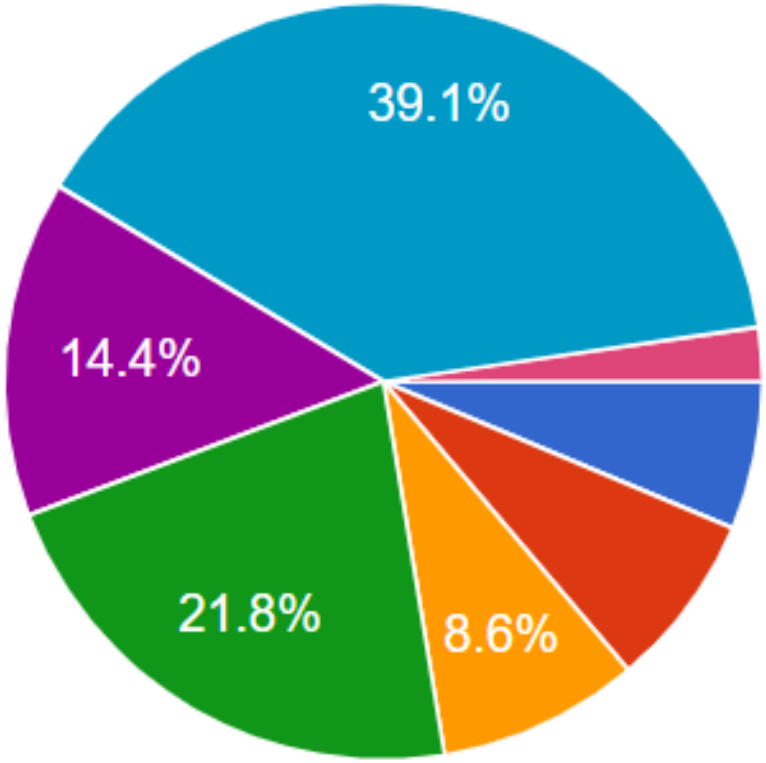
UCRPC Fall 2021
Closing and Award Ceremony

Outlines

- Statistics for participants
- Outline of solutions
- Award ceremony

Which year are you in?



- High school or younger
- Undergraduate freshman
- Undergraduate sophomore
- Undergraduate junior
- Undergraduate senior or higher (year 4 or more)
- Master student
- Ph.D. student

Number of solvers per problem

Problem		# Test cases	Points per test case	Total points	#Solvers	First Solver
A	Senseless Census	20	3	60	84/89	James Rungsawang (0:01)
B	Lost in the Shuffle	20	3	60	82/85	James Rungsawang (0:03)
C	Maths of Glory	20	3	60	77/86	James Rungsawang (0:12)
D	Stake Your Claim	20	4	80	14/26	Omer Eren (0:29)
E	Feeding Friendsy	16	5	80	2/66	Omer Eren (0:58)
F	Trip Navigator	20	5	100	7/34	Sanat Mishra (0:43)
G	Snack Attack	20	5	100	2/15	James Rungsawang (3:48)
H	Fetch Quest	15	8	120	0/15	Omer Eren (104 pts)
I	Sphere Mongers	20	6	120	0/19	Omer Eren (114 pts)
Total				780		

Senseless Census

- **Given a 2D array, count how many “t”s in the array**

- **Solution:**
  - Just count it, probably use doubly nested loops

Lost in the Shuffle

- **Given 5 dolls labeled from 1 to 5 and a list of swaps, decide the final label of doll 3**

- **Solution:**
  - Just simulate it, probably use a for loop

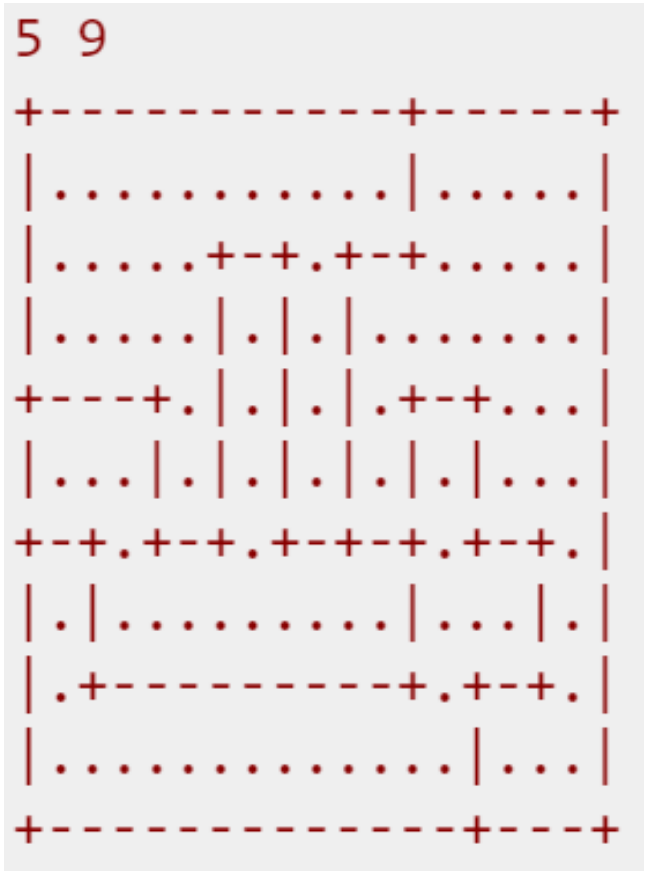
Maths of Glory

- **Given a list of quadruples (a, b, c, d), count whether the sum of $a \cdot b$ reaches n first, or the sum of $c \cdot d$ reaches n first, or at the same time**

- **Solution:**
  - Just simulate it

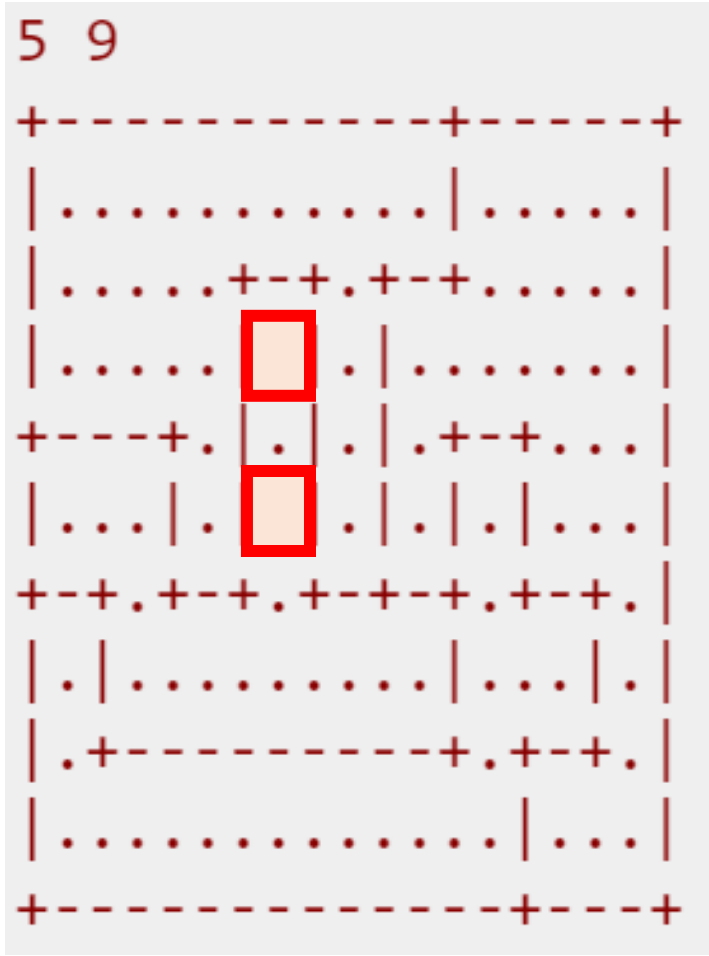
Stake Your Claim

- **Given a 2D maze with exactly four connected regions, compute the areas for the four regions**
- **Solution idea:**
  - Use the graph connectivity algorithms



Stake Your Claim

- **Technical detail 1: how to decide whether two grids are connected?**
  - Just check the wall edge based on their coordinates
- **Technical detail 2: how to implement graph connectivity? (Should be taught in CS 10C)**

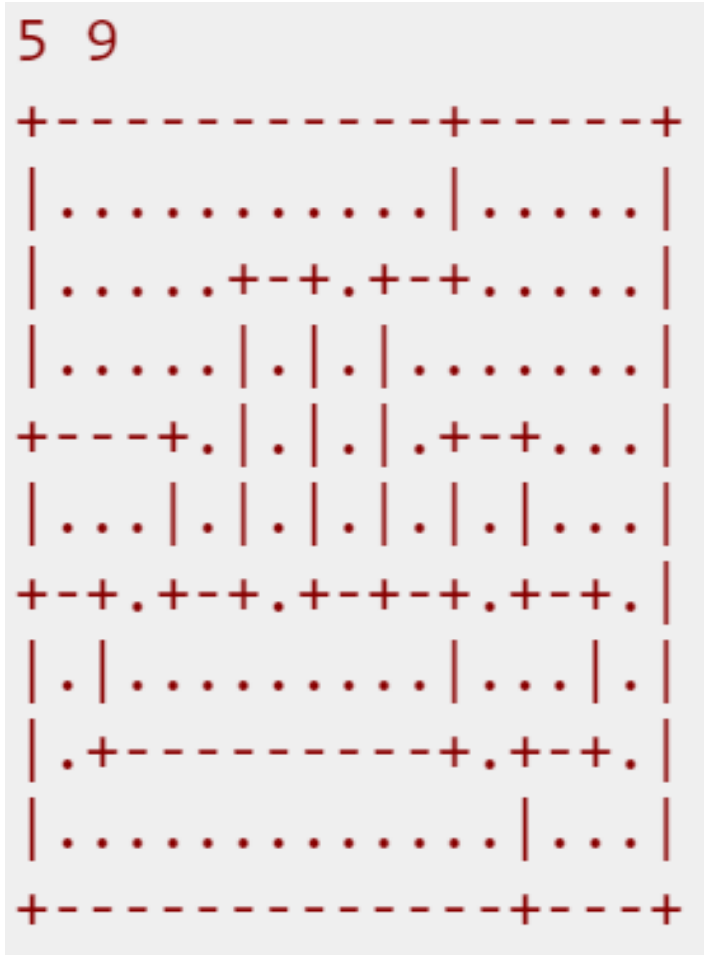


Stake Your Claim

- Floodfill (based on DFS):

```
ff(int x, int y)
    if visited[x][y] return;
    visited[x][y] = true;
    for (dx,dy) in four directions
        if connected ff(x+dx,y+dy);
```

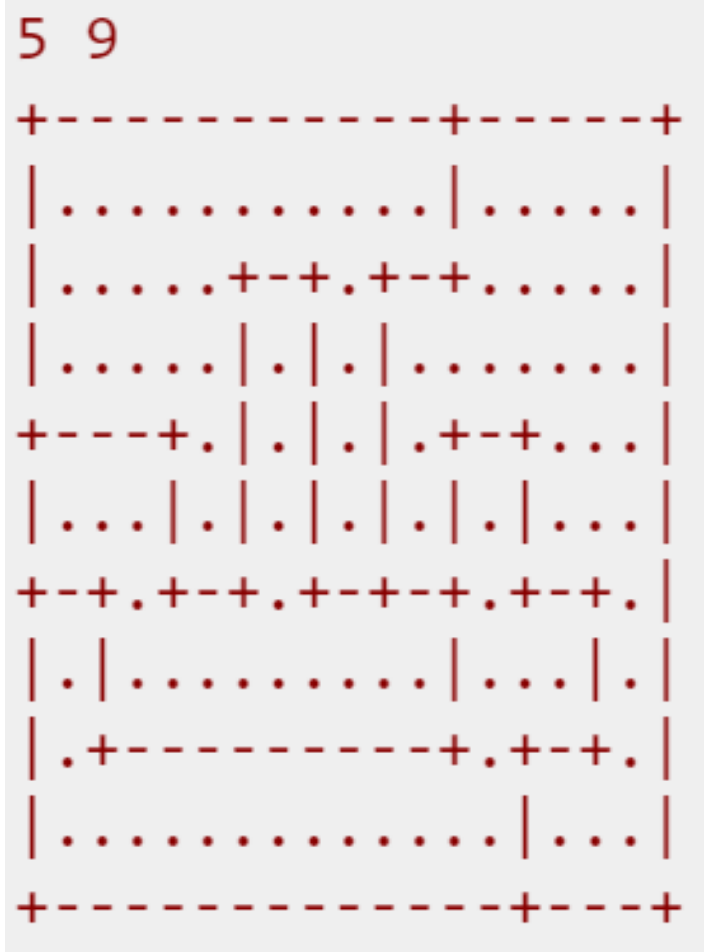
- Time bound: $O(nm)$



Stake Your Claim

- **BFS:**
  - For any unfilled cell, call BFS:
    - Initialize an empty queue and add the first cell
    - While queue is not empty
      - Pop the front of the queue, add area by 1 and add the unvisited connected neighbors to the queue
    - Return the area
  - Sort the four areas

- **Time bound:** $O(nm)$



Feeding Friendsy

- **The two Cheep Chomps open their mouths in certain range, and you can throw a ball to one of the open mouths every second. Decide the maximum points you can get.**

- **Difficulty: time range is large: 10^8, but only 10^6 intervals**

- **Solution:**
  - Only process the events (i.e., mouth open/close), and the state of the interval between two consecutive events remains the same

Trip Navigator

- **Given a 2D maze and each cell is either empty or contains a banana. Go to an empty cell takes 1 second, and a cell with banana takes an additional of t seconds. Compute the shortest path from the top-left cell to the bottom right cell**

- **Solution:**
  - Classic shortest-path problem (taught in CS 141)
  - Using the quadratic algorithm ($O(n^4)$ cost) can get about 50% points
  - Using a binary heap costs $O(n^2 \log n)$ and is sufficient to get full score
  - There exists an $O(n^2)$ algorithm and you are welcome to think about it more

Snack Attack

- **Given an n-by-n grid, and p popcorn kernels and b boulders with their arrival time and coordinates. The player walks at speed 1, and wants to catch popcorns and avoid boulders. Compute the maximum score you can get.**
- **Solution: using dynamic programming (CS 141)**
  - States: f[i,x,y], indicating the maximum score a player can get at time i, location (x,y)
  - Boundary: f[0,x,y]=-infty, f[0,s_r,s_c]=0
  - Decision:
    - Step 1: f[i,x,y]=max(f[i-1,d+dx,y+dy])
    - Step 2: if a popcorn hits this grid at this time, f[i,x,y]+=1, if a boulder hits this grid at this time, f[i,x,y]/=2

Number of solvers per problem

Problem		# Test cases	Points per test case	Total points	#Solvers	First Solver
A	Senseless Census	20	3	60	84/89	James Rungsawang (0:01)
B	Lost in the Shuffle	20	3	60	82/85	James Rungsawang (0:03)
C	Maths of Glory	20	3	60	77/86	James Rungsawang (0:12)
D	Stake Your Claim	20	4	80	14/26	Omer Eren (0:29)
E	Feeding Friendsy	16	5	80	2/66	Omer Eren (0:58)
F	Trip Navigator	20	5	100	7/34	Sanat Mishra (0:43)
G	Snack Attack	20	5	100	2/15	James Rungsawang (3:48)
H	Fetch Quest	15	8	120	0/15	Omer Eren (104 pts)
I	Sphere Mongers	20	6	120	0/19	Omer Eren (114 pts)
Total				780		

Fetch Quest

- Given a graph that each vertex represents a crystal, and each edge corresponds the time to travel between two vertices
- Your team has four players, and you want to partition all crystals into four sets, and ask each player to pick up crystals in one set
- Each play can only carry one crystal at a time
- You want to minimize the maximum time for the four players

Fetch Quest

- Step 0: compute the shortest-paths from vertex 0 to all other vertices, and put all distances in an array

- Goal: distribute the distances into four sets, so the sum of each set is minimized

Fetch Quest: greedy-based solution

- You can try a variety of greedy approaches, which is not optimal but can pass varies test cases, depending on how good your heuristic is

- Example 1 (James's solution at 1:39): sort from large to small, scan, and always put the current one to the smallest batch

- Example 2: random shuffle for 10000 rounds, and run James's heuristic

Fetch Quest: search-based solution

- **Naïve approach: try all 4 possible assignment for each crystal, and check the results (backtracking, CS 10B)**
  - Cost: $O(4^n \cdot n)$
  - Expected score: 30-50 points

- **Can try to prune the search**
  - Boundaries, getting an initial solution as an upper bound, evaluating while searching, avoiding symmetric cases
  - Expected score: 50-112 points

Fetch Quest: dynamic programming

- **The 0/1 Knapsack problem (CS141): given a set of integers, find if a subset can have a given sum**
- **How to extend it to fetch quest?**
  - Add dimension! Add dimension! Add dimension!
- **States: f[i,x,y,z] be if we can choose from the first i elements that has sum x for subset 1, sum y for subset 2, sum z for subset 3**
  - How about the last subset? It has the sum sum(1:n)-x-y-z

Fetch Quest: dynamic programming

- The 0/1 Knapsack problem (CS141): given a set of integers, find if a subset can have a given sum

- States: f[i,x,y,z] be if we can choose from the first i elements that has sum x for subset 1, sum y for subset 2, sum z for subset 3
- Boundary: f[i,0,0,0]=true, false otherwise
- Decision: f[i,x,y,z]=f[i-1,x-w[i],y,z] or f[i,x,y-w[i],z] or f[i,x,y,z[i]]

- Answer: min(x,y,z,sum-x-z-y | f[n][x][y][z]==true)
- Expected score: 80-96

Fetch Quest: dynamic programming

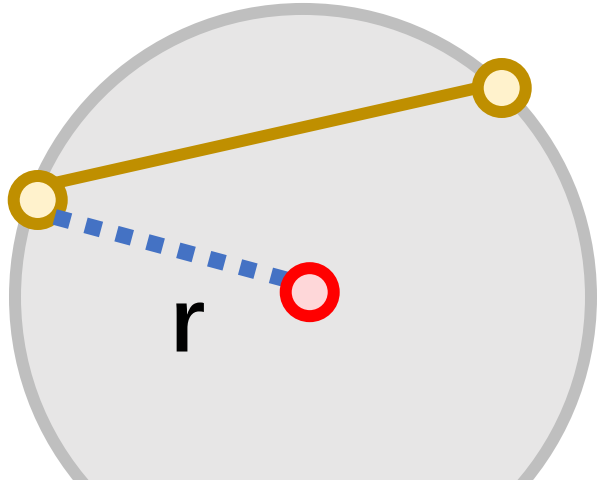
- **The 0/1 Knapsack problem (CS141): given a set of integers, find if a subset can have a given sum**
- **States: f[i,x,y,z] be if we can choose from the first i elements that has sum x for subset 1, sum y for subset 2, sum z for subset 3**
- **Problem: space consumption is too large**
  - Use the space trick for knapsack problem taught in CS 141!
- **Too slow: prune useless states (e.g., only compute x<y<z)**
- **Adding all pieces together: 120 points**

Sphere Mongers

- Given spheres (points) on a 2D plane, and a certain radius, decide the total spheres that can be covered

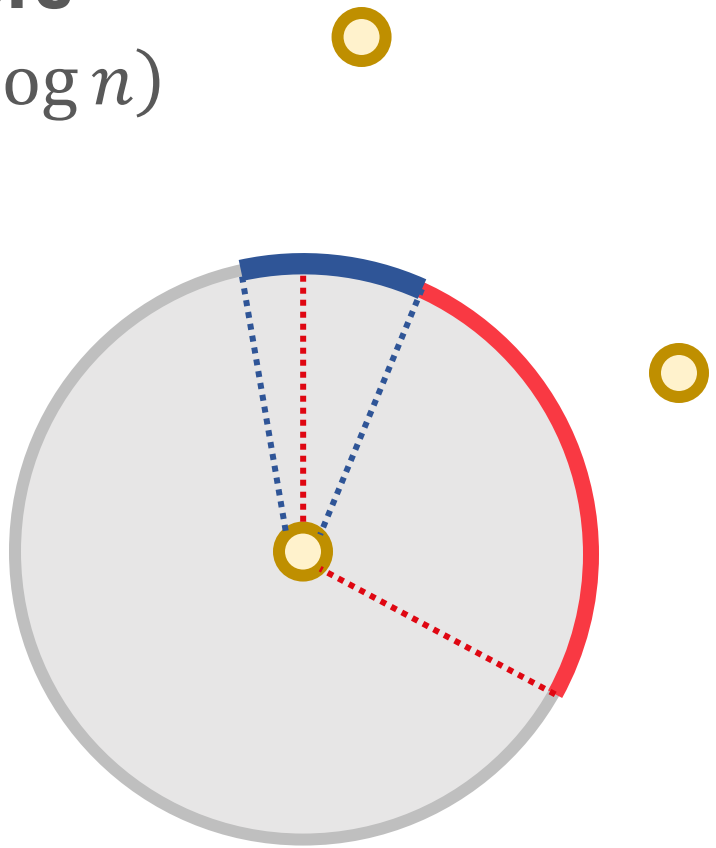
Sphere Mongers

- **How many degree of freedom does a circle has?**
  - Three: three points decide a circle
- **In this problem, the radius is fixed, so we need another two**
- **Solution 1: enumerate two points, and check for all other points**
  - Time complexity: $O(n^3)$
  - Can get 96 points



Sphere Mongers

- **Solution 2: enumerate one points, compute the polar angle interval that covers each other point, apply a scan and get the weighted maximal covers**
  - Time complexity: $O(n^2 \log n)$



Number of solvers per problem

Problem		# Test cases	Points per test case	Total points	#Solvers	First Solver
A	Senseless Census	20	3	60	84/89	James Rungsawang (0:01)
B	Lost in the Shuffle	20	3	60	82/85	James Rungsawang (0:03)
C	Maths of Glory	20	3	60	77/86	James Rungsawang (0:12)
D	Stake Your Claim	20	4	80	14/26	Omer Eren (0:29)
E	Feeding Friendsy	16	5	80	2/66	Omer Eren (0:58)
F	Trip Navigator	20	5	100	7/34	Sanat Mishra (0:43)
G	Snack Attack	20	5	100	2/15	James Rungsawang (3:48)
H	Fetch Quest	15	8	120	0/15	Omer Eren (104 pts)
I	Sphere Mongers	20	6	120	0/19	Omer Eren (114 pts)
Total				780		

Final remark: how to practice in the future?

- CS 141 (Fall): Intermediate Data Structures and Algorithms
- CS 142 (Winter): Algorithm Engineering
- CS 214 (Winter): Parallel Algorithms
- CS 218 (Spring): Design and Analysis of Algorithms
- CS 219 (Spring): Advanced Algorithms

Final remark: how to practice in the future?

- Join CS141 biweekly training: https://codeforces.com/group/xGy7DI5wNo
- Register for CS 142 in Winter 2022
- Participate ACM-ICPC and form your own team!
- Join UCR Competition Programming team and attend weekly practice