Todays Content:

- → permutations
- → N-Queens

{ Every one runs race of life at their own pace }

IIT : { You will have achieving friendly

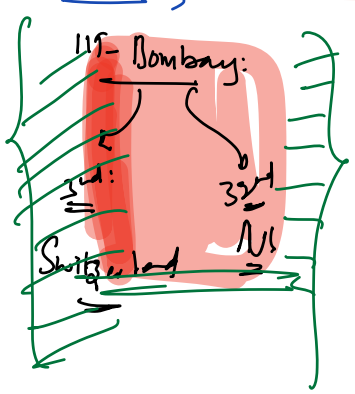
addict

IIT → 400 BITS →

Mains: 170

Rank: 25,000

Expect: 2,000



College

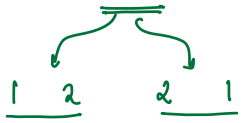
Job: → Microsoft

→ Amazon

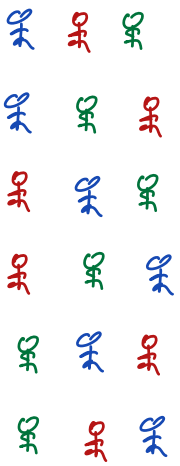
1) Given N distinct numbers, print all permutations

↳ How many different ways we can arrange

N = 2 : 1 2



N = 3 :



Both will have same permutations

N = 3 :

1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

N = 3 :

3	1	2
3	2	1
1	2	3
1	3	2
2	1	3
2	3	1

N = 5 : → Elements can be anything

1 2 3 4 5

1 — 2 3 4 5 → 24
permutate 4 positions

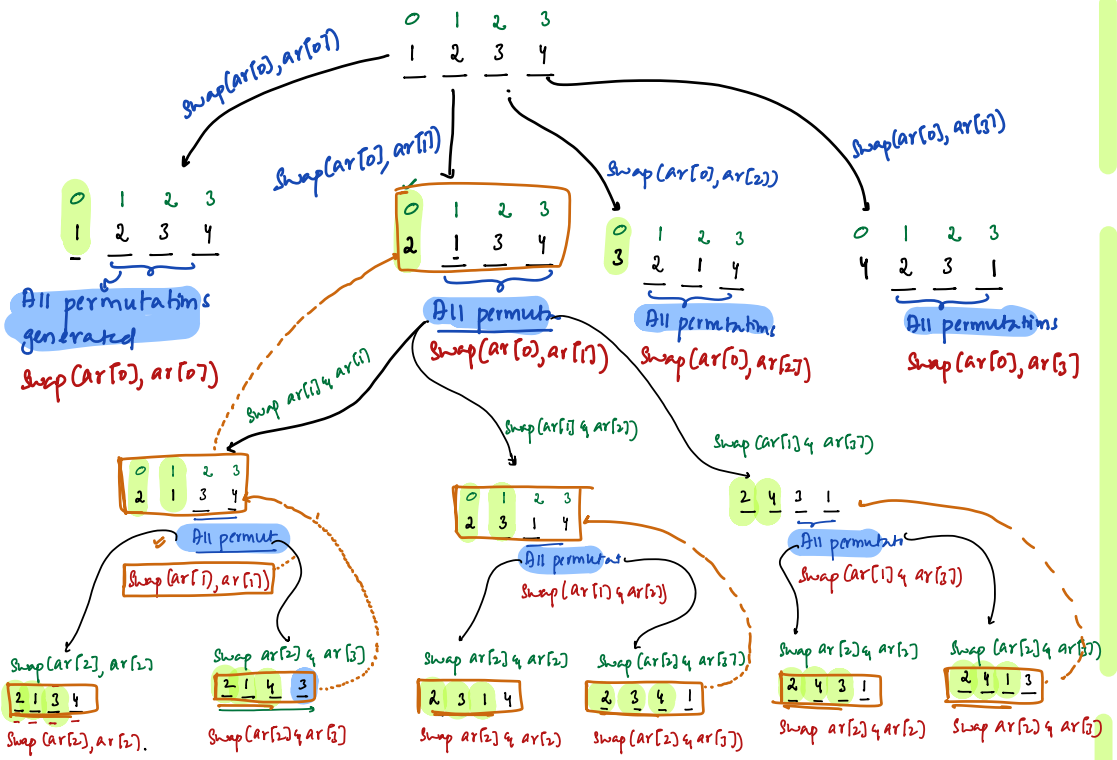
2 — 1 3 4 5 → 24
permutate 4 positions

3 — 1 2 4 5 → 24
permutate 4 positions

4 — 1 2 3 5 → 24
permute 4 positions

5 — 1 2 3 4 → 24
permute 4 positions

24 * 5 ⇒ 120 = 5!



Parameters:

ar[], N, i

- ar[], N: we are changing "given input[] itself"
- i: current index we need to fix

functioncalls:

depends on index we are present

pass array by reference

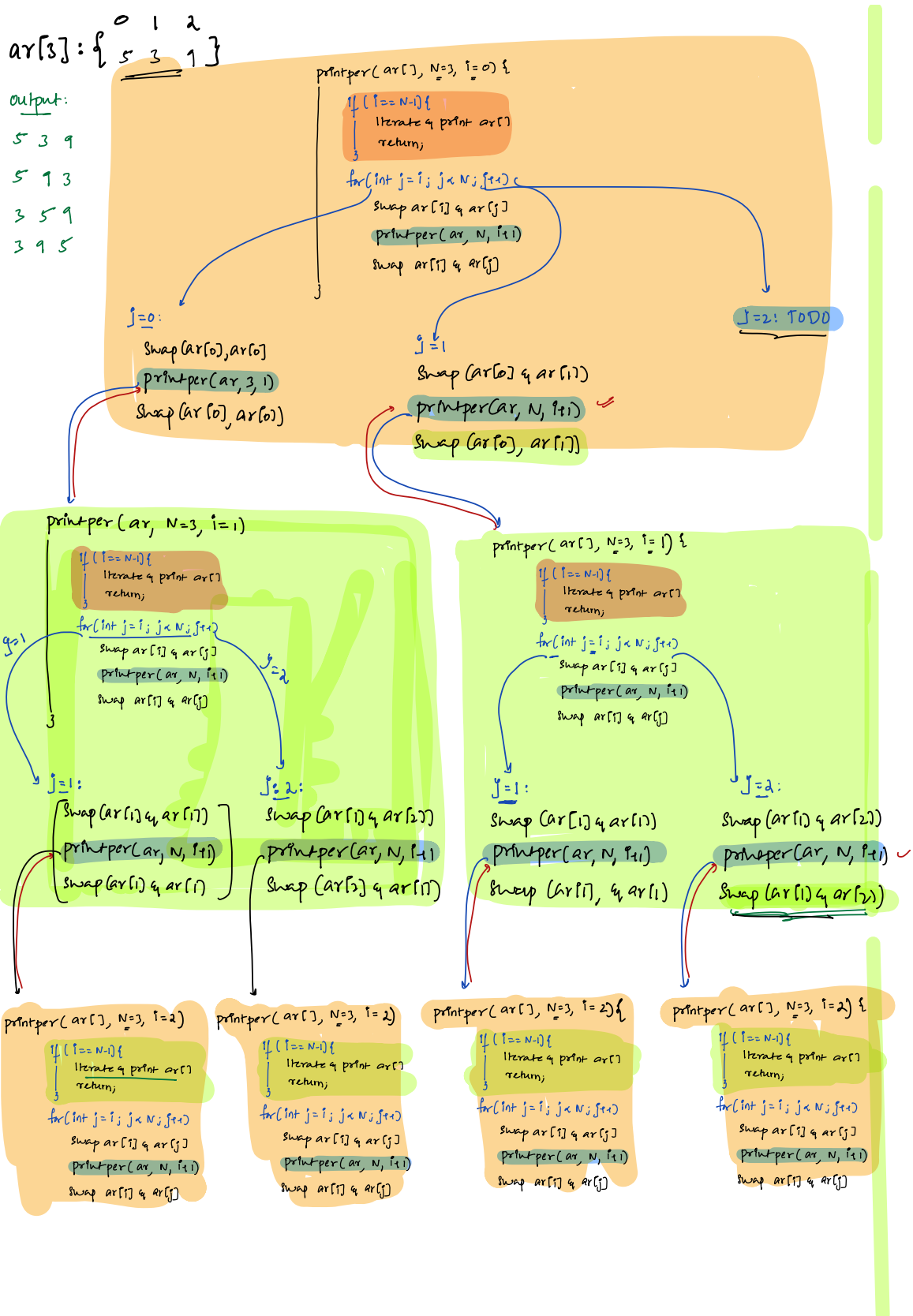
```
void printper(int ar[], int N, int i)
    if (i == N-1) {
        Iterate & print ar[]
        return;
    }

    // We have to fix ith index element
    for(int j = i; j < N; j++) {
        Swap ar[i] & ar[j]
        printper(ar, N, i+1) // generate permutations
        Swap ar[i] & ar[j]
    }
}
```

permutations

TC: N! * N (↳ iterate & print it)

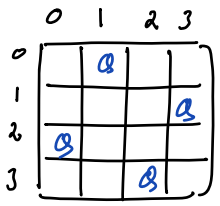
SC: O(N)


ar[3]: { 5 3 1 }
0 1 2
output:
5 3 1
5 1 3
3 5 1
3 1 5
printper(ar[], N=3, i=0) {
if (i == N-1) {
Iterate & print ar[]
return;
}
for(int j = i; j < N; j++)
Swap ar[i] & ar[j]
printper(ar, N, i+1)
Swap ar[i] & ar[j]
}
j=0:
Swap(ar[0], ar[0])
printper(ar, 3, 1)
Swap (ar[0], ar[0])
j=1
Swap (ar[0] & ar[1])
printper(ar, N, i+1)
Swap (ar[0], ar[1])
j=2: TODO
printper (ar, N=3, i=1)
if (i == N-1) {
Iterate & print ar[]
return;
}
for(int j = i; j < N; j++)
Swap ar[i] & ar[j]
printper(ar, N, i+1)
Swap ar[i] & ar[j]
j=1
j=2
j=1:
Swap (ar[1] & ar[1])
printper(ar, N, i+1)
Swap (ar[1] & ar[1])
j=2:
Swap (ar[1] & ar[2])
printper(ar, N, i+1)
Swap (ar[2] & ar[1])
printper(ar[], N=3, i=1) {
if (i == N-1) {
Iterate & print ar[]
return;
}
for(int j = i; j < N; j++)
Swap ar[i] & ar[j]
printper(ar, N, i+1)
Swap ar[i] & ar[j]
j=1:
Swap (ar[1] & ar[1])
printper(ar, N, i+1)
Swap (ar[1], & ar[1])
j=2:
Swap (ar[1] & ar[2])
printper(ar, N, i+1)
Swap (ar[1] & ar[2])
printper(ar[], N=3, i=2)
if (i == N-1) {
Iterate & print ar[]
return;
}
for(int j = i; j < N; j++)
Swap ar[i] & ar[j]
printper(ar, N, i+1)
Swap ar[i] & ar[j]
printper(ar[], N=3, i=2)
if (i == N-1) {
Iterate & print ar[]
return;
}
for(int j = i; j < N; j++)
Swap ar[i] & ar[j]
printper(ar, N, i+1)
Swap ar[i] & ar[j]
printper(ar[], N=3, i=2) {
if (i == N-1) {
Iterate & print ar[]
return;
}
for(int j = i; j < N; j++)
Swap ar[i] & ar[j]
printper(ar, N, i+1)
Swap ar[i] & ar[j]

// Given N*N matrix, print all valid placements of N Queens such that no queen can kill another queen

N=4:

obs1: → At every row we need place 1 queen

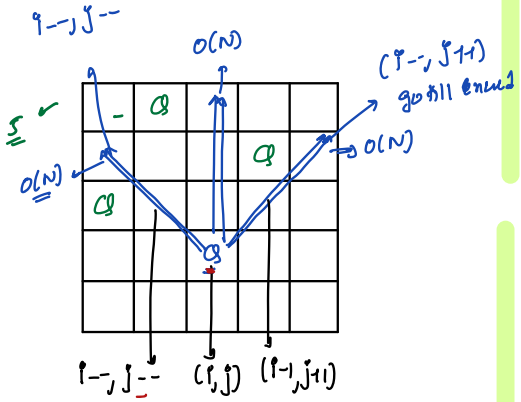


Parameters:
↳ mat[][], N, i → at what row we are keeping a queen

functional calls:
↳ At i^{th} row check if we can place at every column

```
void Queens(int mat[][], int N, int i){        TC → O(N!) * N
                                                  1) check function + Print
                                               SC → O(N)
    if(i == N){ // placed queen at every row
        // Iterate & print entire matrix
        return;
    }
    // At i^th we need to place a Queen
    for(j=0; j<n; j++){
        // We are trying to place at i, j
        { check i^th row, j^th col,
          Both left→right & right→left, diagnols }
        TC: O(N)
        if(check(mat, i, j)){   → return True, if i can place a queen at cell (i,j)
            mat[i][j] = 1; // 1 indicates presence of Queen
            Queen(mat, N, i+1)
            mat[i][j] = 0; // 0 indicate no queen
        }
    }
}
```

```
bool check(int mat[][], int i, int j) { → O(N)
    // whether we can keep queen at i, j
    Step1: Check Column:
    for(int r=0; r<i; r++){
        if(mat[r][j]==1) {return false}
    }
    Step2: left diagnol
    int r=i, c=j;
    while(r>=0 && c>=0)
        if(mat[r][c]==1) {return false}
        r--, c--
    }
    Step3: Right diagnol
    int r=i, c=j
    while(r>=0 && c<N)
        if(mat[r][c]==1) {return false}
        r--, c++
    }
    return true;
}
```

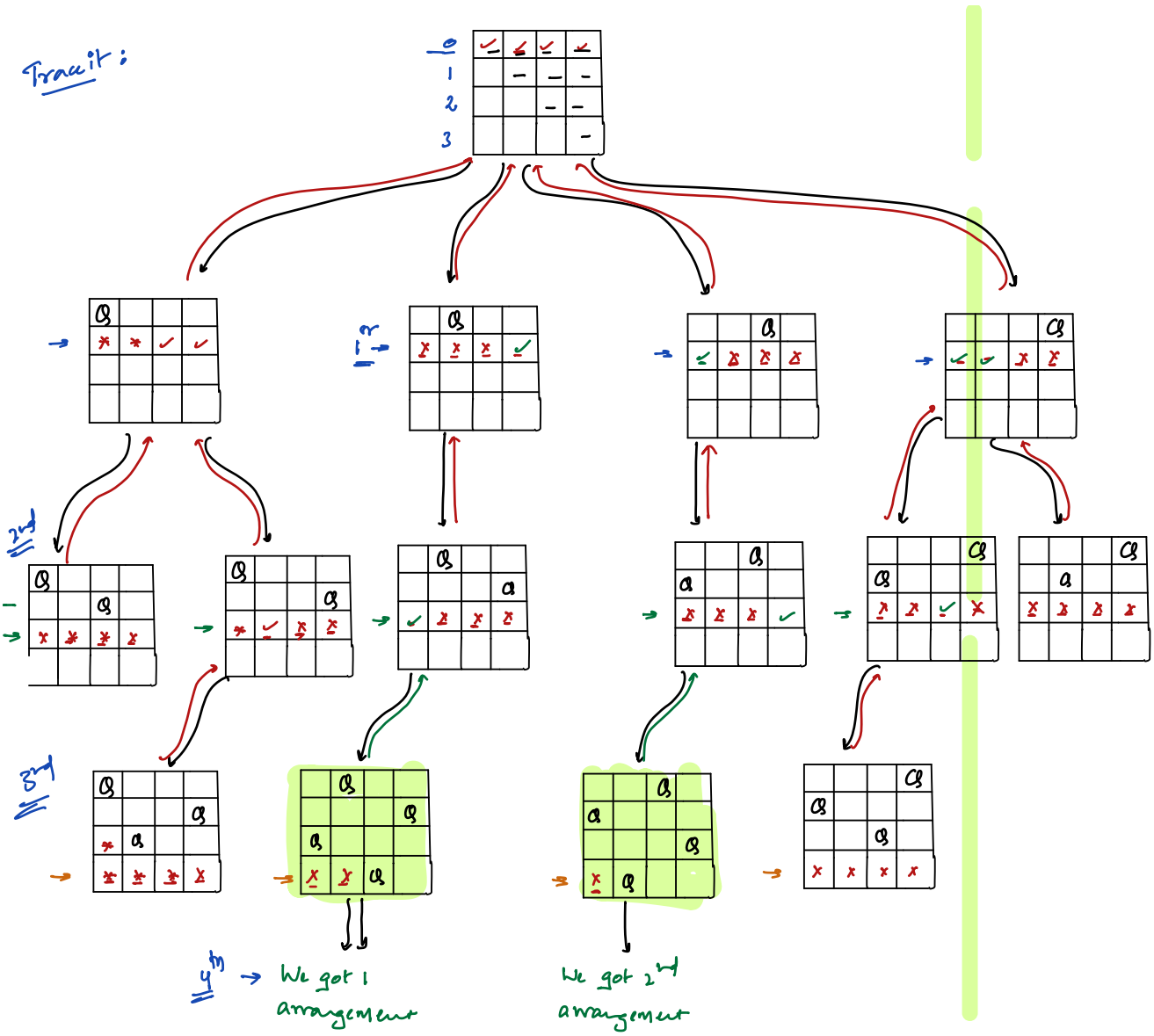


Job / Sue: Parents / Siblings, Teacher

↳ Mech: for loops: DSA + Competitive Dev: Goldman / Amazon
↳ her Big Brother → CTC: 25 → 12 CTC

↳ She: Tech: 4 lakhs, colleague higher → thanker earth → 2/3 DSA, Competitive → 18 CTC
↳ News paper add on her CTC

may backlogs? 40-50
attendance → 90%


Trace it :
0
1
2
3
or
2nd
3rd
4th
We got 1 arrangement
We got 2nd arrangement