Groups

Eliza
Elijah
Nadani

Zeke
Jackson
Bruce

Peter
Alderik
Henry

Lillie
Jacqueline
Ben

Michael
Tommaso
Nick

Graham
Alex
Kai

Christian
Pierce
Chris

Will
Zachary
Dylan

Farhan
Joonwoo
Tenzin

Caroline
Angel
Bryan
**Outline**

Part 1: Problems Computers Can Solve

<table>
<thead>
<tr>
<th>3 Approaches</th>
<th>Divide + Conquer</th>
<th>Dynamic Programming</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prove Correct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze Runtime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 tasks

Interleaved

Tues  Thurs  T/Th

Part 2: Problems Computers Can't Solve (well)
(as far as we know)

NP-Completeness
Course Outline

Part 1: How to Speak Math

- Words = Sets
- Sentences = Statements + predicates
- Essays = Proofs

Part 2: Applications

- Functions
- Counting
- Graphs

Q: When I use a computer/phone in class, it contributes to my learning in that class.

A. Most of the time
B. Some of the time
C. I really just use it to check Instagram and e-mail
D. I don't use a computer/phone in class.
Divide & Conquer ↓ ( & Combine)

- Split big problem into smaller versions of same problem.
- Solve smaller problems via recursion.
- Combine smaller solutions to get big solution.

Already Seen!

Merge Sort

Input: Array A of integers of size n
Output: Sorted array

```
MergeSort(A)
  if length(A) == 1 then return A               \ Base Case
  A1 = MergeSort(A[1 : \lfloor n/2 \rfloor])
  A2 = MergeSort(A[\lfloor n/2 \rfloor + 1 : n])
  p1 = p2 = 1
  for i = 1 to n
      p1++
    else
      A[i] = A2[p2]
      p2++
  return A
```

Combine

A1 sorted
A2 sorted

A
**Divide + Conquer**

- Description:
  - Base Case
  - Recursion
  - Loops/Conditionals

- Correctness:
  - (Strong) Inductive Proof
  - Proof by cases
    - Loop Invariants

- Time Complexity:
  - Recurrence relation + Master method

For rest, we'll do small review, but I'm assuming familiarity

(See PS1 @ go/cs200 for standard inductive proof review.)
Q: Which of the following is a correct recurrence relation for MergeSort?

A) \( T(n) = T\left(\frac{n}{2}\right) + O(n) \)

B) \( T(n) = T(n) + O\left(\frac{n}{2}\right) \)

C) \( T(n) = 2T\left(\frac{n}{2}\right) + O(n) \)

D) None of the above
Q: Which of the following is a correct recurrence relation for MergeSort?

A) $T(n) = T\left(\frac{n}{2}\right) + O(n)$

B) $T(n) = T(n) + O\left(\frac{n}{2}\right)$

C) $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$  $\Leftarrow$ Partly OK, need $T(1) = O(1)$

D) None of the above  $\Leftarrow$ Technically Correct
   All are missing "base case"
Review of Induction (+ application to Algorithm correctness)

Induction

- Useful when problem size decreases by 1 in recursive call

1

Step 1: Show how to get on ladder

K

Step 2: Show how to get from rung K to K+1

K+1

Better for Divide & Conquer: Strong Induction

Step 2: Assume all rungs from 1 to K are true, use to get to K+1

Step 1: Show how to get on ladder

- If problem size in recursive call is smaller than original input, can assume output is correct by any amount, as long as greater than or equal to base case size
Q: Start a Strong Inductive Proof of Correctness of Merge Sort

A: Let P(n) be the predicate that Merge Sort works correctly on arrays of size n. We will prove P(n) is true for all n ≥ 1 using strong induction.

Base Case: P(1) is true because if the array is size 1, it is already sorted, so the algorithm returns it, which is correct.

Inductive Step: Let k ≥ 1. Assume for strong induction P(j) is true for all j: 1 ≤ j ≤ k. Now consider an input of size k+1. Since k+1 ≥ 1, the algorithm goes to the recursive step. It applies MergeSort to the first and second half of the array. Since each half is smaller than k+1, but at least 1 in length, by inductive assumption, the output of these calls are sorted arrays.

... Need Loop Invariants for Rest