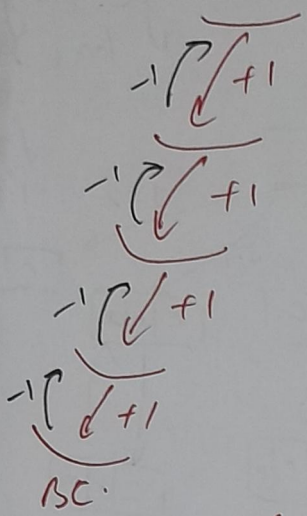
Date-20/10/23 Lecture-23 DnC & Backtracking class-2



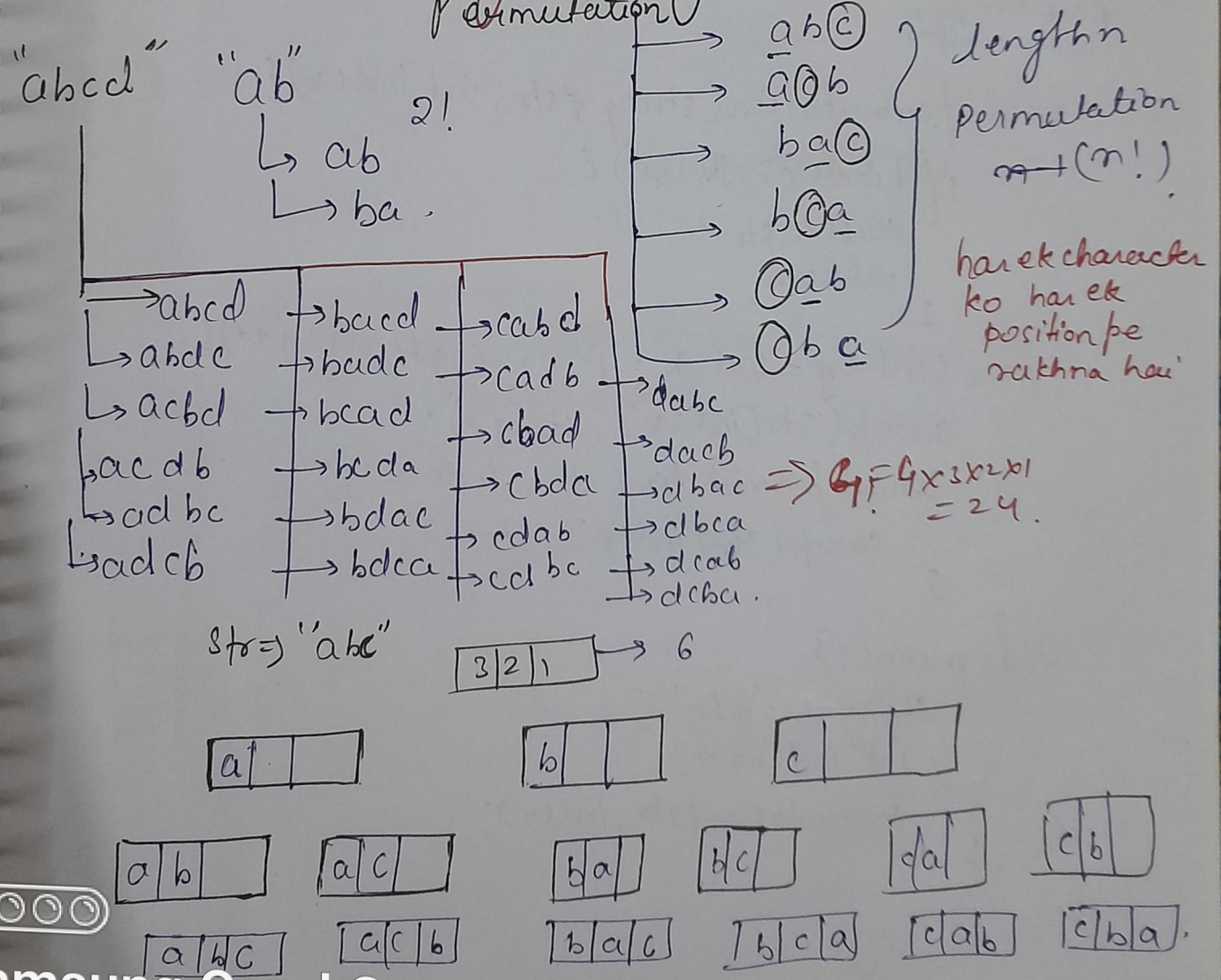
Backtracking → Explore all possible solution only one time.

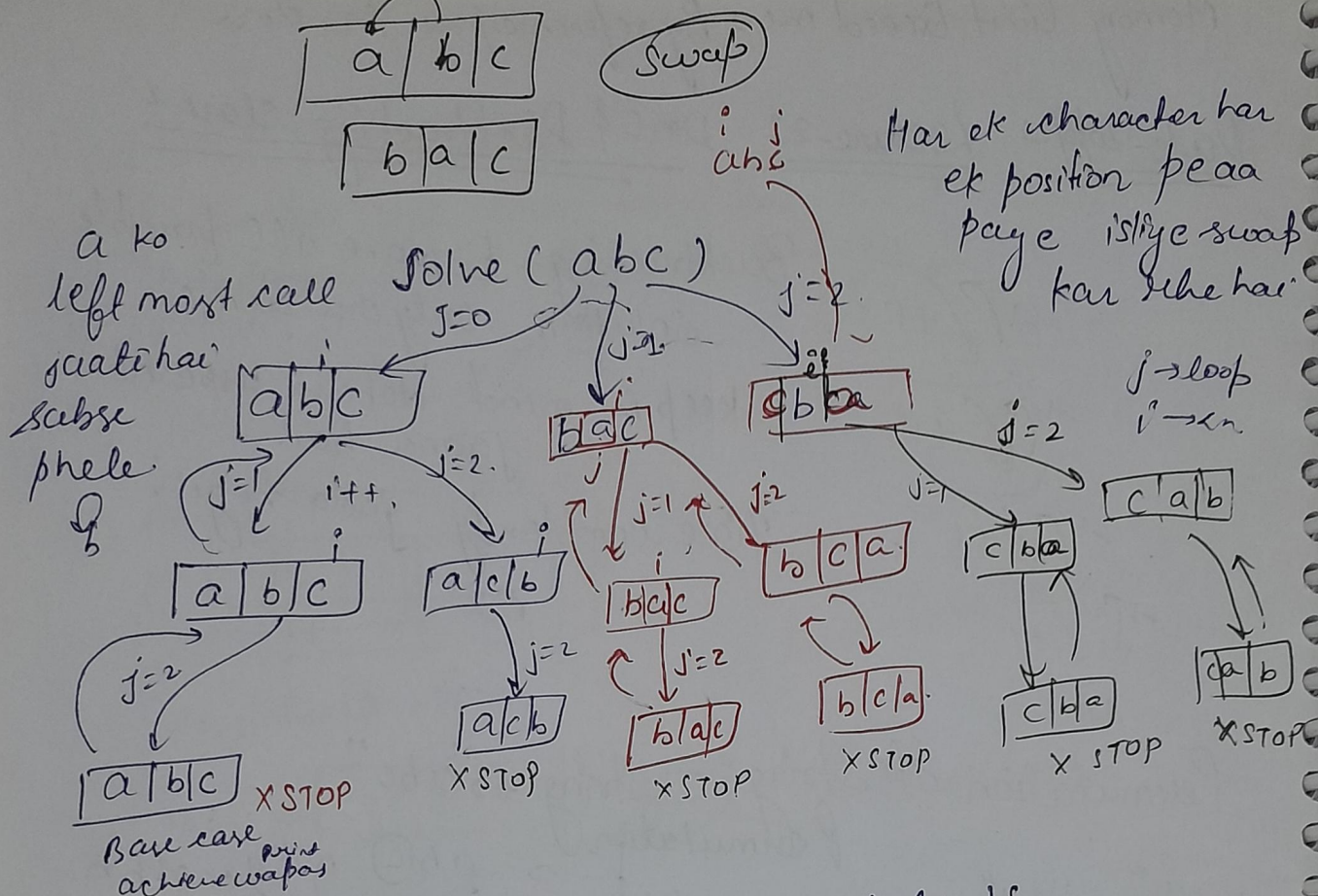
keep in mind dobara uspe nhi jaana hai.

Time complexity jada hogi.

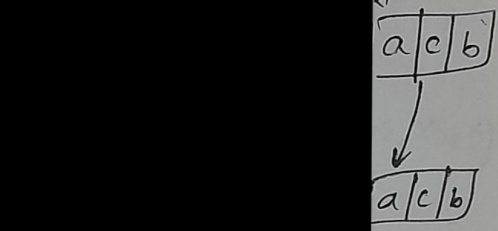
Permutation of String :- String → "abc"

Permutation






```
void permutation(string &str, int index){
    if(index >= str.size()){
        cout << str << " ";
        return;
    }
    for(int j = index; j < str.length(); j++){
        swap(str[index], str[j]);
        permutation(str, index+1);
        swap(str[index], str[j]);
    }
}
int main(){
    string str = "abc";
    int index = 0;
    permutation(str, index);
}
```


j=2
a c b
a c b

* Hard Problem

Rat in Maze Problem

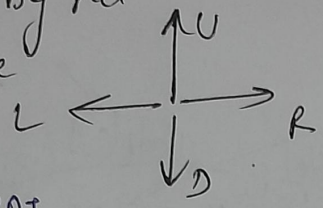
Print all possible ways →

0 → closed path

1 → Open path

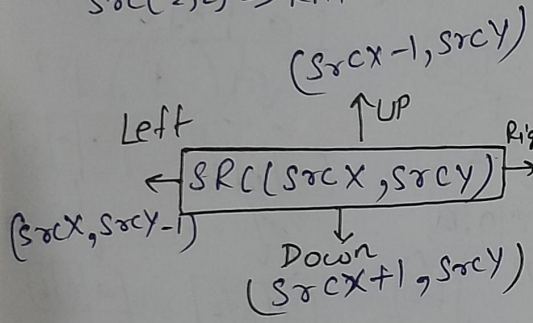
Movement by rat:

4 choice



	0	1	2	3
0	1	0	0	0
1	1	1	0	1
2	1	1	1	0
3	1	1	1	dest

src(2,2) ⇒ RAT



00	01	02	03	
10	11	12 UP	13	(srcx, srcy+1)
20	21 Left	22	23 Right	
30	31	32 Down	33 DES	

Eg.1 → Dry Run

f(0,0) → D|R|L|U

D vis=T

f(1,0)

D|R|L|U

R vis=T

f(1,1) ∞

D|R|L|U

vis=F

f(2,1) D|R|L|U

f(2,2)

vis=F

DR

vis=T

vis=F

vis(1,2)=T

f(1,2) → D|R|L|U

vis(2,1)=T

f(2,2)

DRDR

DRRD

condition

1) visited
2) Index Bound
3) closed Path

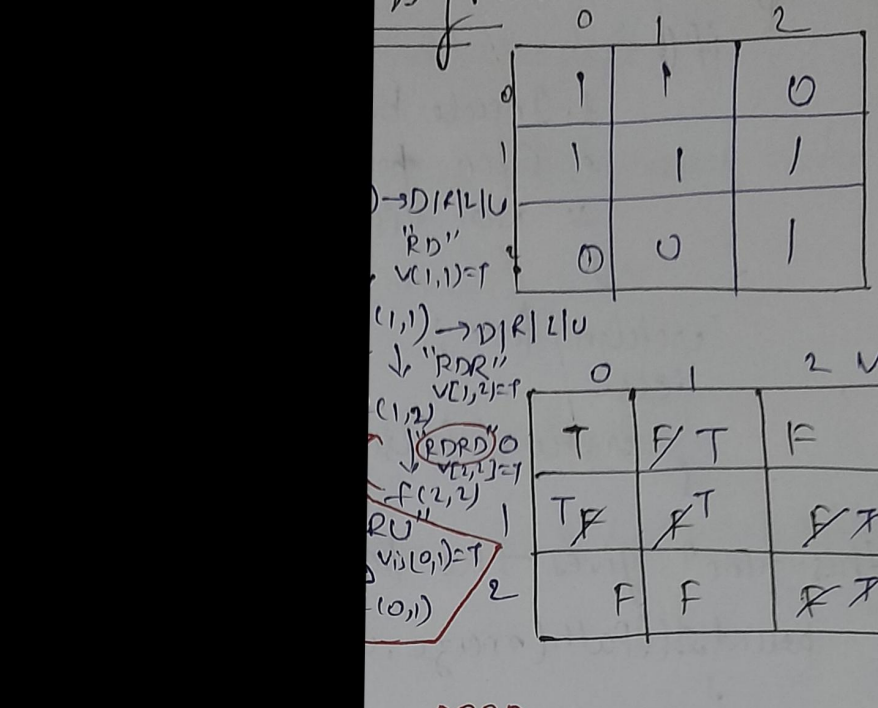
MAZE

1	1	0
1	1	1
0	0	1

VISITED

	0	1	2
0	T F	F	F
1	F T	F T	F T
2	F	F T	F T

Dry Run.



DRRD
RDRD.

```
{ {1,1,0},
  {1,1,0},
  {0,1,1}};
```

1 indicates open path.

(0,0)

= (row-1, col-1)

whether any position in maze
to reach destination or not

isSafe(maze, row, col, srcX, srcY, newX, newY, visited)

{

if(

1. Inside bound
2. Open path
3. Not visited

)

return true;

else

return false;

}

this func^n gives the path to reach destination

printAllPath(maze, row, col, srcX, srcY, output, visited)

{

//base case

if (reached destination)

print O/p and return

1.) //Down → newX = srcX + 1, newY = srcY

if (isSafe) {

// Mark visited

// push_back 'D' (D indicates down)

// recursive call

// backtracking → pop back & unmark visited

}

2.) //right → newX = srcX, newY = srcY + 1

if (isSafe) {

// mark visited

// push back 'R' (R → Right)

// Recusive call

// backtracking → pop back & unmark visited.

}

```
//left → newX = srcX , newY = srcY-1
        if(isSafe){
          //markvisited
          //push_back 'L' (indicates left)
          //recursive call
          //backtracking → pop back & unmark visited.
        }

//UP → newX = srcX-1 , newY = srcY
      if(isSafe){
        // mark visited
        // pushback 'U' (indicates Up)
        // recursive call
        // backtracking → pop back (unmark visited)
      }
```