Welcome ☺

Agenda : Backtracking
3-4 problems.

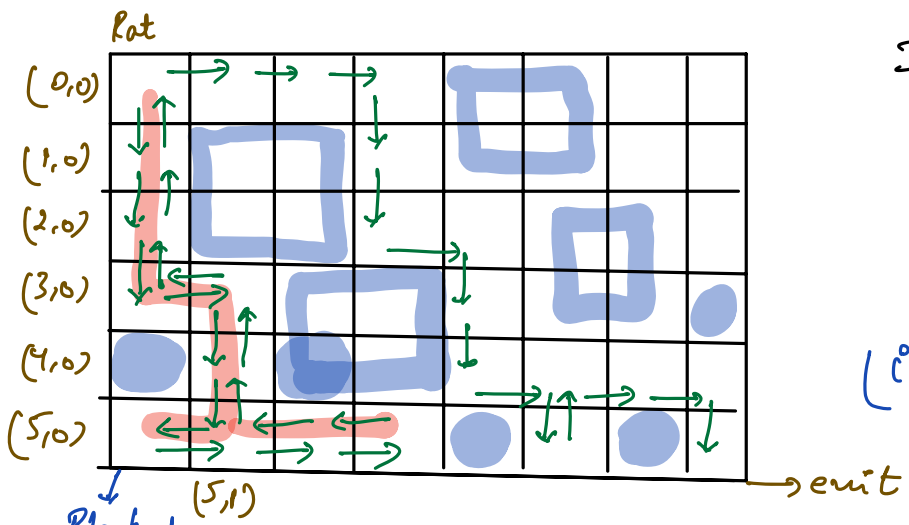
Recursion → Solving a problem using its sub problems.

Backtracking → Trying all possibilities with recursion.
↓
Bruteforce approach

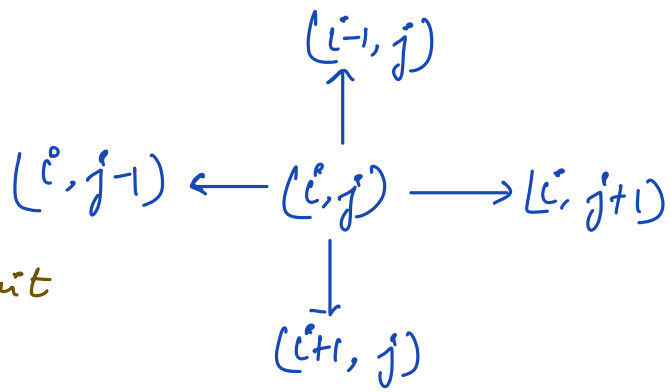
10 bones → Find out which bon has choclate → check all bones.
(bruteforce)

Q Rat in a Maze.

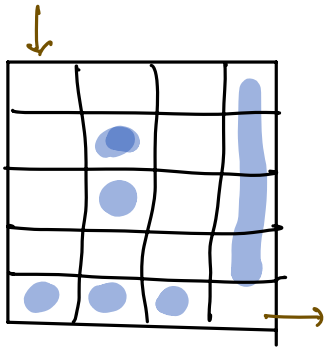
Check if it is possible to go from top left to bottom right cell in a maze with some blocked cells.



I/p ⇒ A[][] → 0 (empty)
→ 1 (blocked)



Blocked
↳ whenever blocked, backtrack one step and try alternate paths.



1) Rat is always inside the cage.
2) Rat should not go to any visited cell.
3) Rat cannot go to any blocked cell.

Code

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

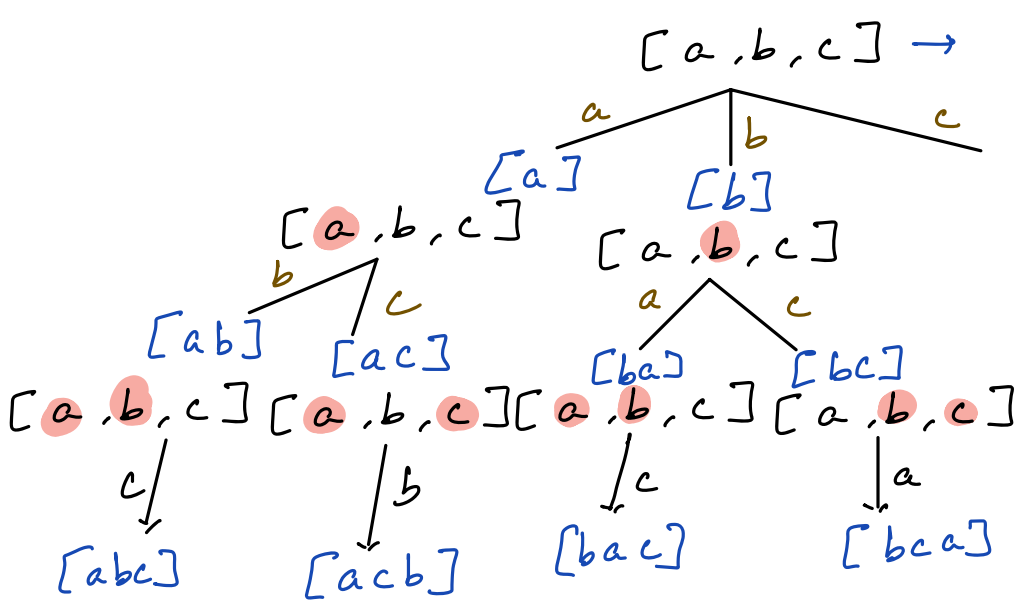
T.C → O(N*M)
S.C < O(N*M)

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

eg: [a b c] ⇒

abc
acb
bac
bca
cab
cba

3 * 2 * 1 = n!



Code

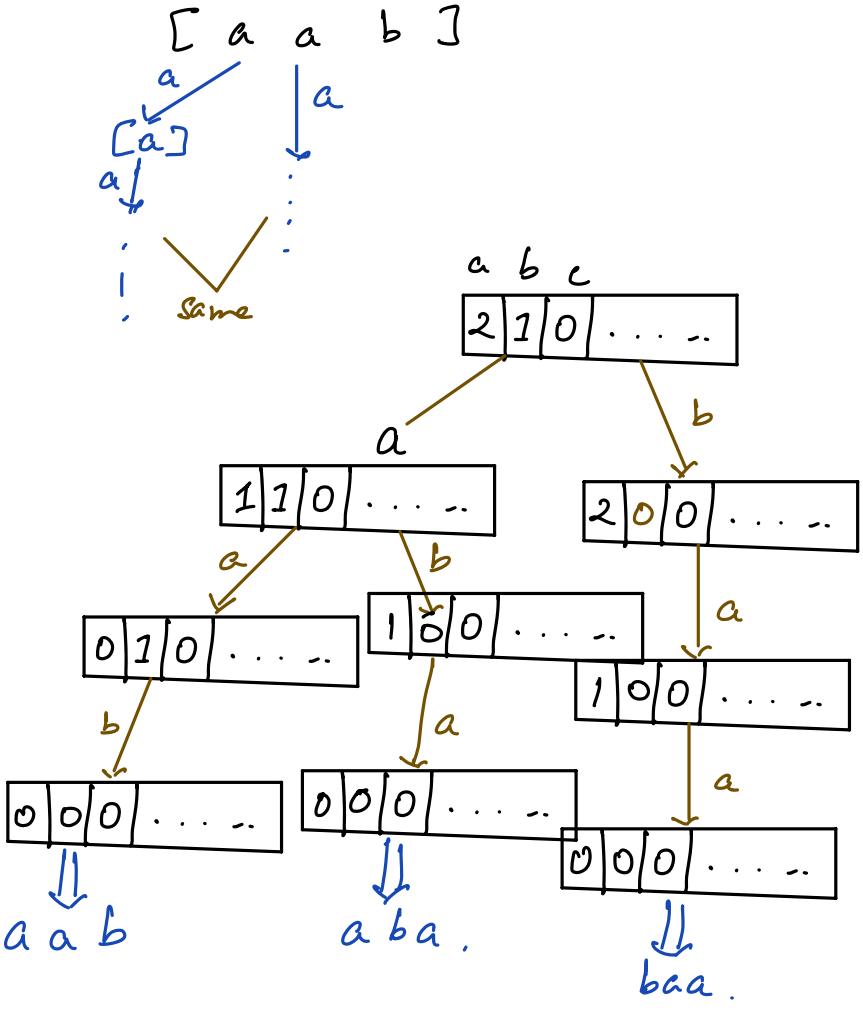
```
void permutation ( A[] , ans [] , vst[] , ind )      // length = N
{   if ( ind == N) {                                  // Base case
        printArray (ans)
        return ;
    }
    for ( i -> 0 to N-1) // trying all possibilities
    {
        if (!vst[i] ){ // valid possibility.
            vst[i] = true                  } DO
            ans[ind] = A[i]                }
            permutation ( A[] , ans [] , vst[] , ind + 1 )|| recursion
            vst[i] = false. // UNDO
        }
    }
}
```

T.C → O(N! * N)

S.C → O(N)

Q Print all unique permutations of the given char array

eg: [a a b] ⇒ a a b b a a a b a



Code

```
void permute ( F[] , ans[] , ind , N)        → length of array.
{  if( ind == N) {
   |     printArray(ans)
   |     return;
   }
   for( i → 0 to 25) // all possibilities.
   {
      if( F[i] > 0) { // valid case.
      |     F[i]--                          } DO
      |     ans[ind] = (char)(i+'a')        }
      |     permute ( F[] , ans[] , ind+1 , N) // recursion.
      |     F[i]++ // undo
      }
   }
}
```

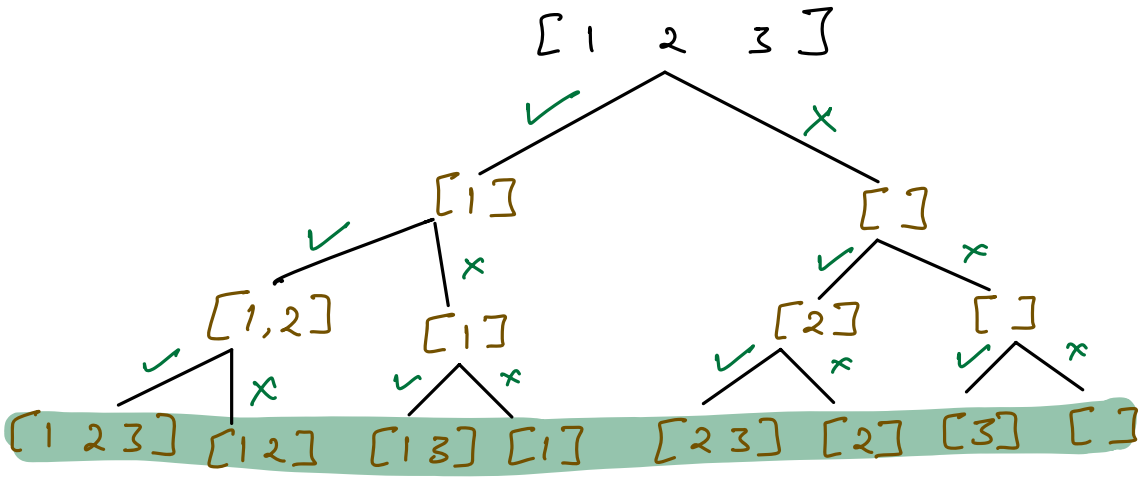
T.C → O(N!*26) = O(N!)

S.C → O(N)

Q Given a set of distinct integers A, return all possible subsets.

eg: A = [1 2 3] ⇒ [ ]
[1] [1 2] [1 3] [1 2 3]
[2] [2 3]
[3]

Select — — —
OR 2 * 2 * 2
NOT SELECT



H.W