3.2

Recursion - More Practice

Lesson Plan

- [10] Icebreaker
- [10] Branching Review
- [10] Helper Functions
- [25] Backtracking
- [20] Recursion in 2D!
- [30] Backtracking Practice
- [???] Boggle

We'll probably save Boggle for the review session next class.

Let me know what other topics you want to review!

Icebreaker

Branching Review

Making change!

Eating chocolate!

Robots!

Helper Functions

Making Change

```
def change(x):
    if x < 0:
        return False
    if x == 0:
        return True
    return change(x - 7) or change(x - 11) or change(x - 19)
```

What if we're only allowed to use 5 coins total?
…*without* changing the function header?

Making Change

```
def change_helper(x):
    if x < 0:
        return False
    if x == 0:
        return True
    return change(x - 7) or change(x - 11) or change(x - 19)
```

Making Change

```
def change_helper(x, n):
    if x < 0 or n > 5:
        return False
    if x == 0:
        return True
    return change(x - 7, n + 1) or change(x - 11, n + 1) or change(x - 19, n + 1)
```

Making Change

```
def change(x):
    return change_helper(x, 0)

def change_helper(x, n):
    if x < 0 or n > 5:
        return False
    if x == 0:
        return True
    return change(x - 7, n + 1) or change(x - 11, n + 1) or change(x - 19, n + 1)
```

It's important not to change function headers when provided!

Backtracking

Backtracking

So far we've always had recursive calls solve sub-problems with 'smaller' inputs

Our recursive calls can have increasingly large inputs as long as there's an upper bound!

We use backtracking to build up every possible combination to try to find solutions

Recursion

- Base case is smallest, simplest version of problem
- Recursive case works down to the base case
- That base case is sent back up the call stack, being built up at each call

Backtracking

???

Binary Numbers [repl.it]

000
001
010
011
100
101
110
111

Recursion

- Base case is smallest, simplest version of problem
- Recursive case works down to the base case
- That base case is sent back up the call stack, being built up at each call

Backtracking

- Each base case is a potential solution
- Recursive case works up to the base case
- The input is usually the empty seed of a solution, to be built up to every possible base case

TL;DR Trying every possible combination!

Binary Numbers - As A List! [repl.it]

[000, 001, 010, 011, 100, 101, 110, 111]

Binary Numbers - As A List! With at most k 1s! [repl.it]

[000, 001, 010, 011, 100, 101, 110, 111]

The Knapsack Problem [repl.it]

Practice Problems: Backtracking

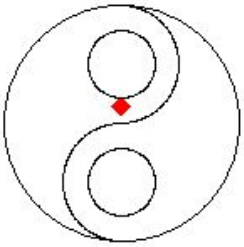
You will be working in teams of 3 or 4. Use the table below to figure out what your role is [https://replit.com/@dgshanker/Recursion-Backtracking#instructions.md]

Role	Responsibilities	Assignment Criteria
Captain	Share screen, write code, keep track of time, ensure all team members participate	Person who has been assigned this role the least number of times
Tester	Plays devil's advocate, design test cases, determine algorithm complexity (time and space)	Person who has been assigned this role the least number of times
Presenter	Explain solution to the class, present the team's algorithm design decisions, state solution's complexity (time and space), share one thing the team learned from the problem	Person who has been assigned this role the least number of times

If there are ties, get creative and come up with a way to break them (i.e., sort yourselves by last name, distance to Google Austin, etc.)
If there are 4 members in your team, you should have two Testers

Recursion in 2D!

Floodfill [repl.it]



Practice Problems: Boggle

You will be working in teams of 3 or 4. Use the table below to figure out what your role is [repl.it]

Role	Responsibilities	Assignment Criteria
Captain	Share screen, write code, keep track of time, ensure all team members participate	Person who has been assigned this role the least number of times
Tester	Plays devil's advocate, design test cases, determine algorithm complexity (time and space)	Person who has been assigned this role the least number of times
Presenter	Explain solution to the class, present the team's algorithm design decisions, state solution's complexity (time and space), share one thing the team learned from the problem	Person who has been assigned this role the least number of times

If there are ties, get creative and come up with a way to break them (i.e., sort yourselves by last name, distance to Google Austin, etc.)
If there are 4 members in your team, you should have two Testers