Permutation Printing

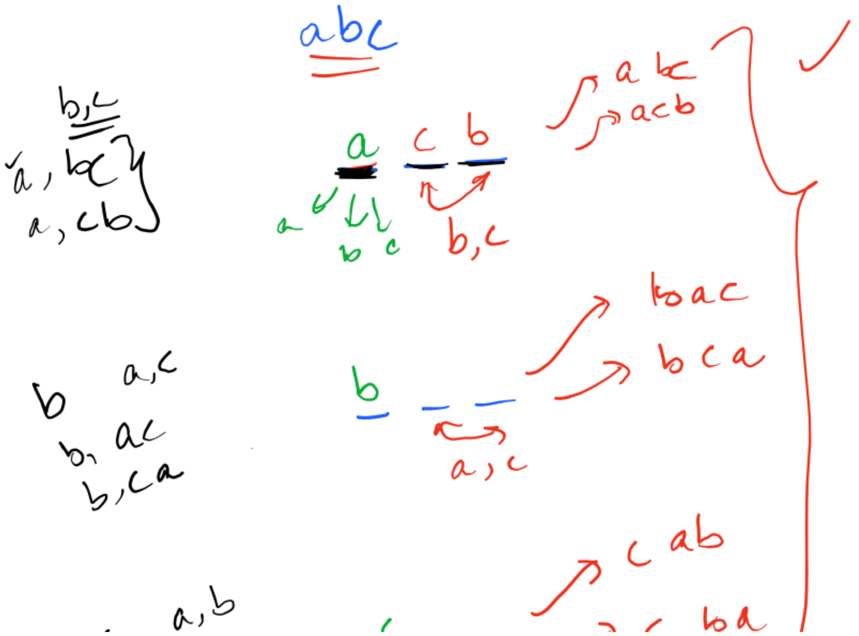
Friday, 16 February 2024 8:17 PM

Arrangements: abc

abc bac cab
acb bca cba

→ Order is not maintained

→ All are of same length



→ We fix a particular index

→ Send remaining characters forward

→ Chose all possible fixed characters.

BP: → To find the permutations

c
cab
cba

a, b

of length 3

SP → To find permutations of length 2

SW → To add the fixed character to them and add possible combinations

BC → Permutations of 0 length

0 1 2
abc

""

remaining = substring (0,0) + substring (0+1) bc

rem-str (1) = subs (0,1) + subs (1+1);
a + c = ac
ab

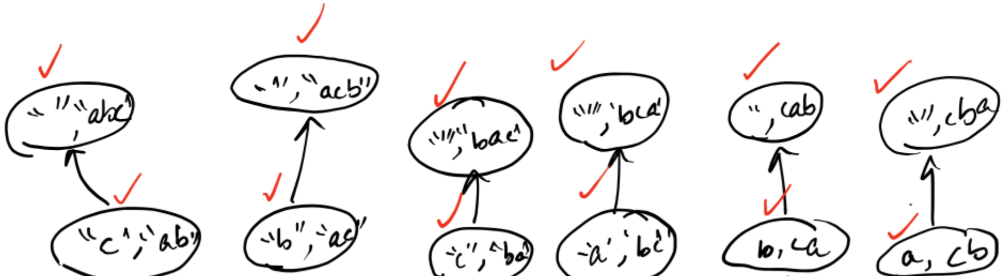
abc
acb
bac
bca
cab
cba

a bb
a, b

bab

```
public static void permutationPrint(String ques, String asf)
  {
     if(ques.length()==0){
        System.out.println(asf);
        return;
     }
     for(int i=0;i<ques.length();i++){
        if(i>0 && ques.charAt(i)==ques.charAt(i-1))
           continue;

        char ch = ques.charAt(i);
        String remaining = ques.substring(0,i)+ques.substring(i+
1);
        permutationPrint(remaining,asf+ch);
     }
```

unique values

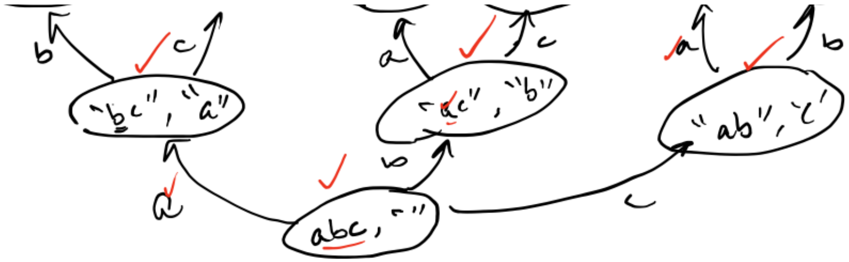


}

abc

"" + "bc"

bc



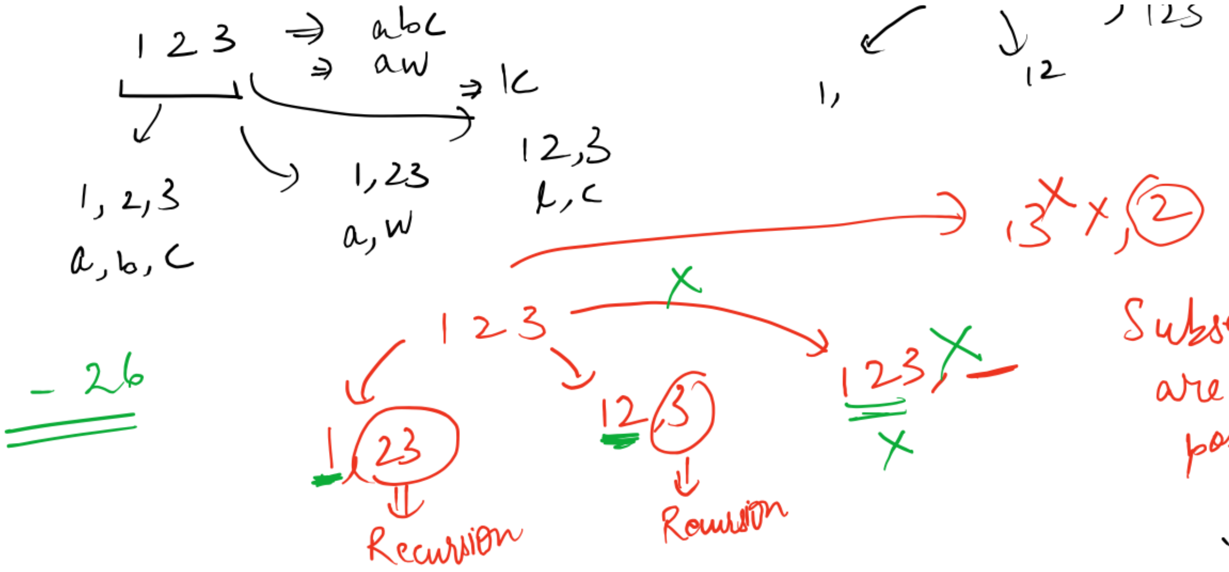
Friday, 16 February 2024
8:51 PM

String Encodings

a, b, c, d, e, f, g, h, i, l .. w, x, y, z

1 2 3 4 5 6 . . . . 23 24 25, 26

1 23 →



1 - 26

Subsequences are not possible

'5' - '0' = $\underset{\uparrow\ \text{int}}{5}$

'a' + 1 = 'b' ✓

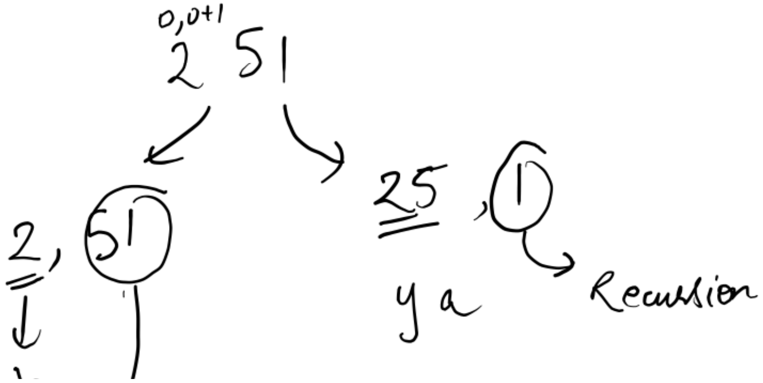
'a' + 3 = 'd' ✓

Steps

↳ Generate numbers from 1 to 26

↳ Ask recursion to get the resultt for remaining numbers

1-26



_bea

_ya

✓

5, 1

e a

findchar (2)

'a' + (2-1)

'a' + 1 → 'b'

a

+1 → b

+2 c

+3 d

+4 e

+5 f

findchar (6)✓

↳ 'a' + (6-1)

+5

✓ 'f'

abc

aw

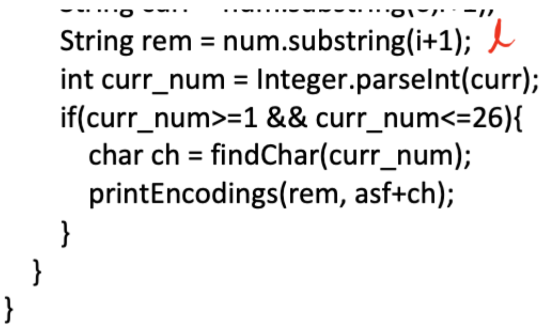
lc

```
public static void printEncodings(String num, String asf) {
        if(num.length()==0){
           System.out.println(asf);
           return;
        }
          if(num.charAt(0)=='0'){
                return;
           }
        for(int i=0;i<num.length();i++){
           String curr = num.substring(0,i+1);
```

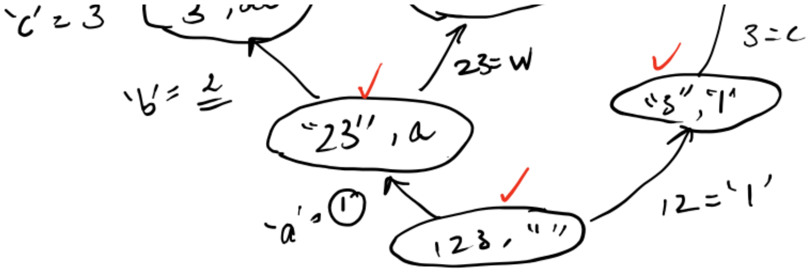
12

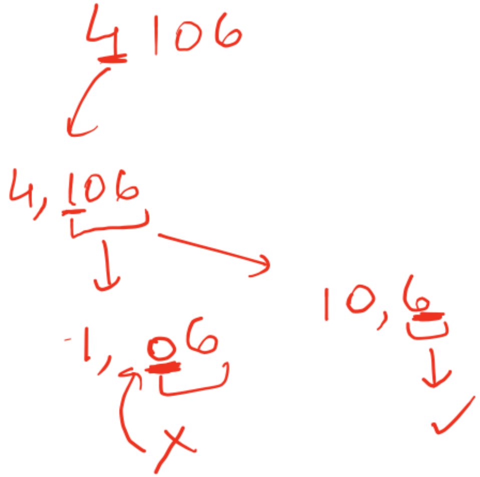
3

12



```
        String curr = num.substring(0,i+1);
        String rem = num.substring(i+1);
        int curr_num = Integer.parseInt(curr);
        if(curr_num>=1 && curr_num<=26){
            char ch = findChar(curr_num);
            printEncodings(rem, asf+ch);
        }
    }
}
```





-ve base case

get all the Subsequences

Print X

Store??

get Subsequences

↳ Get Recursion

↳ Return type ↳ usually an arraylist containing all the possible answers.

abc

a (bc)

↳ ["", b, bc, c,]

sr

mr

↳ [a, ab, abc, ac, "", b, bc, c]

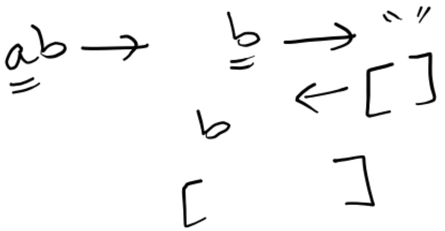
BP : abc

SP : bc

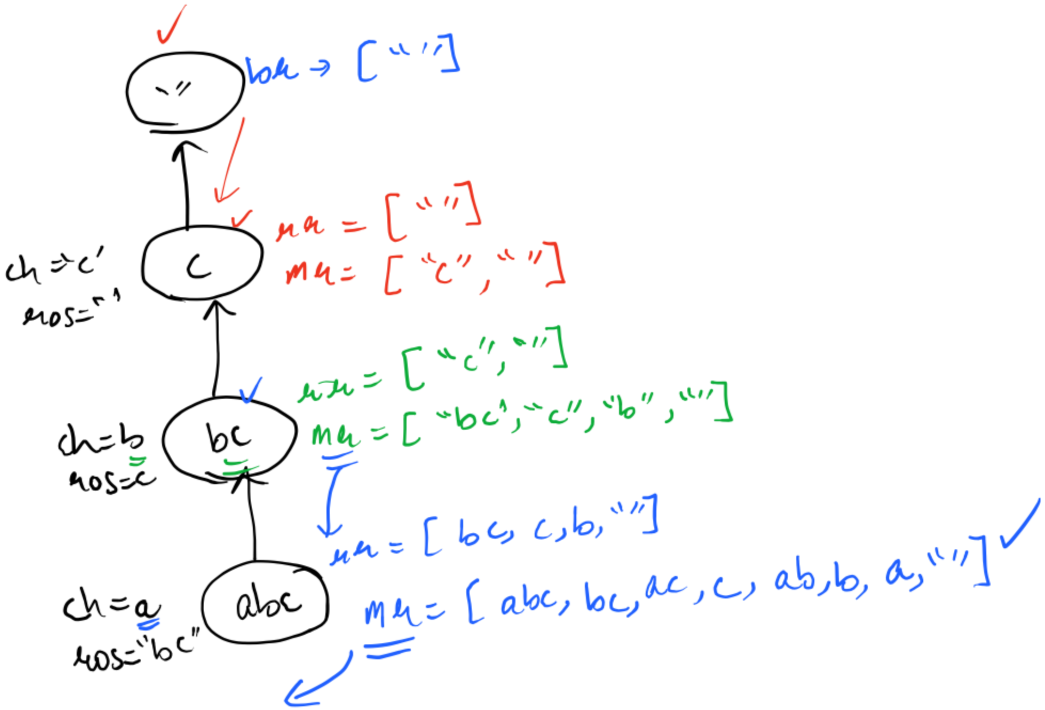
SW : [a + bc + bc]

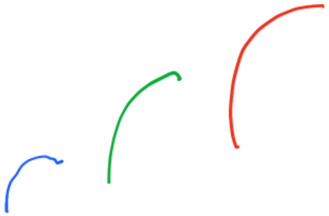
BC : ""

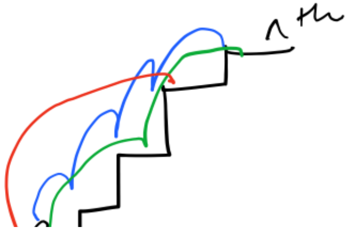
↳ [""] 1

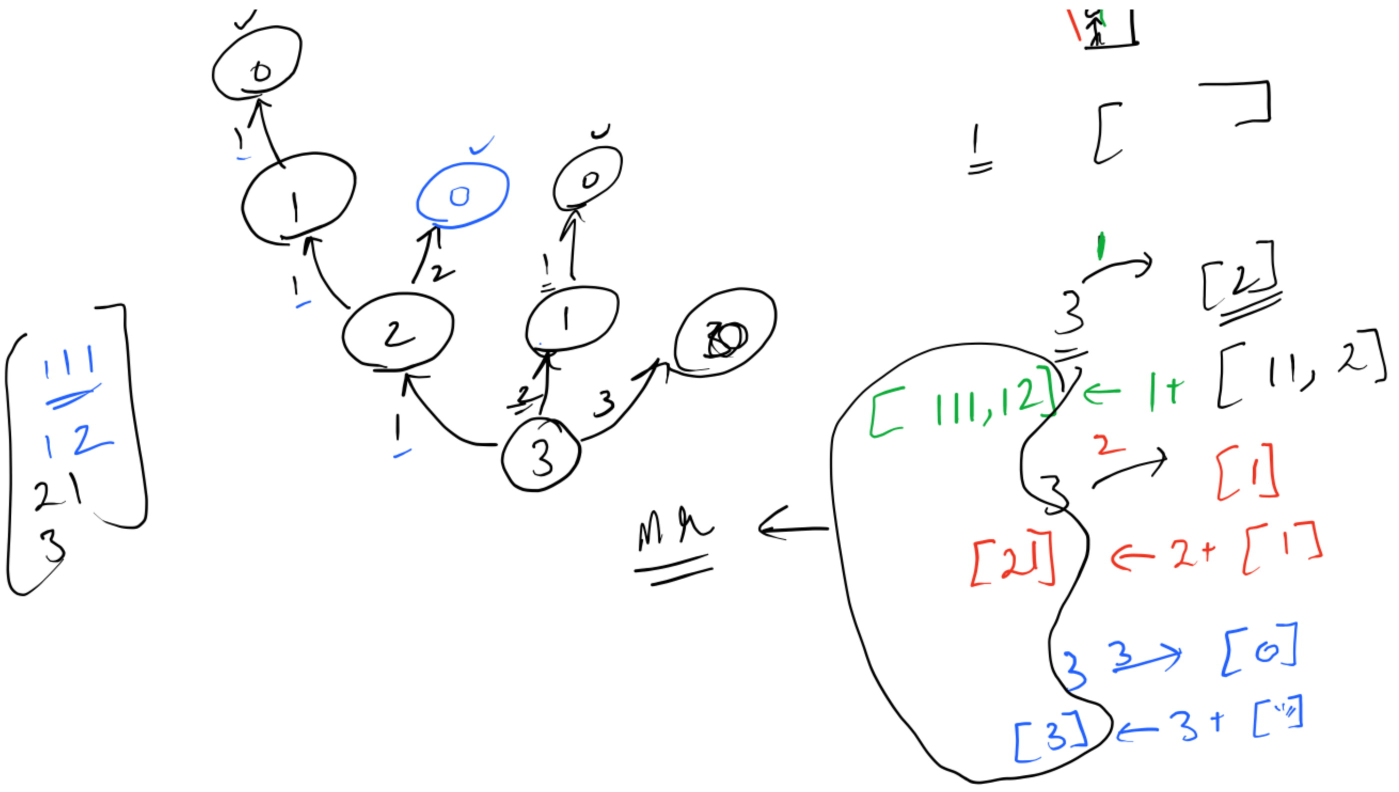


```
public static ArrayList<String> generateSubsequences(String str)
  {
    if(str.length()==0){
      ArrayList<String> br = new ArrayList<>();
      br.add("");
      return br;
    }
    char ch = str.charAt(0);
    String ros = str.substring(1);
    ArrayList<String> rr = generateSubsequences(ros);
    ArrayList<String> mr = new ArrayList<>();
      for(String s:rr){
      mr.add(ch+s);
     Mr.add(s);
    }
     return mr;
  }
```



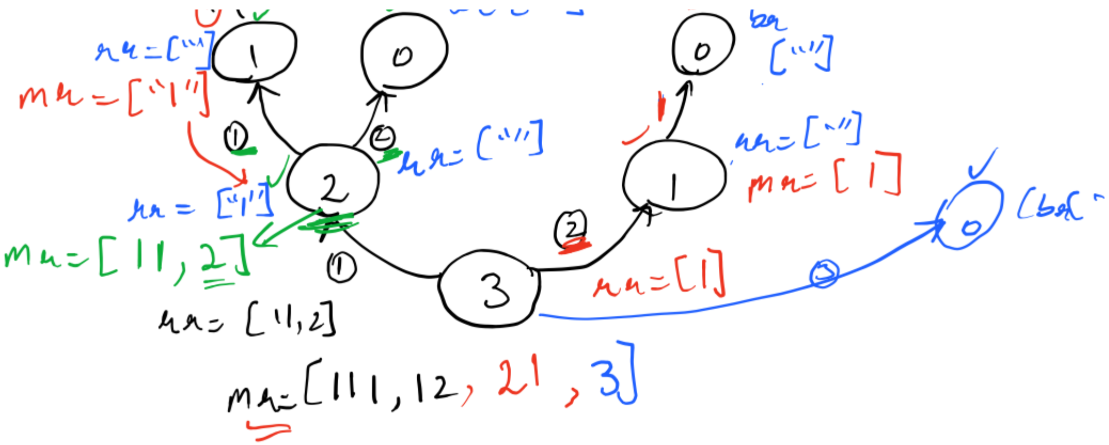






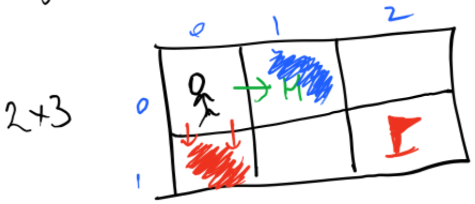
```
public static ArrayList<String> getStairPaths(int n) {
        if(n==0){
            ArrayList<String> br = new ArrayList<>();
            br.add("");
            return br;
        }
        ArrayList<String> mr = new ArrayList<>();
        for(int j=1;j<=3;j++){
            if(n-j>=0){
               ArrayList<String> rr = getStairPaths(n-j);
                for(String s:rr){
                    mr.add(j+s);
```

```
            }
        }
    }
    return mr;
}
```



Friday, 16 February 2024
10:47 PM

get Maze Path



✓ [X1, X2, X3]

sr   sc

dr   dc

✓

→ H → sr, sc → sr, sc+1

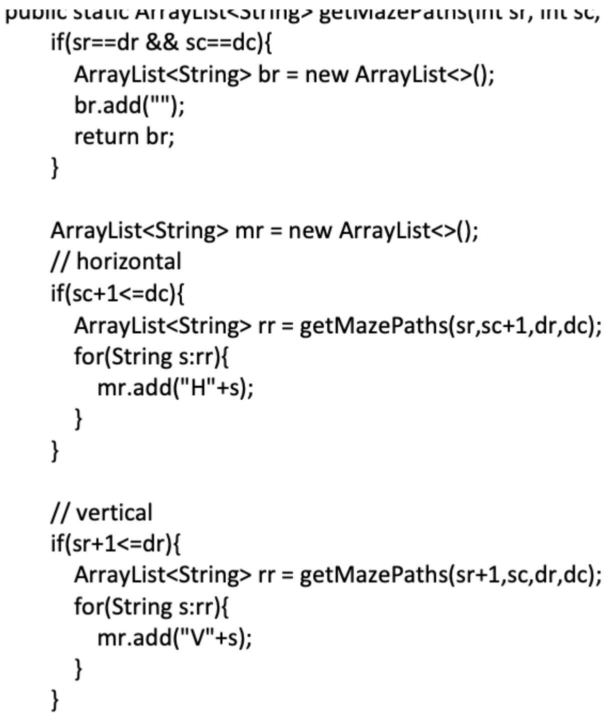
✓
→ V → sr, sc ⇒ sr+1, sc

$0,0 \xrightarrow{H} (0,1)$ [ ]

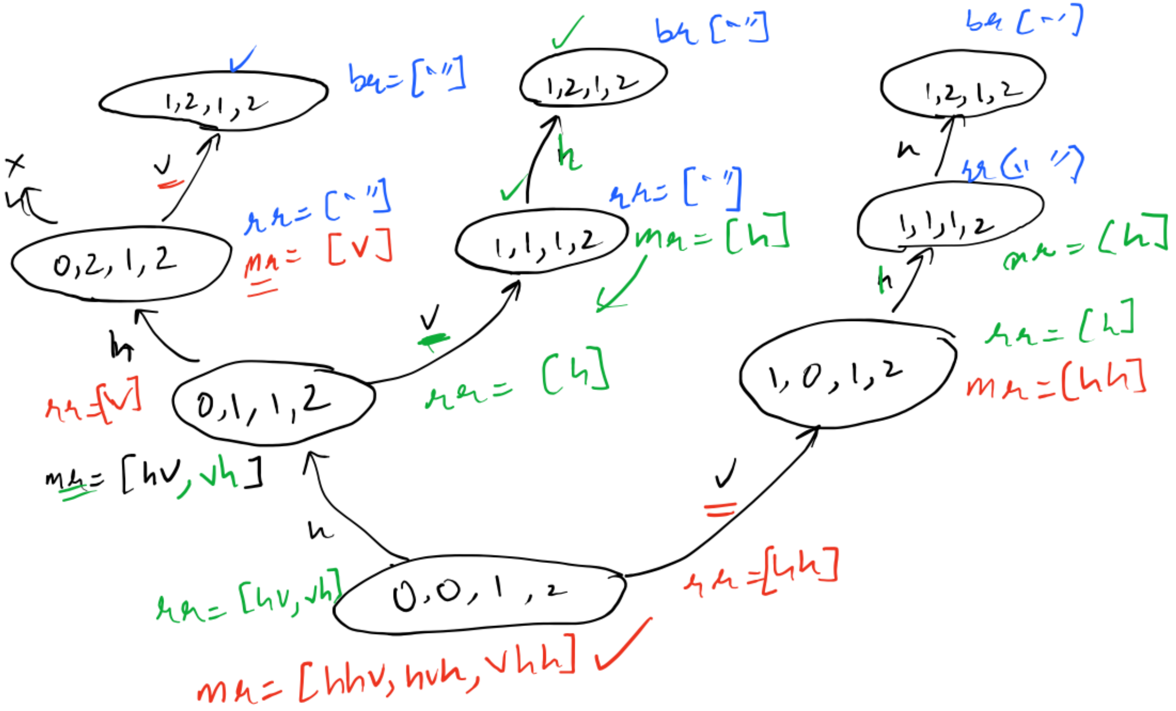
↓V

(1,0) [ ]

public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {

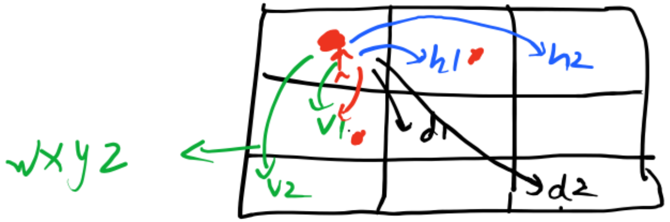


```
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
    if(sr==dr && sc==dc){
        ArrayList<String> br = new ArrayList<>();
        br.add("");
        return br;
    }

    ArrayList<String> mr = new ArrayList<>();
    // horizontal
    if(sc+1<=dc){
        ArrayList<String> rr = getMazePaths(sr,sc+1,dr,dc);
        for(String s:rr){
            mr.add("H"+s);
        }
    }

    // vertical
    if(sr+1<=dr){
        ArrayList<String> rr = getMazePaths(sr+1,sc,dr,dc);
        for(String s:rr){
            mr.add("V"+s);
        }
    }

    return mr;
}
```



Sunday, 18 February 2024
8:59 PM

Get Maze Paths Everydirn



BP

sr = 0
sc = 0

dr = n-1
dc = m-1

h1 → sr → sr
sc → sc+1

h2 → sr → sr
sc → sc+2

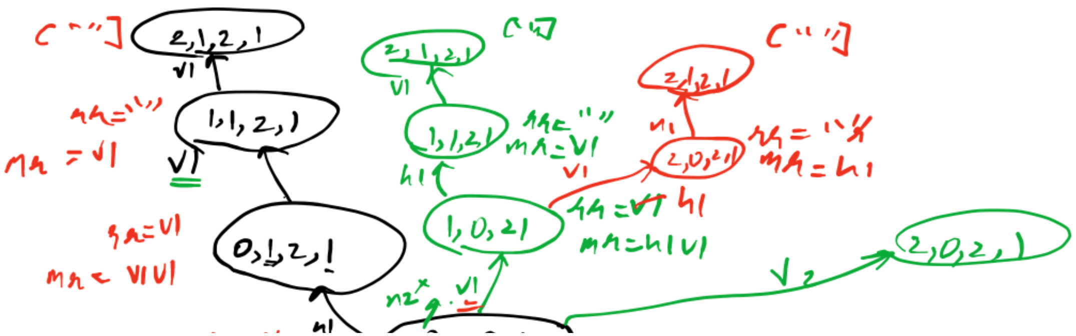
d1 sr → sr+1
sc → sc+1

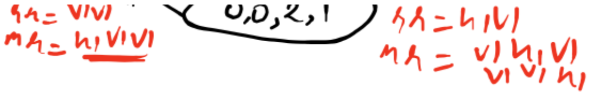
d2 sr → sr+2
sc → sc+2

v1 sr → sr+1
sc → sc

v2 sr → sr+2
sc → sc

```
private static ArrayList<String> solve(int sr, int sc, int dr, int dc){
   if(sr==dr && sc==dc){
      ArrayList<String> br = new ArrayList<>();
      br.add("");
      return br;
   }

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

  // horizontal
   for(int i=1;i<=2;i++){
      if(sc+i<=dc){
         ArrayList<String> h = solve(sr,sc+i,dr,dc);
         updateMR(mr,h,"h"+i);
      }
   }

   // vertical
   for(int i=1;i<=2;i++){
      if(sr+i<=dr){
         ArrayList<String> v = solve(sr+i,sc,dr,dc);
         updateMR(mr,v,"v"+i);
      }
   }

   // diagonal
   for(int i=1;i<=2;i++){
      if(sr+i <= dr && sc+i<=dc){
         ArrayList<String> d = solve(sr+i,sc+i,dr,dc);
         updateMR(mr,d,"d"+i);
```



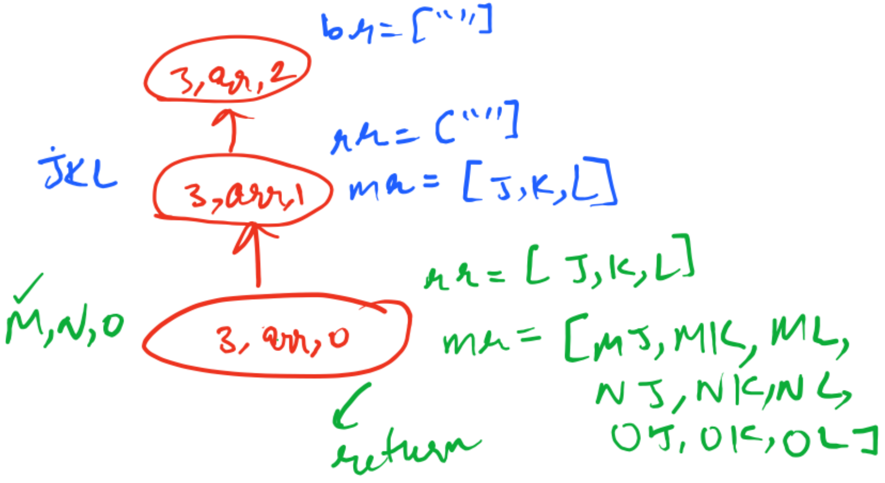


```
            }
        }

        return mr;
    }

    private static void updateMR(ArrayList<String> mr, ArrayList<String>
  rr, String val){
        for(String str:rr){
            mr.add(val+str);
        }
    }
```

```
static String[] table =
{"","ABC","DEF","GHI","JKL","MNO","PQRS","TU","VWX","YZ"};
  static ArrayList<String> solve(int n, int[] keys, int idx){
    if(idx==keys.length){
      ArrayList<String> br = new ArrayList<>();
      br.add("");
      return br;
    }
    ArrayList<String> mr = new ArrayList<>();
    ArrayList<String> rr = solve(n,keys,idx+1);
    String letters = table[keys[idx]];
    for(int i=0;i<letters.length();i++){
      char ch = letters.charAt(i);
      for(String str:rr){→ ""
        mr.add(ch+str);
      }
    }
    return mr;
  }
```

arr = [5, 4]

f(m,n)
m×n



5×4

$\hookrightarrow 5 + (5 \times 3)$

5×3

$\hookrightarrow 5 + (\underbrace{5 \times 2}) \rightarrow 5 + 5 \times 1$

$\hookrightarrow m + (\underbrace{m \times (n-1)})$

$\hookrightarrow f(m, n-1)$

$\hookrightarrow$

$\underbrace{f(m,n)}_{BP} = \underbrace{m +}_{SW} \underbrace{f(m, n-1)}_{SP}$

2

↳ global variable

↳ static variable and use this

→ changing the returntype