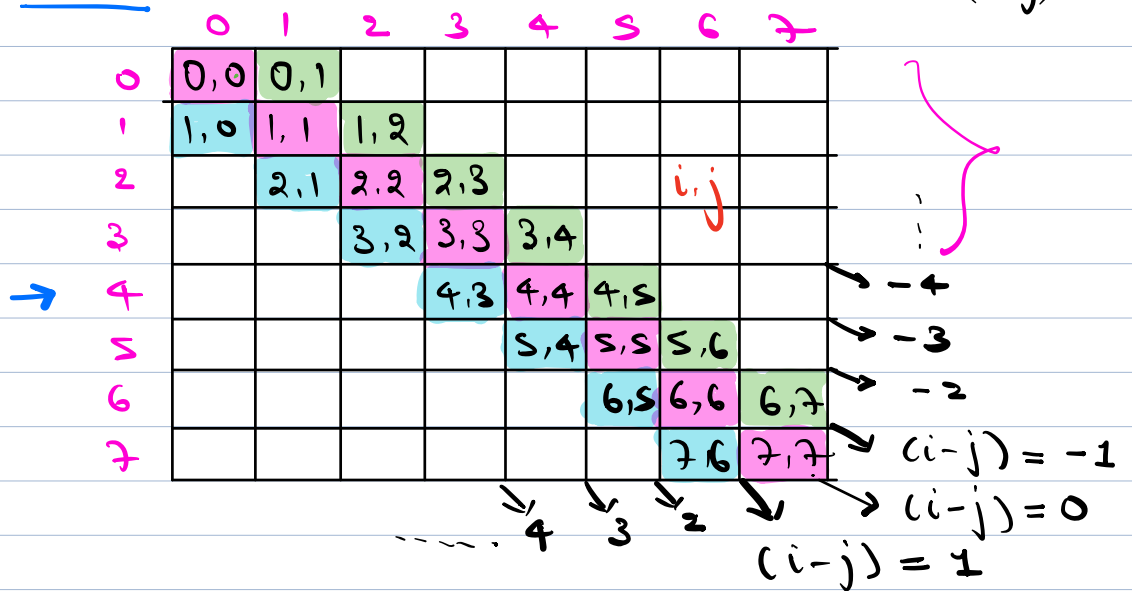
Code

```
int mat[][]

void Queens ( int mat[][] , row) {
```

Reference

L → R



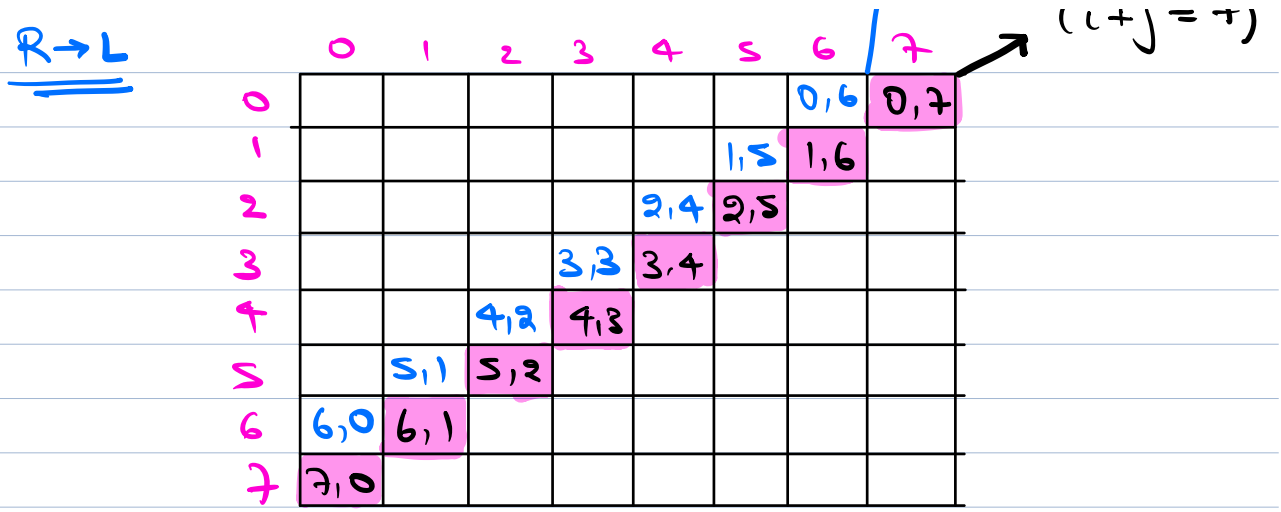
set < int > colums → All the cols in which queen is present

set < int > leftdiagonals

↑

(i-j)

↳ (i+j) = 6



set < integer > right diagonal.

Code

```
mat [ ] [ ] = {0};, set<int> columns,
set< int> ld;  set <int> rt.

void Queens ( mat[][], row ) {
                        ↑ 0

    if ( row == N) {

        print the mat[][] row wise
        return;

}

for (col = 0;  col < N;  col++) {

    if ( check ( row, col)) {

        mat[row][col] = 1;
```

do {
columns.add (col);
ld.add (row - col);
rd.add ((row + col));
Recursive call ← Queens (mat, row+1);
Undo {
mat[row][col] = 0;
columns.remove (col);
ld.remove (row - col);
rd.remove (row + col);
}
}
}

bool check (row, col) {
if (columns.contains (col)) {
return false;
}
if (ld.contains (row - col)) {
return false;
}
if (rd.contains (row + col)) {
return false;
}
return True;
}

Given N distinct numbers.

Print all permutations

↳ No. of arrangements possible

N = 2 → 1 2

1, 2
2, 1 Ans

} (2) ×

N = 3 → 1, 2, 3 ⇒ (3!) = 6

1, 2, 3
1, 3, 2
2, 1, 3
2, 3, 1
3, 1, 2
3, 2, 1

N = 5 → {1, 2, 3, 4, 5}

1 → (2, 3, 4, 5) → 4! = 24
2 → (1, 3, 4, 5) → 24
3 → (1, 2, 4, 5) → 24
4 → (1, 2, 3, 5) → 24
5 → (1, 2, 3, 4) → 24

24 × 5 = 120 = 5!

set<int> elements selected

Code

A[N] ⇒ N Elements.

```
set< integer > selected;
int permutation [N];

void printPermutation ( permutation, index ) {
    if (index == N) {
        print permutation array
        return;
    }

→   for (i=0; i<N; i++) {
        if (!selected.contains (A[i])){
            do {  permutation [index] = A[i];
                  selected.add (A[i]);

                  printPermutation (permutation,
                                    index+1);

                  selected.remove (A[i]);
            }
        }
    }
```

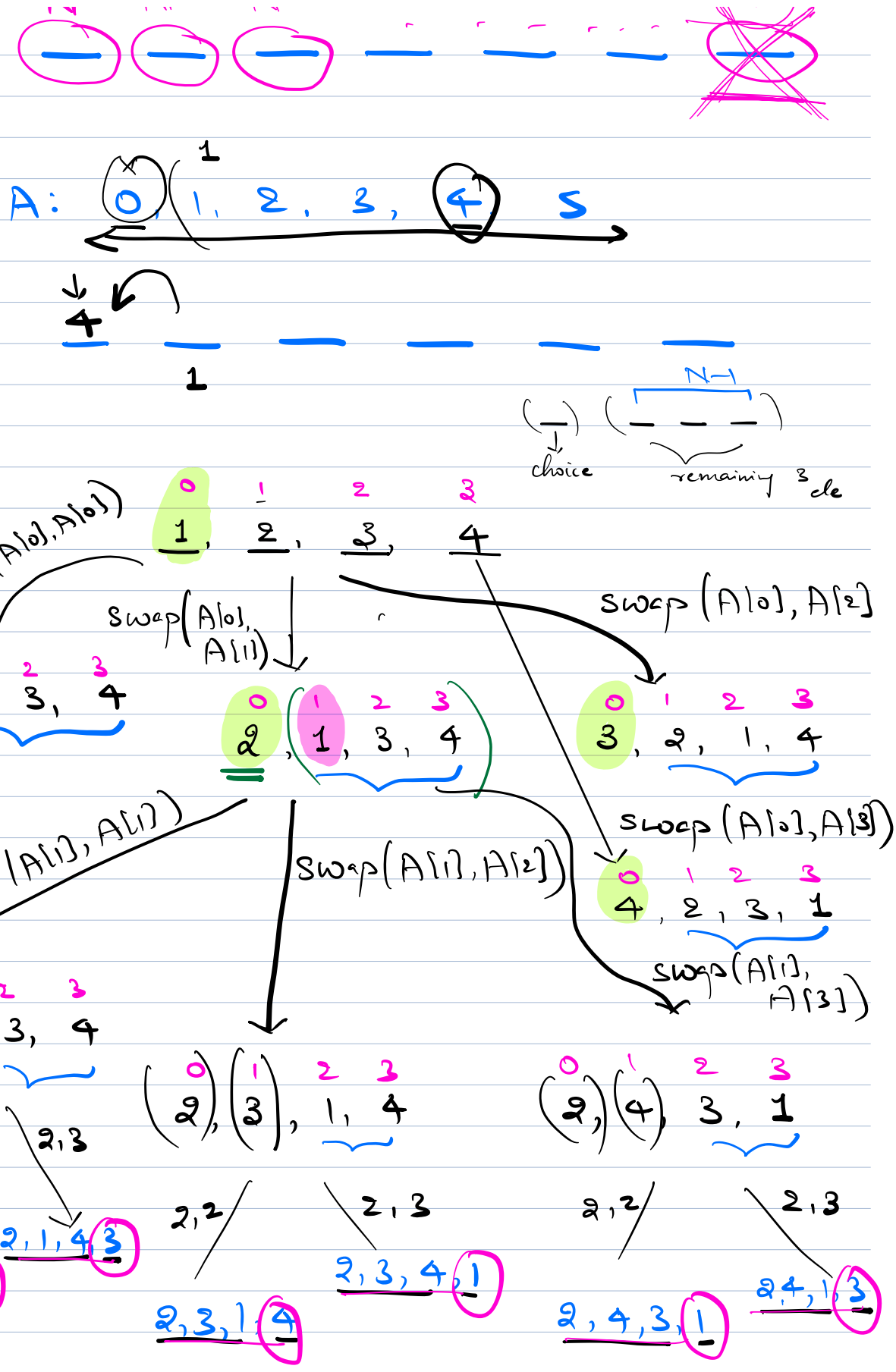
reference

0

N N-1 N-2

O(N)

1


A: 0, 1, 2, 3, 4, 5
choice
remaining cycle
swap(A[0],A[0])
swap(A[0], A[1])
swap(A[0], A[2])
swap(A[0], A[3])
swap(A[1], A[1])
swap(A[1], A[2])
swap(A[1], A[3])
2,2
2,3
2,2
2,3
2,2
2,3
2,1,4,3
2,3,4,1
2,3,1,4
2,4,3,1
2,4,1,3

Code

Global:

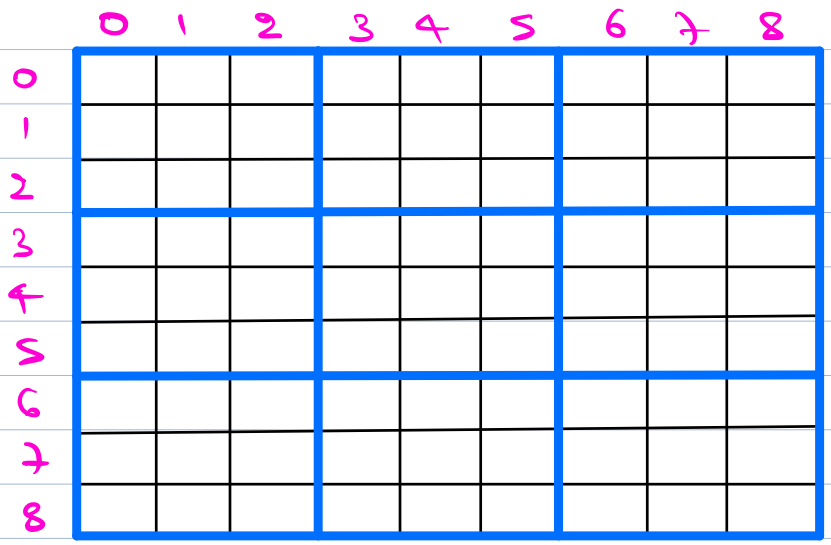
A[], N, index

```
void printPer(A[], N, index) {
    if (index == N-1) {
        print A[] array;
        return;
    }

    for (i = index; i < N; i++) {
        do  swap(A[index] & A[i])
            printPer(A, N, index+1);

        Undo  swap(A[index] & A[i])
    }
}
```

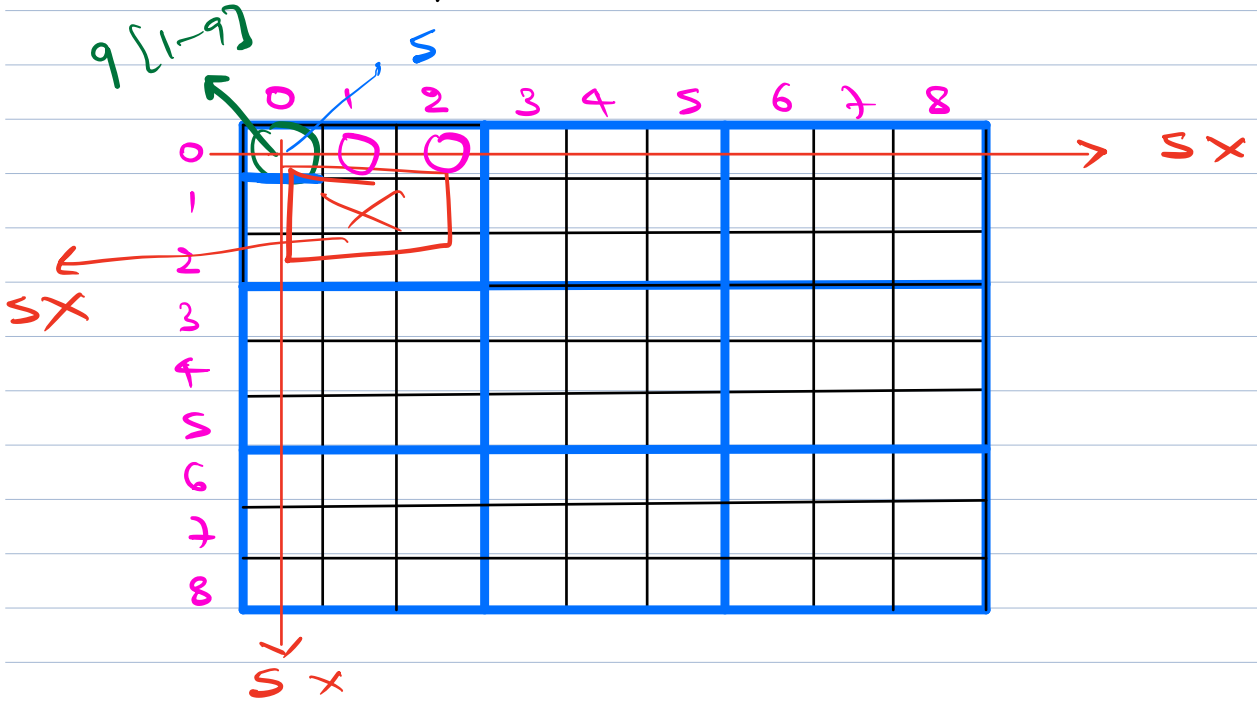
Sudoku (9×9 matrix



[1-9]

[1-9]

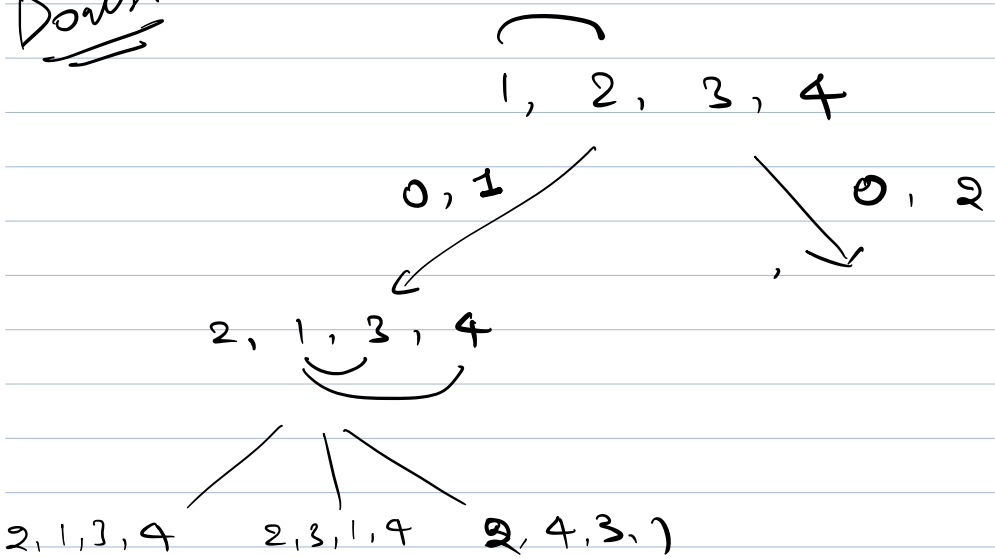
1) In a row data cannot repeat
2) " " column " " "
3) In the smaller 3×3 grid, elements cannot repeat.



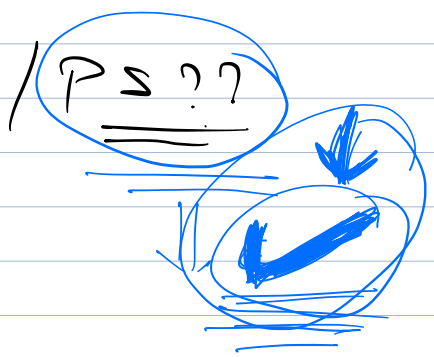
Row Wise from (L → R)

(row, col)

Doubt



Sort ⇒ DP ?? / PS ??

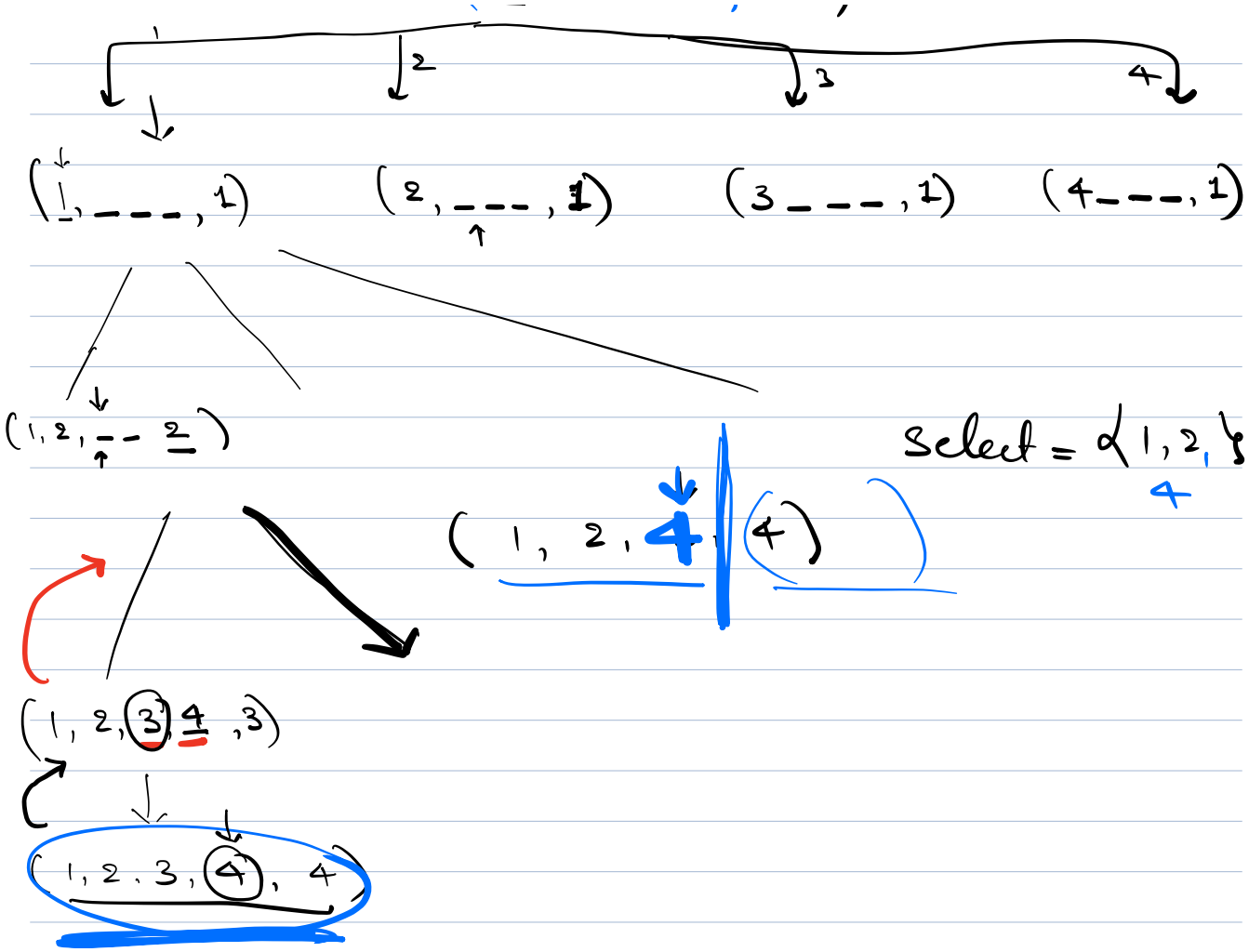


A: 1, 2, 3, 4

i

DFS

(_ _ _ _. 0)


1
2
3
4
(1, ___, 1)
(2, ___, 1)
(3 ___, 1)
(4___, 1)
(1, 2, ___ 2)
select = {1, 2, 4}
(1, 2, 4 | (4))
(1, 2, 3, 4, 3)
(1, 2, 3, (4), 4)