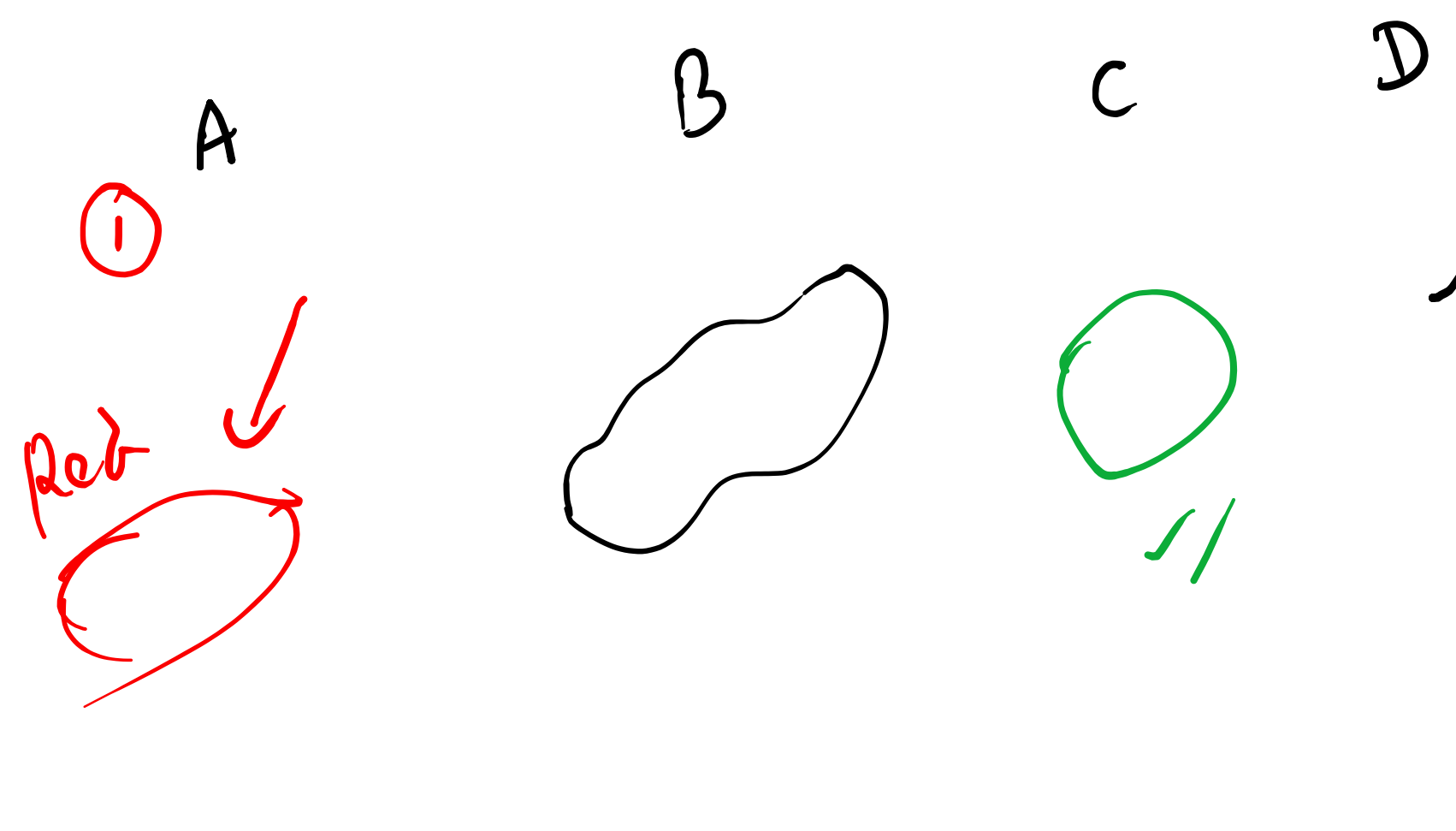
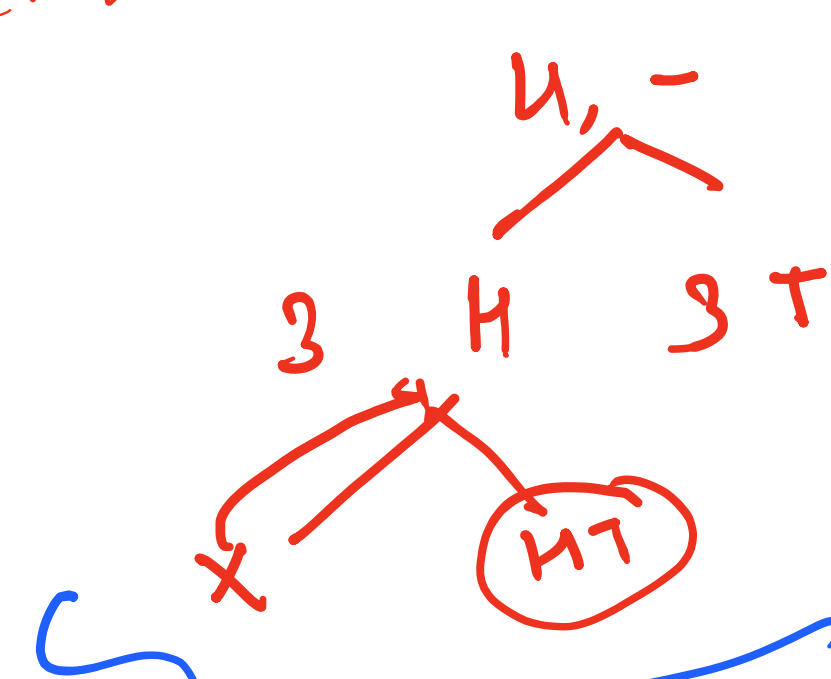
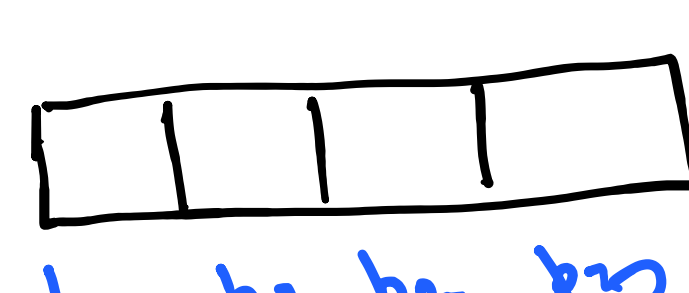


Recursion and Backtracking

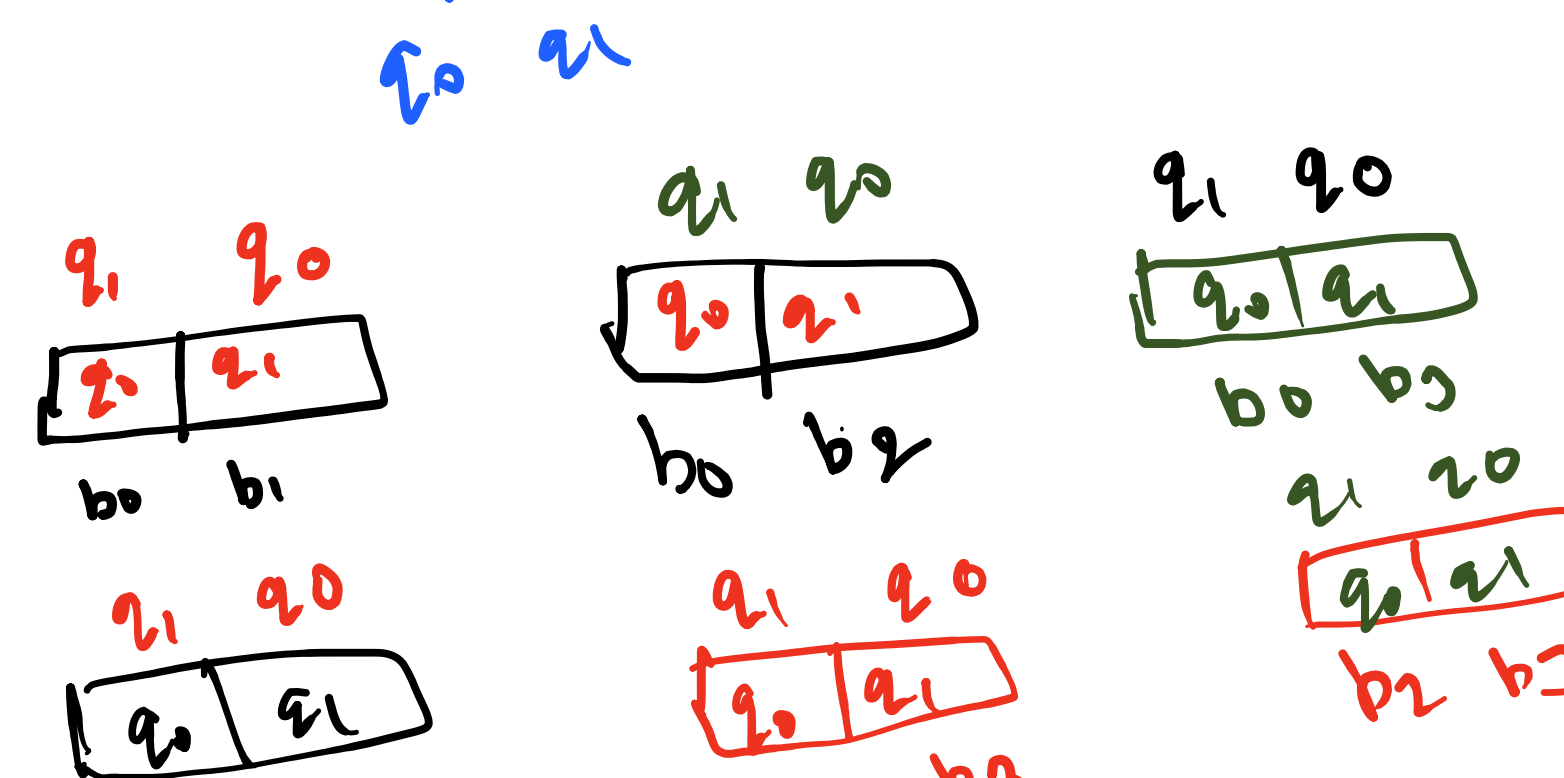


$$N = 4$$

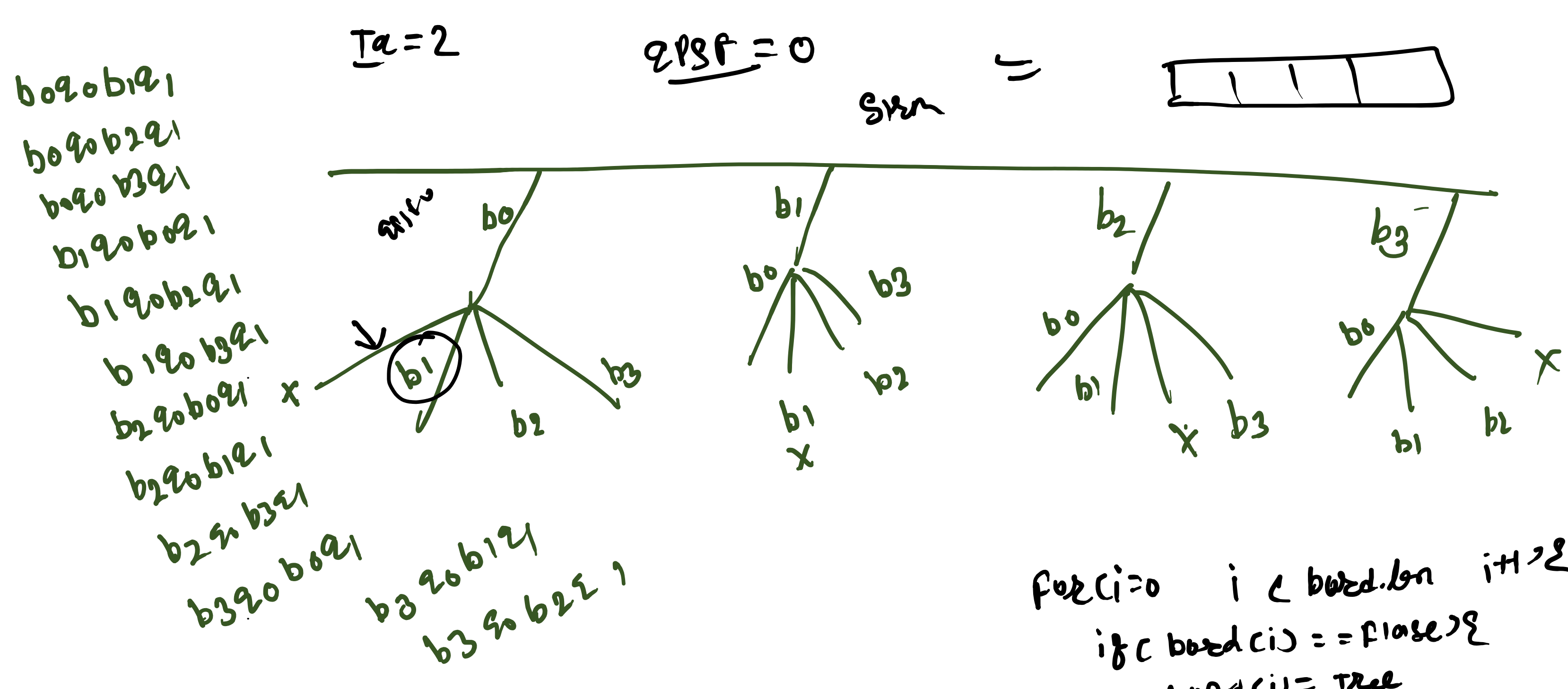
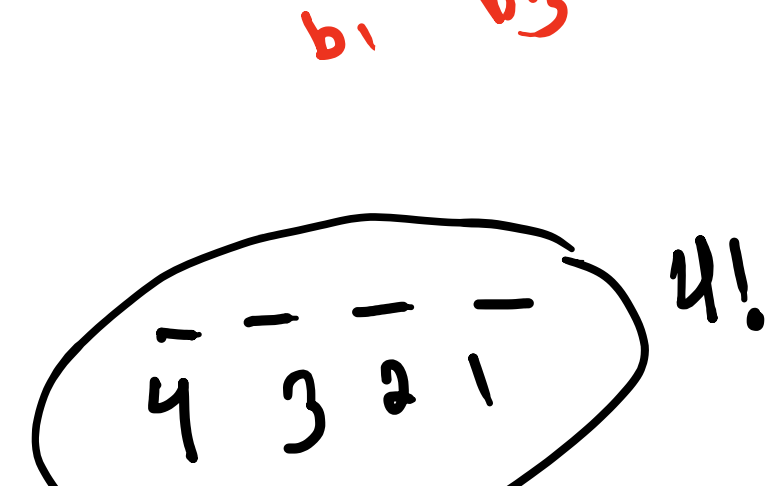


$$u_{c_2} \circ [N_{c_R}] \times R! = \frac{N!}{(N-p)! p!} \rightarrow P_1$$

$$= N! (w_p)$$



$$\left\{ \begin{array}{l} \overline{b o b i} \\ = b o b \bar{z} \\ = c a g \end{array} \right\} \left| \begin{array}{l} b i b \bar{z} \\ b i b \bar{z} \end{array} \right| b \bar{z} b \bar{z} = \frac{n!}{(n-r)!} \left\{ \begin{array}{l} n \\ r \end{array} \right\}$$



```

for (ci = 0; i < boardLen; i++) {
    if (board[ci] == false) {
        board[ci] = true
    }
}

```

```

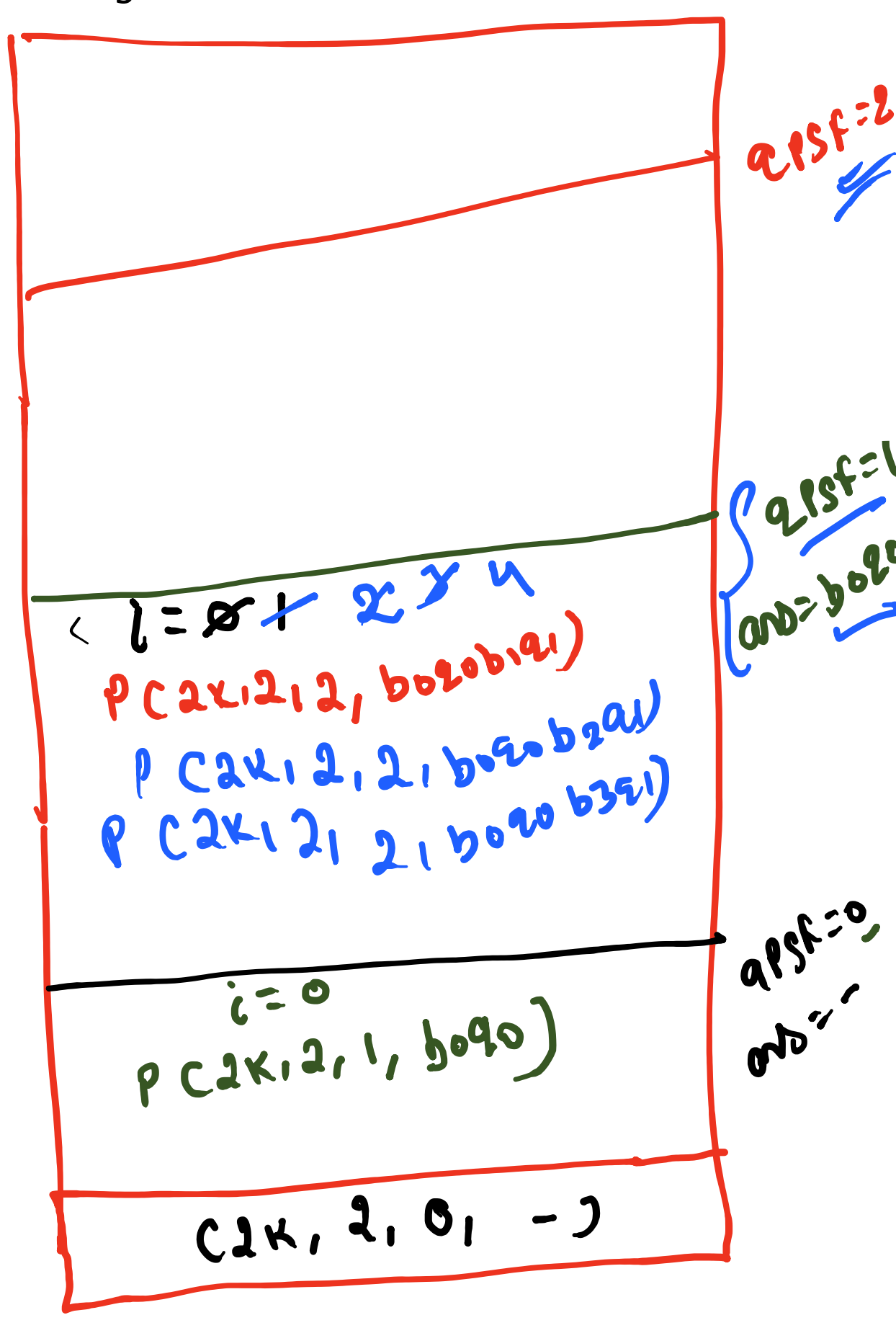
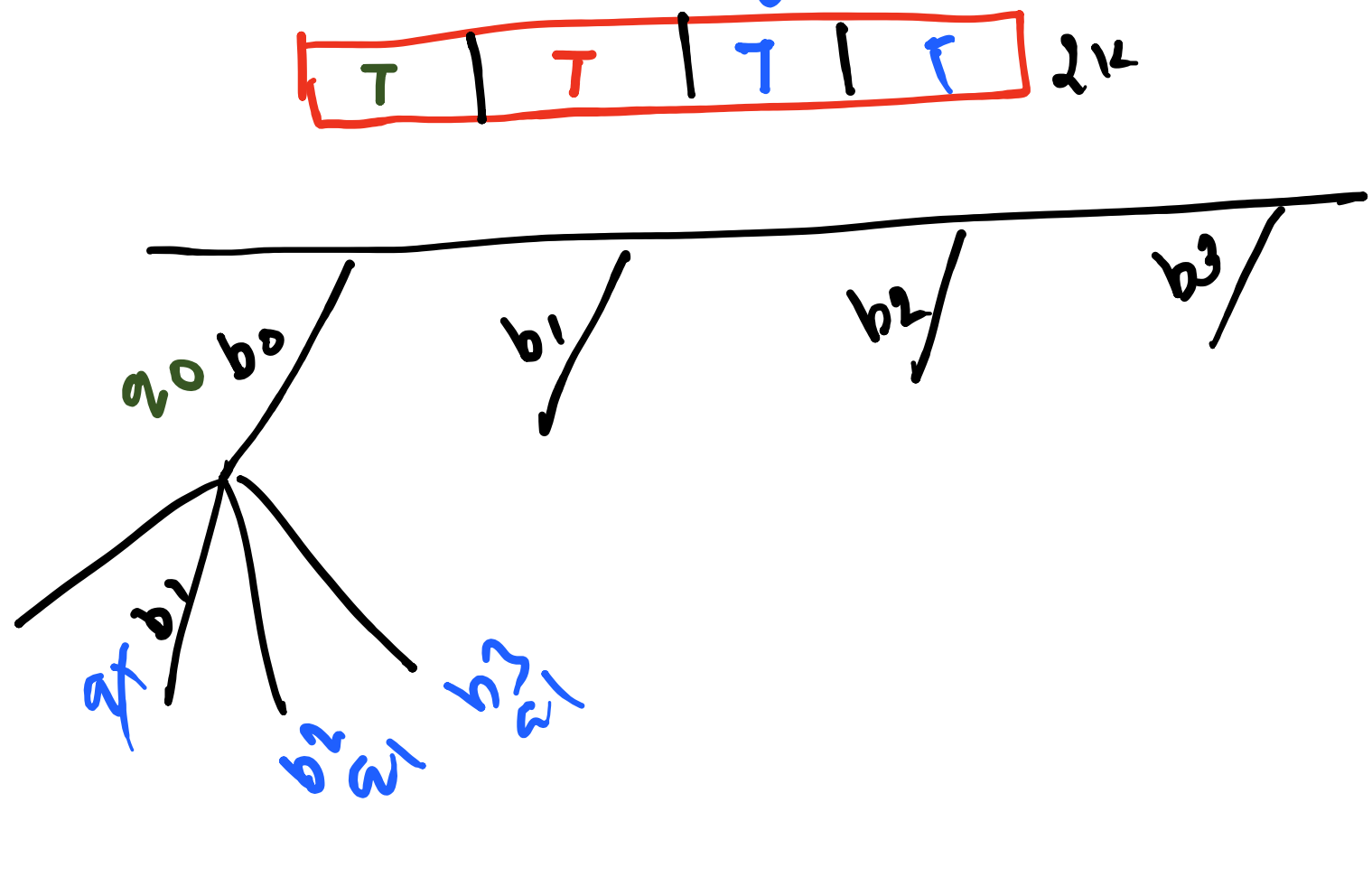
public static void Permutation(boolean[] board, int tq, int qpsf, String ans) {
    if (tq == qpsf) {
        System.out.println(ans);
        return;
    }

    for (int i = 0; i < board.length; i++) {
        if (board[i] == false) {
            board[i] = true;
            Permutation(board, tq, qpsf + 1, ans + "b" + i + "q" + qpsf);
            board[i] = false;
        }
    }
}

```

Handwritten notes and diagram:

- Red text: *badgo 1st*
- Blue text: *badgo 2nd*
- Blue text: *badgo 3rd*
- Blue arrow pointing from the recursive call to the board state.
- Board state diagram: A horizontal array of 8 cells. The 1st, 3rd, and 5th cells contain 'T'. The 4th cell contains 'F'. The 6th, 7th, and 8th cells are empty. To the right of the array is the number 24.



```

public class Queen_Permutation {

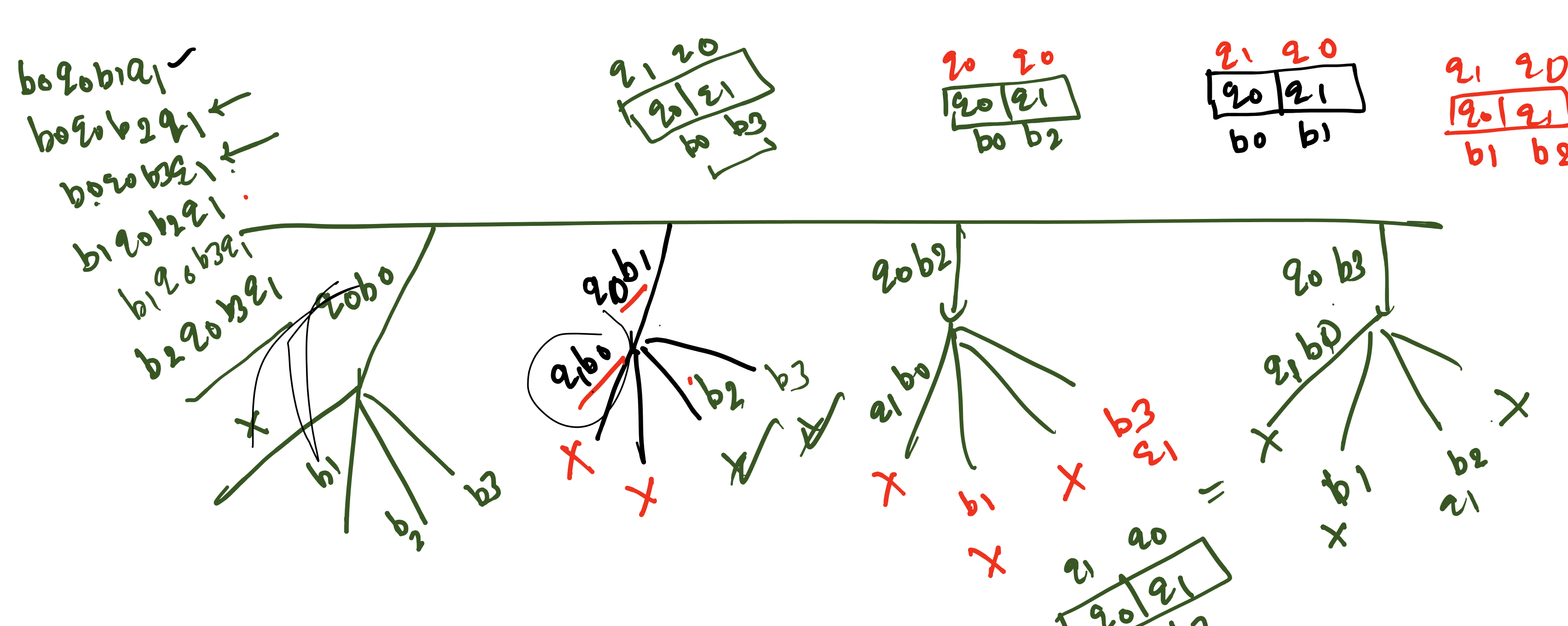
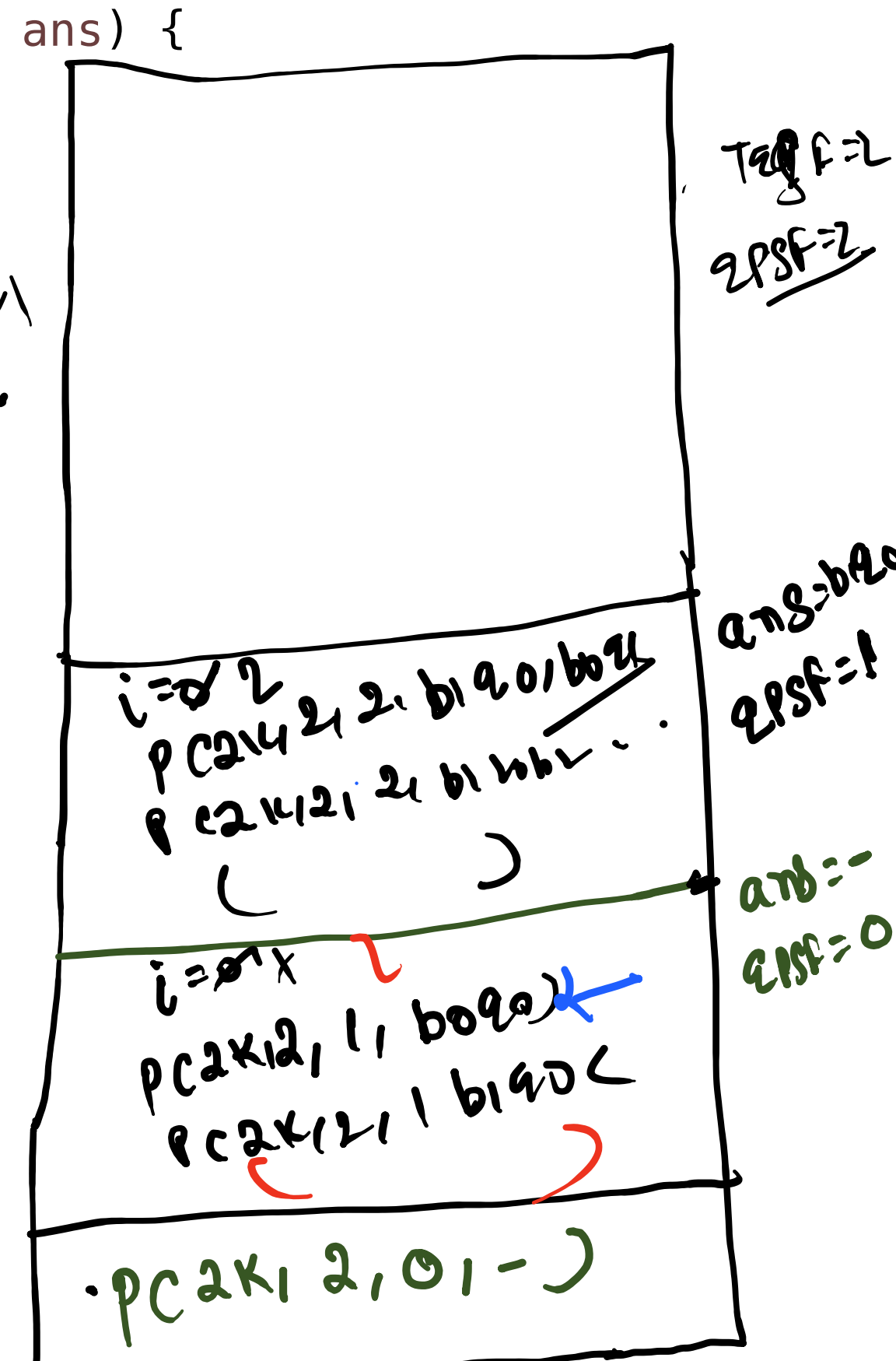
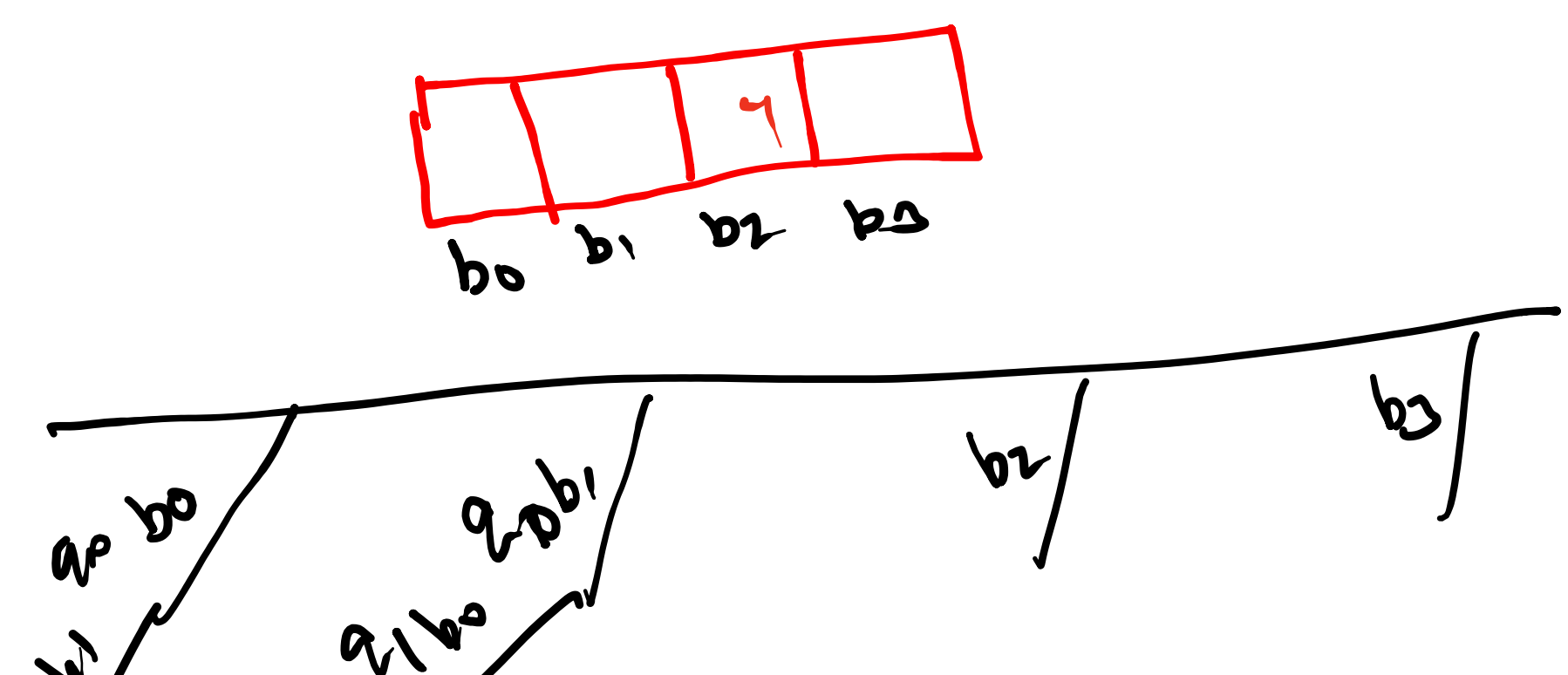
    public static void Permutation(boolean[] board, int tq, int qpsf, String ans) {
        if (tq == qpsf) {
            System.out.println(ans);
            return;
        }

        for (int i = 0; i < board.length; i++) {
            if (board[i] == false) {
                board[i] = true; // marked kr diya Queen yaha hai
                Permutation(board, tq, qpsf + 1, ans + "b" + i + "q" + qpsf);
                board[i] = false; // UNDO
            }
        }
    }
}

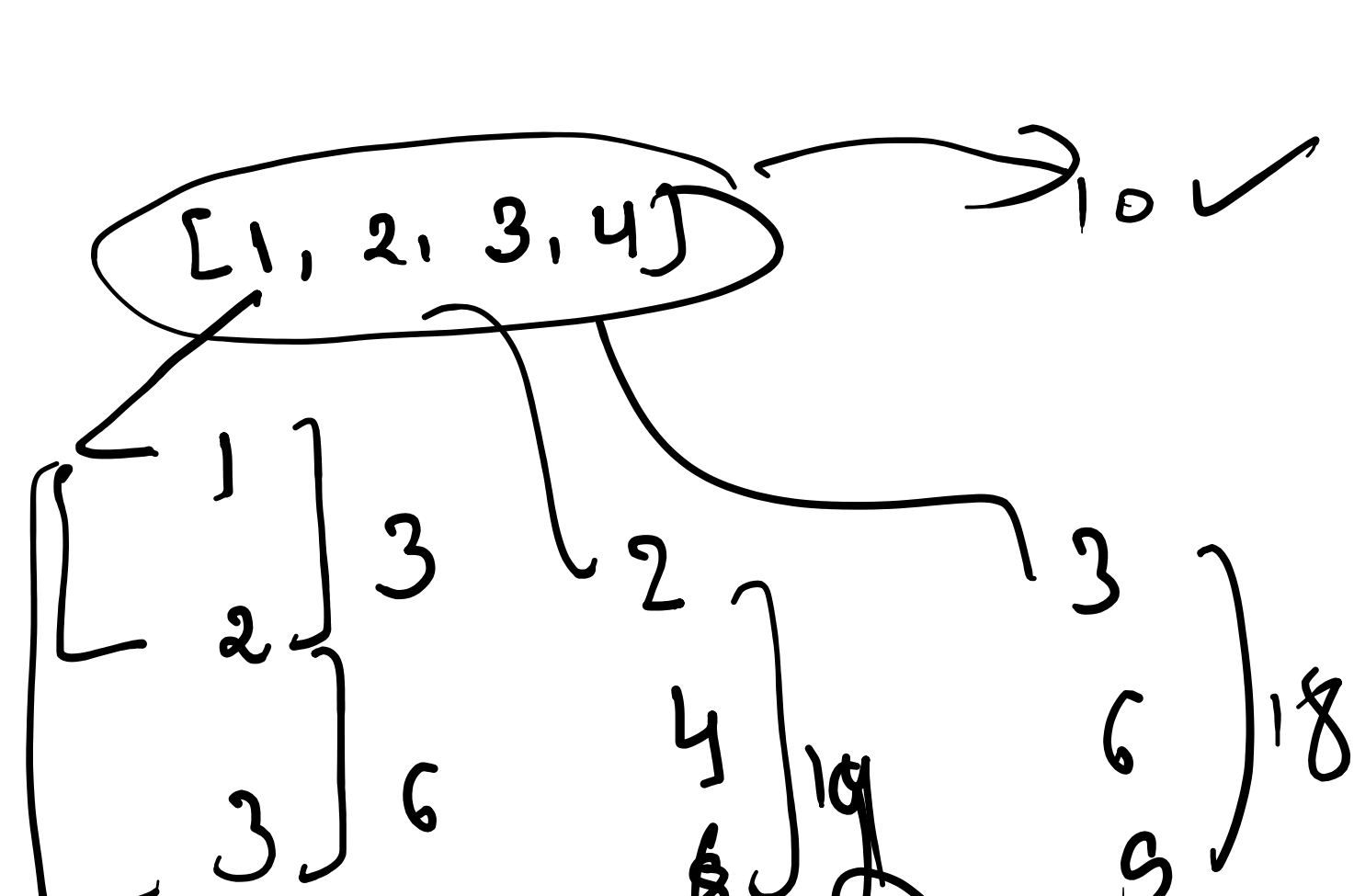
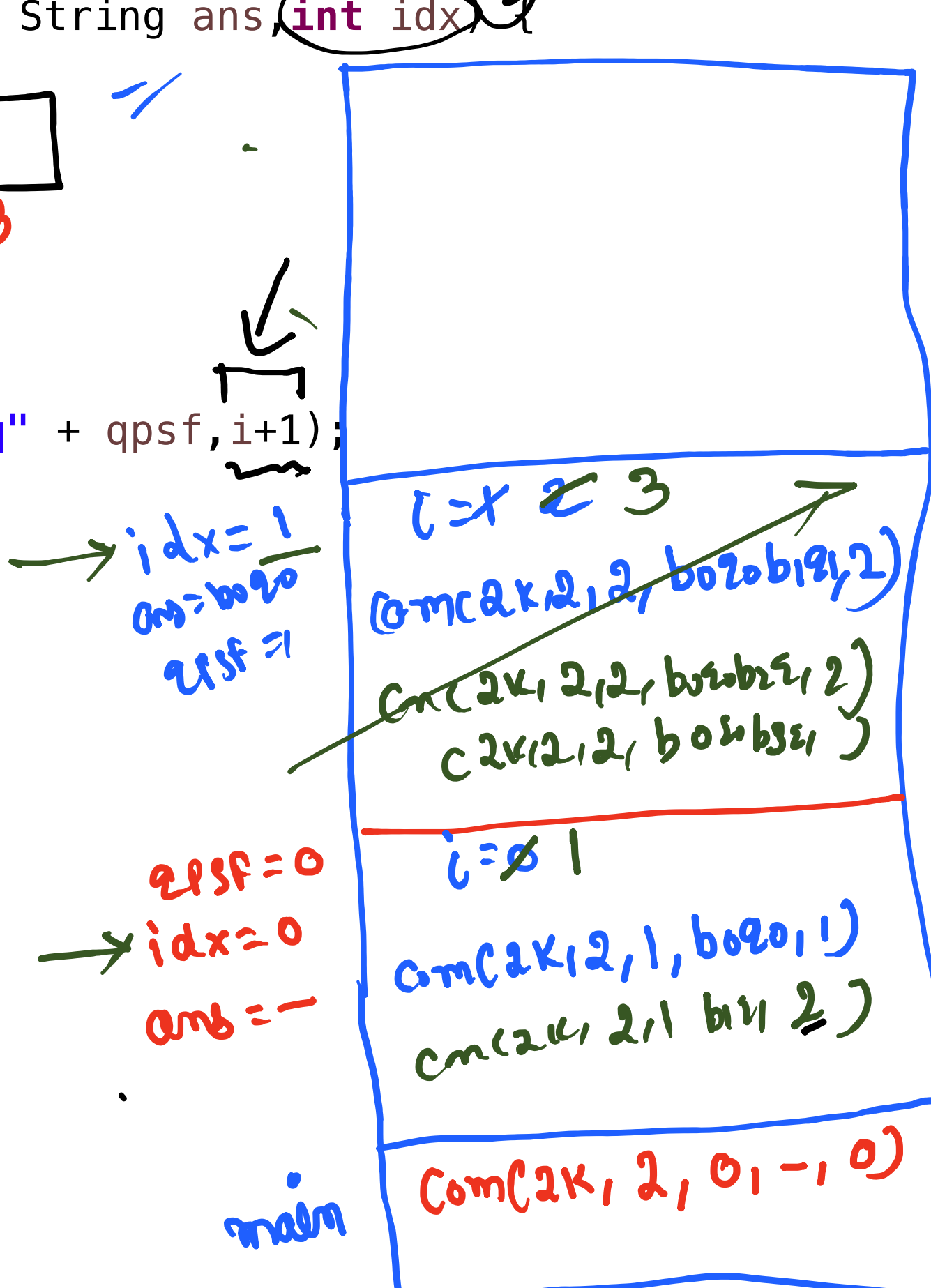
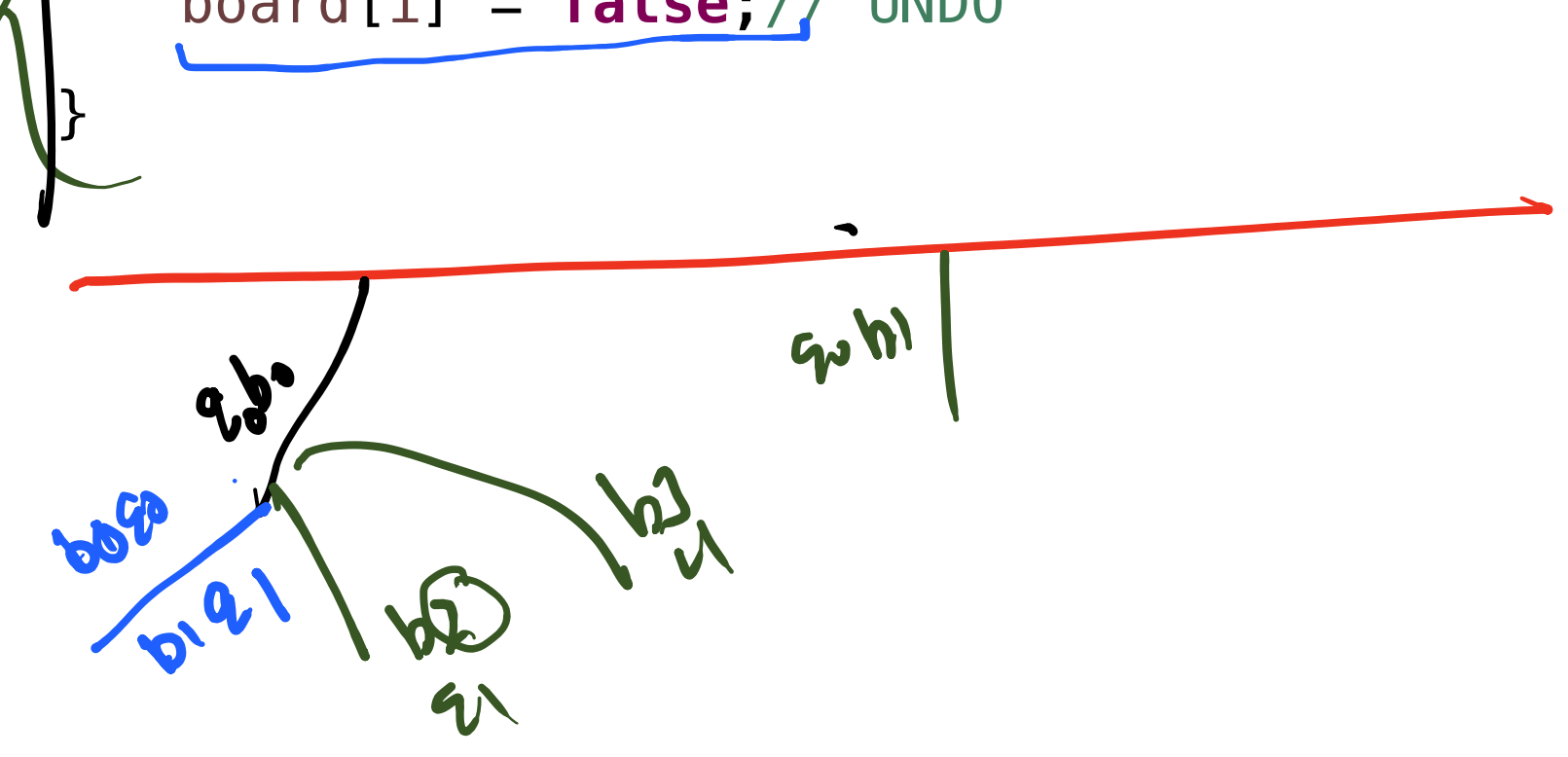
```

Handwritten notes and diagrams:

- Red bracket above `board` parameter: `board`
- Red bracket around `board[i]` and `board[i] = false`: `Do this`
- Blue bracket around `board[i] = true`: `Do this`
- Red bracket around `board[i] = false`: `Undo`
- Diagram of a 4x4 grid with the second column (index 1) highlighted in red, representing the board state.

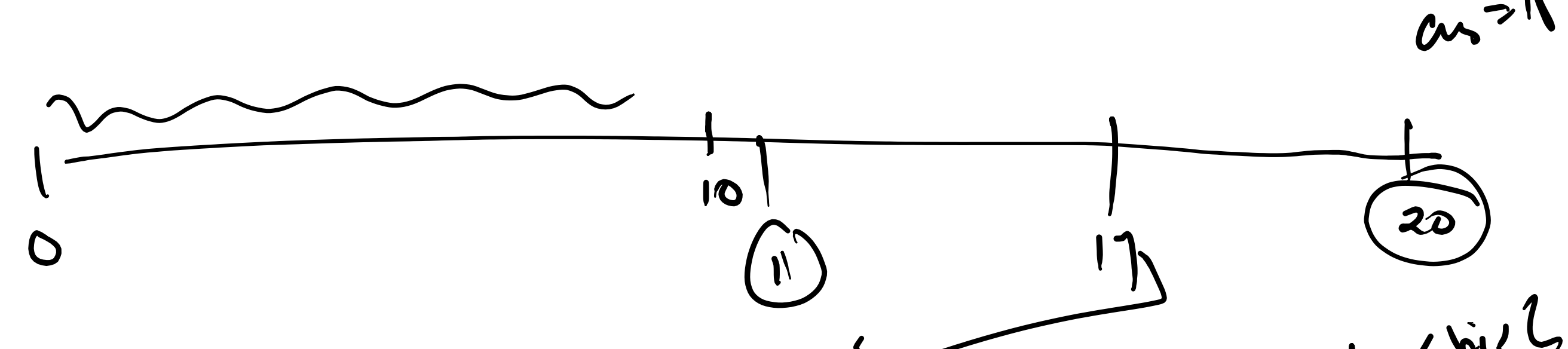
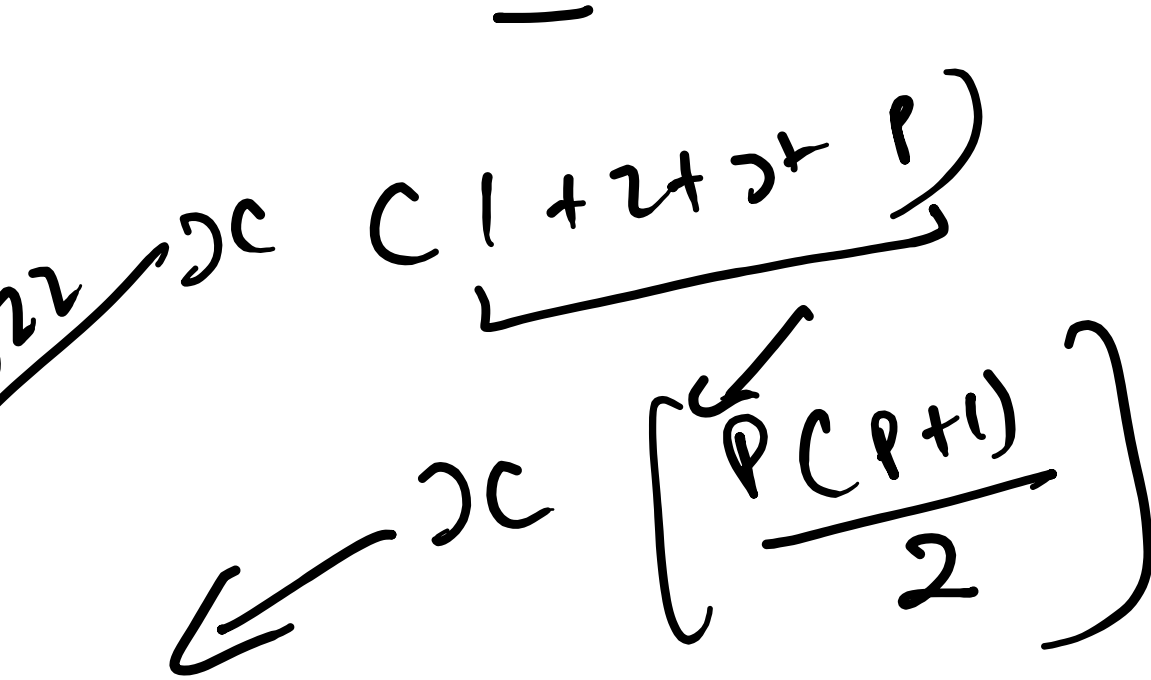
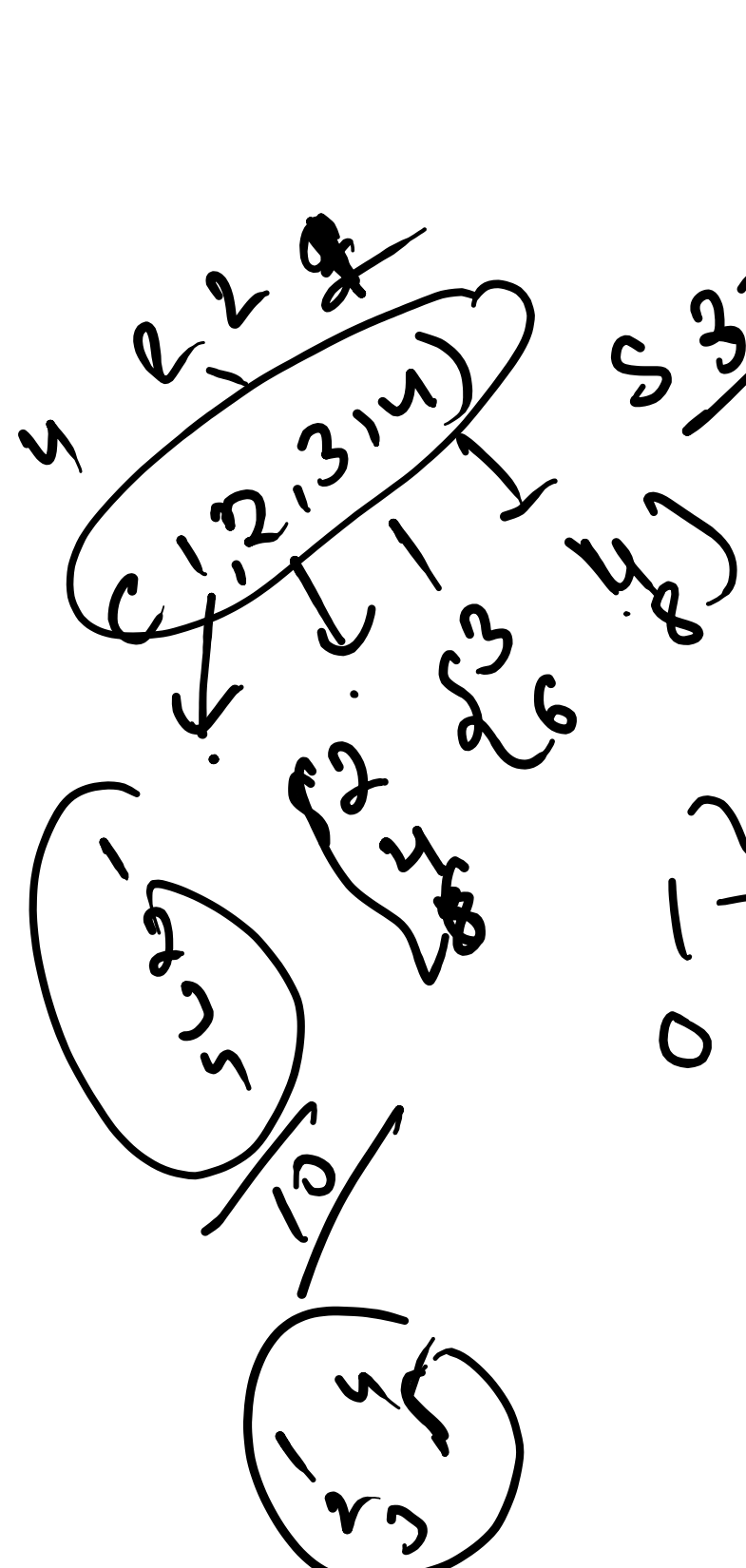


```
public static void Combination(boolean[] board, int tq, int qpsf, String ans, int idx) {
    if (tq == qpsf) {
        System.out.println(ans);
        return;
    }
    for (int i = idx; i < board.length; i++) {
        if (board[i] == false) {
            board[i] = true; // marked kr diya Queen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



$$x + 2x + 3x + 4x + \dots + 9x$$

$$4 \times \frac{\sqrt{10+1}}{2} = \underline{\underline{220}}$$



while (A < B) {
 i = i + 1;
 if (A == B) break;
}

$2 \times 1 = 2$
 $3 \times 1 = 3$
 $2 \times 2 = 4$
 $3 \times 2 = 6$
 $2 \times 3 = 6$
 $3 \times 3 = 9$

$$t = t + \text{ramp}$$

public static char[] arr, int[] m, int[] n
 int cp = 0
 for (int i = 0; i < n; i++)
 {
 cp = cp + arr[i] * m[i];
 }