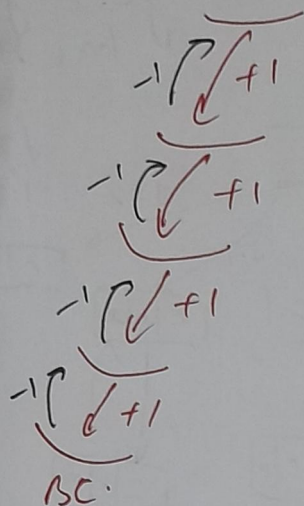


Date-20/10/23 Lecture-23 DnC & Backtracking class-2



Backtracking \rightarrow Explore all possible solution only one time.

keep in mind dobara use nhi jaana hai.

Time complexity jada hogi.

Permutation of String :- String \rightarrow "abc"

Permutation

"abcd" "ab"
 \rightarrow ab
 \rightarrow ba

2!

abcd	bacd	cabd
abdc	badc	cadb
acbd	bcad	cbad
acdb	bdca	cbda
adbc	bdac	cdab
adcb	bdca	cdac

abc
 ab@c
 a@b
 bac@c
 b@ca
 @ab
 @ba

length n
 permutation
 $n! = (n!)$

har ek character ko har ek position pe rakhtna hai

$$\Rightarrow 4! = 4 \times 3 \times 2 \times 1 = 24$$

str \rightarrow "abc"

3 | 2 | 1 \rightarrow 6

a | |

b | |

c | |

a b |

a c |

b a |

b c |

c a |

c b |

a b c

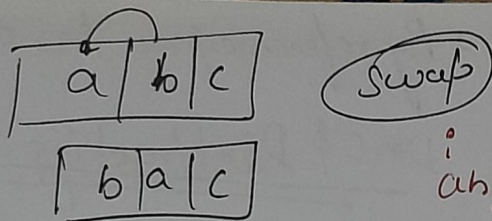
a c b

b a c

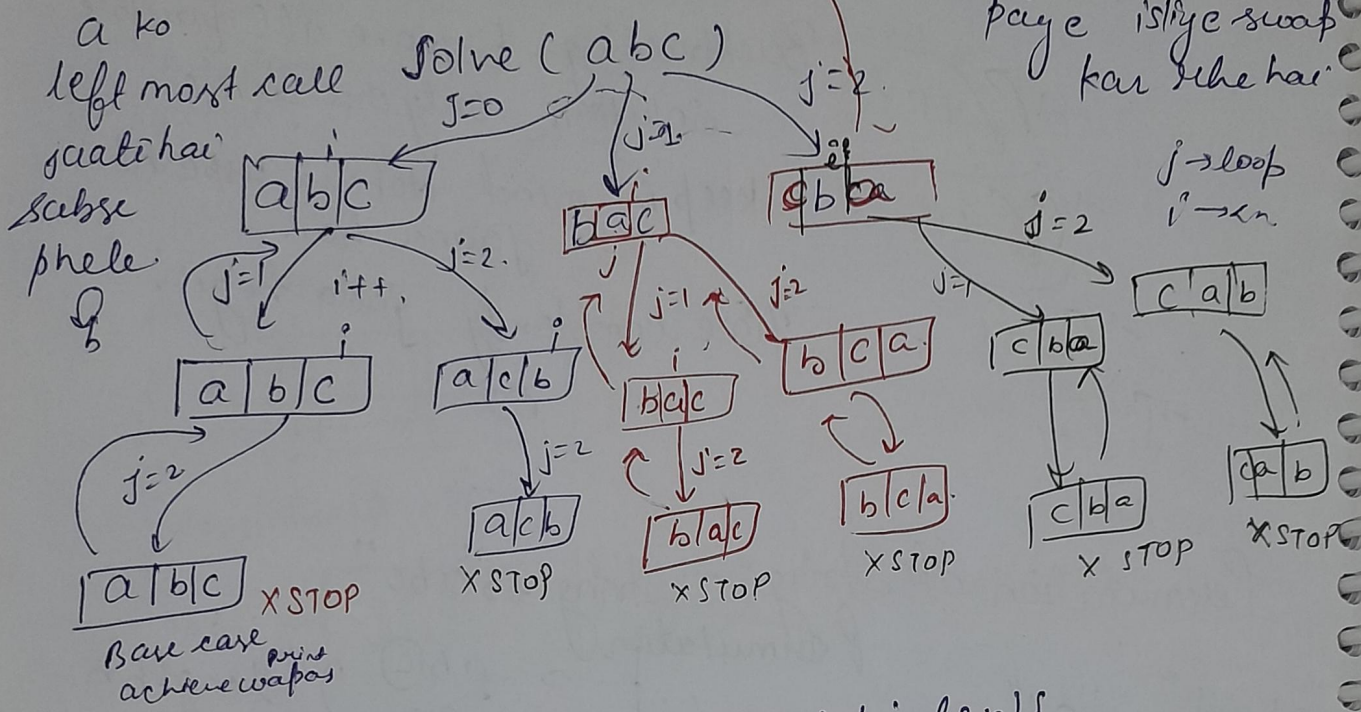
b c a

c a b

c b a



Har ek character har ek position pe aa paye islye swap kar rhe hai



```
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);
}
```



as
e here
code type.

aa

n,

2.



j=2

a	c	b
---	---	---



a	c	b
---	---	---

rekan

ction
perform
iya

* Hand Problem

Rat in Maze Problem

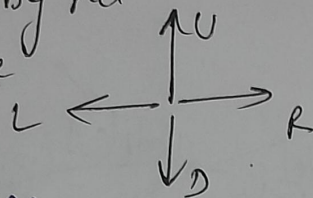
Print all possible ways →

0 → closed path

1 → Open path

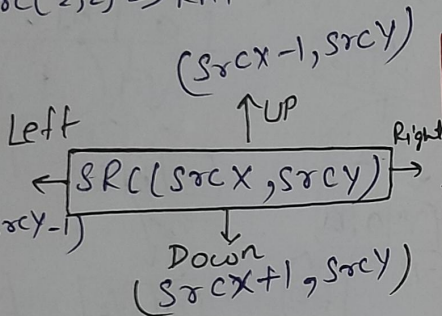
Movement by rat.

4 choice



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

src(2,2) ⇒ RAT



00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

up, left, right, down arrows from center (22) to (21), (23), (32), (12) respectively. (33) is labeled DES.

(srcx, srcy)

Eg. 1 Dry Run

f(0,0) → D|R|L|U

vis = F, D vis = T

f(1,0)

vis = F, R vis = T

f(1,1)

vis = F, L vis = T

f(2,1)

vis = F, U vis = T

f(2,2)

vis = T

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
1	F	F	F
2	F	F	F

DRDR, DRRD

dry run.

	0	1	2
0	1	1	0
1	1	1	1
2	0	0	1

→ D|R|L|U

"RD"

$V(1,1)=T$

(1,1) → D|R|L|U

↓ "RDR"

$V(1,2)=F$

(1,2) → D|R|L|U

↓ "RRDR" 0

$V(2,1)=T$

↓ "RU"

$V(0,1)=T$

(0,1)

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

src = (0,0)

dest = (2,2)

more

D|R|L|U

DRRD

RDRD.

{ 1, 1, 0 },

{ 1, 1, 0 },

{ 0, 1, 13 };

1 indicates open path.

(0,0)

$n = (row-1, col-1)$

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 funcⁿ gives the path to reach destination

printAllPaths(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)

// Recursive call

// backtracking → pop back & unmark visited.

}

// left \rightarrow new $x = \text{src } x$, new $y = \text{src } y - 1$

if (isSafe) {

// mark visited

// push back 'L' (indicates left)

// recursive call

// backtracking \rightarrow pop back & unmark visited.

}

// UP \rightarrow new $x = \text{src } x - 1$, new $y = \text{src } y$

if (isSafe) {

// mark visited

// push back 'U' (indicates up)

// recursive call

// backtracking \rightarrow pop back (unmark visited)

}