Snake and Board (Day 28)

Problem | Submissions | Leaderboard | Discussions

1.Take as input N, a number. N is the size of a snakes and ladder board (without any snakes and ladders).
2.Take as input M, a number. M is the number of faces of the dice.

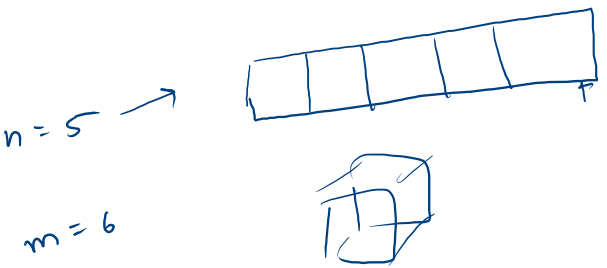
- Write a recursive function which returns the count of different ways the board can be travelled using the dice. Print the value returned.
- Write a recursive function which returns an ArrayList of dice values for all valid paths across the board. Print the value returned.
- Write a recursive function which prints dice values for all valid paths across the board

Sample Input 0

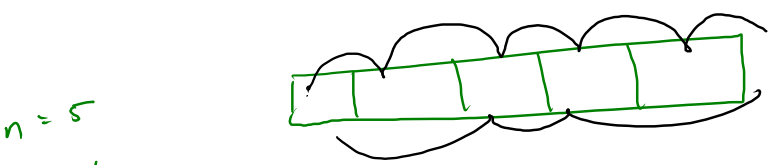
```
5
6
```

Sample Output 0

```
16
[11111, 1112, 1121, 113, 1211, 122, 131, 14, 2111, 212, 221, 23, 311, 32, 41, 5]
11111
1112
1121
113
1211
122
131
14
2111
212
221
23
311
32
41
5
```



no. of paths
paths in arraylist →
printpaths →



1 1 1 1 1
1 1 1 2
1 2 2
1 1 3
1 4
5
2 1 2

Loops

n = 5

m = 6

curpos == n → answer → positive base case.

curpos > n → negative base case.

p.s. void sBoard(curpos, arr, n, m, a)

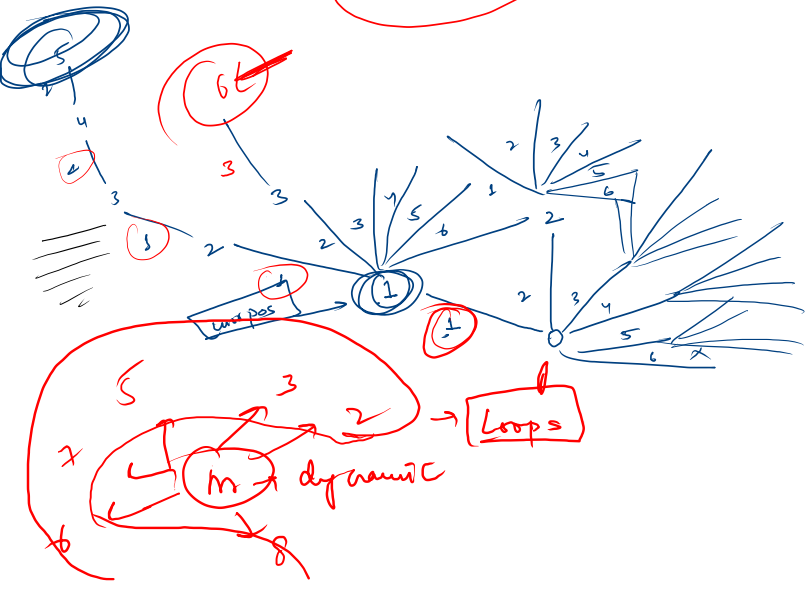
{

// negative & positive base case.

// → Logic

}

String, array list



adding integer to string

↓

No conversion needed

```java
public class Solution {
    public static void sBoard(int currpos, String ans, int n, int m, ArrayList<String> paths){
        if(currpos > n) return ;

        if(currpos == n){
            paths.add(ans);
            return;
        }

        for(int i=1;i<=m;i++){
            sBoard(currpos+i, ans+i, n , m, paths);
        }
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int m = scn.nextInt();

        ArrayList<String> paths = new ArrayList<>();

        sBoard(0,"",n,m,paths);
        System.out.println(paths.size());
        System.out.println(paths);

        for(int i=0;i<paths.size();i++){
            System.out.println(paths.get(i));
        }
    }
}
```

negative base case

positive base case

m choice

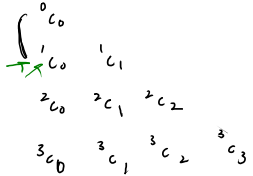
5-6 minutes

Sample Input 0

```
4
```

Sample Output 0

```
1
11
121
1331
```



```
for (int i=0; i<= n ; i++ )
    int val=1;
    for (int j=0; j<=i; j++)
    {
        cout<<(val);
        val= val * (i- j) / (j+1);
    }
```

```
p.s. void pattern (int row, int n)
{
    if(i==n) return;
    int val = 1;
    for( int j = 0; j<= row; j++) {
        cout (val);
        val = val * (row-j)/(j+1);
    }
    pattern(i+1, n);
}
```

val = 1

$$^{n}C_{r} = \frac{n!}{r!\,(n-r)!}$$

$$= \frac{n!}{r!\,(n-r)(n-r-1)!}$$

$$^{n}C_{r+1} = \frac{n!}{(r+1)!\,(n-(r+1))!}$$

$$^{n}C_{r+1} = \frac{n!}{(r+1)(r)!\,(n-r-1)!}$$

$$\frac{^{n}C_{r+1}}{^{n}C_{r}} = \frac{\frac{n!}{(r+1)(r)!\,(n-r-1)!}}{\frac{n!}{r!\,(n-r)(n-r-1)!}} \Rightarrow \frac{^{n}C_{r+1}}{^{n}C_{r}} = \frac{(n-r)}{(r+1)}$$

$$^{n}C_{r+1} = \frac{(n-r)}{(r+1)} * val$$

5-1 mins

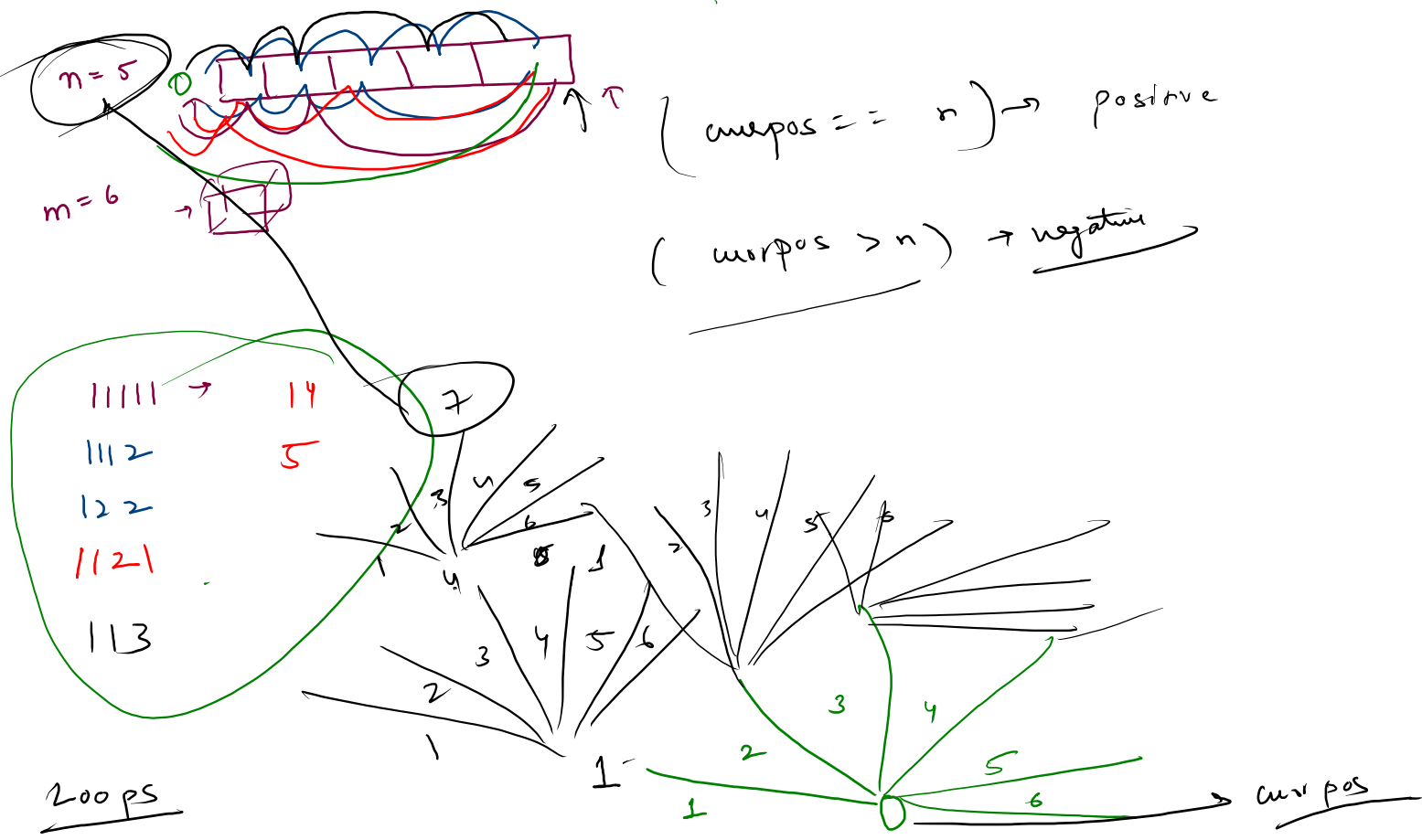
```java
import java.io.*;
import java.util.*;

public class Solution {
    public static void pattern(int row, int n){
        if(row == n) return;

        int val = 1;
        for(int j=0;j<=row;j++){
            System.out.print(val);
            val = val*(row-j)/(j+1);
        }
        System.out.println();
        pattern(row+1,n);
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT.
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        pattern(0,n);

    }
}
```

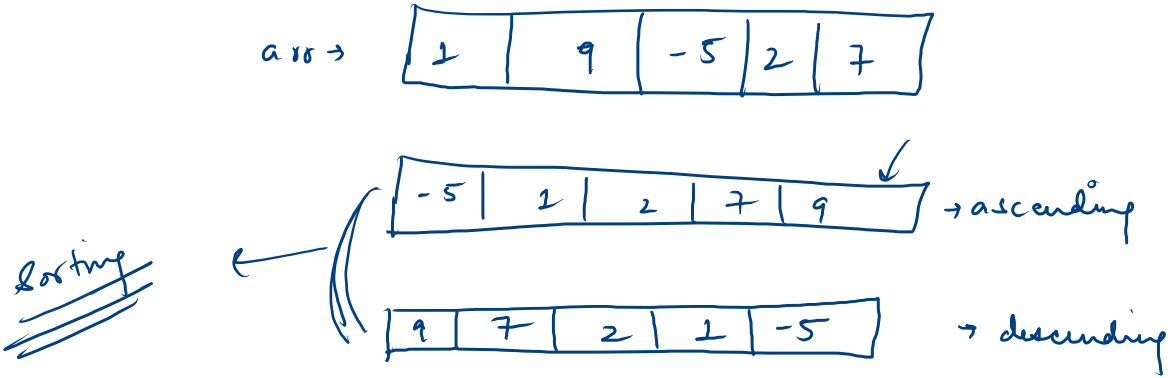

n = 5
m = 6
curpos == n → positive
curpos > n → negative
11111 → 14
1112 5
1122
11211
113
200 ps
curr pos

Searching and Sorting

- All the important sorting algo's.
  - Built sorting fn in Java.
  - Binary search Algo^n → Very tricky questions
  - Some practice questions on searching & sorting.

Sorting → ??

↳ arranging elements in ascending / descending order

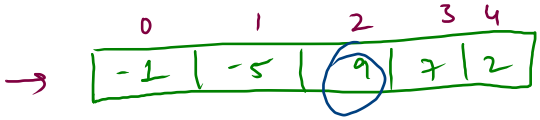


1) Bubble Sort → ascending order

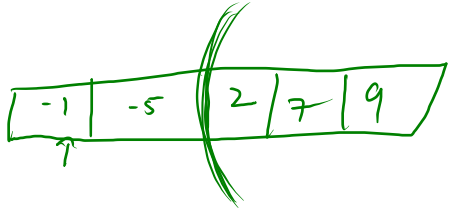
↳ Idea → sort the largest element first

n → iterations

(n-1)

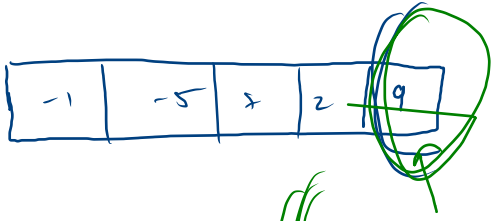


itr-03

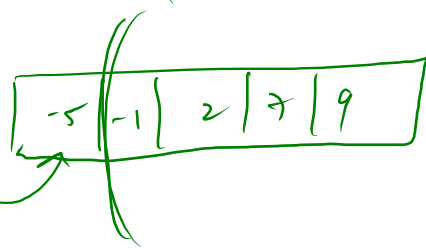


iteration -01

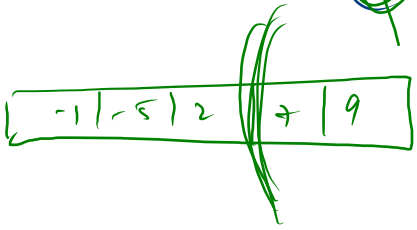
swapping



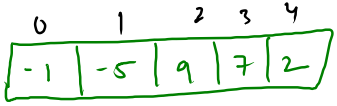
itr-04



itr-02



n → (n-1)



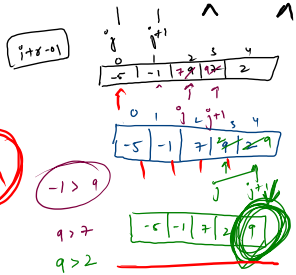
1 loop - iteration

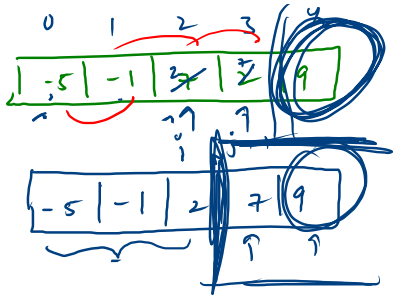
if (element at index j greater element at(j+1))
swap the element

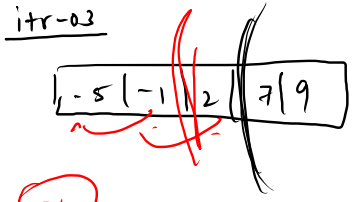
-1 > -5

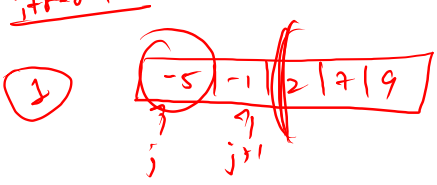
else

no need to swap the elements.









int j=0; j< n-itr; j++

itr=1 j < 5-1 = 4

itr=2 j < 5-2 = 3

n → (n-1)

Time complexity

Space complexity

```java
public class Solution {
    public static void swap(int arr[],int i,int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void bubbleSort(int arr[],int n){
        // Logic for bubble sort algo
        for(int itr=1;itr<n;itr++){
            for(int j=0;j<n-itr;j++){
                if(arr[j]>arr[j+1]){
                    swap(arr,j,j+1);
                }
            }
        }

        // For Printing
        for(int i=0;i<n;i++){
            System.out.print(arr[i] + " ");
        }
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print out
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        bubbleSort(arr,n);
    }
}
```

S.C = O(1)

T.C = $O(n^2)$

n →

n-1 →

n-2 →

n-3 →

1

1 + 2 + 3 + ---- n-3 + n-2 + n-1 + n

→ $\frac{n(n+1)}{2}$ → $\frac{n^2+n}{2}$

T.C = $O(n^2)$