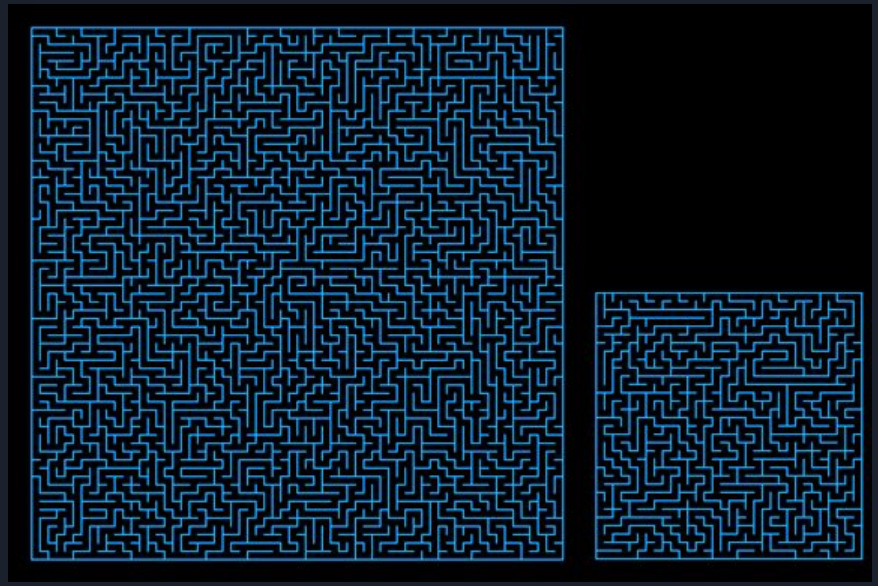
CS 314 Discussion

Problems

- Main Problem
  - Recursive Backtracking
- Extra Problem
  - (Harder) Recursive Backtracking



Proto Power-stop

If anyone knows how to fix pls tell me

In case you're interested!

7

TOMORROW AT 8 PM – 11 PM

Mueller Lake Park Trail Skate Meetup

1830 Simond Ave, Austin, TX 78723-4603, United States

About | Discussion | Interested | Invite

Details

16 people responded

Event by **All Wheels Wanted**

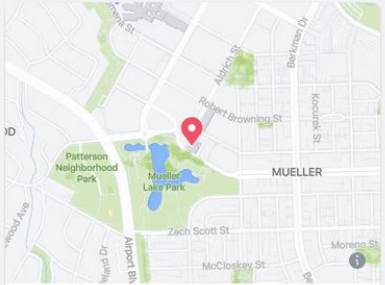
1830 Simond Ave, Austin, TX 78723-4603, United States

Duration: 3 hr

Public · Anyone on or off Facebook

All Wheels are Wanted at this nighttime beginner-friendly trail skate meetup. We rendezvous at Halcyon Mueller and The Thinkery at 8pm and rollout is at 9pm. It's roughly 3.5 miles with a spot to chill and jam at the end and usually wraps up before 11pm.

We recommend outdoor wheels and anything you have that lights up!



1830 Simond Ave, Austin, TX 78723-4603, United States

Y'all make me feel bad

- I took off a lot of experiments that otherwise were perfect because of no timing data
- If your feedback file looks like:

```
*****************************************************
CS 314 Assignment 5 Grading - Experiment (0/1)

-1 Missing timing data (always include!)
```

- Email me your timing data by Tuesday (3/8) by 11:00 PM for this point back!

Code Collab!

codecollab.io/@proj/SpadeVoyageRoute

Dice Problem

- Base case?
  - We're limited by the number of dice we have
- Recursive step
  - What are the different paths we can take?

Dice Problem

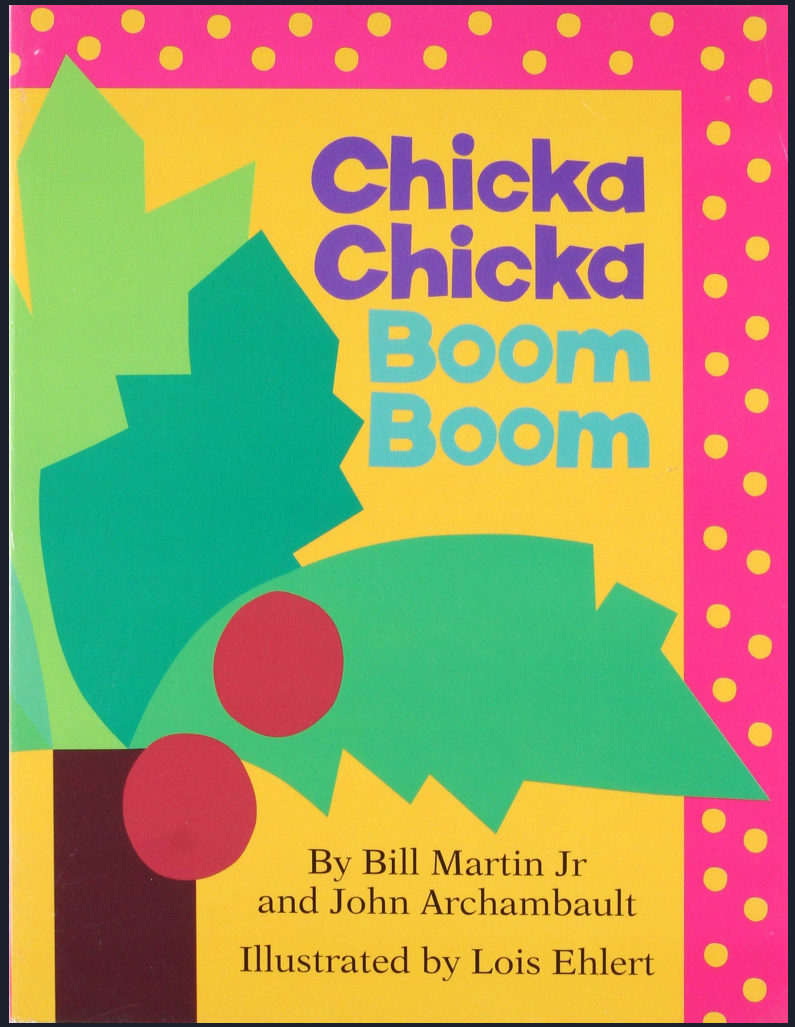
- Base case:
  - If we have 0 dice there's nothing we can do
  - If we have 1 di(?) then check if we can get to numToRoll
- Recursive step
  - We can roll a 1, 2, ..., 5, 6

```java
public int waysToRoll(int numToRoll, int numDice) {
    if (numDice == 0) {
        return 0 if numToRoll > 0 else 1;
    }
    if (numDice == 1 |) {
        return numToRoll >= 1 && numToRoll <= 6 ? 1 : 0;
    }
    int numWays = 0;
    for (int side = 1; side <= 6; side++) {
        numWays += waysToRoll(numToRoll - side, numDice - 1);
    }
    return numWays;
}
```

```java
public int waysToRoll(int numToRoll, int numDice) {
    // Examples: If we have to roll 4 with 5 dice or roll 13 with 2 dice
    if (numToRoll < numDice || numToRoll > 6 * numDice) {
        return 0;
    }
    // Annoying edge case
    if (numDice == 1 || (numDice == 0 && numToRoll == 0)) {
        return 1;
    }
    int numWays = 0;
    for (int side = 1; side <= 6; side++) {
        numWays += waysToRoll(numToRoll - side, numDice - 1);
    }
    return numWays;
}
```

Forest Problem

- Base case?
  - Come back later ...
- Recursive step
  - How can we check surrounding areas without counting the same plant twice?



	0	1	2	3	4	5	column indices
0	2	5	2	**5**	9	1	
1	5	2	2	**5**	**5**	**5**	
2	5	6	7	**5**	9	9	
3	1	**5**	7	**5**	7	**5**	
4	**5**	**5**	**5**	**5**	**5**	**5**	
row indices							

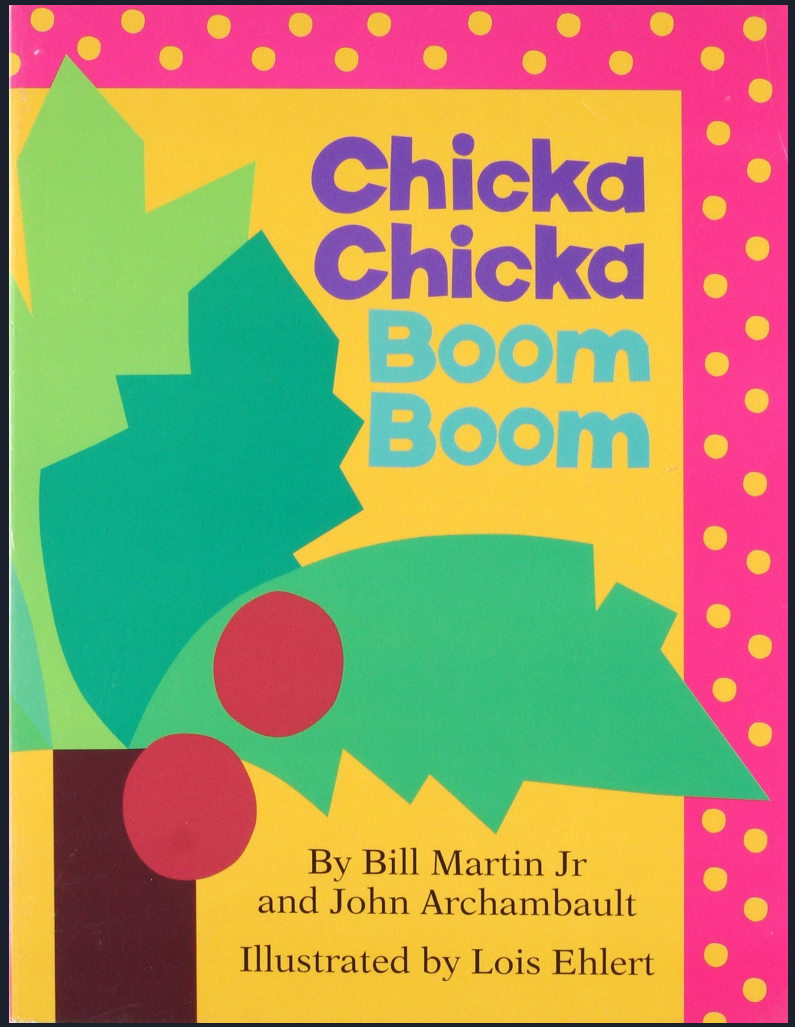
[redacted]

[redacted]

	0	1	2	3	4	5	column indices
0	2	5	2	**5**	9	1	
1	5	2	2	**5**	**5**	**5**	
2	5	6	7	[illegible]	9	9	
3	1	**5**	7	**5**	7	**5**	
4	**5**	**5**	**5**	**5**	**5**	**5**	
row indices							

Forest Problem

- Base case?
  - Check it's the right plant & hasn't been visited
  - Check it's in bounds
- Recursive step
  - Mark current step
  - Check 4 adjacent sides



```java
public static int sizeOfPlantArea(int[][] map, int row, int col) {
    boolean[][] visited = new boolean[map.length][map[0].length];
    return areaHelper(map, visited, row, col, map[row][col]);
}
private static int areaHelper(int[][] map, boolean[][] visited, int row,
    int col, int tgt) {
    if (row < 0 || row >= map.length || col < 0 || col >= map[0].length ||
        visited[row][col] || map[row][col] != tgt) {
        return 0;
    }
    visited[row][col] = true;
    return 1 + areaHelper(map, visited, row - 1, col, tgt)
        + areaHelper(map, visited, row + 1, col, tgt)
        + areaHelper(map, visited, row, col - 1, tgt)
        + areaHelper(map, visited, row, col + 1, tgt);
}
```

About Assignment 6

- 0/1/2 in the Maze this time around have meaning! They ***should*** be magic numbers!
  - They don't represent the quantities/indices 0/1/2, they represent a state of completion in the maze
- If there is not an exit in the maze, then we don't need to recurse through it!
- Remember to exit early from the maze if you've found an optimal exit!