23/8/2023

Backtracking - 1

Recursion → Solving a problem using its subproblem.

Backtracking → Trying all possibilities with recursion.

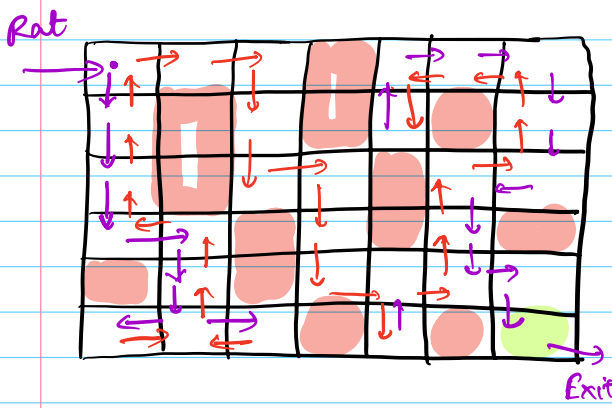
↓

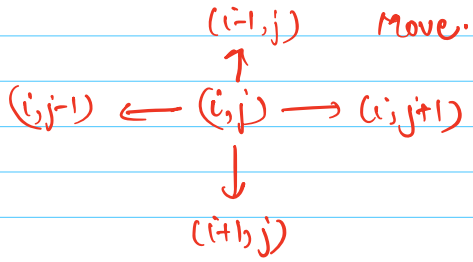
Bruteforce.

3 boxes → Find box with max chocolates → check all 3

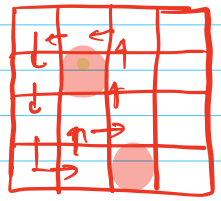
Q Rat in a maze → check if it is possible to go from top-left to bottom-right cell in a maze with some blocked cells.

I/P → A[][] → 0 (empty)
↳ 1 (blocked)





Ans = true

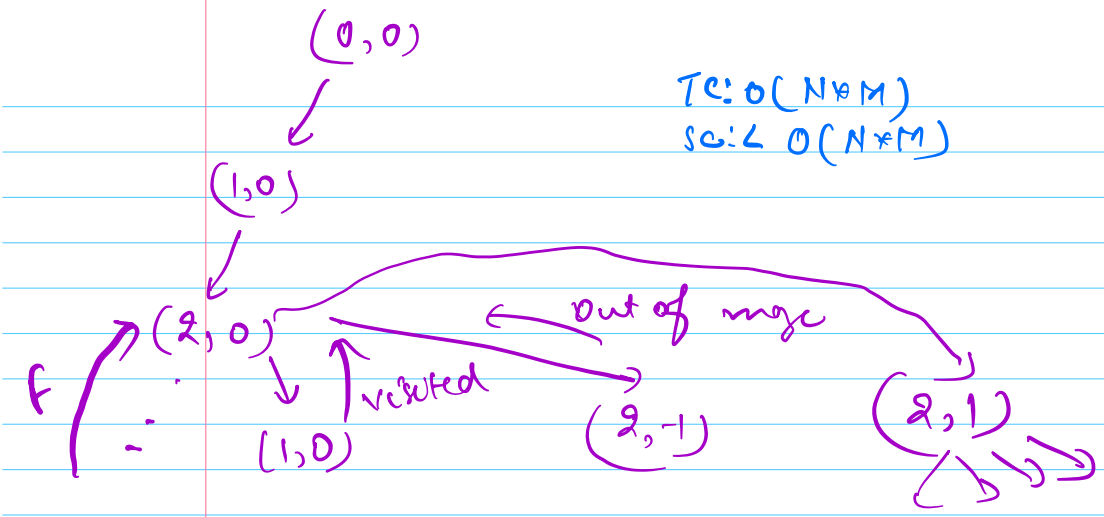


1> Rat is always inside the maze.
2> Rat is not going to any visited cell.
3> Rat is not going to any blocked cell.

```
// A[][] , N, M
if (A[N-1][M-1] == 1) return false;
boolean check(i,j){
    if (i == N-1 && j == M-1)
        return true //reached exit.
    if (i<0 || i>=N || j<0 || j>=M){
        return false;
    }
    if (A[i][j] == 1 || A[i][j] == 2)
        return false // blocked or visited
    A[i][j] = 2
    return check(i+1,j) || check(i-1,j) ||
           check(i,j-1) || check(i,j+1);
}
```

A[][] → 0 (empty)
↳ 1 (blocked)
↳ 2 (visited)

order is important



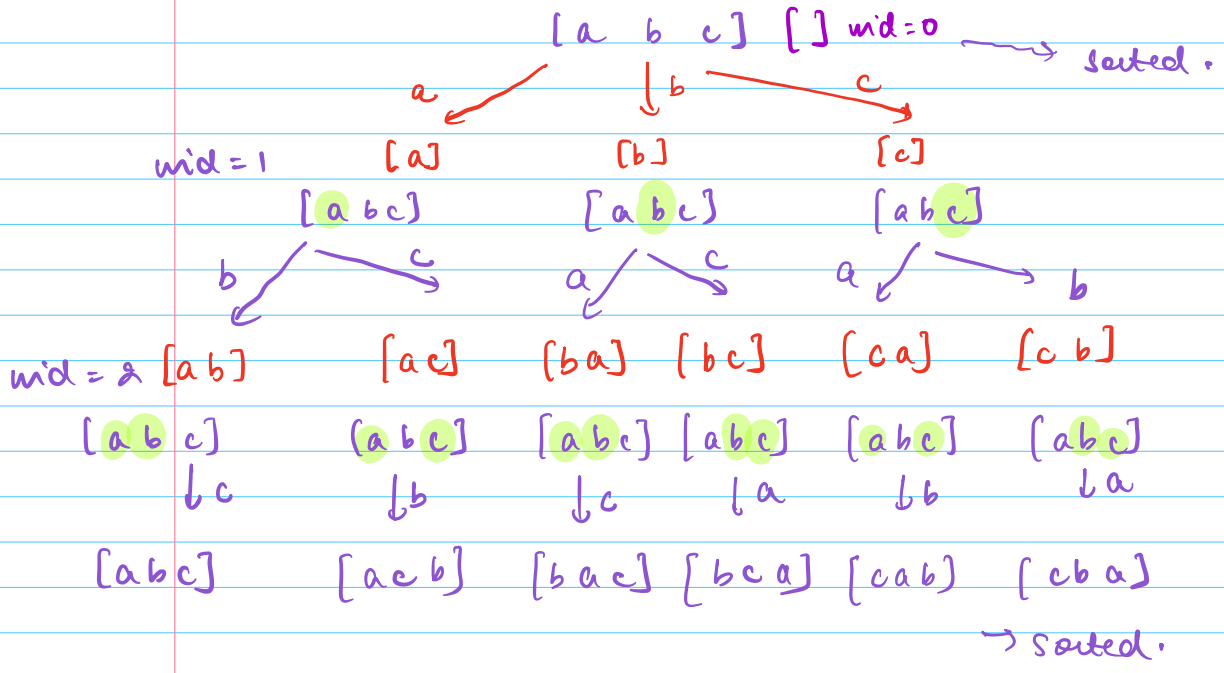
TC: O(N*M)
SC: O(N*M)

Q Given a char array with distinct elements. Print all the permutations of the array without modifying the i/p array.

A = [a b c] o/p ⇒ abc bac cab
acb bca cba

3 × 2 × 1 = 3! = 6

// permutation with distinct chars = N!



```
void permutation (A[], vst[], ans[], ind) {      N -> A.length
                                       ↑0
    if (ind == N){                     // Base case
        printArray(ans);
        return;
    }

    for i -> 0 to (N-1) {              // All possibilities.
        if (!vst[i]) {                 // Valid possibility
            vst[i] = true              // do
            ans[ind] = A[i]

            permutate(A, vst, ans, ind+1)    // Recursion.

            vst[i] = false;            // Undo
        }
    }
}
```

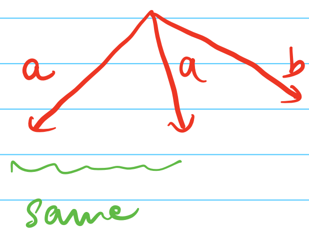
TC: O(N! * N)
SC: O (N)

Print the permutations in sorted order → Initially sort the i/p array.

Meet at 8:35 am IST

Q Print all unique permutations of the given char array.

A = [a a b] o/p → a a b   a b a   b a a

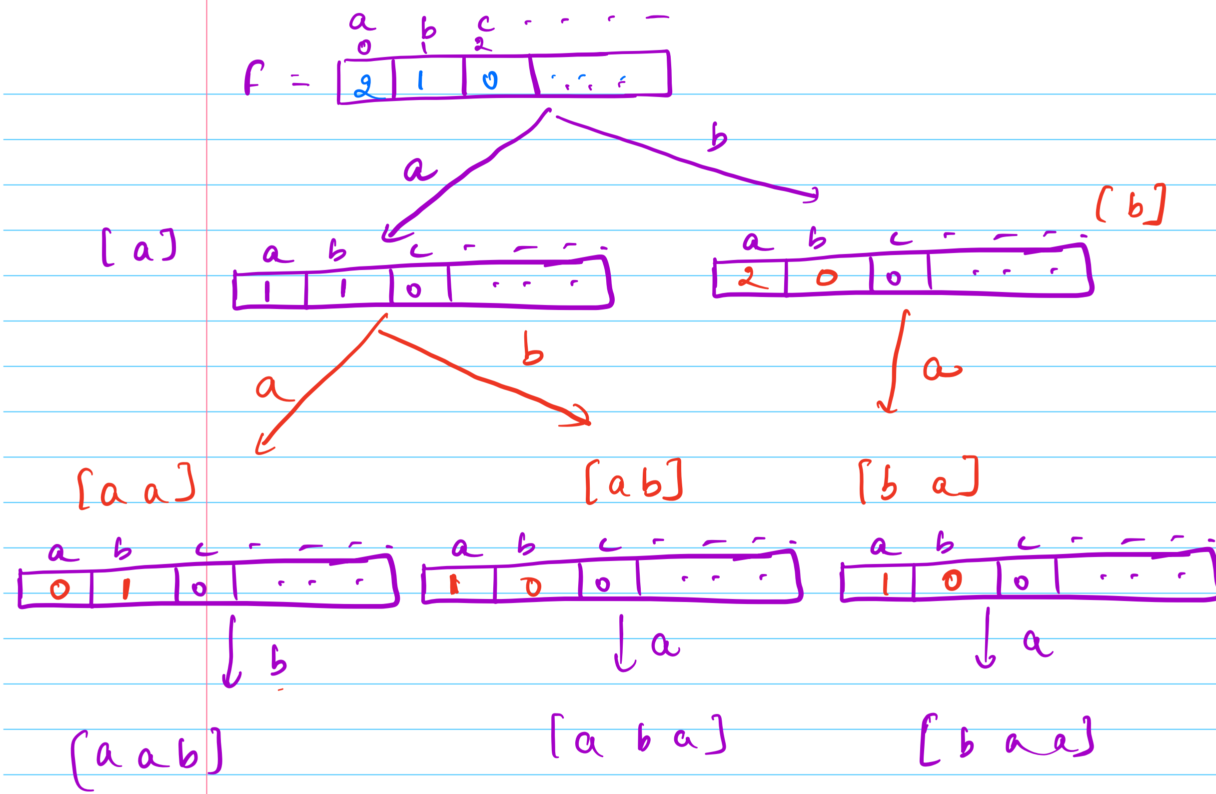



```
void permutation (F [] , N , ans[] , ind) {     N → A.length
                                      ↑0
    if ( ind == N){                // Base case
        printArray (ans);
        return;
    }

    for i → 0 to 25               // All possibilities.
        if (F[i] > 0) {           // Valid possibility
            F[i]--;               // do
            ans [ind] = char ('a' + i);

            permutate (A, N , ans , ind + 1)   // Recursion.

            F[i]++;               // Undo
        }
    }
}
```



Tc: O(N! * ~~N~~ 26) ≃ O (N!)

Sc: O (N)

Q4 Given a set of distinct integer A, return all possible subsets.

A = [1, 2, 3]
↓ ↓ ↓
2 × 2 × 2 = 8

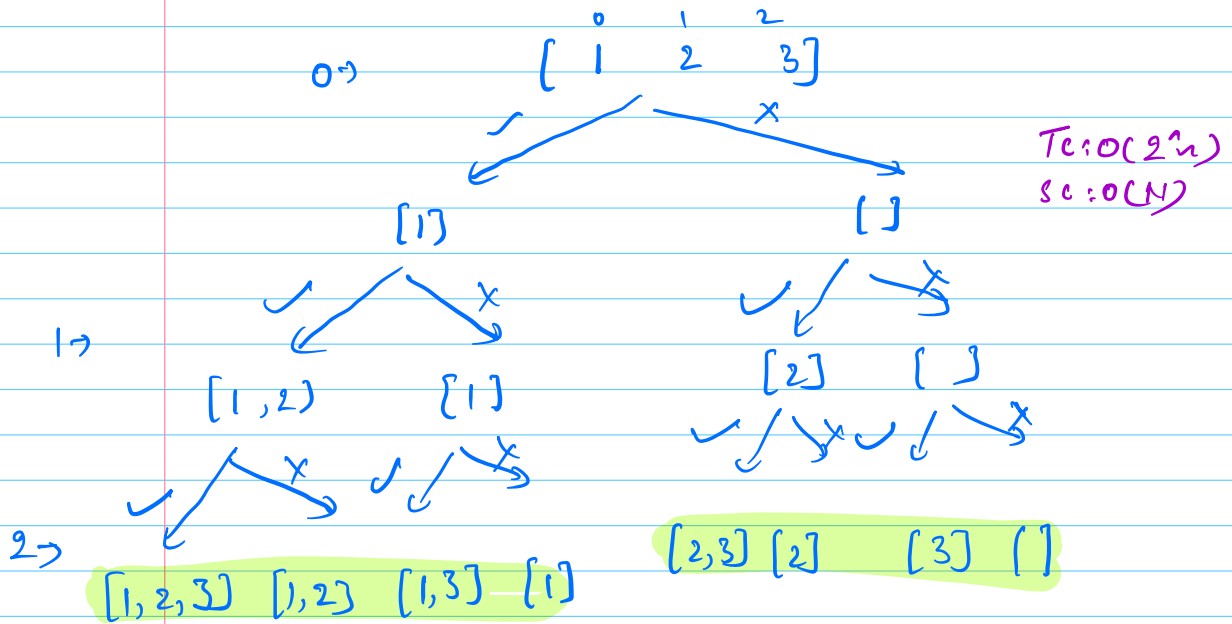
[]
[1]
[1, 2]
[1, 2, 3]
[1, 3]
[2]
[2, 3]
[3]

a b c ✓
a b d

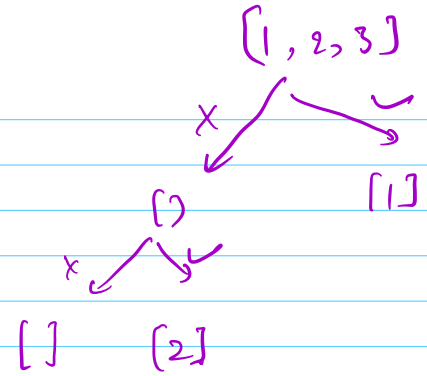
,

a b c ✓
∅

a ✓
a b



Sorted ans → 1) sort i/p
2) sort list of answer.


[1, 2, 3]
x
[1]
[]
x
[]
[2]