $y$.




- Draw the residual graph for this flow.
- Find an augmenting path in the residual graph and use it to increase the flow.
- Find a minimum cut and prove (not by exhaustion) that it is a minimum cut.

# Dynamic Programming!

- Suppose that roads in a city are laid out in an $n \times n$ grid, but some of the roads are obstructed.
- For example, for $n = 3$, the city may look like this:




where we have only drawn the roads that are not blocked. You want to count the number of ways to get from $(0, 0)$ to $(n-1, n-1)$, using paths that only go up and to the right. In the example above, the number of paths is 3.

- Design a DP algorithm to solve this problem.

# Divide and Conquer!

- Given an array $A$ of length $n$, we say that an array $B$ is a *circular shift* of $A$ if there is an integer $k$ between 1 and $n$ (inclusive) so that

$$B = A[k : n] + A[1 : k],$$

  where $+$ denotes concatenation.
- For example, if $A = [2, 5, 6, 8, 9]$, then $B = [6, 8, 9, 2, 5]$ is a circular shift of $A$ (with $k = 2$). The sorted array $A$ itself is also a circular shift of $A$ (with $k = 1$).
- Design a $O(\log(n))$-time algorithm that takes as input an array $B$ which is a circular shift of a sorted array which contains distinct positive integers, and returns the value of the largest element in $B$. For example, give $B$ as above, your algorithm should return 9.

# Greedy Algorithms!

There are $n$ final exams on Dec. 13 at Stanford; exam $i$ is scheduled to begin at time $a_i$ and end at time $b_i$. Two exams which overlap cannot be administered in the same classroom; two exams $i$ and $j$ are defined to be *overlapping* if $[a_i, b_i] \cap [a_j, b_j] \neq \emptyset$ (including if $b_i = a_j$, so one starts exactly at the time that the other ends). Design an algorithm which solves the following problem.

- **Input:** Arrays $A$ and $B$ of length $n$ so that $A[i] = a_i$ and $B[i] = b_i$.
- **Output:** The smallest number of classrooms necessary to schedule all of the exams, and an optimal assignment of exams to classrooms.
- **Running time:** $O(n \log(n) + nk)$, where $k$ is the minimum number of classrooms needed.
- **For example:** Suppose there are three exams, with start and finish times as given below:

| i | 1 | 2 | 3 |
|---|---|---|---|
| $a_i$ | 12pm | 4pm | 2pm |
| $b_i$ | 3pm | 6pm | 5pm |

Then the exams can be scheduled in two rooms; Exam 1 and Exam 2 can be scheduled in Room 1 and Exam 3 can be scheduled in Room 2.

# Universal Hash Families

- Definition: A hash family $\mathcal{H}$ (mapping $\mathcal{U}$ into $n$ buckets) is **2-universal** if for all $x \neq y \in \mathcal{U}$ and for all $a, b \in \{1, \ldots, n\}$,

$$\mathbb{P}((h(x), h(y)) = (a, b)) = \frac{1}{n^2}.$$

(a) Show that if $\mathcal{H}$ is 2-universal, then it is universal.

(b) Show that the converse is not true. That is, there is a universal family that's not 2-universal.

# More universal hash families

Say that $\mathcal{H}$ is a universal hash family, containing functions $h : \mathcal{U} \to \{1, \ldots, n\}$. Consider the following game.

- You choose $h \in \mathcal{H}$ uniformly at random and keep it secret.
- A bad guy chooses $x \in \mathcal{U}$, and asks you for $h(x)$. (You give it to them).
- The bad guy chooses $y \in \mathcal{U} \setminus \{x\}$, and tries to get $h(y) = h(x)$.
- If $h(x) = h(y)$, the bad guy wins. Otherwise, you win.

One of the following two is true.

1. There is a universal hash family $\mathcal{H}$ so that the bad guy wins with probability 1.
2. For any universal hash family $\mathcal{H}$, the probability that the bad guy wins is at most $1/n$.

Which is true and why?

# Red-Black Trees

Which of the following can be colored as a red-black tree? Either give a coloring or explain why not.