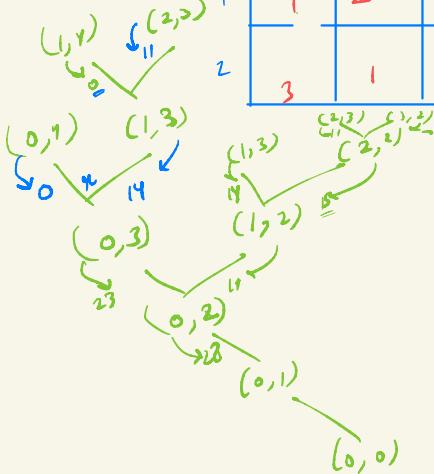



(i, j)



0	1	2	3
0	10	15	9
1	-1	2	4
2	3	1	9
3	11		

max path sum in matrix

```

public static int maxSum(int i, int j, int[][] matrix, int n, int m){
    if(i>n || j>m || i<0 || j<0){
        return 0;
    }

    if(i==n-1 && j==m-1){
        return matrix[i][j];
    }

    // right direction
    int a = maxSum(i, j+1, matrix, n, m);

    // downwards direction
    int b = maxSum(i+1, j, matrix, n, m);

    int ans = Math.max(a, b) + matrix[i][j];
    return ans;
}

```

Flood fill

0	1	0	1	1	1
0	1	0	1	1	0
0	1	0	0	0	0
0	0	0	1	1	0
0	1	0	1	1	0
0	0	0	1	1	0

```

public static void floodFill(int i, int j, int[][] mat, int n, int m, String psf, boolean[] visited) {
    if(i<0 || j<0 || i>n || j>m || mat[i][j]==1 || visited[i][j]){
        return;
    }

    if(i==n-1 && j==m-1){
        System.out.println(psf);
        return;
    }

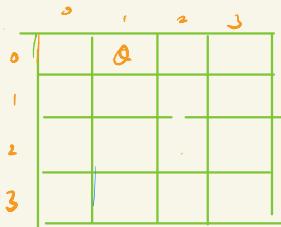
    visited[i][j] = true;
    floodFill(i-1, j, mat, n, m, psf+"t", visited);
    floodFill(i, j-1, mat, n, m, psf+"l", visited);
    floodFill(i+1, j, mat, n, m, psf+"d", visited);
    floodFill(i, j+1, mat, n, m, psf+"r", visited);
    visited[i][j] = false;
}

```

break till
10:15

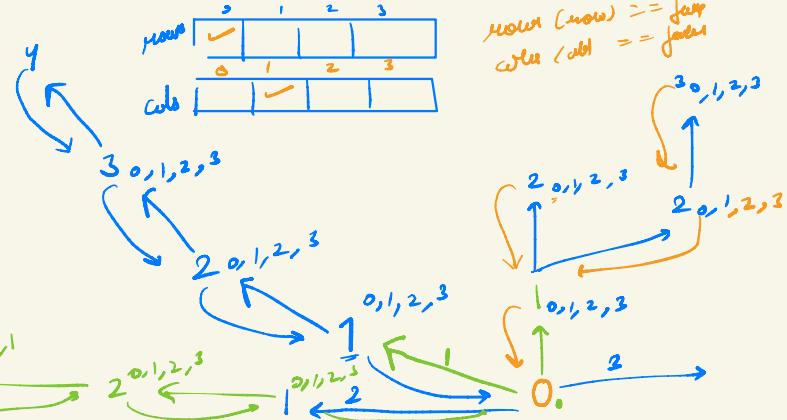
Every row
will have
exactly 1 queen

$(0,1) (1,3) (2,0) (3,2)$
 $(0,2) (1,0) (2,3) (3,1)$

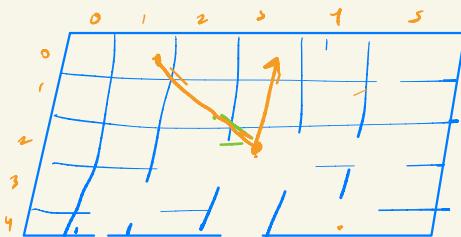


```
public static void NQueens(int row, int n, boolean[][] board, String asf){
    if(row==n){
        System.out.println(asf);
        return;
    }

    for(int col=0; col<n; col++){
        if(isQueenSafe(row, col, board)==true){
            board[row][col] = true;
            NQueens(row+1, n, board, asf+" "+row+" "+col+" ");
            board[row][col] = false;
        }
    }
}
```



int [][] dirs = { {-1,-1}, {-1,0}, {-1,1}, {0,1}, {1,1}, {1,0}, {-1,-1}, {0,-1} };
 $\Rightarrow \begin{cases} (-1, -1) & (-1, 0) \\ (-1, 1) & (0, 1) \\ (1, 1) & (1, 0) \\ (1, -1) & (0, -1) \end{cases}$



row=2
col=3

dir = $\frac{6}{\underline{2}}, j$

$$\begin{aligned} x_{\text{ad}} &= 2 & \text{dir} &= (-1, 0) \\ i &= 2 + 2 * (-1) \Rightarrow 2 - 2 = 0 & & \\ j &= 3 + 2 * (0) \Rightarrow 3 + 0 = 3 & & \end{aligned}$$

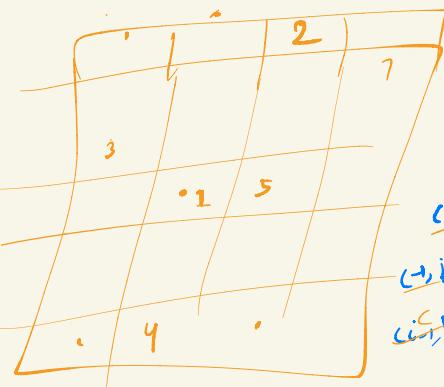
```
public static boolean isQueenSafe(int row, int col, boolean[][] board){
    int n = board.length;

    int[][] dirs = {{-1,-1}, {-1,0}, {-1,1}, {0,1}, {1,1}, {1,0}, {-1,-1}, {0,-1}};

    for(int rad=1; rad<n; rad++){
        for(int[] dir : dirs){
            int i = row + rad * dir[0];
            int j = col + rad * dir[1];

            if(i<=0 && j<=0 && i>n && j>n && board[i][j]==true) return false;
        }
    }

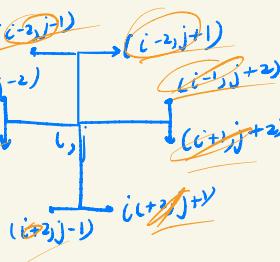
    return true;
}
```



8×8

$\forall (move == 2^r)$
points to array
 $chess(x)(c) = move$

move + 1

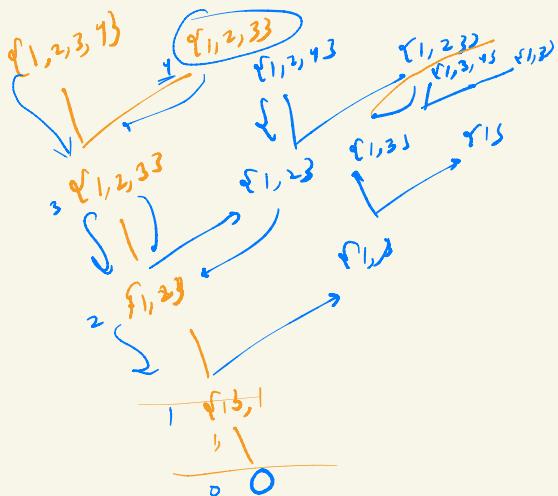


$chess(x)(c) = 0$

$\forall \{1, 2, 3\}$

$ans = \{1, 2, 3, 4\}$

$t_{ave} = 6$ "abc-def"



```

public static void solve(int[] arr, int tar, int idx, int csum,
    if(csum>tar) return;
    if(idx == arr.length){
        if(tar==csum){
            ans.add(new ArrayList<>(curr));
        }
        return; // ans.add(curr)
    }
    // include
    curr.add(arr[idx]);
    solve(arr,tar,idx+1,csum+arr[idx],curr,ans);
    curr.remove(curr.size()-1);

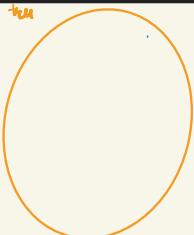
    // not include
    solve(arr,tar,idx+1,csum,curr,ans);
}

```

curr ↳ address
ans ↳ address

$TC = (\text{number of calls})^{\log N \text{ of tree}}$

$\underline{2^n}$



0	1	2	3
0			
1			
2			
3			

row-visited \Rightarrow

✓			
0	1	2	3

cols-visited \Rightarrow

✓						
0	1	2	3	4	5	6

diag \Rightarrow

ad diag \Rightarrow

1,0,1
1
0,0
[1, 1]

```
public static void NQueens(int row, int n, boolean[] row_visited, boolean[] col_visited) {
    if(row==n){
        System.out.println(asf);
        return;
    }

    for(int col=0; col<n; col++){
        if(row_visited[row] == false && col_visited[col] == false){
            row_visited[row] = true;
            col_visited[col] = true;
            NQueens(row+1, n, row_visited,col_visited, asf+"("+row+", "+col+")");
            row_visited[row] = false;
            col_visited[col] = false;
        }
    }
}
```

0	1	2	3	4	5	6	7	8
5	3			7				
1	6			1	9	5		
2		9	8				6	
3	8			6		4		3
4			8	3			.	1
5	7			2			5	
6		6			2	8		
7			4	1	9			5
8			8			7	9	

$\left(\frac{8}{3}\right) \times 3 \Rightarrow 2 \times 3 \Rightarrow 6$

$\left(\frac{2}{3}\right) \times 3 + 1 \times 3 \Rightarrow 3$

$\text{row} = 4$, $\text{col} = 1$, $\text{num} = 8$

$5, 8$

$\left(\frac{5}{3}\right) \times 3 \Rightarrow 1 \times 3 = 3$

$\left(\frac{8}{3}\right) \times 3 \neq 6$

```
public static boolean canPlace(int row, int col, int num, char[][] board) {
    // check row
    for(int i=row, j=0; j<9; j++){
        if(board[i][j]-'0'==num){
            return false;
        }
    }

    // check col
    for(int i=0, j=col; i<9; i++){
        if(board[i][j]-'0'==num){
            return false;
        }
    }

    // check subGrid
}
```

$$\left(\frac{4+6}{3}\right) \times 3 \Rightarrow \left(\frac{4}{3}\right) \times 3 \Rightarrow 1 \times 3 = 3$$

$$\left(\frac{1+2}{3}\right) \times 3 \Rightarrow 0 \times 3 = 0$$

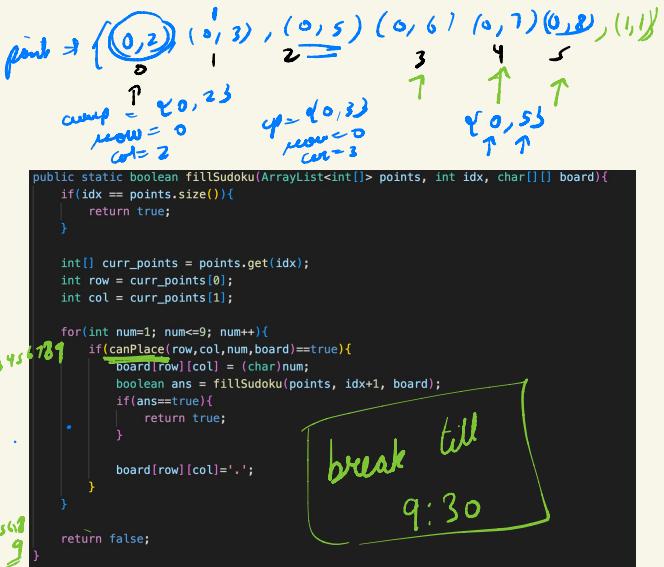
$$5r=3 \\ sc=3$$

$$i=sr \\ j=sc \\ j < sc+3$$

$$i < sr+3$$

	0	1	2	3	4	5	6	7	8
0	5	3	1	7					
1	6			1	9	5			
2	9	8				6			
3	8			6					3
4	4		8	3					1
5	7			2					6
6	6				2	8			
7			4	1	9				5
8			8			7	9		

0 → 1 → 2 → 3 → 4 → 3 → 5 → 4 → 3 → 2 → 1 → 0



$$\text{coins} = \{2, 3, 1, 3\} \quad t = 6$$

(1, 7) → (2, 1)

2 2 2
2 2 1 1

(0, 6)
↓
(0, 4)

↓
(0, 2)

↓
(0, 1)

```

public static void combination_ManyCoins(int[] arr, int idx, int tar, int csum, String asf){
    if(csum>tar){
        return;
    }
    if(idx == arr.length){
        if(csum==tar){
            System.out.println(asf);
        }
        return;
    }

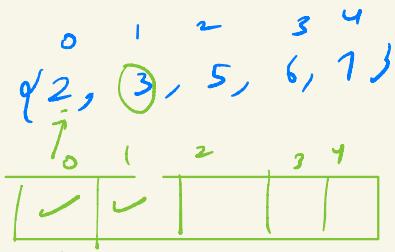
    // include (I can include the same coin again, so next call is with the same id)
    combination_ManyCoins(arr, idx, tar, csum+arr[idx], asf+arr[idx]+",");

    // not include
    combination_ManyCoins(arr, idx+1, tar, csum, asf);
}

```

j=0, j=1

0 → 1



valid subsets
 \rightarrow permutation 2
dry run every thing

$"32"$
 $"3, 2"$
 $"3, 13"$
 $"0, 0"$
 $(2, "2")$

```

public static void getPermutations(int[] coins, int tar, int csum, boolean[] vis, String asf) {
    if(csum > tar){
        return;
    }
    if(tar == csum){
        System.out.println(asf);
        return;
    }
    for(int j=0; j<coins.length; j++){
        if(vis[j]==false){
            vis[j] = true; // using this coin
            getPermutations(coins, tar, csum+coins[j], vis, asf+coins[j]+" ");
            vis[j] = false;
        }
    }
}

```

$\{2, 3, 6\}$
 $\{7\}$

$tar=12$

$2, 2, 2, 2, 2$
 $2, 2, 3, 3, 2$
 $2, 3, 3, 2, 2$

$6, 6$
 $3, 3, 6$
 $6, 3, 3$
 $3, 6, 3$

$\text{for } (\text{int } j=1; j < \text{arr.length})$

Permutations and Combinations

4 items in 2 boxes

$$\begin{aligned} \text{boxes} &= 4 \\ \text{clans} &= 2 \end{aligned}$$

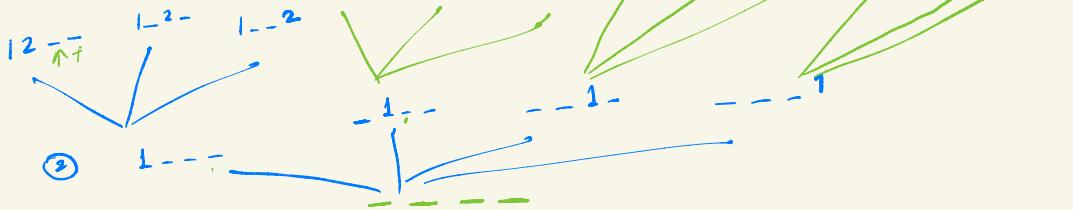
$${}^n P_x \Rightarrow {}^4 P_2 \Rightarrow \frac{n!}{(n-x)!}$$

$$4! \leftarrow \frac{4 \times 3 \times 2 \times 1}{2 \times 1} \Rightarrow \frac{4!}{2!} \Rightarrow \frac{4!}{(4-2)!} \text{ empty}$$

2 items

1 item

3rd, 4th

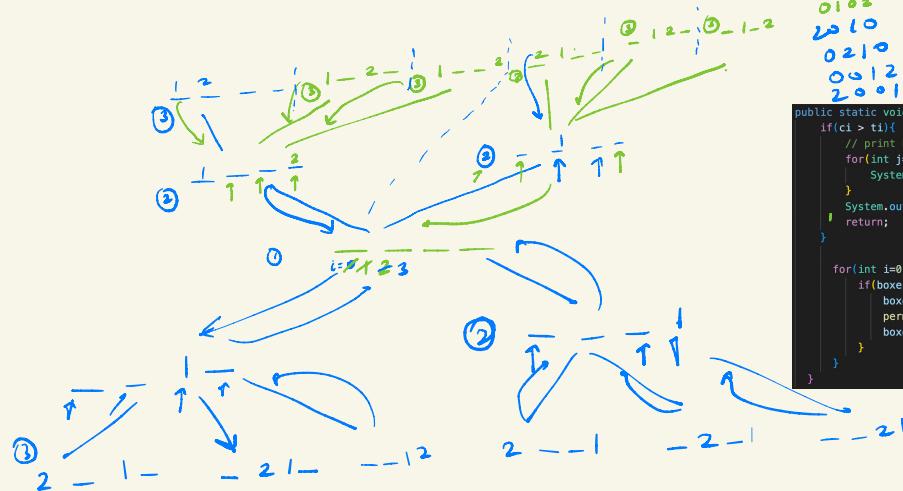


12--	1-2-	21--	-12-	-1-2	2-1-	-21-	-12
0201	0211						

b
b

```
public static void permutations(int[] boxes, int ci, int ti){
    if(ci > ti){
        // print boxes and return
        for(int j=0; j<boxes.length; j++){
            System.out.print(boxes[j]);
        }
        System.out.println();
        return;
    }

    for(int i=0; i<boxes.length; i++){
        if(boxes[i]==0){
            boxes[i] = ci;
            permutations(boxes, ci + 1, ti);
            boxes[i] = 0;
        }
    }
}
```



bones = 4
items = 2

- ✓ 1, 2, 3
- ✓ 2, 3, 1
- ✓ 3, 2, 1
- ✓ 3, 1, 2
- ✓ 2, 1, 3
- ✓ 1, 3, 2

$${}^4C_2 \Rightarrow \frac{4!}{2!2!} \Rightarrow \frac{4 \times 3}{2} = 6$$

$$\begin{matrix} {}^4P_2 \\ \cancel{\{1, 2, 3\}} \\ \cancel{\{3, 1, 2\}} \end{matrix} \Rightarrow 12$$

permutation ↳ arrangements

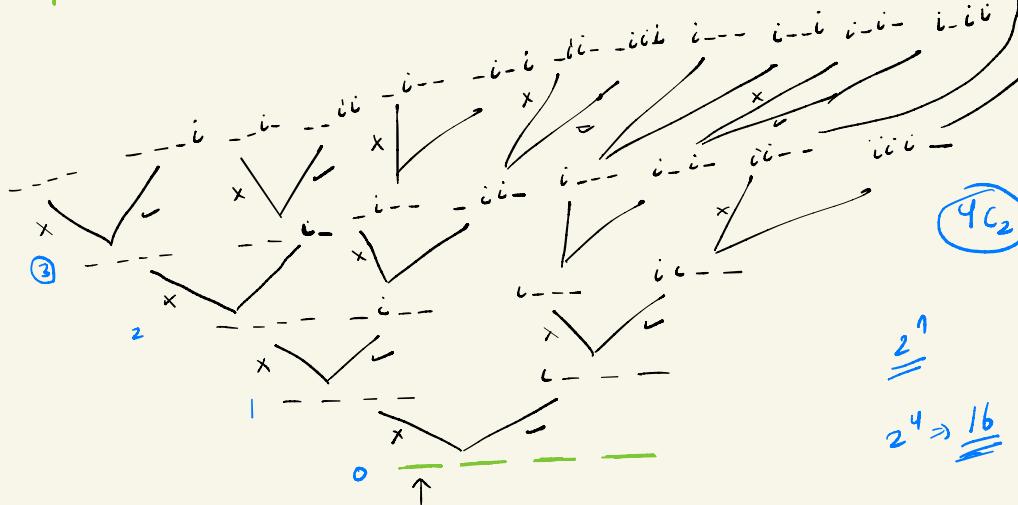
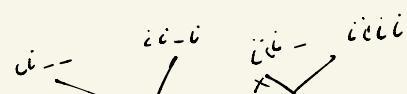
1	2	-	-	1	2	1	-	-
1	-	2	-	1	-	1	-	-
1	-	-	2	1	-	-	1	-
1	-	-	-	2	1	-	-	-
-	1	-	-	-	2	-	1	-
-	-	1	-	-	-	2	1	-
-	-	-	1	-	-	-	-	1

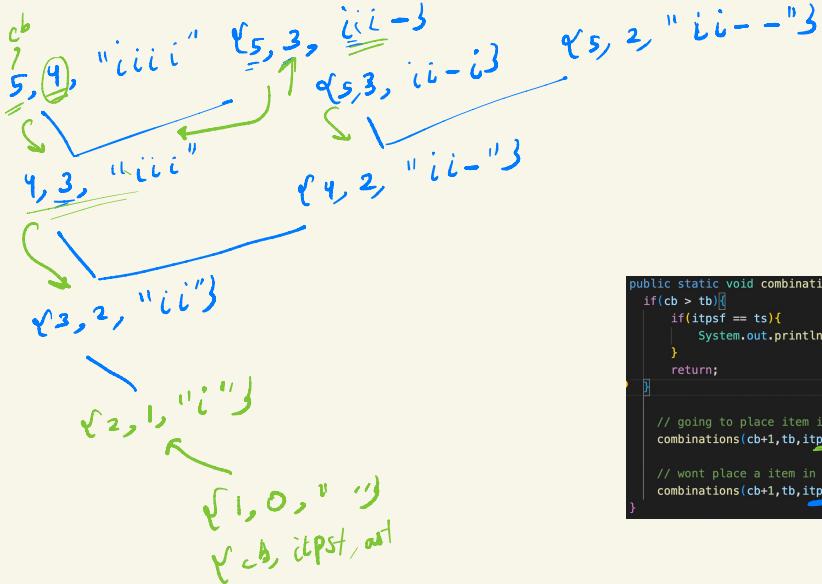
Combination ↳ elements, selection

i	i	-	-
i	-	i	-
i	-	-	i
-	i	i	-
-	i	-	i
-	-	i	i

$$\begin{aligned} 2^n &= {}^nC_0 + {}^nC_1 + {}^nC_2 + {}^nC_3 + \dots \\ 2^4 &= {}^4C_0 + {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4 \\ &\quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ 1 &\quad 4 \quad 6 \quad 4 \quad 1 \end{aligned}$$

4 bones, 2 items





break till
10:05

```
public static void combinations(int cb, int tb, int itpsf, int ts, String asf){
    if(cb > tb){
        if(itpsf == ts){
            System.out.println(asf);
        }
        return;
    }

    // going to place item in this box
    combinations(cb+1, tb, itpsf+1, ts, asf+"i");

    // wont place a item in this box
    combinations(cb+1, tb, itpsf, ts, asf+"-");
}
```

3C_2

$${}^nPr_n = \frac{n!}{(n-r)!}$$

$${}^nCr_n = \frac{n!}{r!(n-r)!}$$

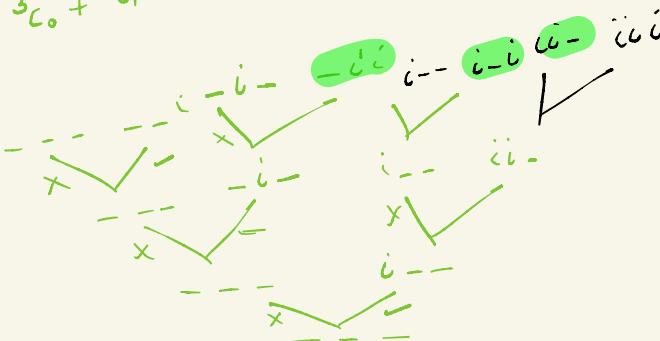


$$2^3 = {}^3C_0 + {}^3C_1 + {}^3C_2 + {}^3C_3$$

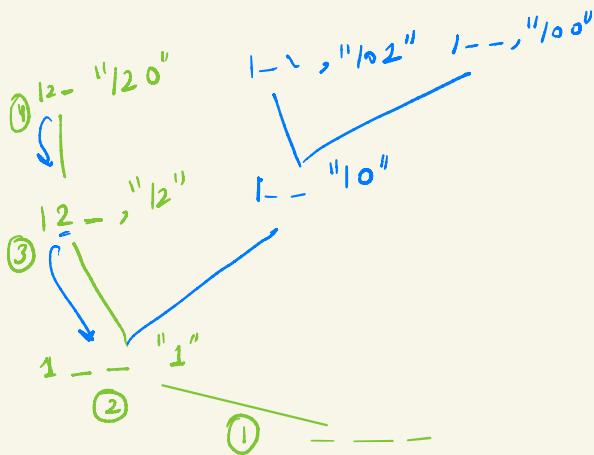
$$\text{permutation} = \underline{\text{combination}} \times \frac{(n!)^r}{r!}$$

$$2! \Rightarrow 2$$

$$\frac{3!}{1!} \times \frac{3!}{2!} = 3 \times 3 = 9$$



$$\begin{aligned} {}^3C_2 &\Rightarrow 3 \\ {}^3P_2 &\Rightarrow 6 \end{aligned}$$



```

public static void permutations(int cb, int tb, boolean[] items,
    if(cb > tb){
        if(itpsf == ts){
            System.out.println(asf);
        }
        return;
    }

    // i can pick any of the remaining items
    for(int i=0; i<items.length; i++){
        if(items[i]==false){ // we have not placed current item
            items[i] = true;
            permutations(cb+1,tb,items,itpsf+1,ts,asf+(i+1));
            items[i] = false;
        }
    }

    // keep the current box empty
    permutations(cb+1,tb,items,itpsf,ts,asf+"0");
}

```

