Today's content

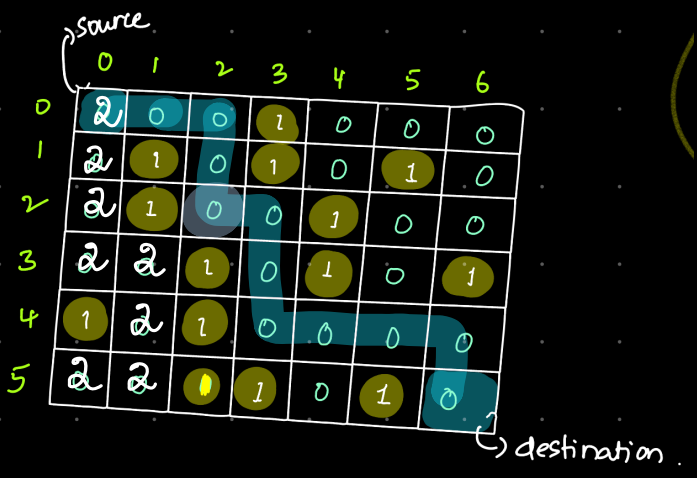
1. Introduction
2. Rat in a maze
3. Generate all permutations. (unique chars)
4. Generate all permutations. (dupl chars)
5. Subset.

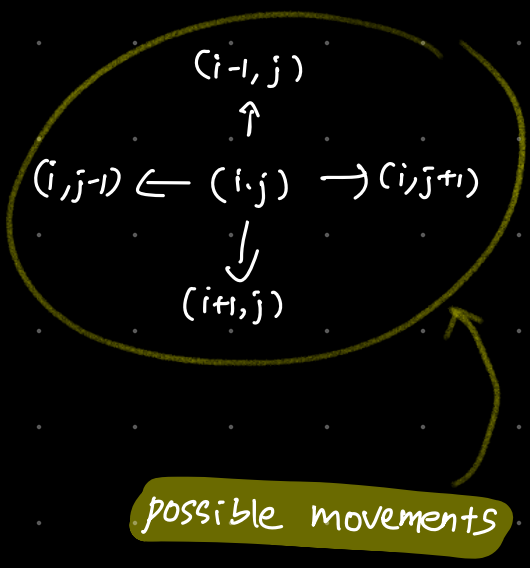
Will not be covered

Backtracking → subset of recursion + Generate all possibilities

Recursion: solving problems using subproblems.

Rat in a maze





1. check if we can move from source to destination.

You should not visit the same box again.

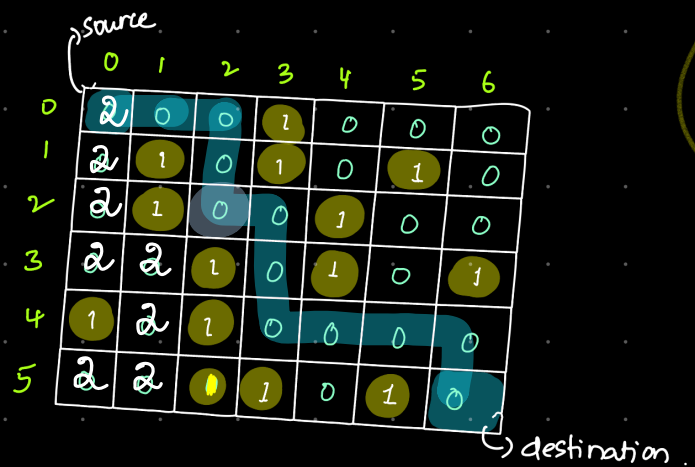
→ Keep track of visited cells.

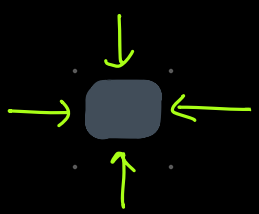
→ boolean [N][M] — true (cell is visited), false (cell is not visited)

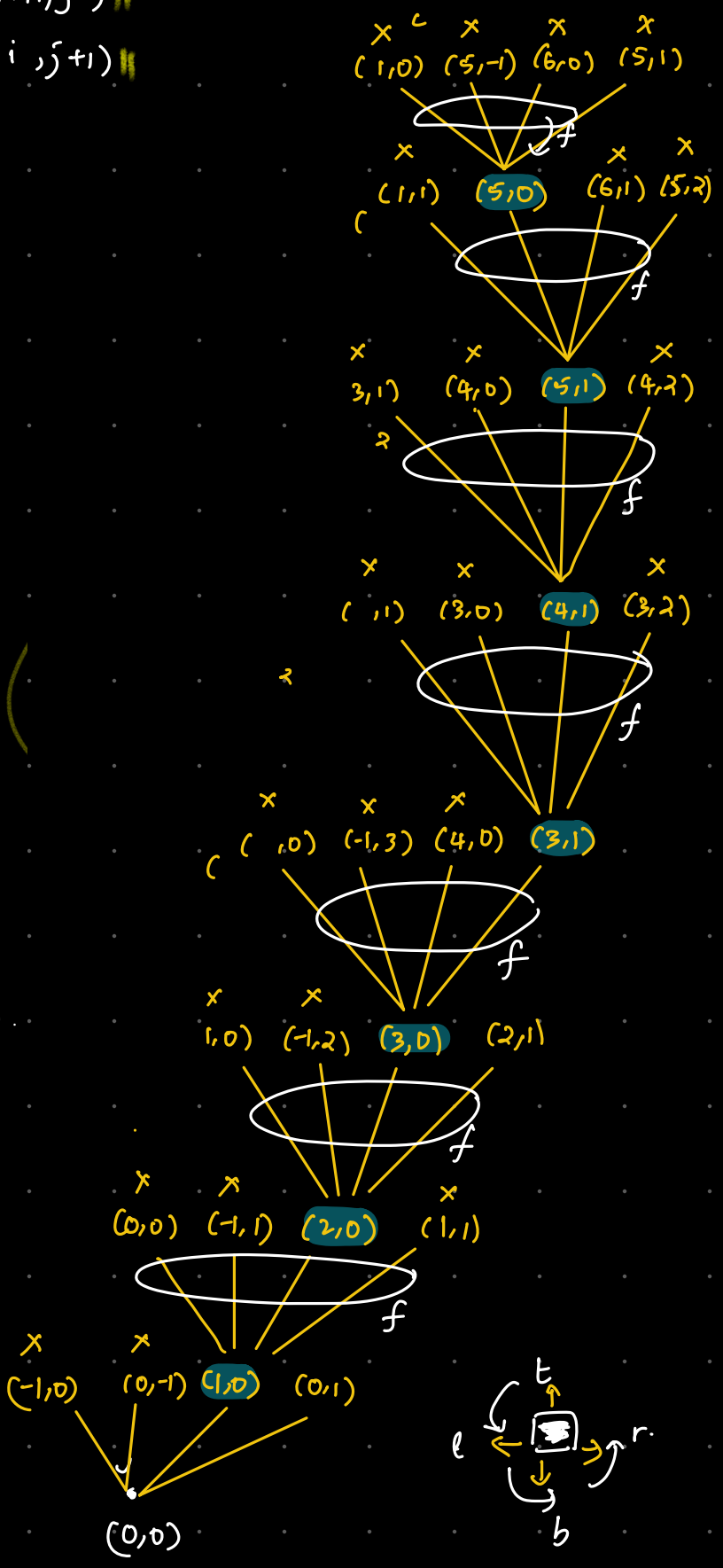
→ arr [N][M] — 0 (empty cell), 1 (blocked cell), 2 (visited)

```
boolean checkPath (arr[N][M], i, j)
    if (i==N-1 && j=M-1)
        return true
    if (i<0 || j<0 || i>=N || j>=M || arr[i][j]==1 || arr[i][j]==2)
        return false
    arr[i][j] = 2
    return checkPath (arr, i-1, j) ||
           checkPath (arr, i, j-1) ||
           checkPath (arr, i+1, j) ||
```

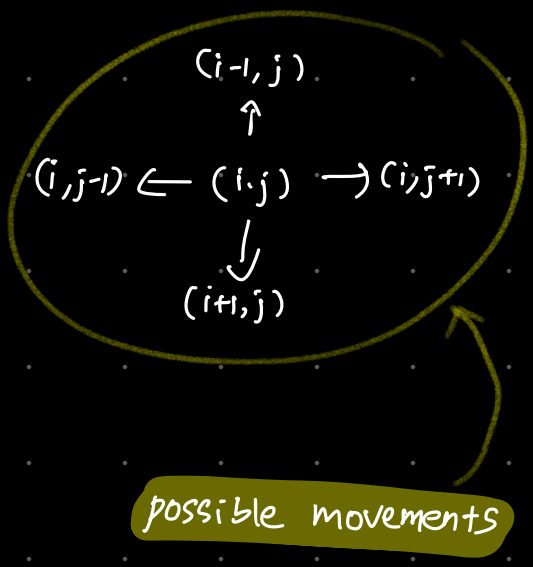
checkPath (arr, i, j+1)







```
dx = [-1, 0, 1, 0]
dy = [0, -1, 0, 1]
```

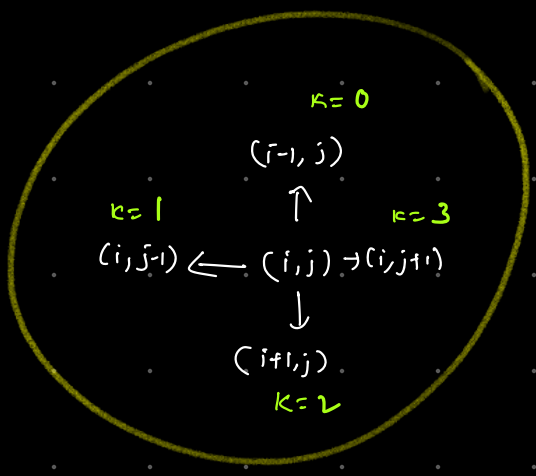



```
boolean checkPath(arr[N][M], i, j)
    if(i==N-1 && j==M-1)
        return true

    arr[i][j] = 2.
    dx = [-1, 0, 1, 0]
    dy = [0, -1, 0, 1]

    for(k=0; k<4; k++)
        nI = i + dx[k]
        nJ = j + dy[k]
        if(nI>=0 && nJ>=0 && nI<N && nI<M && arr[nI][nJ]==0)
            return checkPath(arr, nI, nJ)

    return false
```



TC: $O(4*N*M) = O(N*M)$

SC: $O(N*M)$ (∵ In worst case, every fn call will be stored)

① Generate all permutations of a string, without modifying the string.

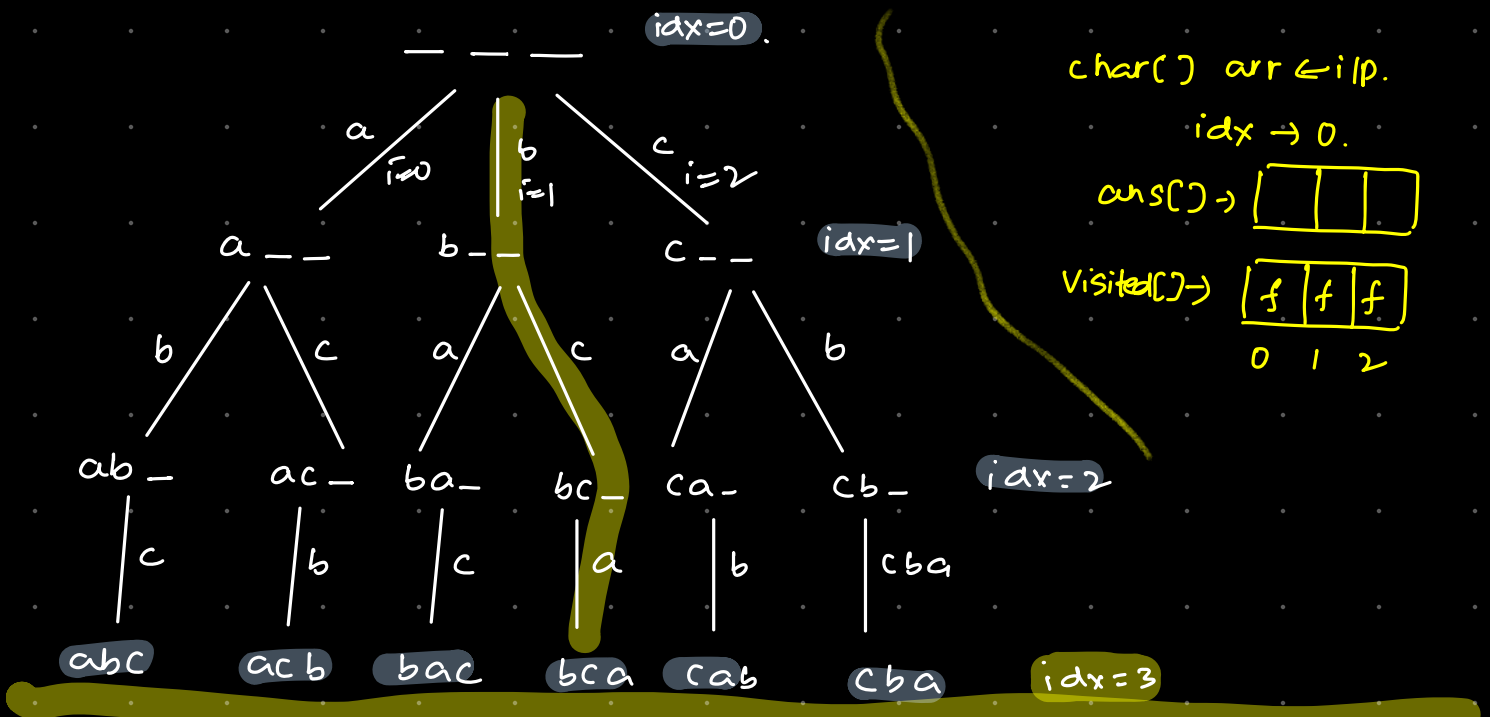
char[] arr → contains unique characters.

A: [a b c]

o/p: [abc acb bac bca cab cba]

3 * 2 * 1 = 6 => 3!

N N-1 N-2 ... *1 = N!



char[] arr ← i/p.

idx → 0.

ans[] →

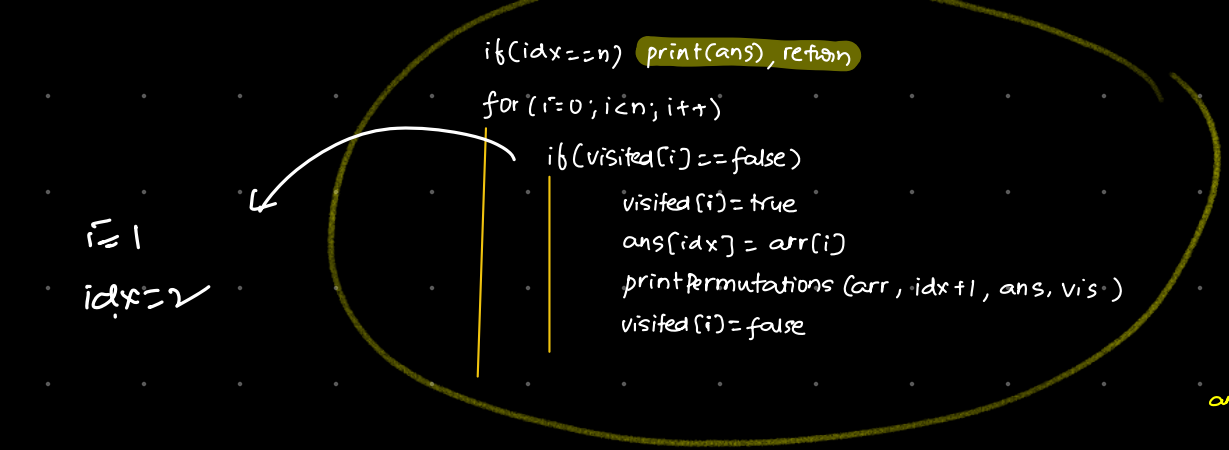
Visited[] → | f | f | f |

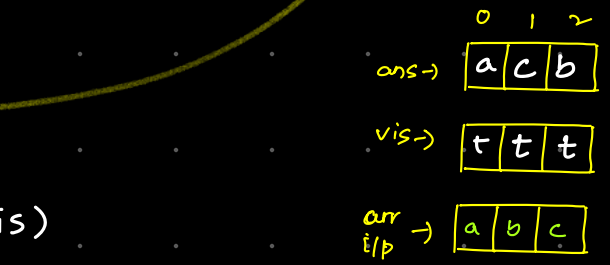
0 1 2

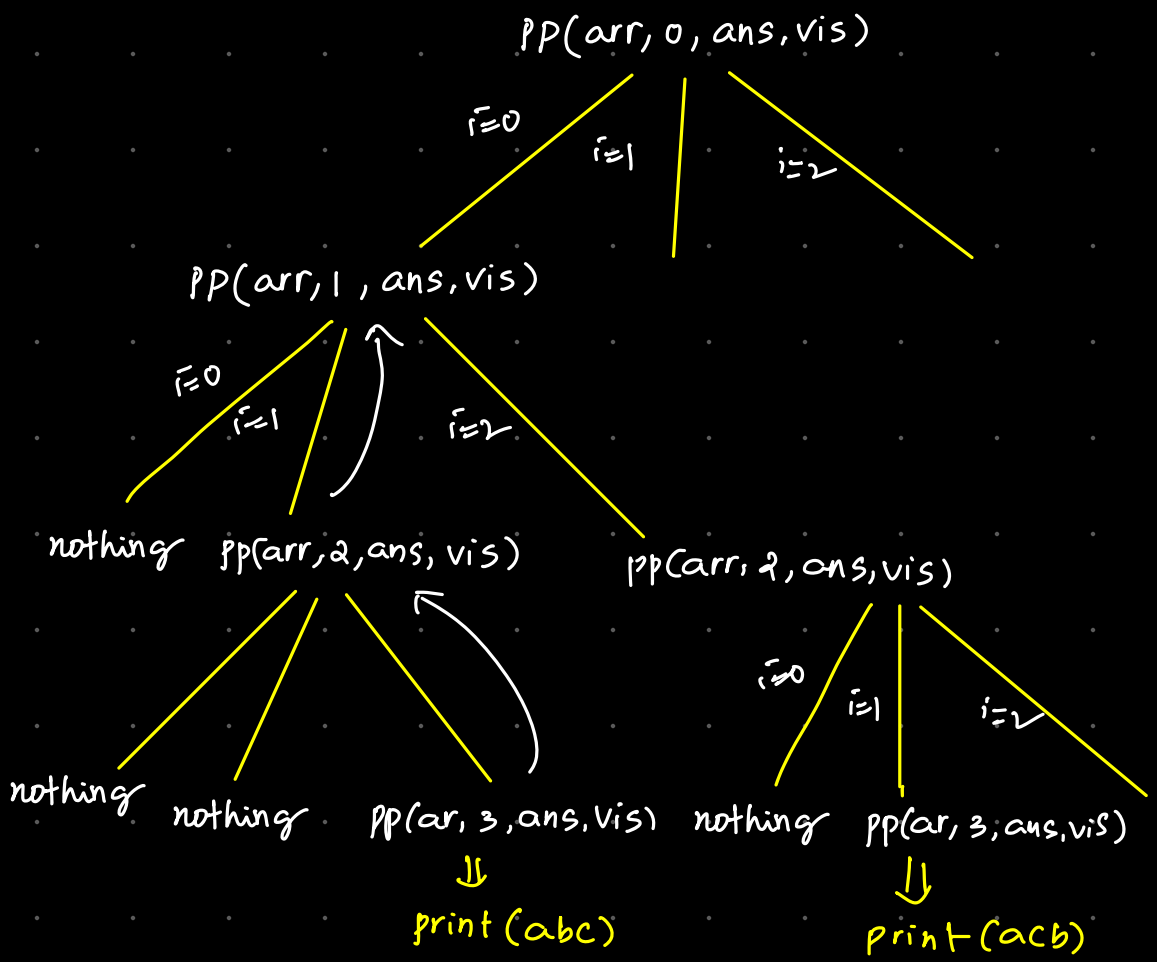
```
Void printPermutations(char[] arr, int idx, char[] ans, bool[] vis)
    if(idx==n) print(ans), return
    for(i=0; i<n; i++)
        if(visited[i]==false)
            visited[i]=true
            ans[idx] = arr[i]
            printPermutations(arr, idx+1, ans, vis)
            visited[i]=false
```

TC: O(n!)

SC: O(n)


if(idx==n) print(ans), return
for(i=0; i<n; i++)
if(visited[i]==false)
visited[i]=true
ans[idx] = arr[i]
printPermutations (arr, idx+1, ans, vis)
visited[i]=false
i=1
idx=2


0 1 2
ans→ a c b
vis→ t t t
arr i/p → a b c


pp(arr, 0, ans, vis)
i=0
i=1
i=2
pp(arr, 1, ans, vis)
i=0
i=1
i=2
nothing
pp(arr, 2, ans, vis)
pp(arr, 2, ans, vis)
nothing
nothing
pp(ar, 3, ans, vis)
print (abc)
i=0
i=1
i=2
nothing
pp(ar, 3, ans, vis)
print (acb)