Lec-18
06/05/25 4:42 PM

Recursion and Backtracking

① Notes ① S.O ② N.O ③ Dry Run ④ Formula

Recursion

$^nC_R = \frac{n!}{(n-R)!R!}$

$= \frac{n!}{(n-R)!}$

$^nP_R = \frac{n!}{(n-R)!}$

$N=4$ $TQ = 2$ $^{4!}_{(4-2)!}$

Permutation

$4_{P_2} = 6$

bo b1
bo b2
bo b3
b1 b2
b1 b3
b2 b3

bo b1 b2
bo b1 b3
bo b2 b1
bo b2 b3
bo b3 b1
bo b3 b2
b1 b0 b2
b1 b2 b3
b1 b3 b2
b2 b0 b1
b2 b1 b3
b2 b3 b1
b3 b0 b1
b3 b1 b2
b3 b2 b1

```
public static void Permutation(boolean[] board, int tq, String ans, int qpsf) {
    if (qpsf == tq) {
        System.out.println(ans);
        return;
    }
    for (int i = 0; i < board.length; i++) {
        if (board[i] == false) {
            board[i] = true;
            Permutation(board, tq, ans + "b" + i + "q" + qpsf, qpsf + 1);
            board[i] = false;
        }
    }
}
```

T	T	T	T
0	1	2	3

qpsf==tq

board[i] == false

board[i] = true

Permutation(board, tq, ans + "b" + i + "q" + qpsf, qpsf + 1);

b0q0b1q1
b0q0b2q2
b0q0b3q3

qpsf==tq

i=0 b0q0
P(2x2, b0q0b1q1, 2)
P(2x2, b0q0b2q2, 2)
P(2x2, b0q0b3q3, 2)

i=1 b1q1
P(2x2, b1q1b0q0, 2)

i=2 b2q2
P(2x2, b2q2b0q0, 2)

main (2x2, "", 0)

```
public static void Permutation(boolean[] board, int tq, String ans, int qpsf) {
    if (qpsf == tq) {
        System.out.println(ans);
        return;
    }
    for (int i = 0; i < board.length; i++) {
        if (board[i] == false) {
            board[i] = true;
            Permutation(board, tq, ans + "b" + i + "q" + qpsf, qpsf + 1);
            board[i] = false;
        }
    }
}
```

0	1	2	3

qpsf==tq

board[i] == false

board[i] = true

Permutation(board, tq, ans + "b" + i + "q" + qpsf, qpsf + 1);

b0q0b1q1
b0q0b2q2
b0q0b3q3
b1q1b0q0
b2q2b0q0
b3q3b0q0

qpsf==tq

i=0 b0q0
P(2x2, b0q0b1q1, 2)
P(2x2, b0q0b2q2, 2)
P(2x2, b0q0b3q3, 2)

i=1 b1q1
P(2x2, b1q1b0q0, 1)
P(2x2, b1q1b2q2, 1)

main (2x2, "", 0)

Permutation

b0q0b1q1
b0q0b2q2
b0q0b3q3
b1q1b0q0
b1q1b2q2
b2q2b0q0
b2q2b1q1
b3q3b0q0
b3q3b1q1
b3q3b2q2

```
public static void Combination(boolean[] board, int tq, String ans, int qpsf,int idx) {
    if (qpsf == tq) {
        System.out.println(ans);
        return;
    }
    for (int i = idx; i < board.length; i++) {
        if (board[i] == false) {
            board[i] = true;
            Combination(board, tq, ans + "b" + i + "q" + qpsf, qpsf + 1,i+1)
            board[i] = false;
        }
    }
}
```

0	1	2

qpsf==2

i=1
C(2x2, b0q0b1q1, 2)
C(2x2, b0q0b2q2, 3)

idx=1
ans = b0q0
qpsf=1

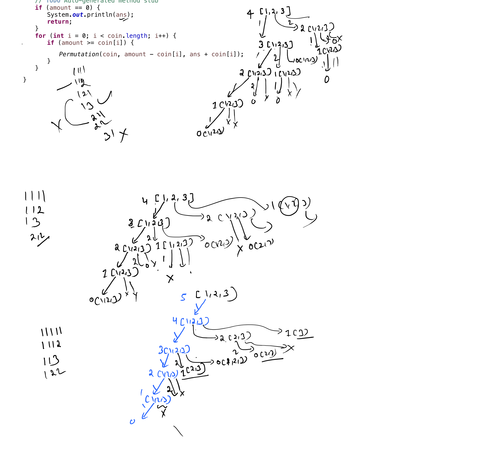
i=0
C(2x2, b0q0, 1, 2)

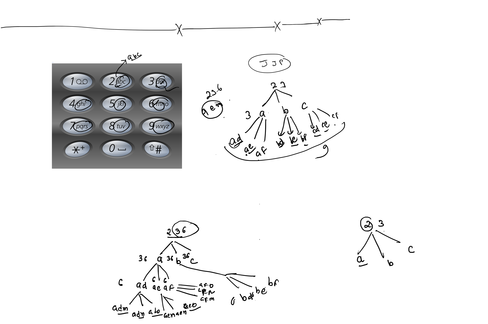
idx=0
qpsf=0
ans=""

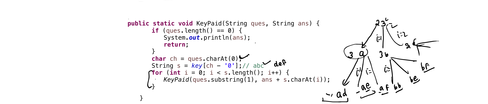
Combination

C(2x2, "", 0, 0)

Permutation(int[] coin, int amount, String ans) {
    // TODO Auto-generated method stub
    if (amount == 0) {
        System.out.println(ans);
        return;
    }
    for (int i = 0; i < coin.length; i++) {
        if (amount >= coin[i]) {
            Permutation(coin, amount - coin[i], ans + coin[i]);
        }
    }
}








```
public static void KeyPaid(String ques, String ans) {
    if (ques.length() == 0) {
        System.out.println(ans);
        return;
    }
    char ch = ques.charAt(0);
    String s = key[ch - '0'];// abc
    for (int i = 0; i < s.length(); i++) {
        KeyPaid(ques.substring(1), ans + s.charAt(i));
    }
}
```