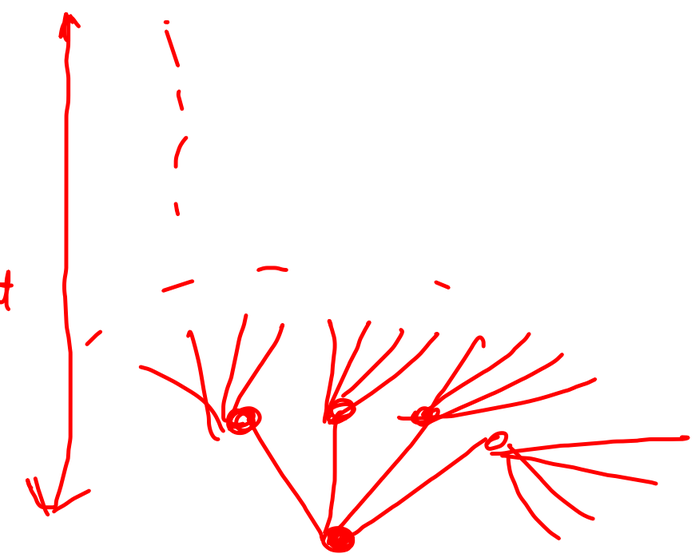
Recursion

1. Expectation
2. Faith
3. Meeting expectation with faith
4. Base Case

Time Complexity

(calls)^{height} + preorder * height
+ postorder * height

levels
⇒ height



① Print Increasing

1.1 fun(n) : 1 - - - n

1.2 fun(n-1) : 1 - - - (n-1)

1.3 M.E ⇒ Postorder : Syso(n);

n=0 : return;

② Print decreasing

1.1 fun(n) : n - - - 1

1.2 fun(n-1) : (n-1) - - 1

1.3 M.E ⇒ Preorder : Syso(n);

n=0 : return;

TC ⇒ $(1)^n + k*n = O(n)$

C++

Mlmr

③ Print Increasing Decreasing

fun(5) : 5 [4 3 2 1 1 2 3 4] 5

fun(4) : 4 3 2 1 1 2 3 4

M.E :→ Pre : Syso(n) fun(n-1) Post Syso(n)

$$TC \Rightarrow \overset{n}{(1)} + k * n + k * n$$

$$= 2kn + 1 \Rightarrow O(n)$$

Power - linear

Power - logarithmic

power(x, n)

$x * x * x * \cdots$ n times

power(x, n-1)

$x^{n-1} = x * x * \cdots (n-1)$ times

return $x * x^{n-1}$;

$x^{n/2}$ = power(x, n/2)

if (n % 2 == 0)
return $x^{n/2} * x^{n/2}$;

if (n % 2 == 1)
return $x^{n/2} * x^{n/2} * x$;

Base Case : n = 0 : return 1;

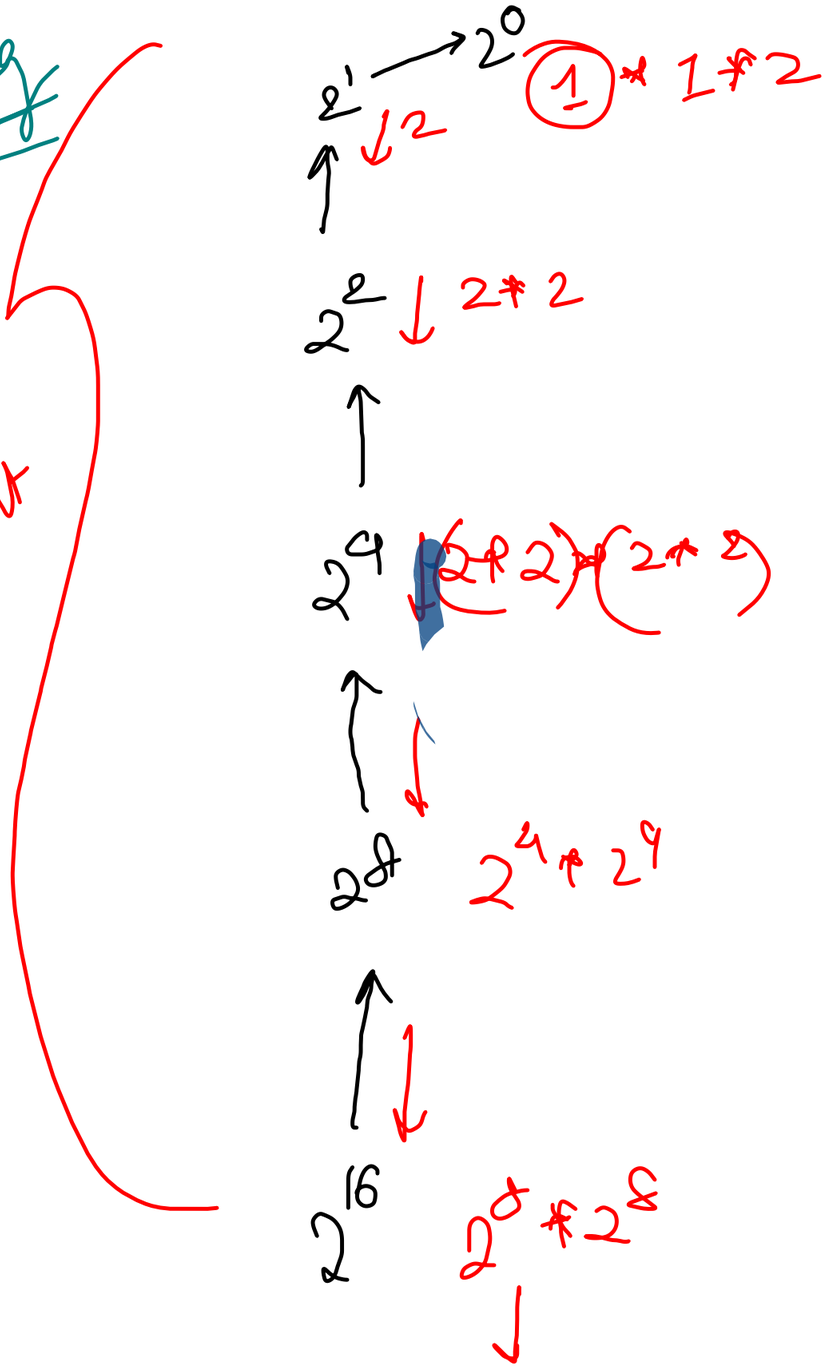
TC ⇒ $(1)^n + k * n \Rightarrow O(n)$

TC ⇒ $(1)^{\log_2 n} + k * \log_2 n$

⇒ $O(\log n) \approx O(1)$

Power-log

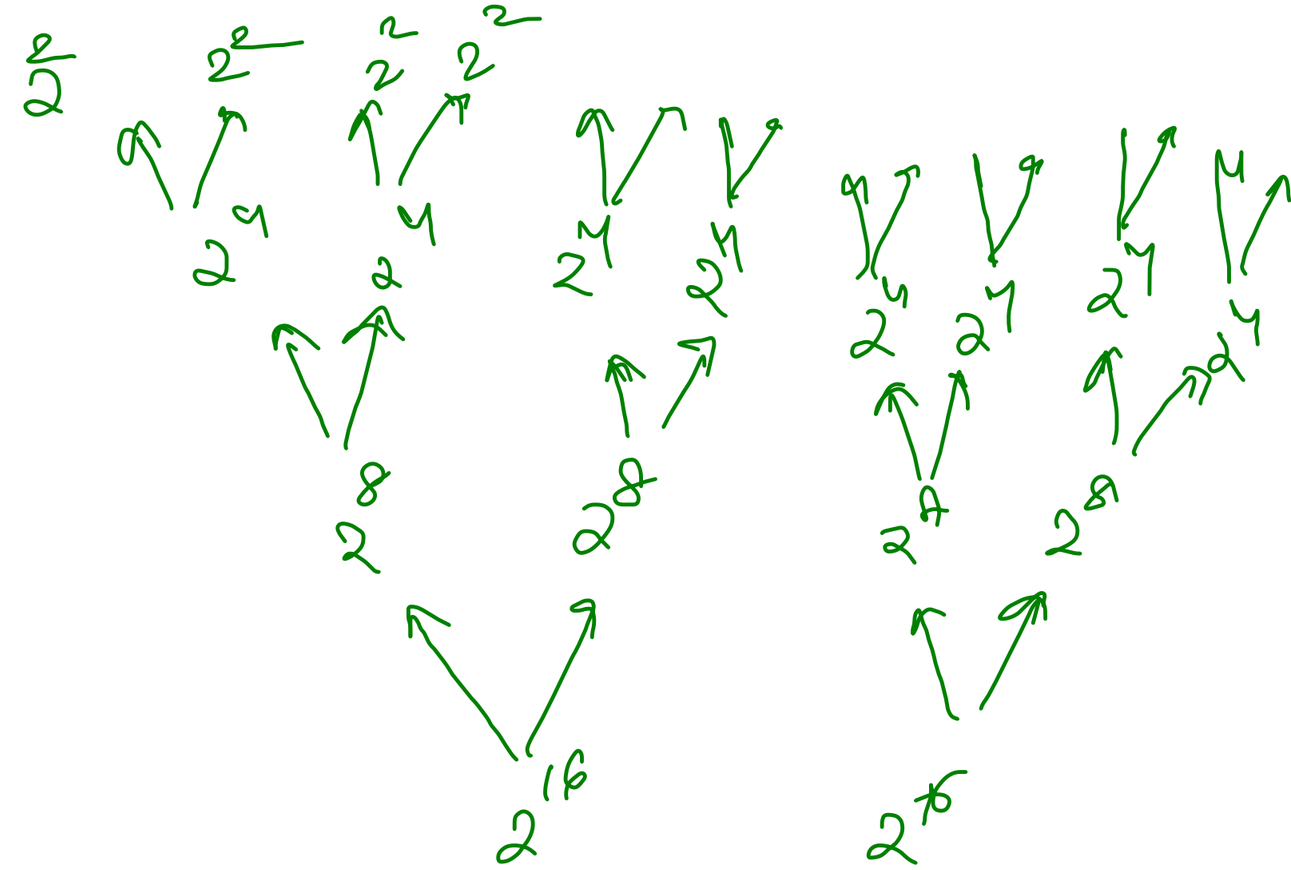
log n height




```java
public static int power(int x, int n){
    // Base Case x^0 = 1
    if(n == 0) return 1;

    // 1. Faith : x ^ n/2
    int xpnb2 = power(x, n/2);

    // 2. x^n = x^n/2 * x^n/2
    int xpn = xpnb2 * xpnb2;

    // 3. If n is odd
    if(n % 2 == 1) xpn = xpn * x;

    // 4. Return
    return xpn;
}
```

Power - 2

```
if(n%2==0)
  return power(x,n/2) * power(x,n/2)
else
  return pow(x,n/2) * pow(x,n/2) * x;
```



height $\Rightarrow \log_2 n$

calls $\Rightarrow 2$

$$(2)^{\log_2 n} + k * \log_2 n$$

$$= n + k \log_2 n$$

$$\Rightarrow O(n)$$

long

① 10^{18} % mod

Because ab

int max value

$10^9 + 7$

long ⇒ $\approx 10^{18}$

int ⇒ $2^{31} - 1 \approx 10^9$

② → $(a+b) \% m$

$(a-b) \% m$

$(a*b) \% m$

Modulus formulas

1. $(a+b)\%m = (a\%m + b\%m)\%m$

2. $(a-b)\%m = ((a\%m - b\%m)\%m + m)\%m$

3. $(a*b)\%m = ((a\%m) * (b\%m))\%m$

4. Division $\Rightarrow$ $\boxed{\text{Modulo Inverse}}$

{} → {} → {2} → {2} → {2} → {2} → {2, 6} → {2,6,7} → {2,6,7} num = 6

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

{2,6,7} ← {6,7} ← {7} ← {}

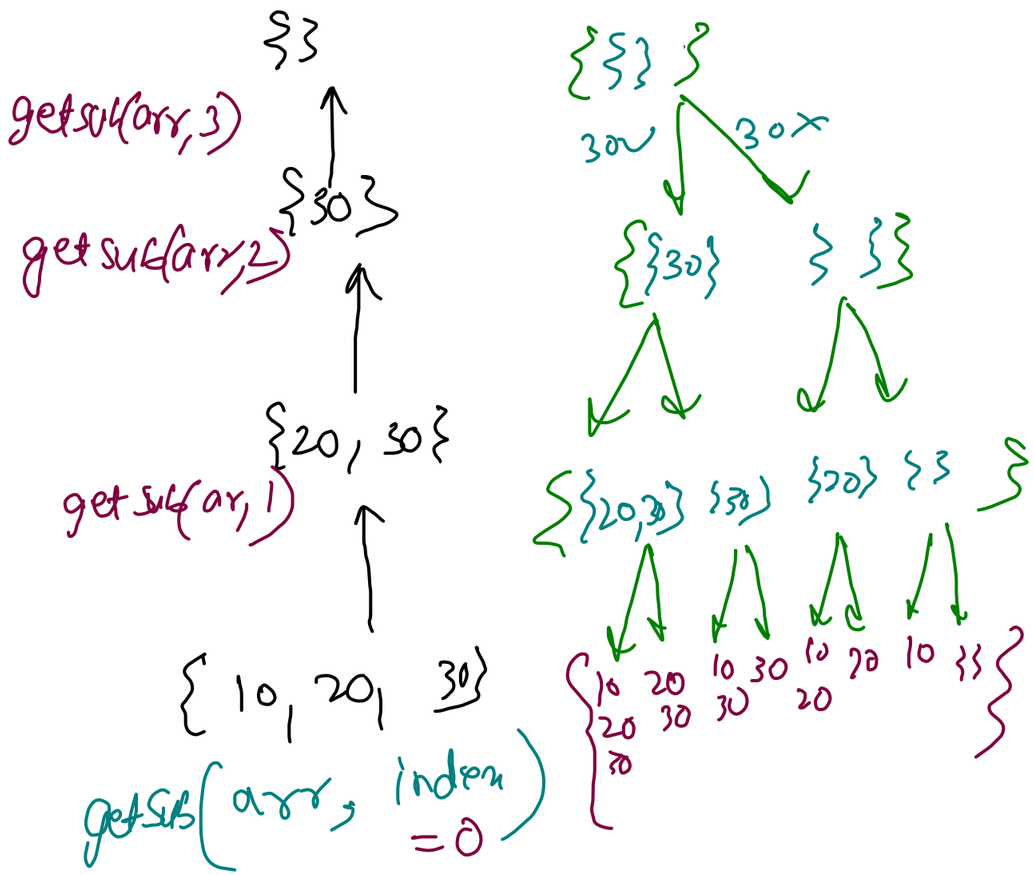
{2, 6, 7}

① preorder: append at last

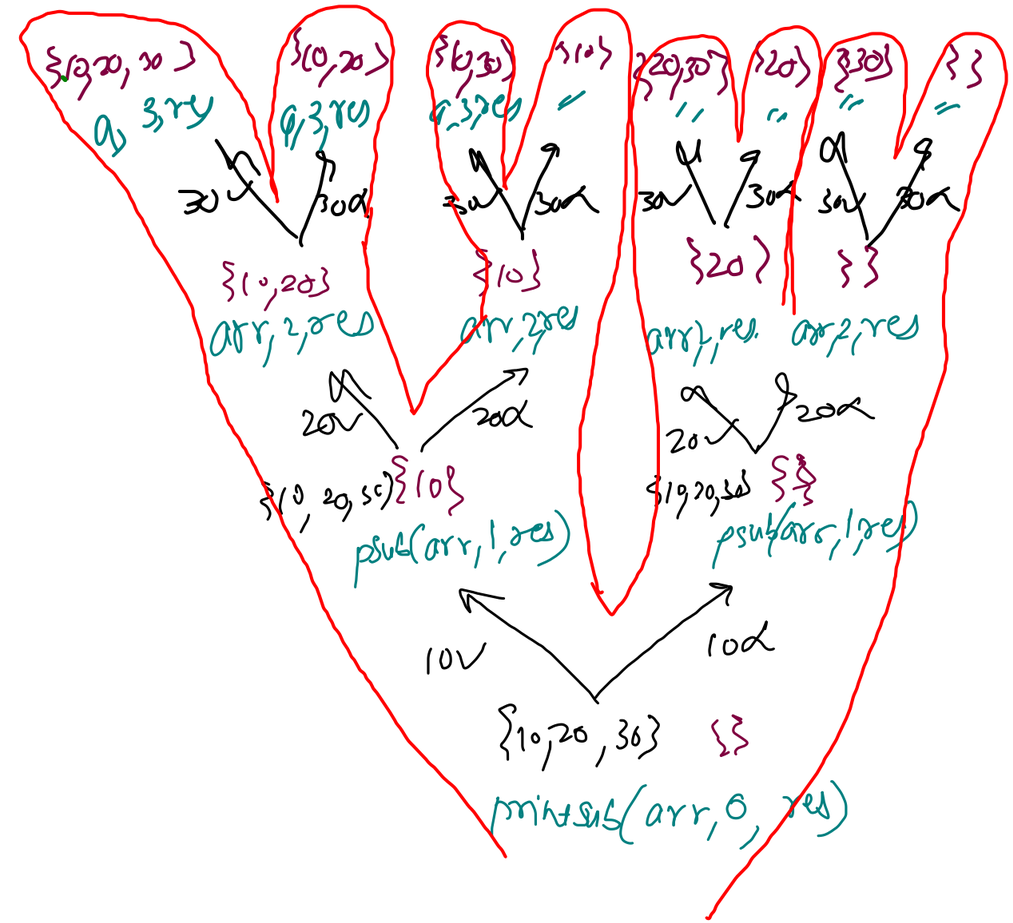
fun(arr; 0; res, num)

② postorder: append at first

Get Subsequence



Print Subsequence



$$TC \Rightarrow \left((2)^n + k*n + k*n \right) \Rightarrow O(2^n)$$

Today's Target

1. Permutations
   - → Box on level
   - → Item on level
   - → Swap method
2. Combinations
   - → Box on level
   - → Item on level

15-20 → follow up } → 30 Q

3. String Permutations {Unique}
   - → Box on level
   - → Item on level
4. Palindromic Permutations

Homework :-

- Factor combinations
- Dictionary Order
  - → small
  - → large

Combinatn

$${}^nC_r \rightarrow \frac{n!}{(n-r)!\,r!}$$

i i — —

i — i —

i — — i

— i i —

— i — i

— — i i

$${}^4C_2$$

identical items

Permutation

selectn + Arrangement

$${}^nP_r \rightarrow \frac{n!}{(n-r)!}$$

12 — — 21 — —

1 — 2 — 2 — 1 —

1 — — 2 2 — — 1

— 12 — — 21 —

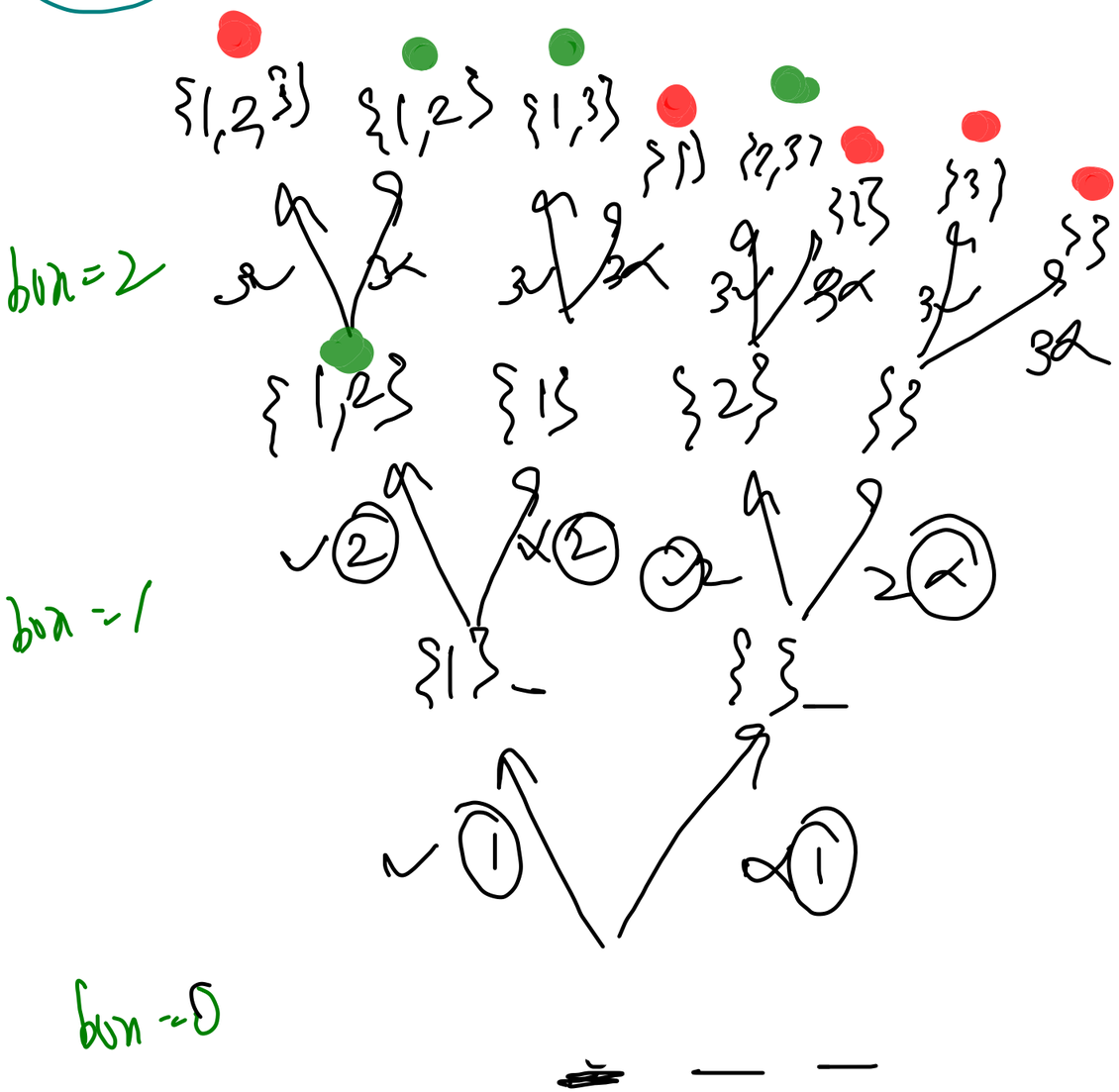
— 1 — 2 — 2 —

— — 12 — — 21

non-identical items

$${}^4P_2 = {}^4C_2 * 2!$$

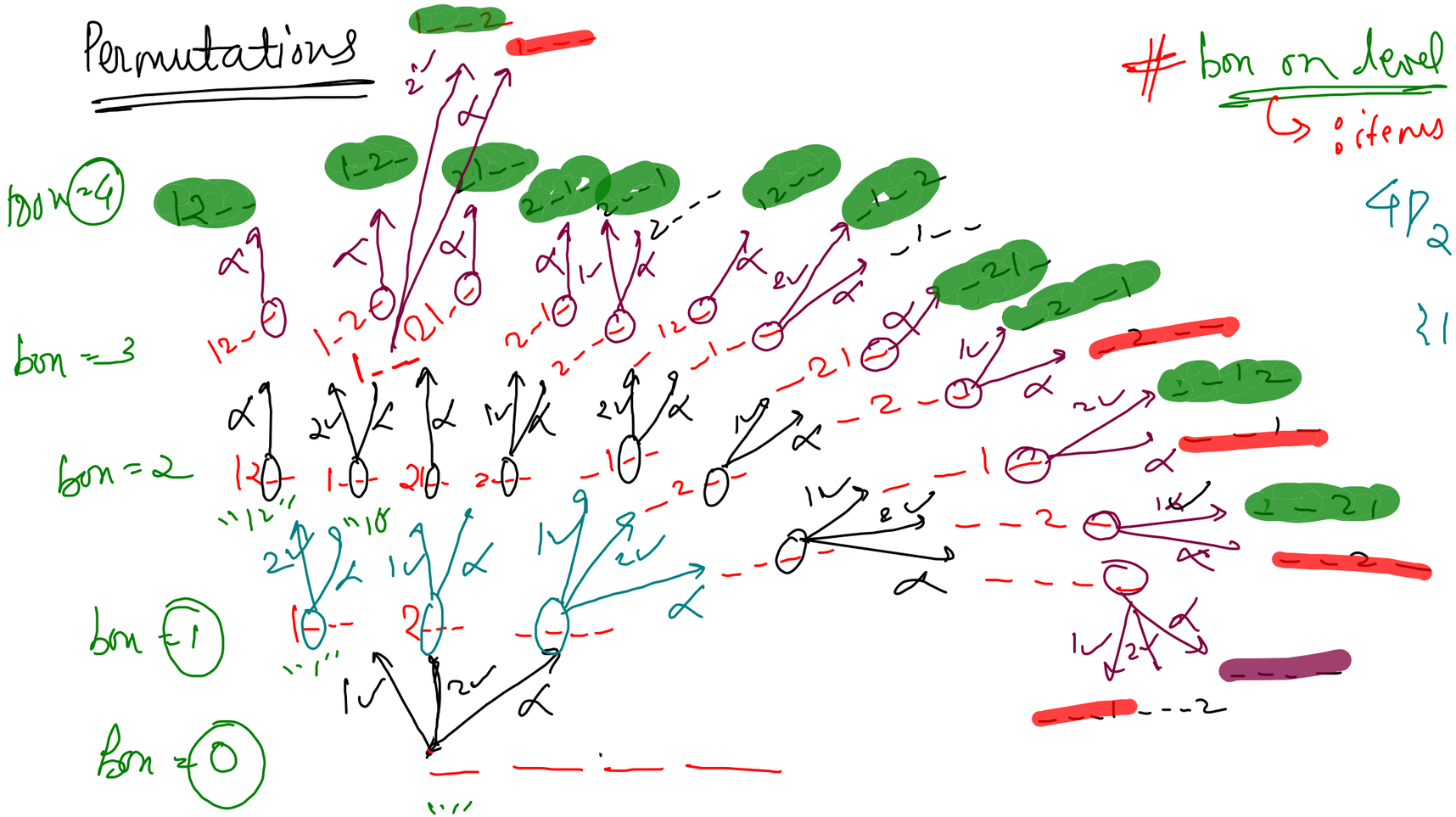
$^3C_2 = 3$




```java
public void combine(List<List<Integer>> combinations,
    List<Integer> combination, int currentBox, int n, int k){
    if(currentBox == n){
        if(combination.size() == k){
            // deep copy
            List<Integer> temp = new ArrayList<>(combination);
            combinations.add(temp);
        }
        return;
    }

    // options -> current Box -> item should be placed or not

    // yes
    combination.add(currentBox + 1);
    combine(combinations, combination, currentBox + 1, n, k);
    combination.remove(combination.size() - 1);

    // no
    combine(combinations, combination, currentBox + 1, n, k);
}
```

{1,2}, {1,3}, {2,3}

Permutations



bon on level

↳ : items were on opt. bon + α

$^4P_2$

{1, 2}

$^4P_0 + {}^4P_1 + {}^4P_2$

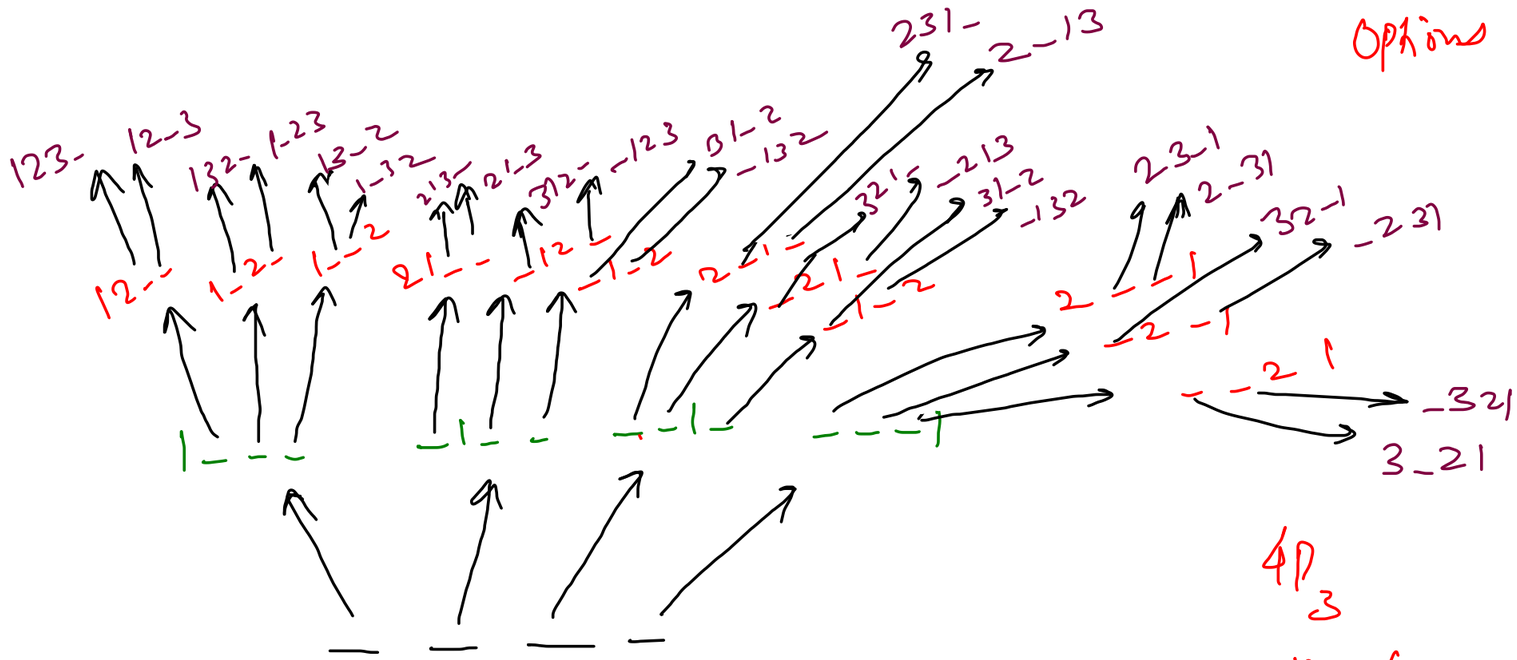
Permutation - 2

items on level: boxes are options

item = 3

item = 2

item = 1



$^4P_3$

n = 4

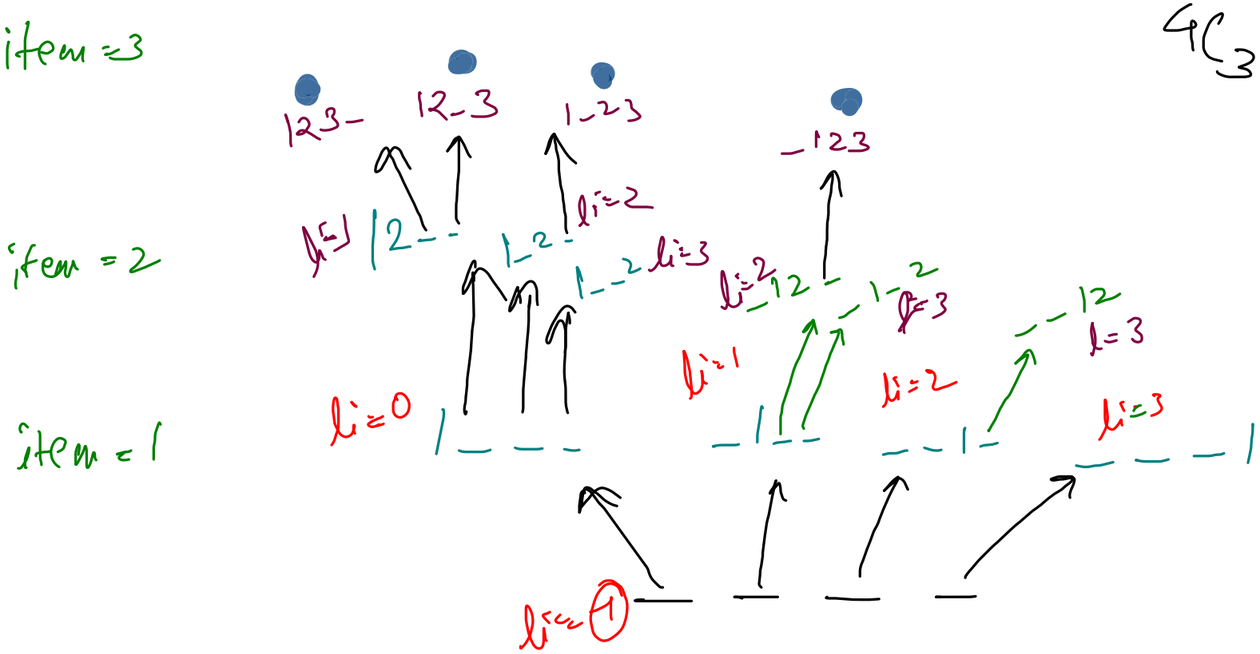
r = 3

```java
public static void permutations(int[] boxes, int ci, int ti){
  // write your code here
  if(ci == ti){
      for(int val: boxes) System.out.print(val);
      System.out.println();
  }

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

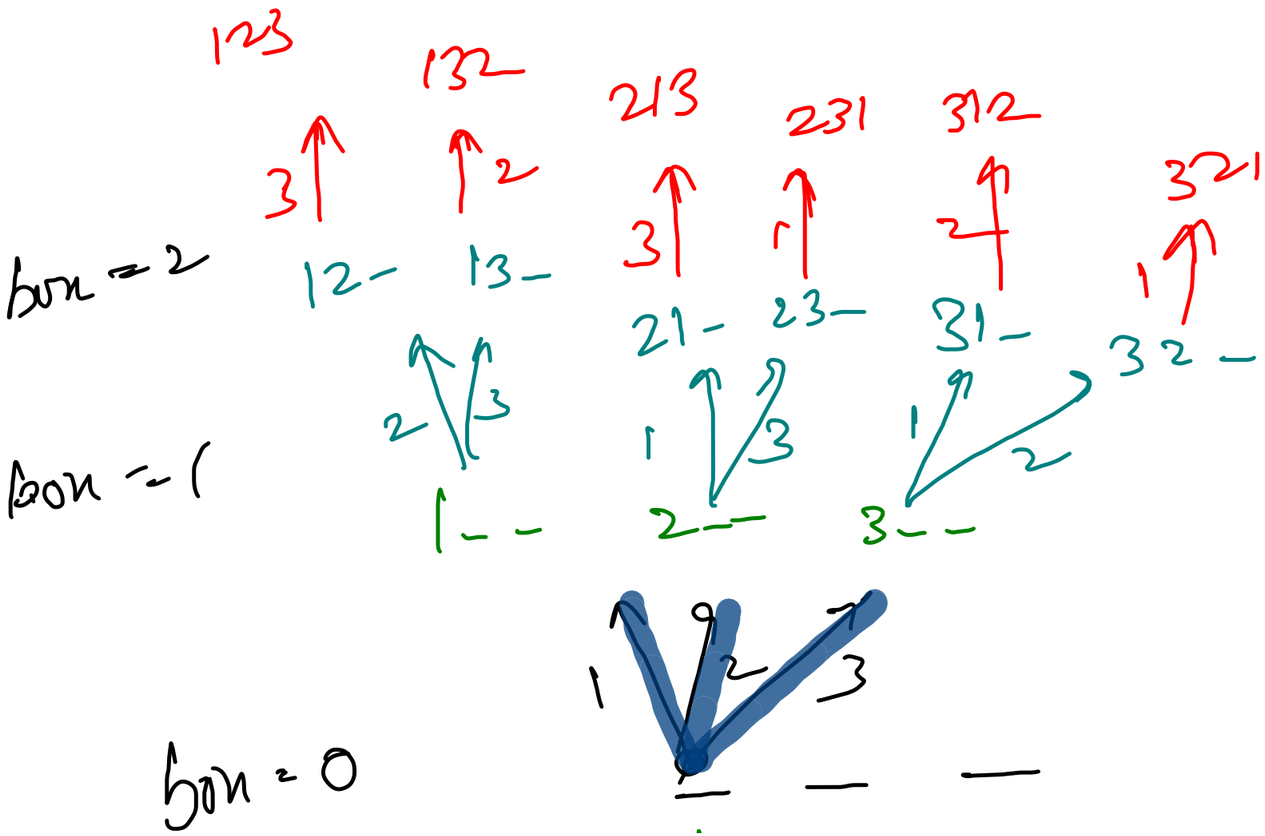
Combinations -2

{Items ⇒ level ; Option ⇒ box}

$^4C_3 = 4$




```java
public static void combunations(int[] boxes, int ci, int ti, int lastItemIdx){
  // write your code here
  if(ci == ti){
      for(int val: boxes){
          if(val == 0) System.out.print("-");
          else System.out.print("i");
      }

      System.out.println();
  }

  // Item -> Chose box
  for(int i=lastItemIdx + 1; i<boxes.length; i++){
      if(boxes[i] == 0){
          boxes[i] = ci + 1;
          combunations(boxes, ci + 1, ti, i);
          boxes[i] = 0;
      }
  }
}
```

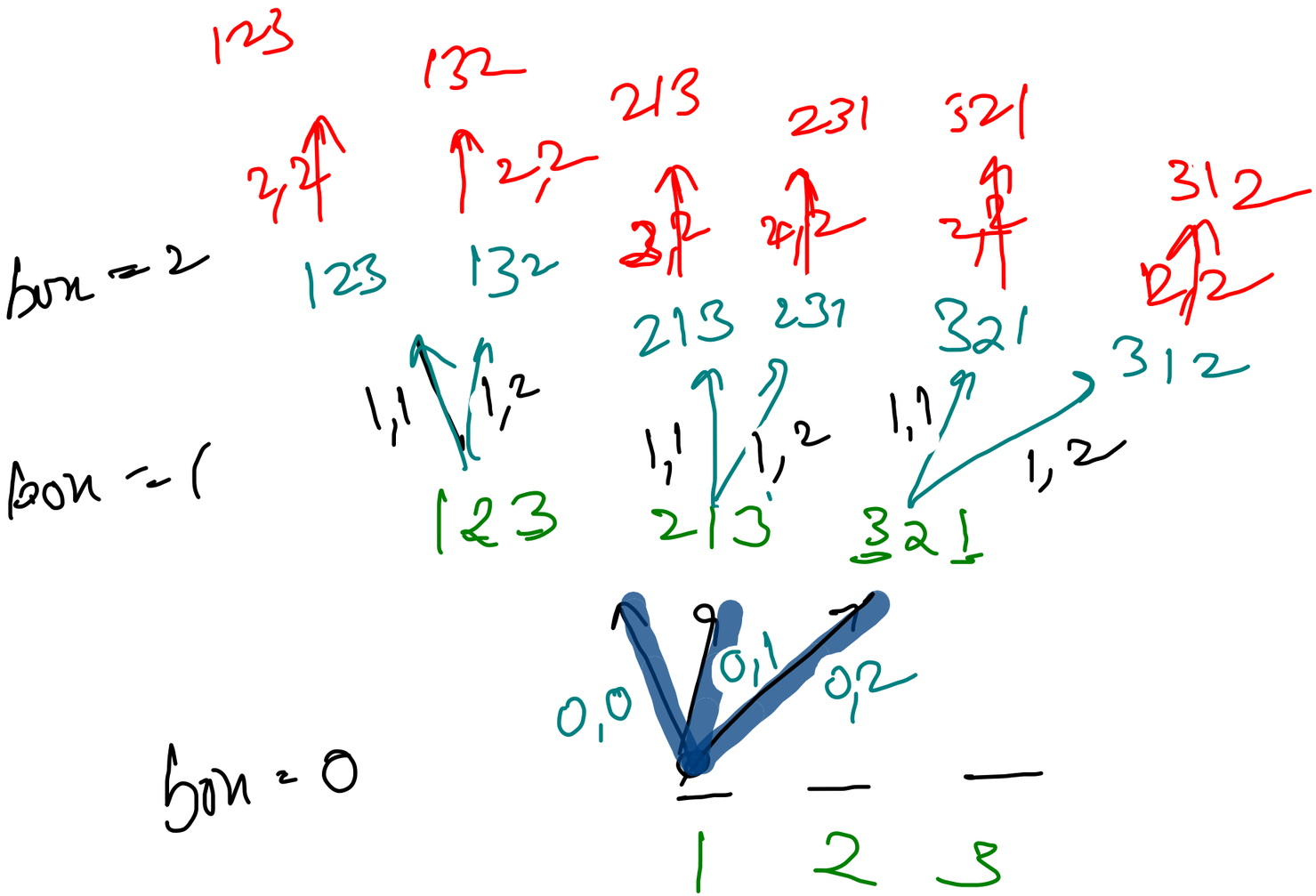


~~boxes~~

items

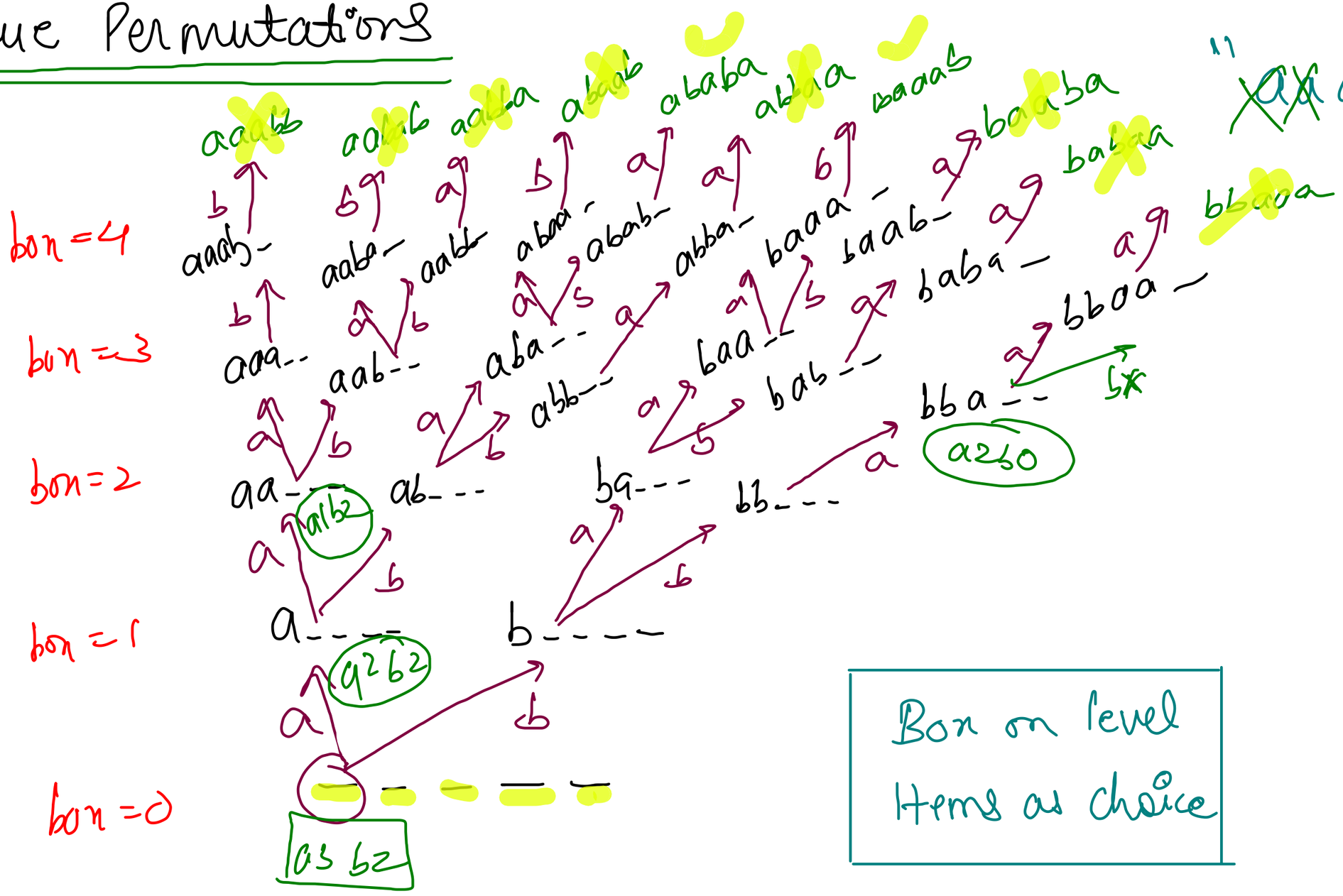
boxes = items

{1, 2, 3}




```java
public void permutations(int currentBox, int[] nums){
    if(currentBox == nums.length){
        List<Integer> copy = new ArrayList<>();
        for(int i=0; i<nums.length; i++){
            copy.add(nums[i]);
        }
        res.add(copy);
        return;
    }

    for(int i=currentBox; i<nums.length; i++){
        swap(nums, currentBox, i);
        permutations(currentBox + 1, nums);
        swap(nums, currentBox, i);
    }
```

Unique Permutations



"aaabb"

$= 5!$

$\frac{3!}{2!}$

$= \frac{5 \times 4 \times 3 \times 2}{3 \times 2 \times 2}$

$= 10$

Box on level
Items as choice

helper

```java
public void permutations(HashMap<Integer, Integer> freq, int[] nums,
        int currentBox, List<Integer> ans){
    if(currentBox == nums.length){
        List<Integer> copy = new ArrayList<>(ans);
        res.add(copy);
    }

    for(Integer key: freq.keySet()){
        int val = freq.get(key);
        if(val > 0){
            freq.put(key, val - 1);
            ans.add(key);

            permutations(freq, nums, currentBox + 1, ans);

            ans.remove(ans.size() - 1);
            freq.put(key, val);
        }
    }
}
```

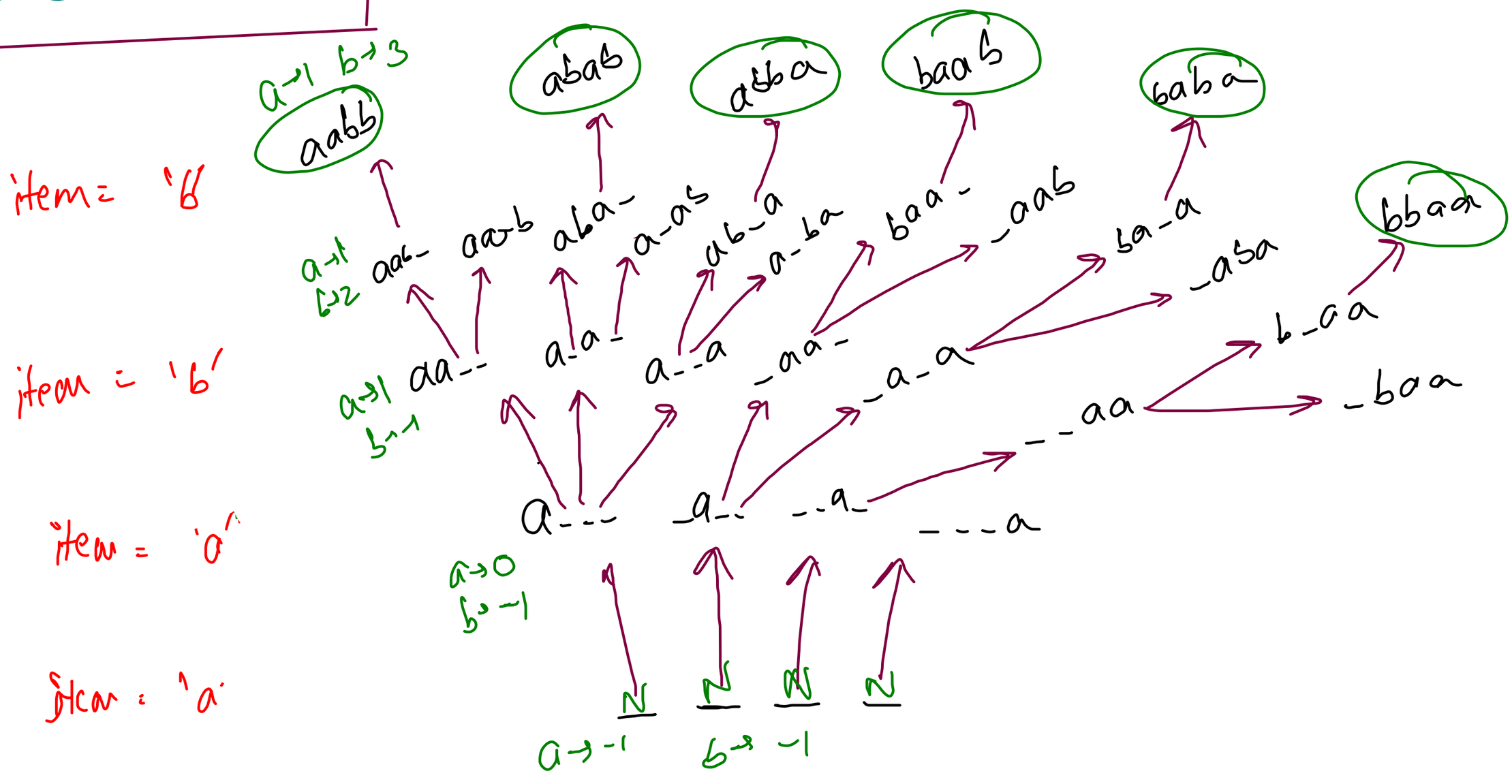
main

```java
HashMap<Integer, Integer> freq = new HashMap<>();
for(int i=0; i<nums.length; i++){
    if(freq.containsKey(nums[i])){
        int val = freq.get(nums[i]);
        freq.put(nums[i], val + 1);
    } else {
        freq.put(nums[i], 1);
    }
}
res = new ArrayList<>();
permutations1(freq, nums, 0, new ArrayList<>());
```

Item on level
Box as choice

"aabb"

$$\frac{4C}{2!\,2!} = \frac{\cancel{4}\times 3\times 2}{\cancel{2\times 2}} = 6$$



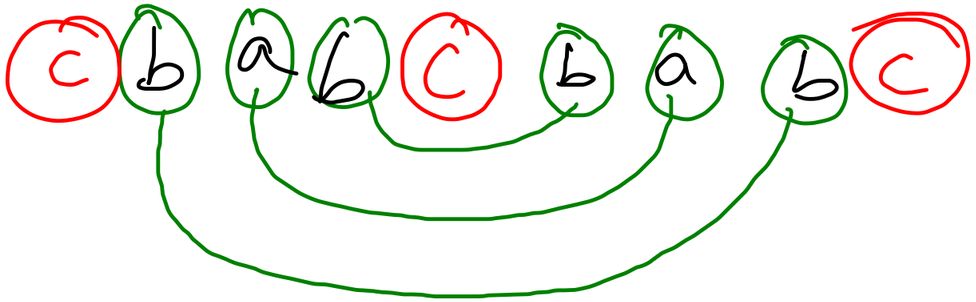
helper

```java
public void permutations2(HashMap<Integer, Integer> lastIdx, int[] nums,
        int currentItem, List<Integer> ans){
    if(currentItem == nums.length){
        List<Integer> copy = new ArrayList<>(ans);
        res.add(copy);
        return;
    }

    int st = lastIdx.get(nums[currentItem]);

    for(int i=st+1; i<ans.size(); i++){
        if(ans.get(i) == null){
            ans.set(i, nums[currentItem]);
            lastIdx.put(nums[currentItem], i);

            permutations2(lastIdx, nums, currentItem + 1, ans);

            lastIdx.put(nums[currentItem], st);
            ans.set(i, null);
        }
    }
}
```

main

```java
HashMap<Integer, Integer> lastIdx = new HashMap<>();
for(int i=0; i<nums.length; i++){
    lastIdx.put(nums[i], -1);
}

List<Integer> ans = new ArrayList<>();
for(int i=0; i<nums.length; i++){
    ans.add(null);
}

res = new ArrayList<>();
permutations2(lastIdx, nums, 0, ans);
return res;
```

#Palindrome Permutat^n - 1



a → 2
b → 4
c → 3

: odd freq. = 0

: odd freq = 1

all char ✓ even freq

atmost 1 char with odd freq

```java
public boolean canPermutePalindrome(String s) {
    HashMap<Character, Integer> freq = new HashMap<>();
    for(int i=0; i<s.length() ; i++){
        if(freq.containsKey(s.charAt(i)) == false){
            freq.put(s.charAt(i), 1);
        } else {
            freq.put(s.charAt(i), freq.get(s.charAt(i)) + 1);
        }
    }

    int oddCount = 0;
    for(Character c: freq.keySet()){
        if(freq.get(c) % 2 == 1){
            oddCount++;
        }
    }

    if(oddCount > 1) return false;
    return true;
}
```

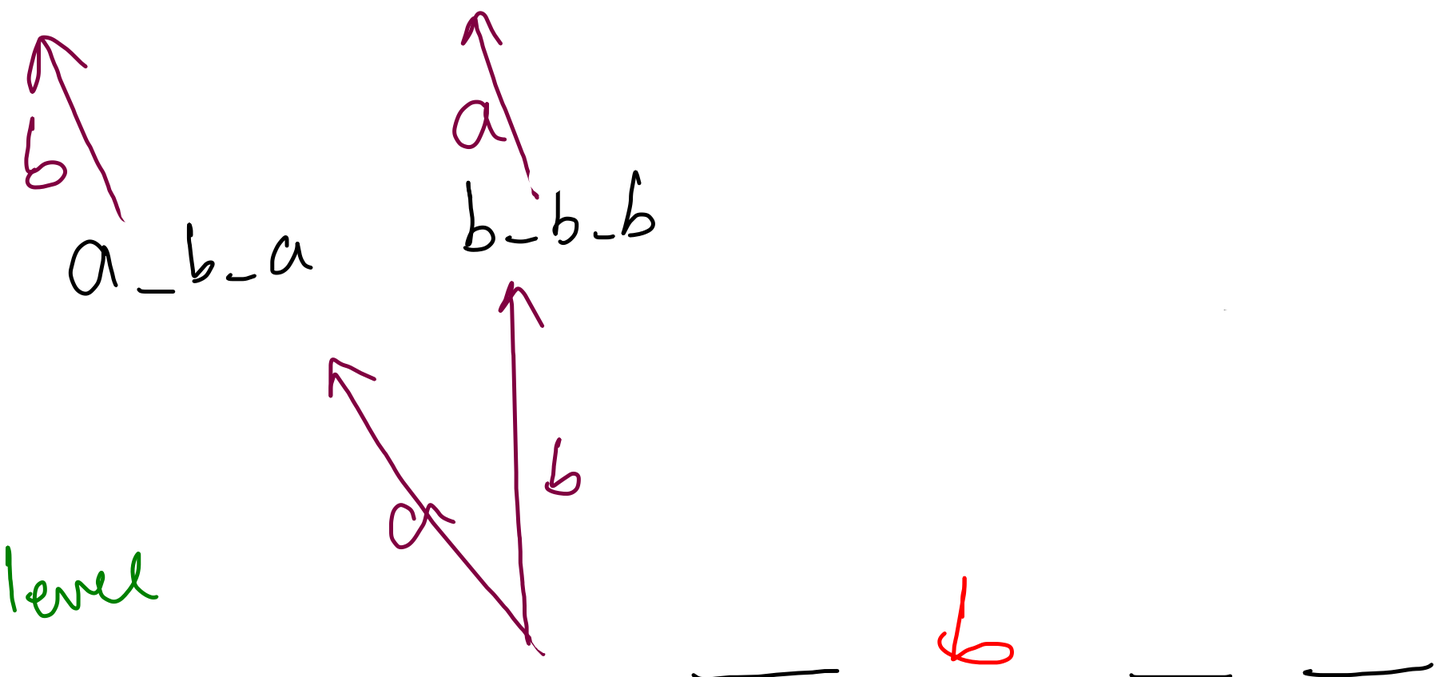
All Palindromic Permutations {PP-11}

aabbb

a2b2

abba

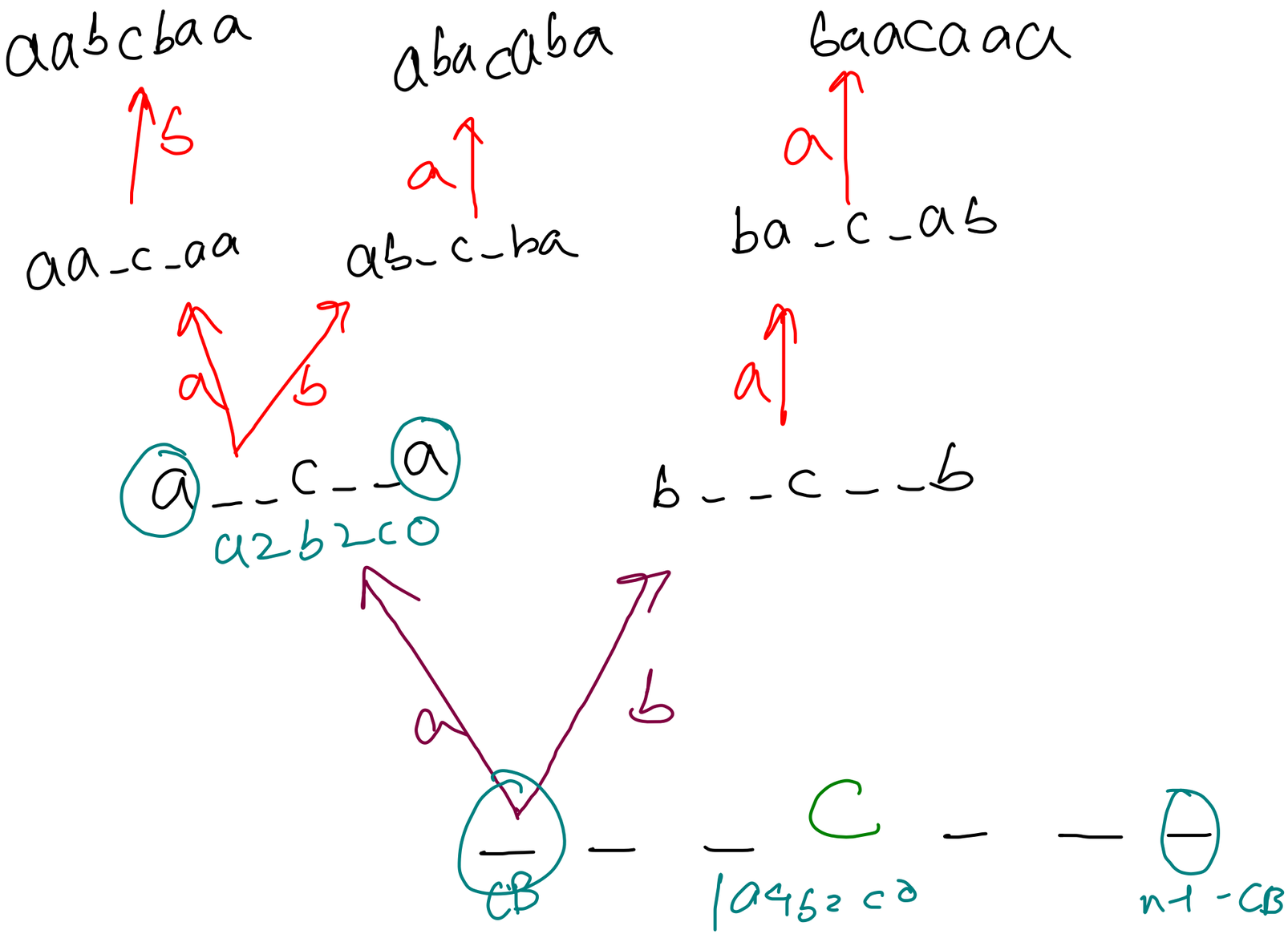
baab



box on level

aaaabbc

a4b2c



bon on level

item as choice

K words {Repitition not allowed} {Combinations}

a2 b3 c2 d2 e

aabbbccdde

3

unique characters : 5 boxes

1. Box on level

2. Item on level :→ 3 items

→ k characters

a, b, c, d, e

↓

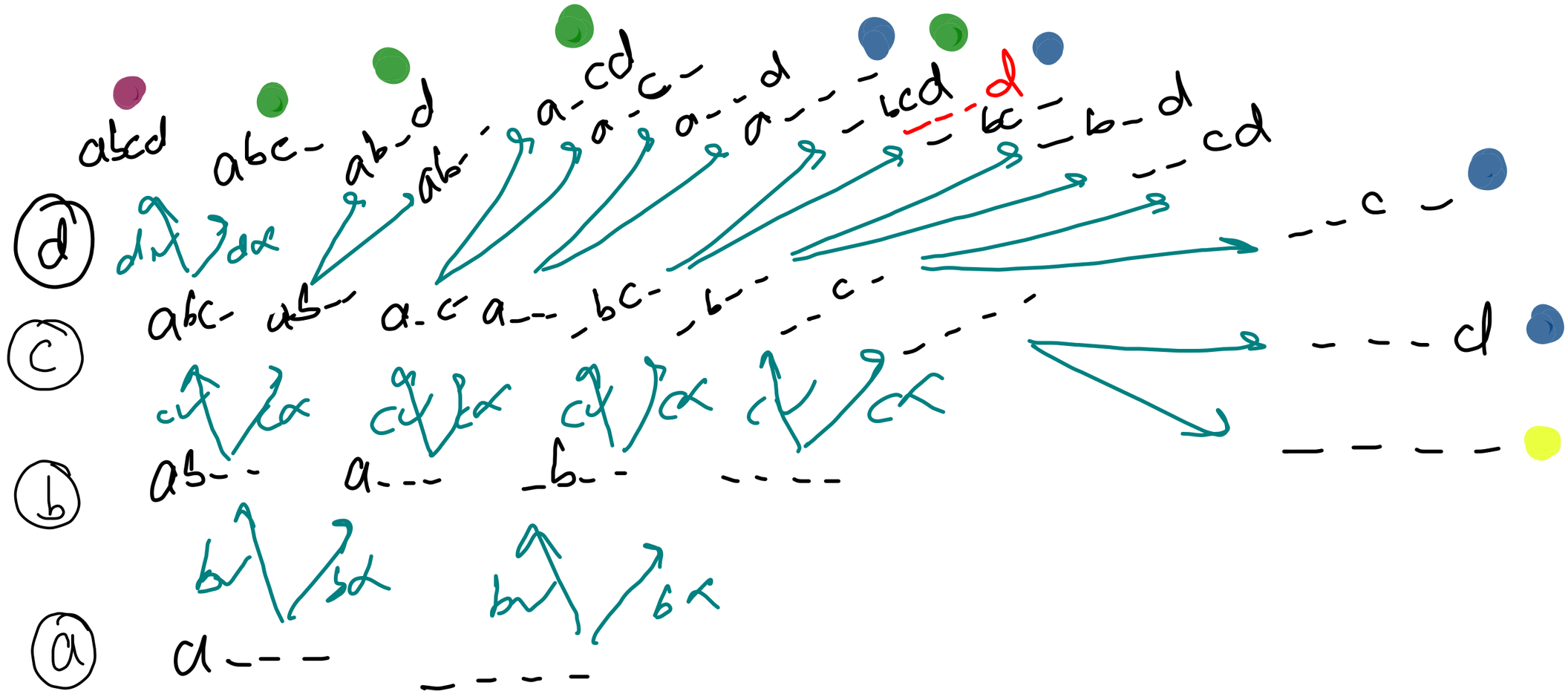
3 items

{ a b c d e }

_ _ _ _ _

$^5C_3 = \frac{5 * 4 * 3 * 2}{3 * 2 * 2} = 10$

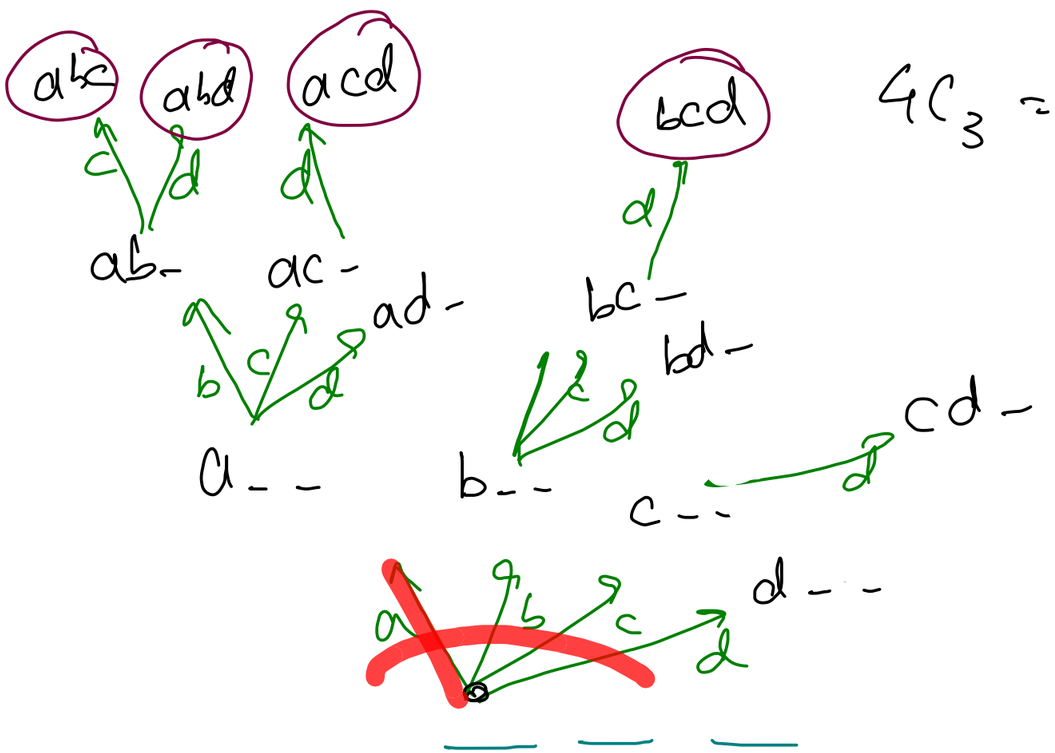
{ }



$$2^4 = {}^4C_0 + {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4$$

a2b3c2d2

{a, b, c, d}



$4C_3 = 4$

$O((k)^{\text{unique char}})$

a2b3c2d

{a,b,c,d}

```
public static void combination(ArrayList<Character> chs, int lastIdx, String res, int k){
    if(res.length() == k){
        System.out.println(res);
        return;
    }

    for(int i=lastIdx + 1; i<chs.size(); i++){
        combination(chs, i, res + chs.get(i), k);
    }
}
```

Today's Questions

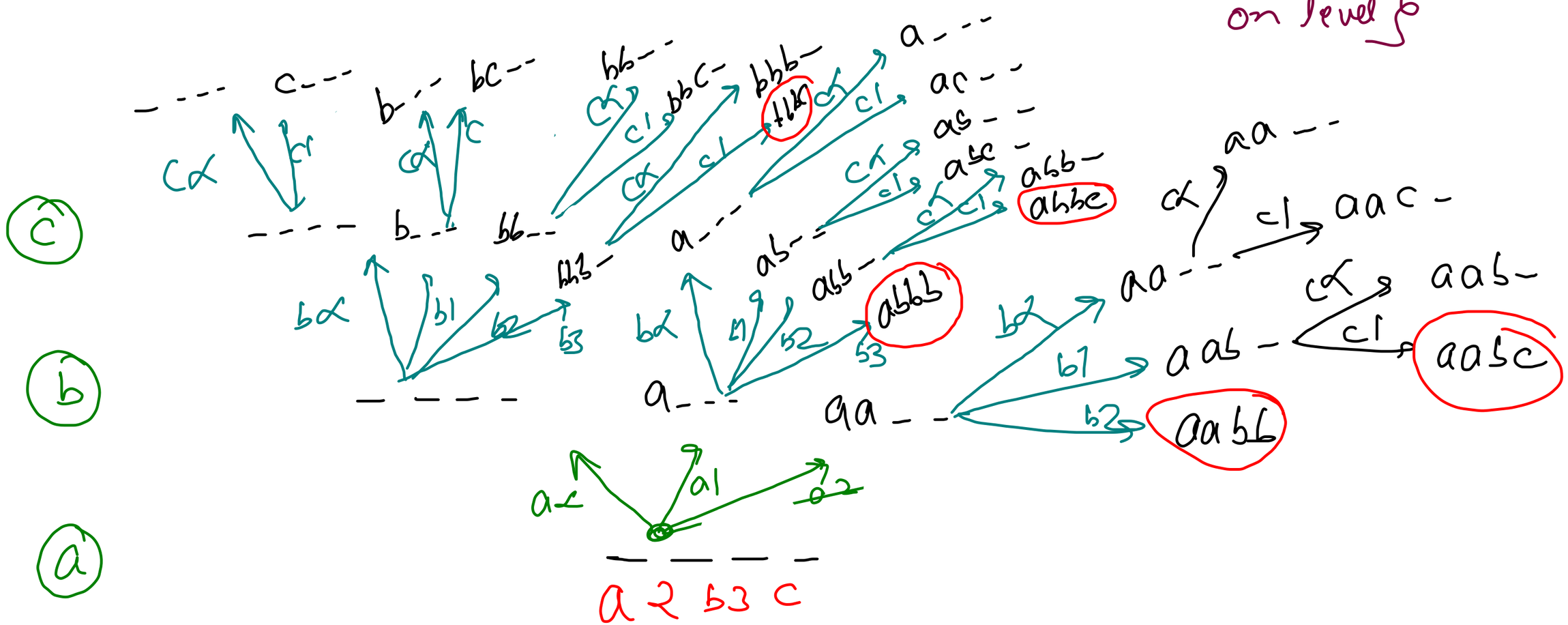
1. Words - K Selection - 1
   Words - K Selection - 2
   } String Combination → Repitition not allowed
2. Words - K Length Words - 1
   Words - K Length Words - 2
3. Words - K Selection - 3
   Words - K Selection - 4
   } String Combination → Repitition allowed
4. Words - K Length Words - 3
   Words - K Length Words - 4
5. Coin Change - Combinations - 1
   Coin Change - Combinations - 2
6. Coin Change - Permutations - 1
   Coin Change - Permutations - 2

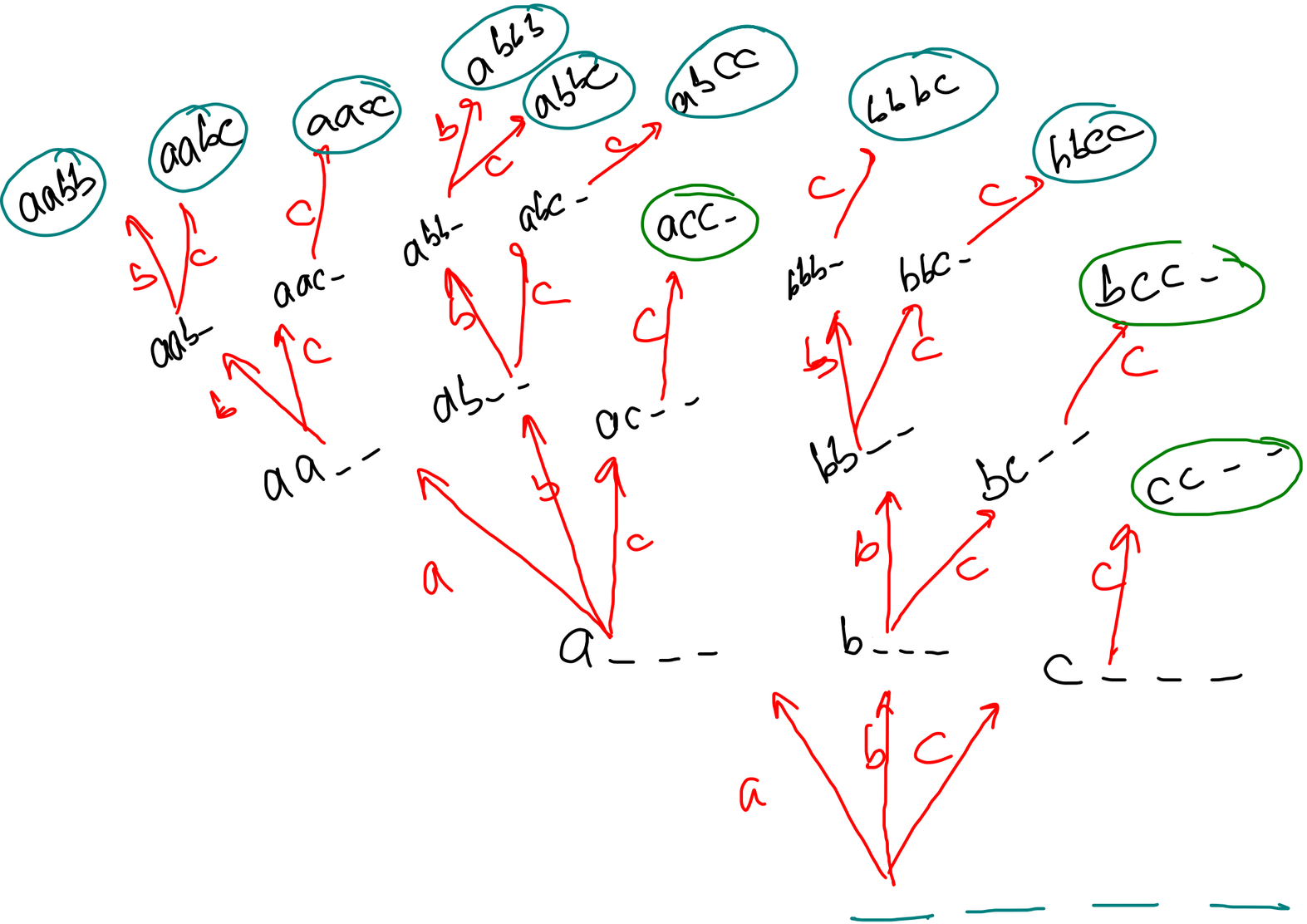
K word select (III)

Repition allowed

$k = 4$

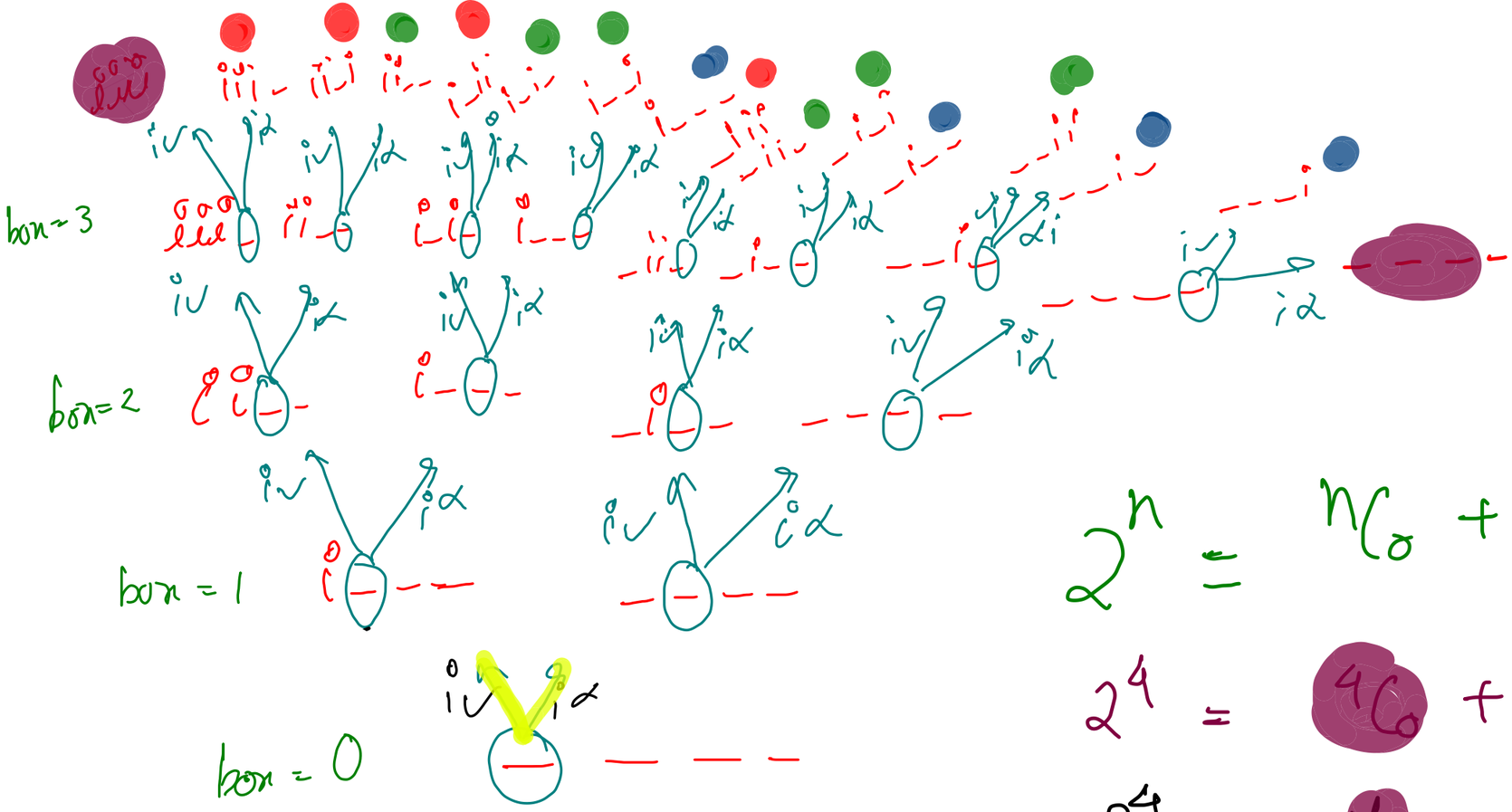
{unique character on level}





K words
Selection - IV
Combination
Repitition allowed

Combinations → Box on level



$\{1, 2\}$

$n = 4$

$r = 2$

$$2^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + \cdots {}^nC_n$$

$$2^4 = {}^4C_0 + {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4$$

$$2^4 = 1 + 4 + 6 + 4 + 1$$

TC ⇒ $(2)^n + \cdots + \cdots$

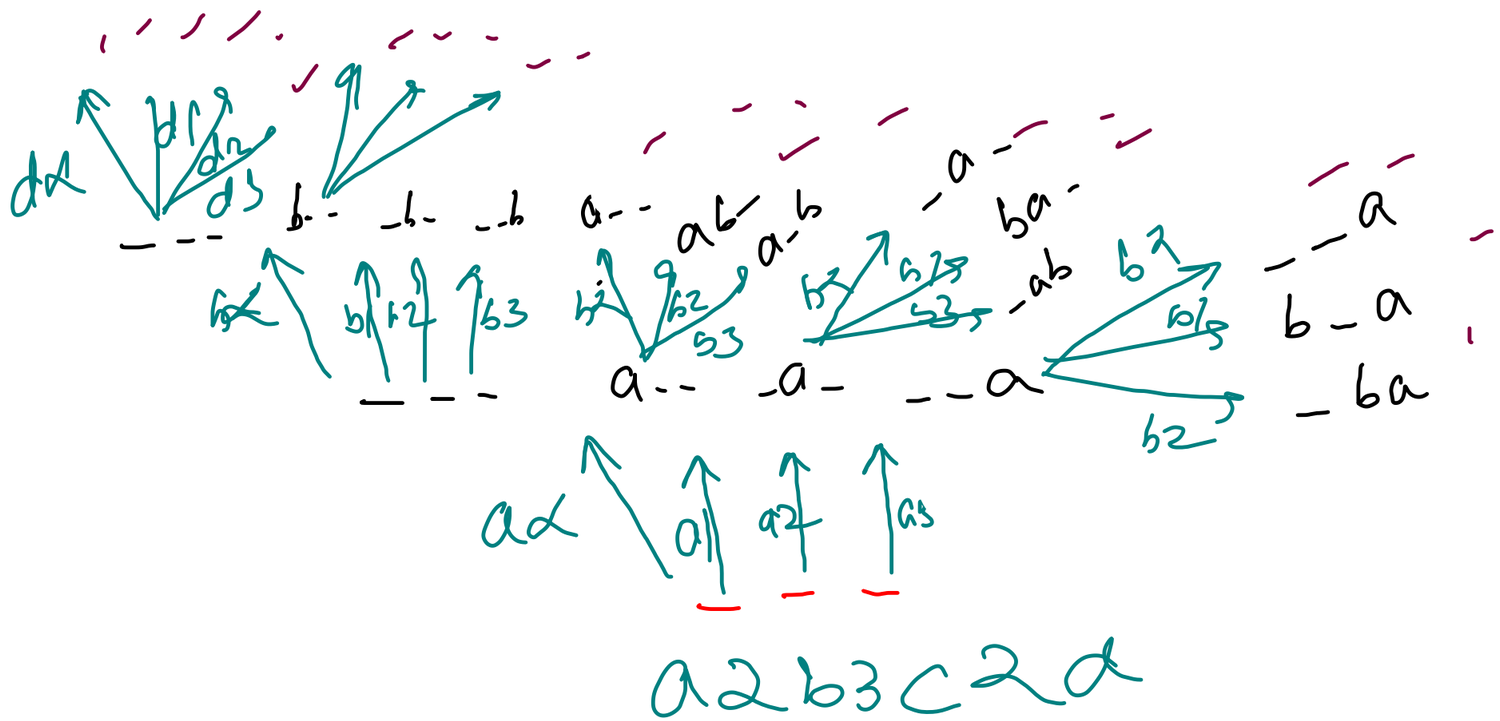
```java
public static void generateSelection(String ustr, int cs, int ts, int lc, HashMap<Character, Integer> freq, String asf) {
  if (cs > ts) {
    System.out.println(asf);
    return;
  }
  if(lc == ustr.length()){
      return;
  }

  for (int i = lc; i < ustr.length(); i++) {
      int oldFreq = freq.get(ustr.charAt(i));
      if(oldFreq > 0){
          freq.put(ustr.charAt(i), oldFreq - 1);
          generateSelection(ustr, cs + 1, ts, i, freq, asf + ustr.charAt(i));
          freq.put(ustr.charAt(i), oldFreq);
      }
  }
}
```

k words Permutation

① Repitition not allowed
→ Unique char
→ Slot on level

② Repitition allowed
→ Unique char
→ Slot on level

Unique character on level

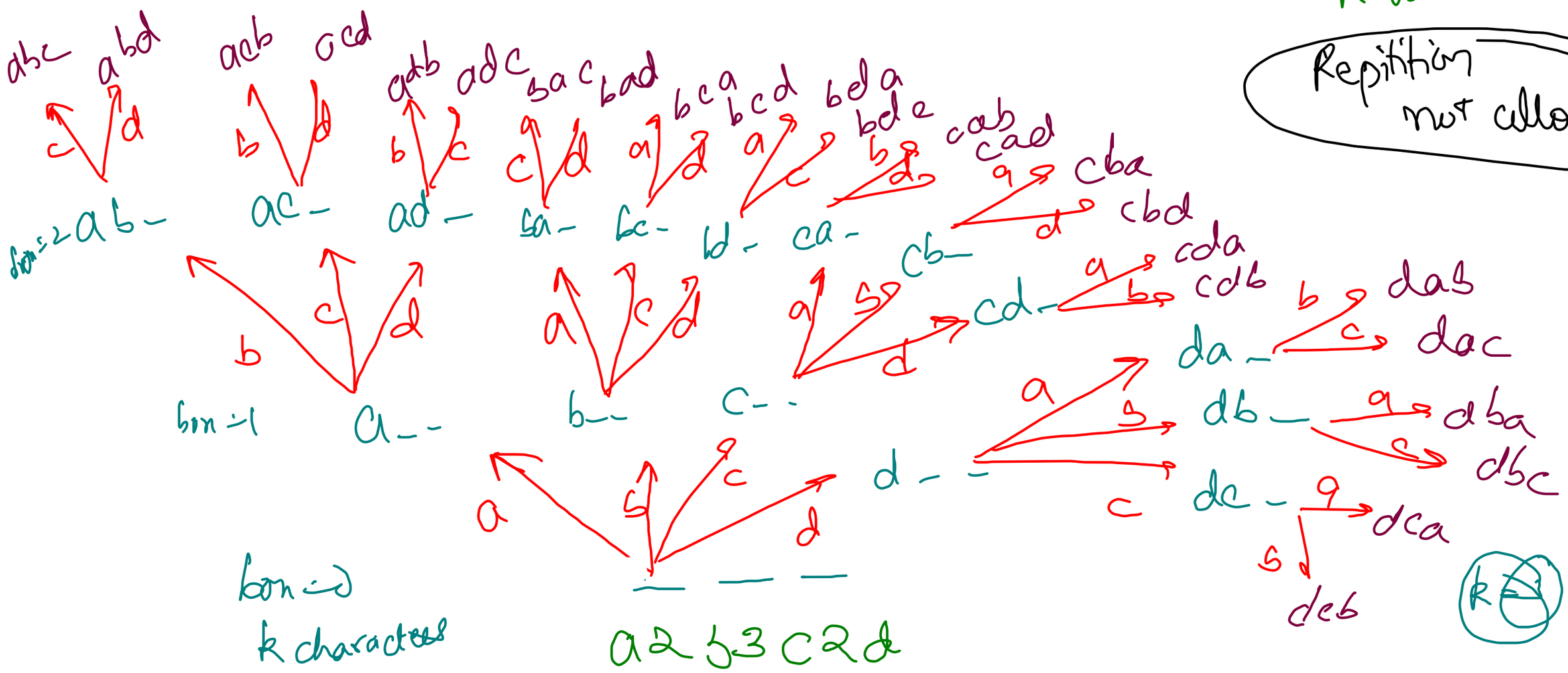


k = 3

Bon on level

k words

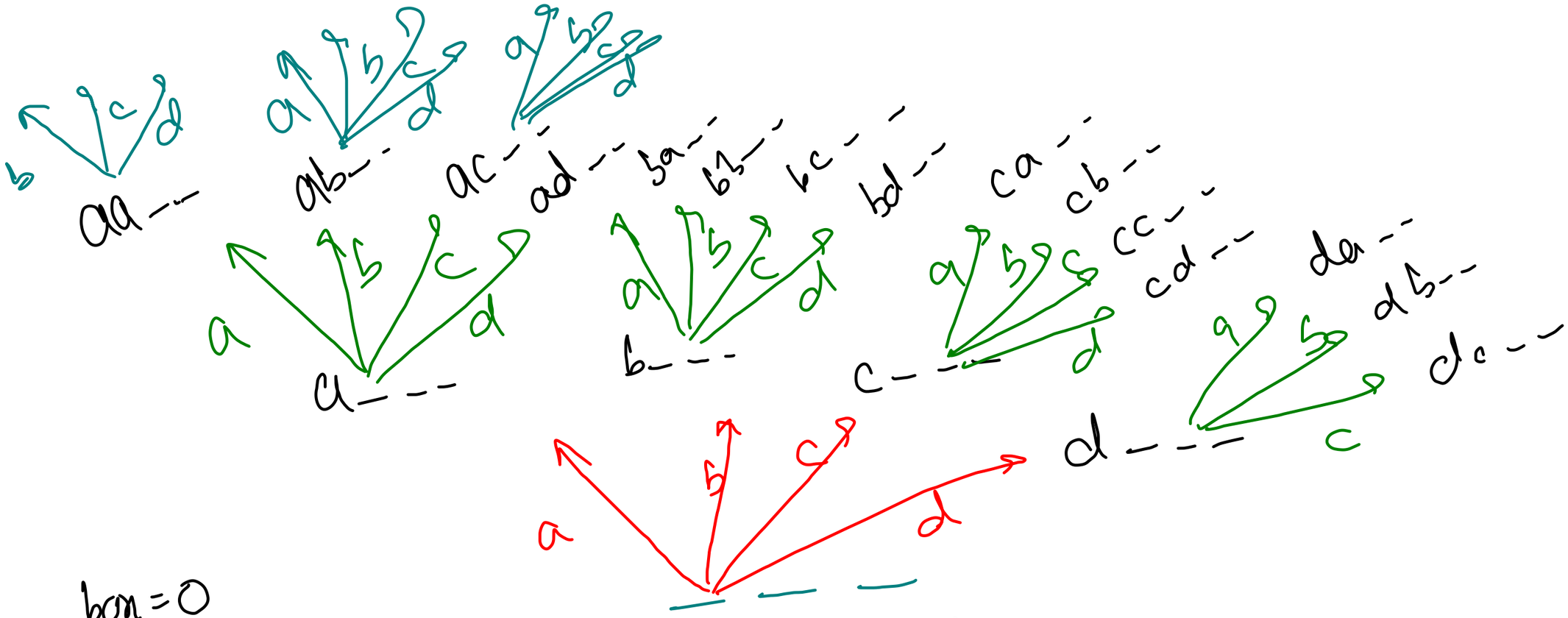
Repitition not allowed



```java
public static void permutation(String res, String ustr, HashSet<Character> unique, int k){
    if(res.length() == k){
        System.out.println(res);
        return;
    }

    for(Character ch: ustr.toCharArray()){
        if(unique.contains(ch) == true){
            unique.remove(ch);
            permutation(res + ch, ustr, unique, k);
            unique.add(ch);
        }
    }
}
```

Press Esc to exit full screen

Repetition allowed



box = 0

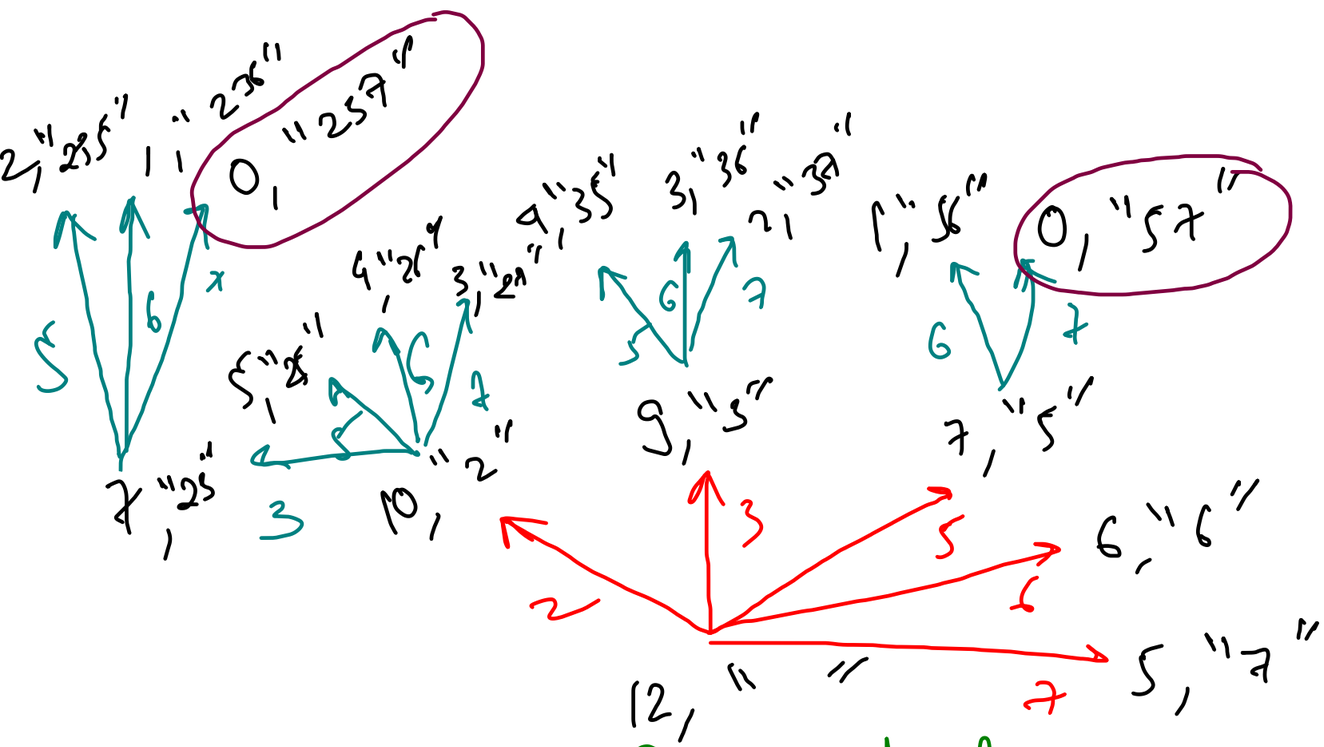
a2b3c2d

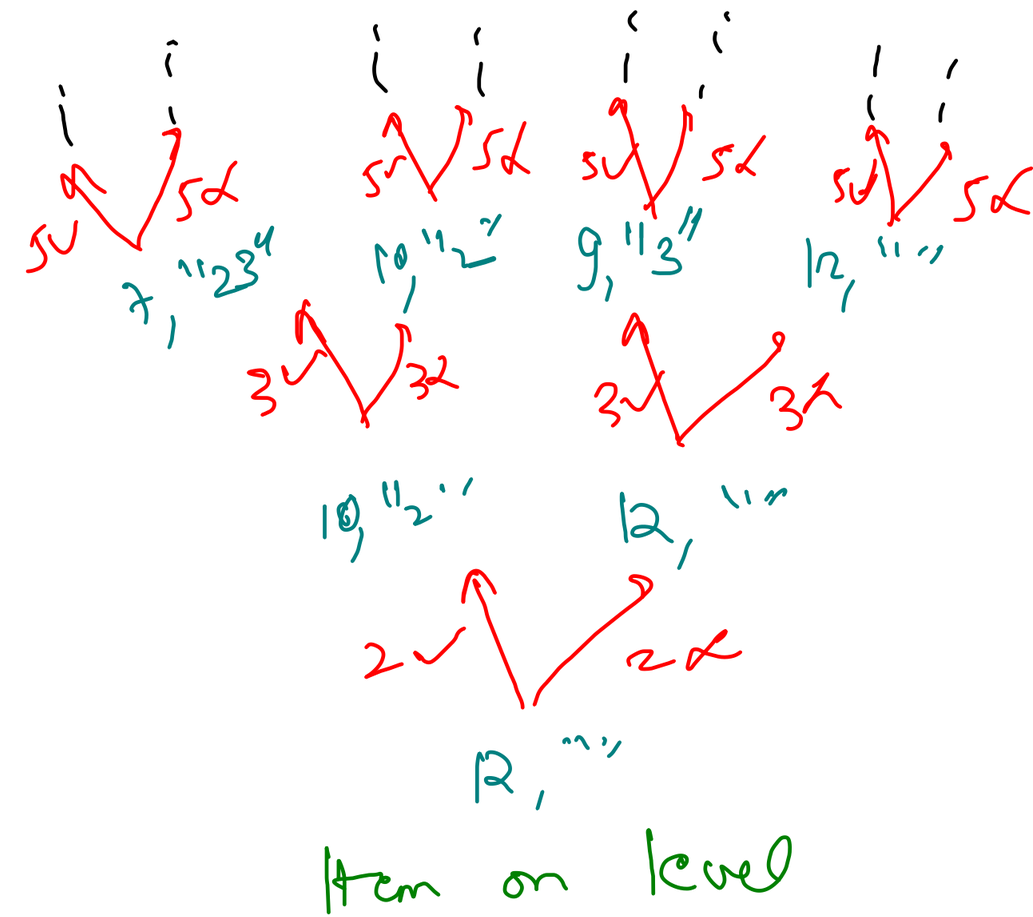
```java
public static void permutation(String res, String ustr, HashMap<Character, Integer> unique, int k){
    if(res.length() == k){
        System.out.println(res);
        return;
    }

    for(Character ch: ustr.toCharArray()){
        int oldFreq = unique.get(ch);
        if(oldFreq > 0){
            unique.put(ch, oldFreq - 1);
            permutation(res + ch, ustr, unique, k);
            unique.put(ch, oldFreq);
        }
    }
}
```

Coin Change Combination (II) {2, 3, 5, 6, 7}

Finite Supply

12



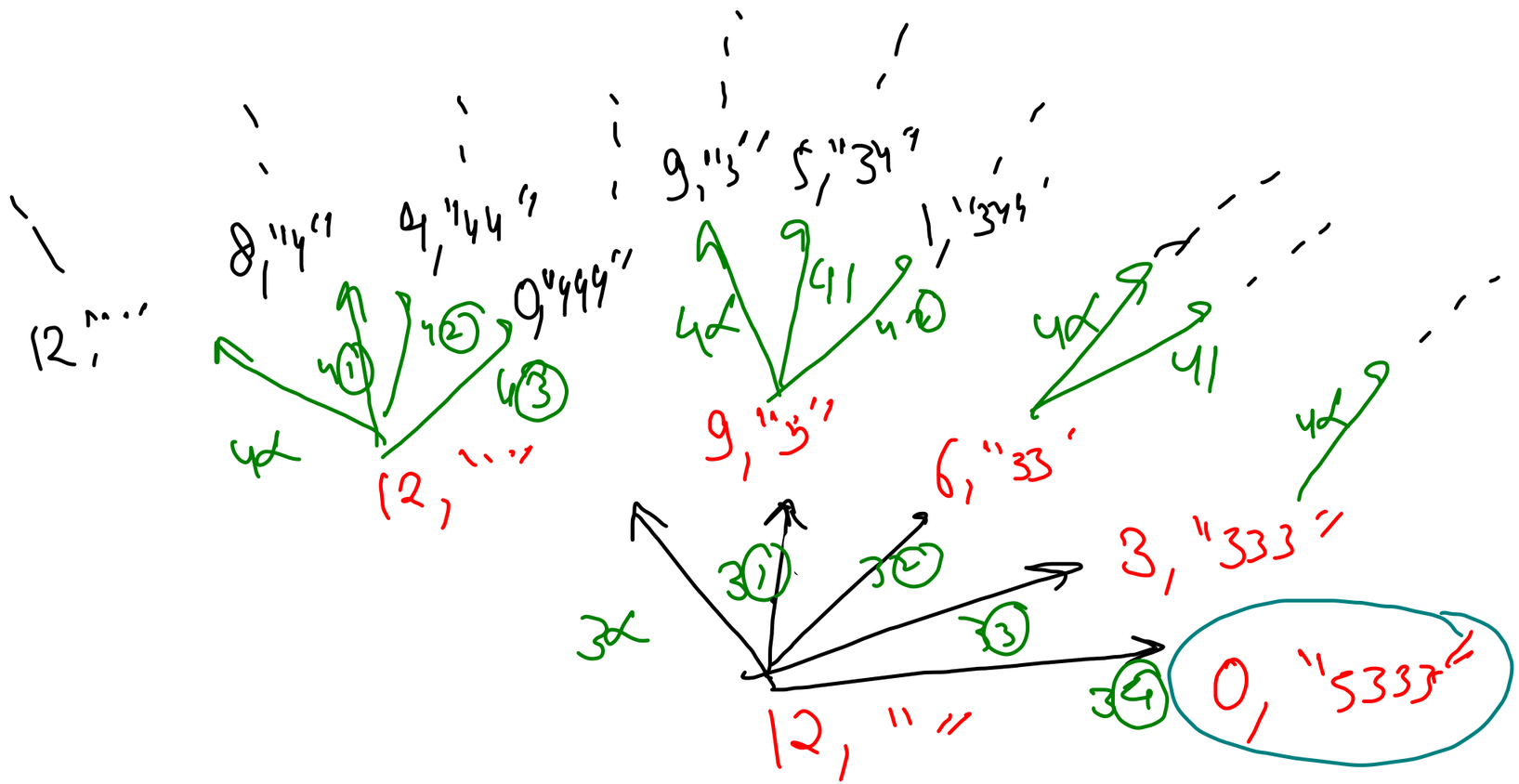


Finite Supply

```java
public void combinations(List<Integer> ans, HashMap<Integer, Integer> freq,
                         ArrayList<Integer> uniqueCoins, int currentCoin, int target){
    if(target == 0){
        res.add(new ArrayList<>(ans));
        return;
    }
    if(currentCoin == uniqueCoins.size()){
        return;
    }

    int val = uniqueCoins.get(currentCoin);
    for(int i=0; i<=freq.get(val); i++) {
        if(target - i * val >= 0){
            for(int j=1; j<=i; j++){
                ans.add(val);
            }

            combinations(ans, freq, uniqueCoins, currentCoin + 1, target - i * val);

            for(int j=1; j<=i; j++){
                ans.remove(ans.size() - 1);
            }
        }
    }
}
```

Infinite Supply

$\{3, 4, 5\}$ 12



Infinite supply

```java
public void combinations(List<Integer> ans, int[] uniqueCoins, int currentCoin, int target){
    if(target == 0){
        res.add(new ArrayList<>(ans));
        return;
    }
    if(currentCoin == uniqueCoins.length){
        return;
    }

    int val = uniqueCoins[currentCoin];

    for(int i=0; target - i * val >= 0; i++) {
        for(int j=1; j<=i; j++){
            ans.add(val);
        }

        combinations(ans, uniqueCoins, currentCoin + 1, target - i * val);

        for(int j=1; j<=i; j++){
            ans.remove(ans.size() - 1);
        }
    }
```

Today's Questions

1. Coin change - Permutat^n → I
   Coin change - Permutat^n → II

2. Queens Permutations - 2d As 2d - Queen Chooses
   Queens Permutations - 2d As 2d - Box Chooses

3. Queens Combinations - 2d As 2d - Queen Chooses
   Queens Combinations - 2d As 2d - Box Chooses
   Queens Combinations - 2d As 1d - Queen Chooses

4. Nqueens Combinations - 2d As 1d - Queen Chooses
   Nqueens Permutations - 2d As 1d - Queen Chooses

5. Nknights Combinations - 2d As 1d - Knight Chooses

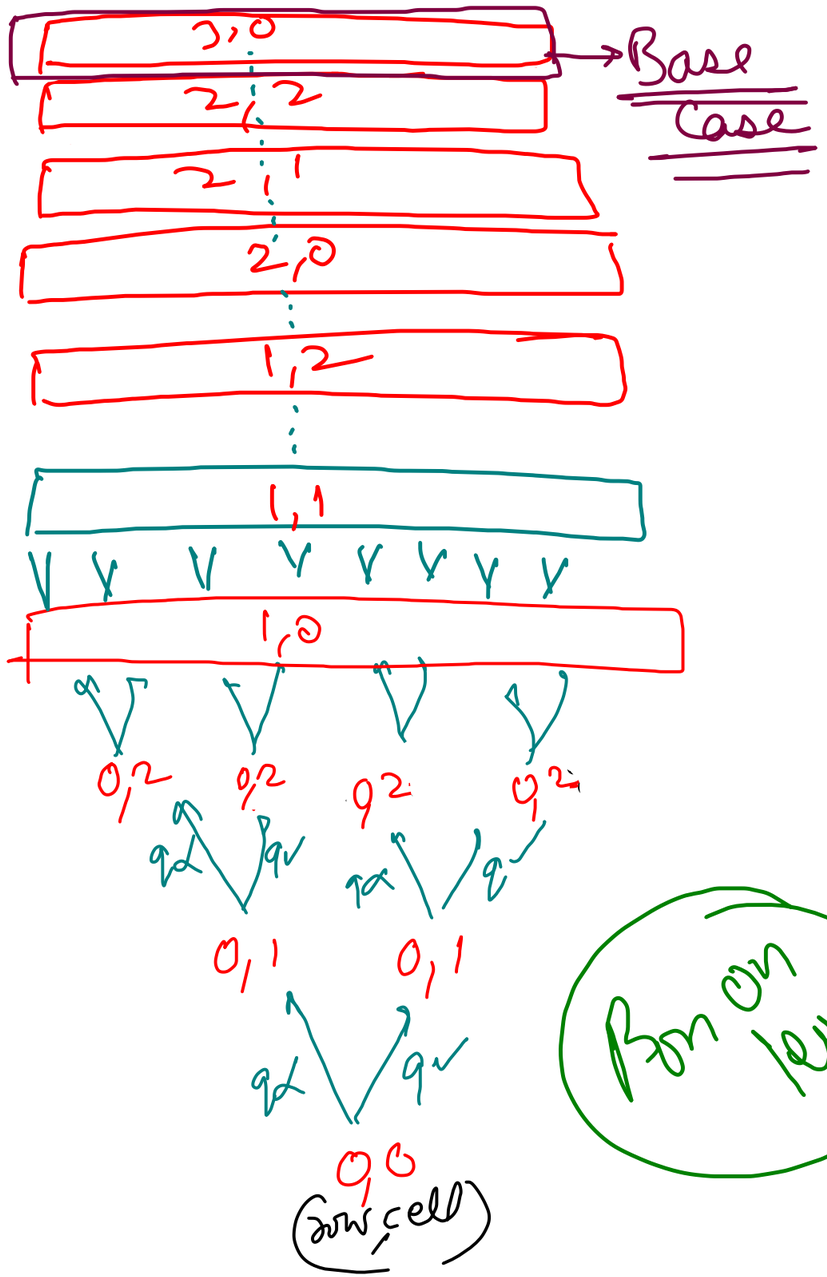
1. Recursion Tree {RT}
2. RT → level → parameter
   RT → option → for
3. Backtracking
4. Rcs → order
   Rcs → +ve & -ve base case
5. Dry run

Dictionary Order Small "cab"

```
void permute(string &input, vector<bool> &vis, string output, string &s){
    if(output.size() == s.size()){
        if(output < s){
            cout << output << endl;
        }
        return;
    }
    for(int i=0; i<input.size(); i++){
        if(vis[i] == false){
            vis[i] = true;
            permute(input, vis, output + input[i], s);
            vis[i] = false;
        }
    }
}
```

2D Combination - 1

$\{q, q, q\}$ $k = 3$



Row on level

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

3,0

$k = n$

$k = n$ ⇓ queens

n^2 ⇓ boxes

Total combinat^n $= 2^{N*N} = 2^9$

$2^9 = {}^9C_0 + {}^9C_1 + {}^9C_2 + \boxed{{}^9C_3} + \cdots {}^9C_9$ → Result

```java
// qpsf -> Queens placed so far, tq -> total Queens
public static void queensCombinations(int qpsf, int tq, int row, int col, String mat){
    if(row == tq){
        if(qpsf == tq){
            System.out.println(mat);
        }
        return;
    }

    if(col == tq - 1){
        // Apni Row ka last element, agli row ke pehle element par jana hai
        queensCombinations(qpsf + 1, tq, row + 1, 0, mat + 'q' + '\n'); // yes
        queensCombinations(qpsf, tq, row + 1, 0, mat + '-' + '\n'); // no
    } else {
        queensCombinations(qpsf + 1, tq, row, col + 1, mat + 'q'); // yes
        queensCombinations(qpsf, tq, row, col + 1, mat + '-'); // no
    }
}
```

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	6	7	8	9	10	11
2	12	13	14	15	16	17
3	18	19	20	21	22	23

4 rows :- N = 4

6 columns :- M = 6

(1) cell $\xrightarrow{\text{row no}}$ cellNo / M

cell $\xrightarrow{\text{col no}}$ cellNo % M

(2) row, col $\xrightarrow{\text{cell no}}$ rowNo * M + colNo

2 * 6 + 4 = 16

3 * 6 + 1 = 19

16 / 2 = 2

16 % 2 = 4

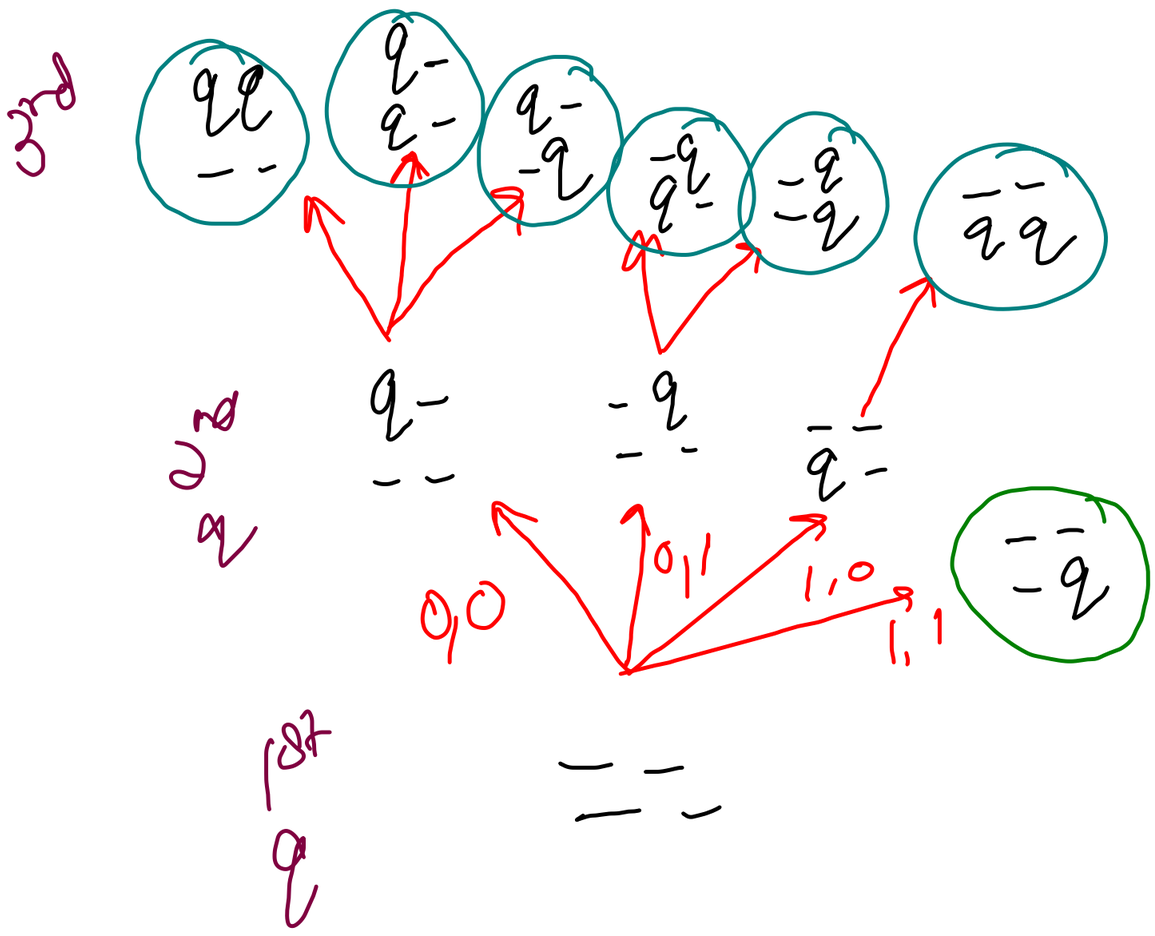
16 = 6 * 2 + 4

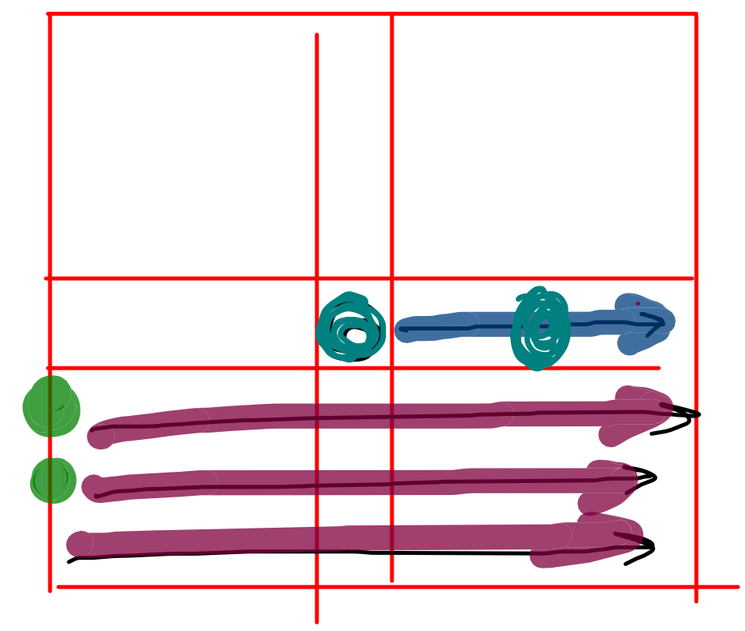
```java
// qpsf -> Queens placed so far, tq -> total Queens
public static void queensCombinations(int qpsf, int tq, int cellNo, String mat){
    if(cellNo == tq * tq){
        if(qpsf == tq){
            System.out.println(mat);
        }
        return;
    }

    if(cellNo % tq == tq - 1){
        // Apni Row ka last element, agli row ke pehle element par jana hai
        queensCombinations(qpsf + 1, tq, cellNo + 1, mat + 'q' + '\n'); // yes
        queensCombinations(qpsf, tq, cellNo + 1, mat + '-' + '\n'); // no
    } else {
        queensCombinations(qpsf + 1, tq, cellNo + 1, mat + 'q'); // yes
        queensCombinations(qpsf, tq, cellNo + 1, mat + '-'); // no
    }
}
```

2D Combinatn : Queen on level

$k = 2$

$n = 2 \Rightarrow$ boxes $= n^2 = 4$



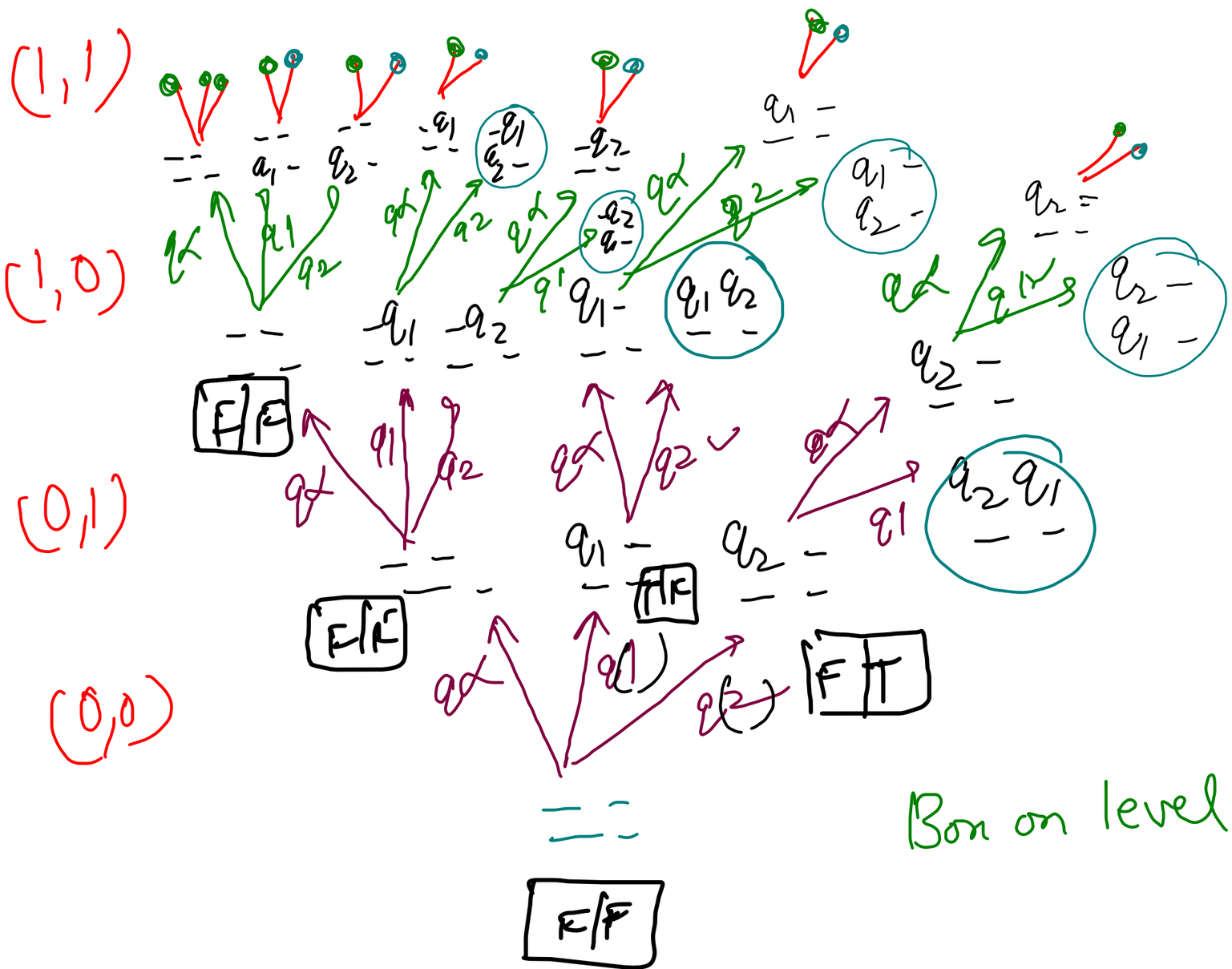


```java
public static void queensCombinations(int qpsf, int tq, Character[][] chess, int lastQueenRow, int lastQueenCol){
    if(qpsf == tq){
        for(int i=0; i<tq; i++) {}
        System.out.println();
        return;
    }

    for(int j=lastQueenCol + 1; j < tq; j++){
        if(chess[lastQueenRow][j] == '-'){
            chess[lastQueenRow][j] = 'q';
            queensCombinations(qpsf + 1, tq, chess, lastQueenRow, j);
            chess[lastQueenRow][j] = '-';
        }
    }

    for(int i=lastQueenRow + 1; i < tq; i++){
        for(int j=0; j<tq; j++){
            if(chess[i][j] == '-'){
                chess[i][j] = 'q';
                queensCombinations(qpsf + 1, tq, chess, i, j);
                chess[i][j] = '-';
            }
        }
    }
}
```

2D queen – Permutation

$\{q_1, q_2\}$ $k = 2$

$n = 2 \Rightarrow n^2 = 4$



Box on level

```
if(cellNo % tq == tq - 1){
    // Apni Row ka last element, agli row ke pehle element par jana hai
    for(int i=0; i<tq; i++){
        if(vis[i] == false){
            vis[i] = true;
            queensPermutations(qpsf + 1, tq, cellNo + 1, mat + "q" + (i + 1) + "\t\n", vis); // yes
            vis[i] = false;
        }
    }
    queensPermutations(qpsf, tq, cellNo + 1, mat + "-\t\n", vis); // no
} else {
    for(int i=0; i<tq; i++){
        if(vis[i] == false){
            vis[i] = true;
            queensPermutations(qpsf + 1, tq, cellNo + 1, mat + "q" + (i + 1) + "\t" , vis); // yes
            vis[i] = false;
        }
    }
    queensPermutations(qpsf, tq, cellNo + 1, mat + "-\t", vis); // no
}
```

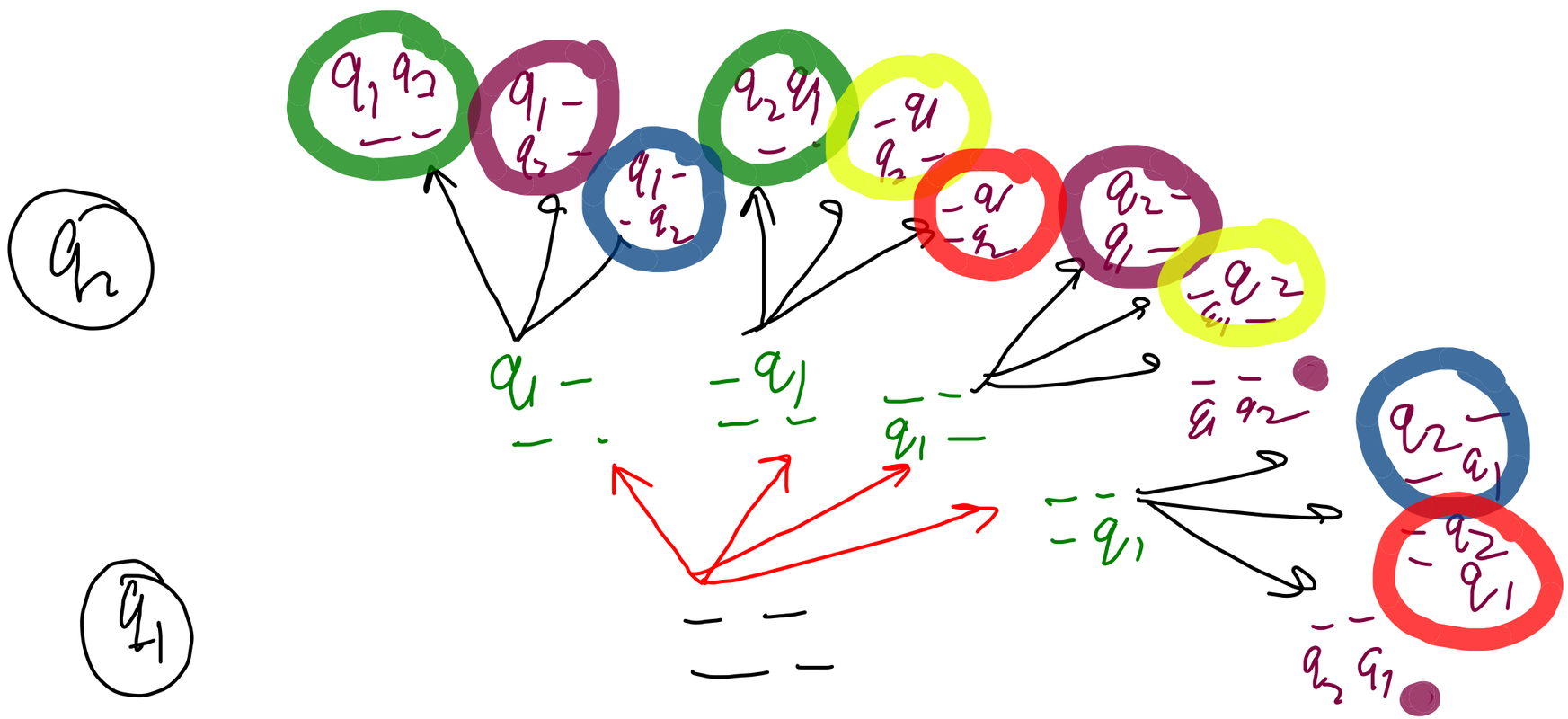
2D queen – Permutation

$\{q_1, q_2\}$ $k = 2$

$n = 2 \Rightarrow n^2 = 4$

$^4P_2 = {}^4C_2 * 2!$

Queen chosen

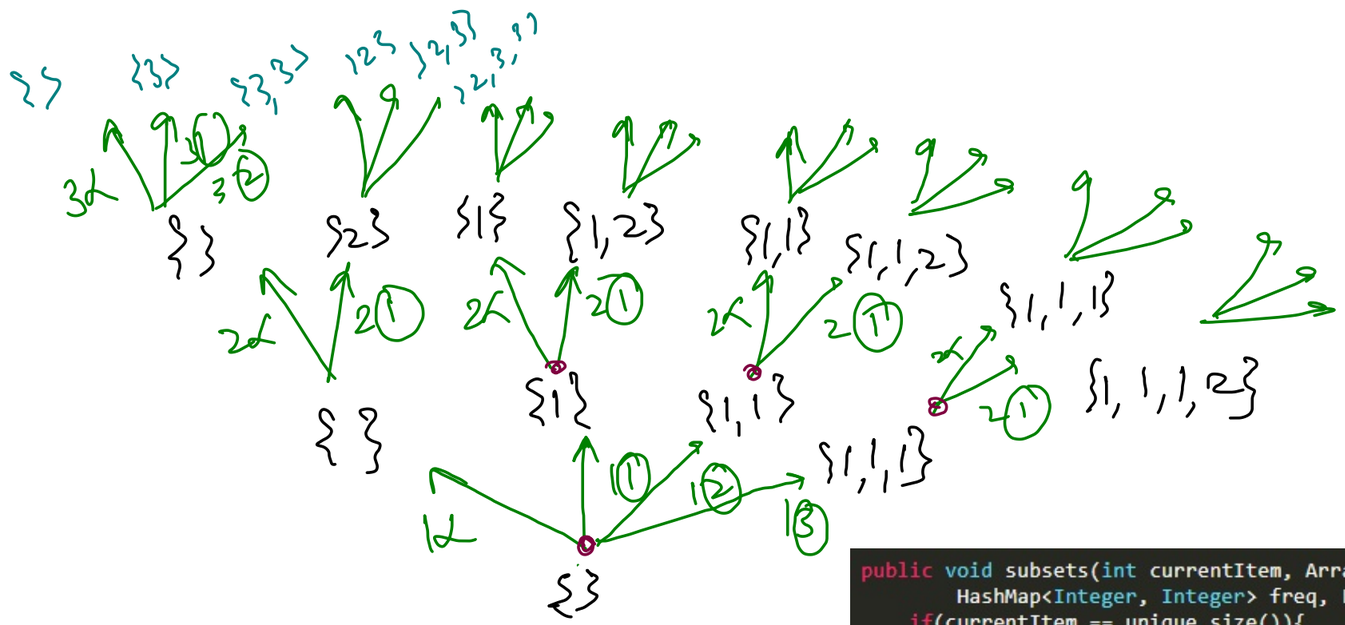


```java
    public static void queensCombinations(int qpsf, int tq,
Character[][] chess, int lastcellNo){
        if(qpsf == tq){
            for(int i=0; i<tq; i++) {}
            System.out.println();
            return;
        }

        for(int c=lastcellNo + 1; c < tq * tq; c++){
            int rowNo = c / tq;
            int colNo = c % tq;

            chess[rowNo][colNo] = 'q';
            queensCombinations(qpsf + 1, tq, chess, c);
            chess[rowNo][colNo] = '-';
        }
    }
```

Unique subsets {subsets-II}

{1, 1, 1, 2, 3, 3}

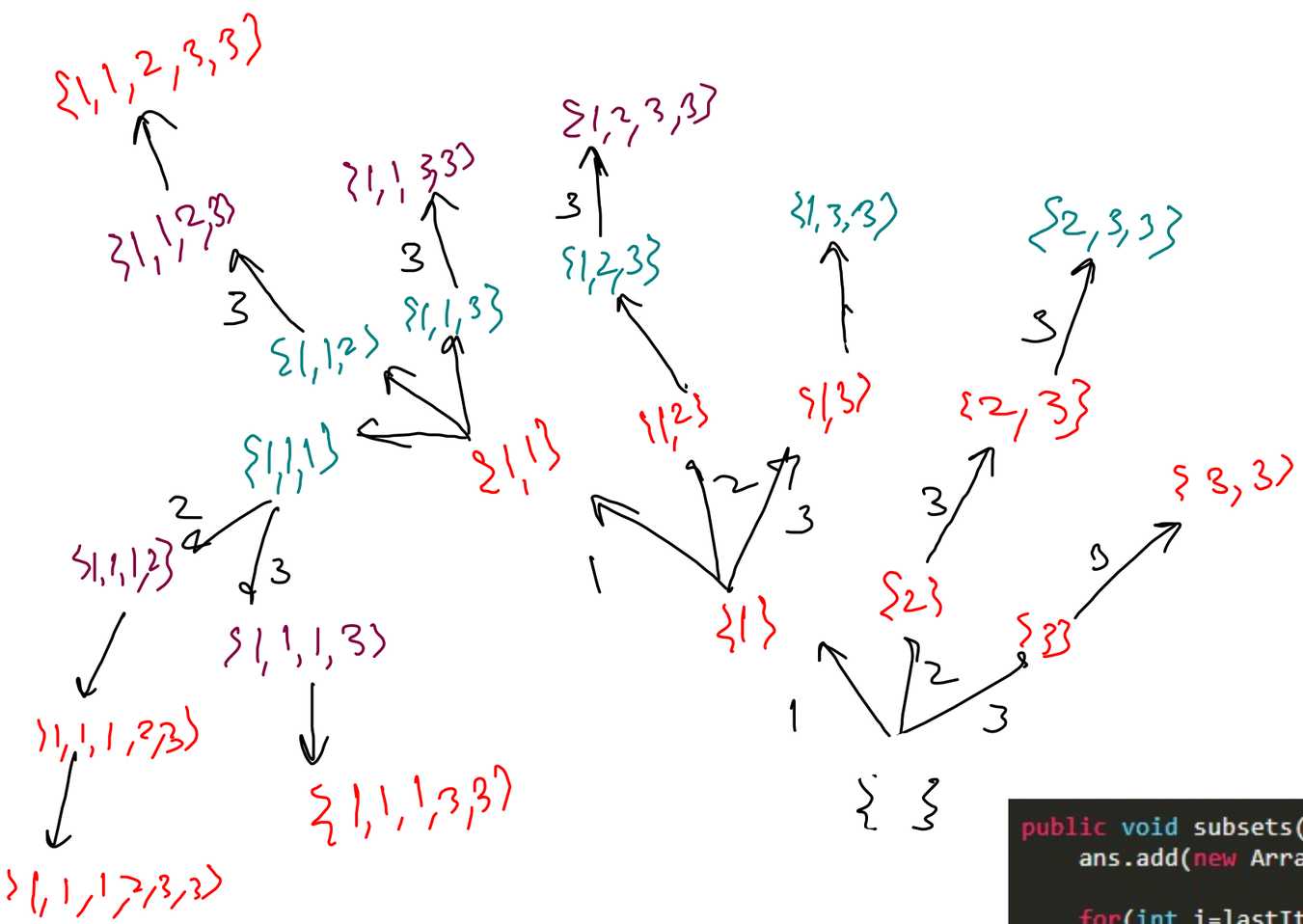
Hashmap of
⇓
Array



Item on level

1 → 3
2 → 1
3 → 2

```
public void subsets(int currentItem, ArrayList<Integer> unique,
        HashMap<Integer, Integer> freq, List<Integer> res){
    if(currentItem == unique.size()){
        ans.add(new ArrayList<>(res));
        return;
    }

    int val = unique.get(currentItem);
    subsets(currentItem + 1, unique, freq, res); // no

    for(int f=0; f<freq.get(val); f++){
        res.add(val);
        subsets(currentItem + 1, unique, freq, res);
    }

    for(int f=0; f<freq.get(val); f++){
        res.remove(res.size() - 1);
    }
}
```

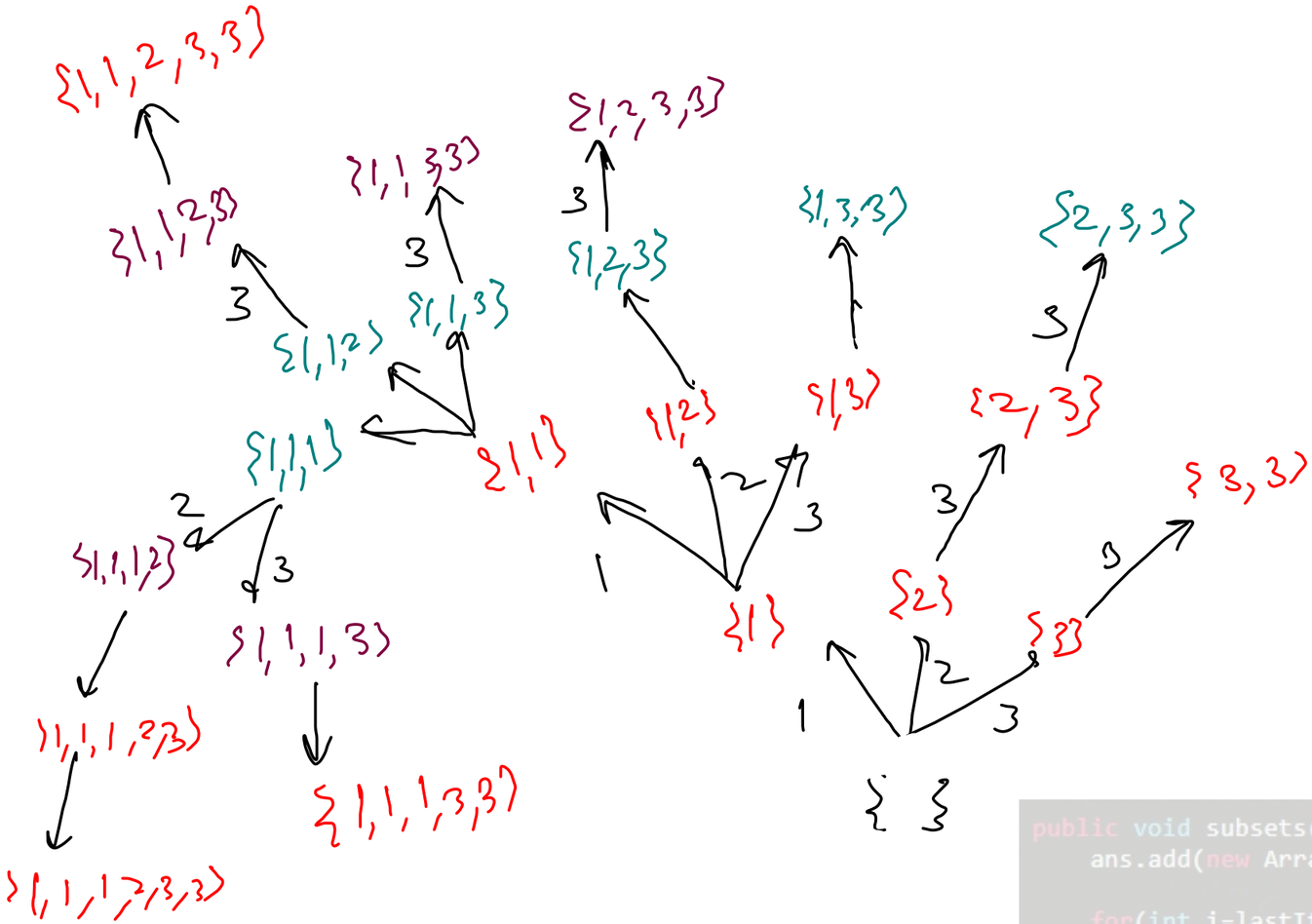


{1, 1, 1, 2, 3, 3}

$\frac{7!}{3!\,1!\,3!}$

Box on level

```
public void subsets(int lastItem, ArrayList<Integer> unique, HashMap<Integer, Integer> fre
    ans.add(new ArrayList<>(res));

    for(int i=lastItem; i<unique.size(); i++){
        int val = unique.get(i);
        int oldFreq = freq.get(val);

        if(oldFreq > 0){
            freq.put(val, oldFreq - 1);
            res.add(val);

            subsets(i, unique, freq, res);

            res.remove(res.size() - 1);
            freq.put(val, oldFreq);
        }
    }
}
```



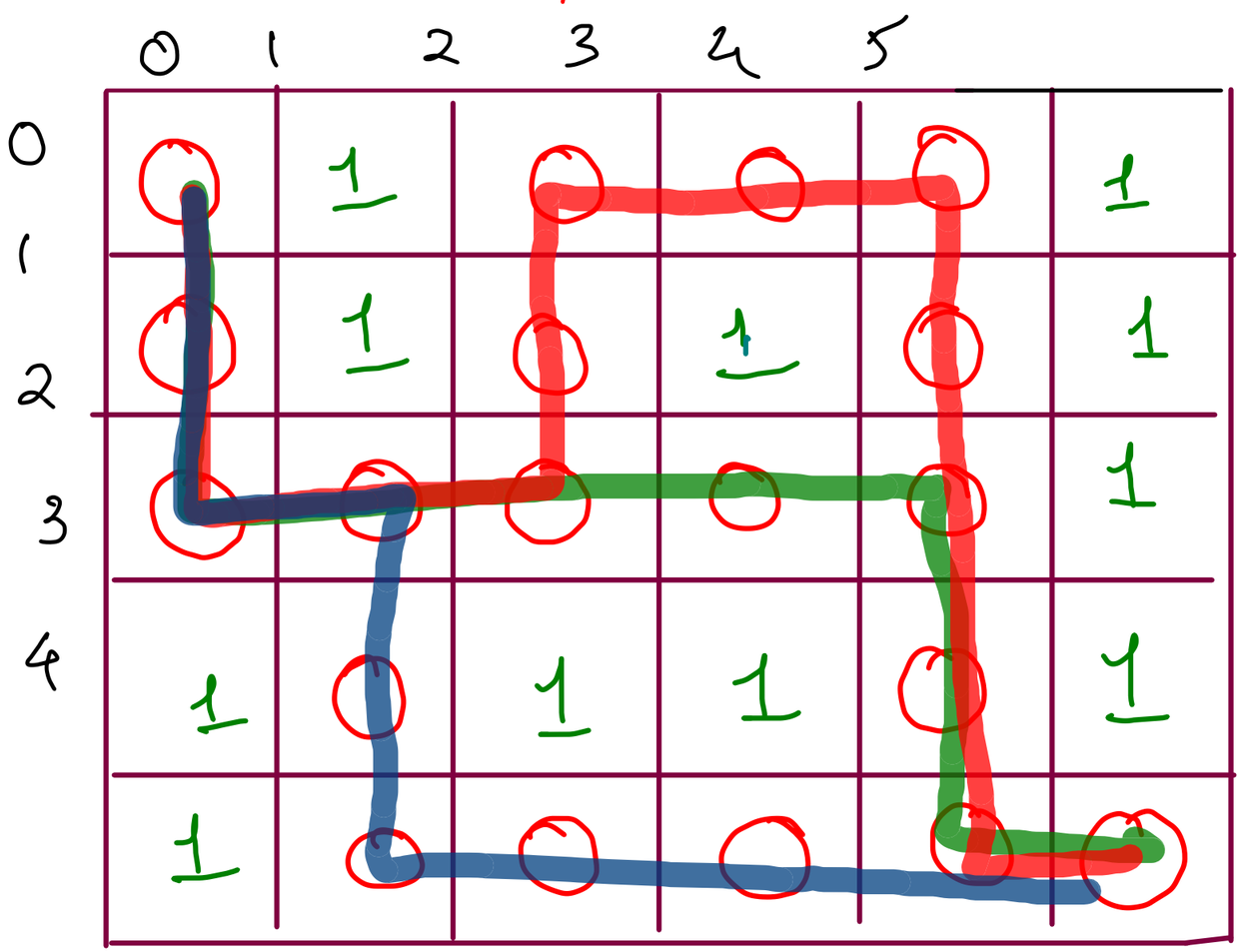
{1, 1, 1, 2, 3, 3}

$$\frac{7!}{3!\,1!\,3!}$$

Box on level

```
public void subsets(int lastItem, ArrayList<Integer> unique, HashMap<Integer, Integer> fre
    ans.add(new ArrayList<>(res));

    for(int i=lastItem; i<unique.size(); i++){
        int val = unique.get(i);
        int oldFreq = freq.get(val);

        if(oldFreq > 0){
            freq.put(val, oldFreq - 1);
            res.add(val);

            subsets(i, unique, freq, res);

            res.remove(res.size() - 1);
            freq.put(val, oldFreq);
        }
    }
}
```

Rat in Maze / Flood Fill



① Preorder {Node-Pre}

vis[r][c] = true

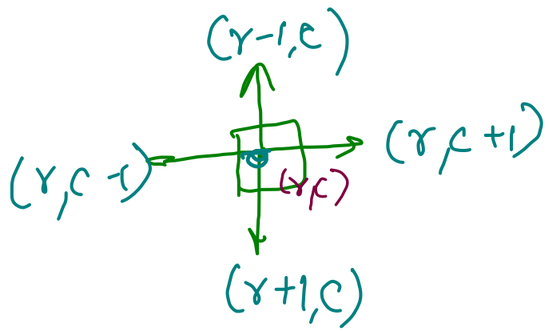
② Calls & TLDR}

③ Postorder {Node-Post}

vis[r][c] = false

4 way adjacent

{TRDL}



$x :\to \{-1, 0, +1, 0\}$

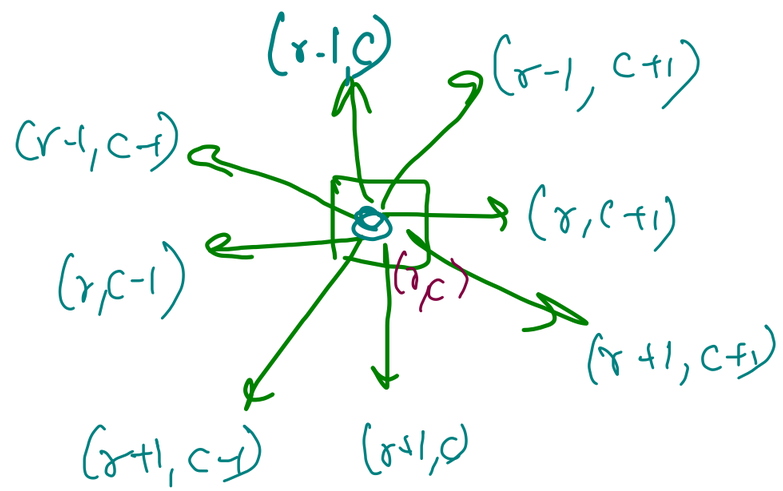
$y :\to \{0, +1, 0, -1\}$

```
for(i=0; i<4; i++)
{
    ff(r+x[i], c+y[i])
}
```

8 way adjacent

{ TLDR ; TL, TR, DL, DR }

edge adjacent corner adjacent



$x : \{-1, -1, 0, 1, 1, 1, 0, -1\}$

$y : \{0, 1, 1, 1, 0, -1, -1, -1\}$

```
for (i=0; i<8; i++)
{
    ff(r+x[i], c+y[i])
}
```

Path with Maximum Gold {Gold Mine}

```
[[0,6,0],
 [5,8,7],
 [0,9,0]]
```

All paths

587
586
589

685
687
689

786
785
789

985
986
987

85
86
87
89

(1) you can revisit cell ⟹ visited matrix

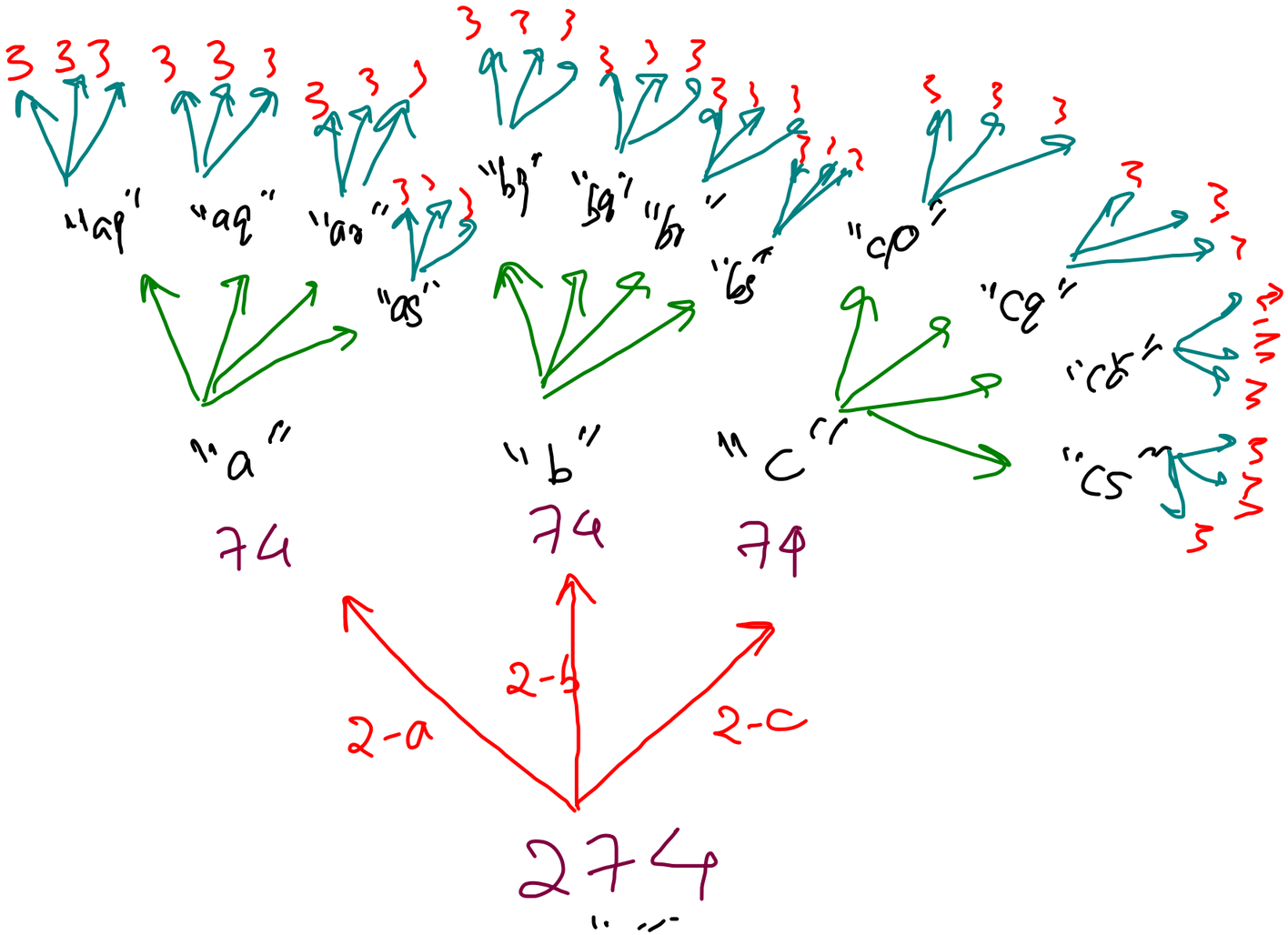
(2) Any node can be source node ⟹ All nodes can act as a source node.

(3) Treat 0 as a blockage.

```java
public int DFS(int r, int c, int[][] grid){
    if(r < 0 || c < 0 || r >= grid.length || c >= grid[0].length
      || grid[r][c] == 0 || grid[r][c] == -1){
        return 0;
    }

    int val = grid[r][c];
    grid[r][c] = -1; // visited mark

    int maxGold = 0;
    for(int i=0; i<4; i++){
        maxGold = Math.max(maxGold, DFS(r + x[i], c + y[i], grid));
    }

    grid[r][c] = val;
    return maxGold + grid[r][c];
}

public int getMaximumGold(int[][] grid) {

    int maxGold = 0;
    // DFS Starting from every node as source node
    for(int i=0; i<grid.length; i++){
        for(int j=0; j<grid[0].length; j++){
            int currentAns = DFS(i, j, grid);
            maxGold = Math.max(maxGold, currentAns);
        }
    }

    return maxGold;
}
```

```
[[0,6,0],
 [5,8,7],
 [0,9,0]]
```

Keypad



3

4

74

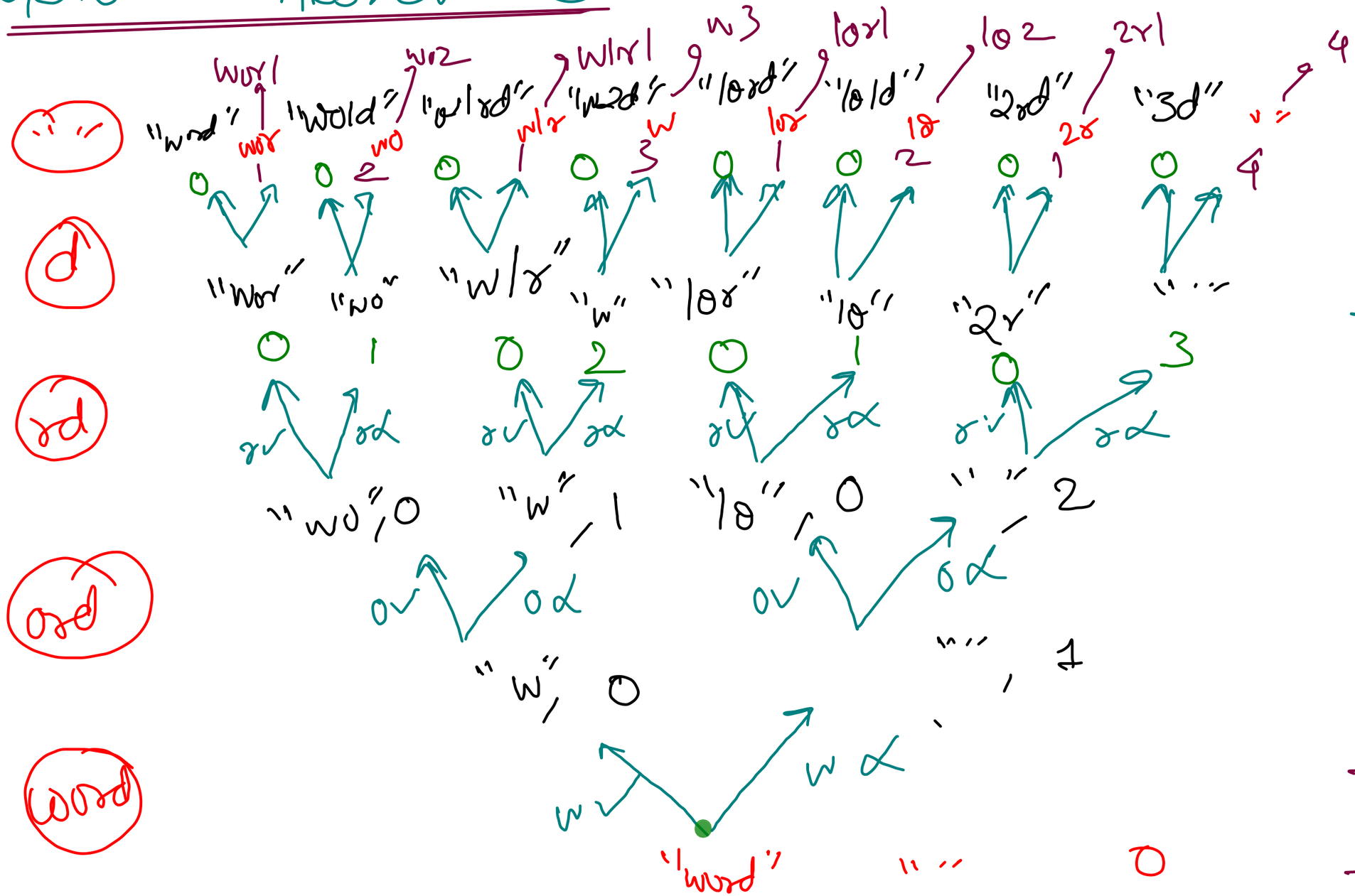
274

Base case

3 * 4 * 3

```java
List<String> res;

String[] keypad = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
public void letterCombinations(int idx, String input, String output){
    if(idx == input.length()){
        if(output.length() > 0){
            res.add(output);
        }
        return;
    }

    char digit = input.charAt(idx);
    for(char letter: keypad[digit - '0'].toCharArray()){
        letterCombinations(idx + 1, input, output + letter);
    }
}

public List<String> letterCombinations(String digits) {
    res = new ArrayList<>();
    letterCombinations(0, digits, "");
    return res;
}
```

Generate Abbreviations



Base case
{count > 0}
↓
append in result

① Yes

→ append count if > 0 in output

→ append characters

→ call count = 0

② No

→ pass output as it is

→ count + 1

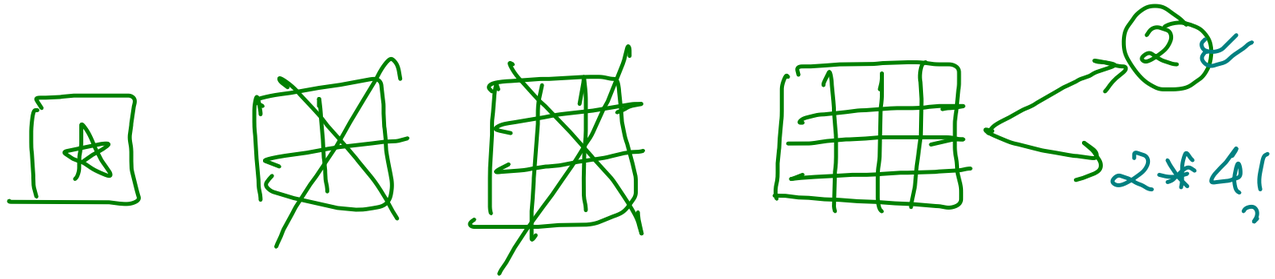
Code

```java
class Solution {
    List<String> res;
    public void generateAbb(String input, int idx, String output, int count){
        if(idx == input.length()){
            if(count > 0) res.add(output + count);
            else res.add(output);
            return;
        }

        char ch = input.charAt(idx);
        generateAbb(input, idx + 1, output + ((count == 0) ? "" : count) + ch, 0); // yes
        generateAbb(input, idx + 1, output, count + 1);  // no
    }

    public List<String> generateAbbreviations(String word) {
        res = new ArrayList<>();
        generateAbb(word, 0, "", 0);
        return res;
    }
}
```

Recursion & Backtracking

Today's Class {lecture - 7}

N Queen - Combinations	
N Queen - Permutations	
N-Queens - I (IB)	
N-Queens - II	
Using Backtracking	Code
Using Branch & Bound	
Using Bit Manipulation	
N Knight	
Knight's Tour	GfG Article
Knight's Tour	Code



	☆	q_1	
q_2			☆
☆			q_3
	q_4	☆	

$N^2 = 4*4$

N = 4 queens

Combination

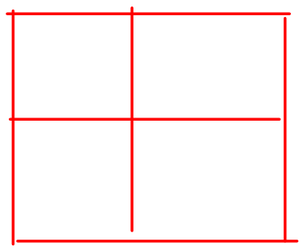
$\{q_1, q_1, q_1, q_1\}$

constraint → No queen pair should kill each other.

2D P&C

- Permutation
  - Box onlevel
  - Item onlevel
- Combination
  - Box
  - Item

① N Queen - Combination

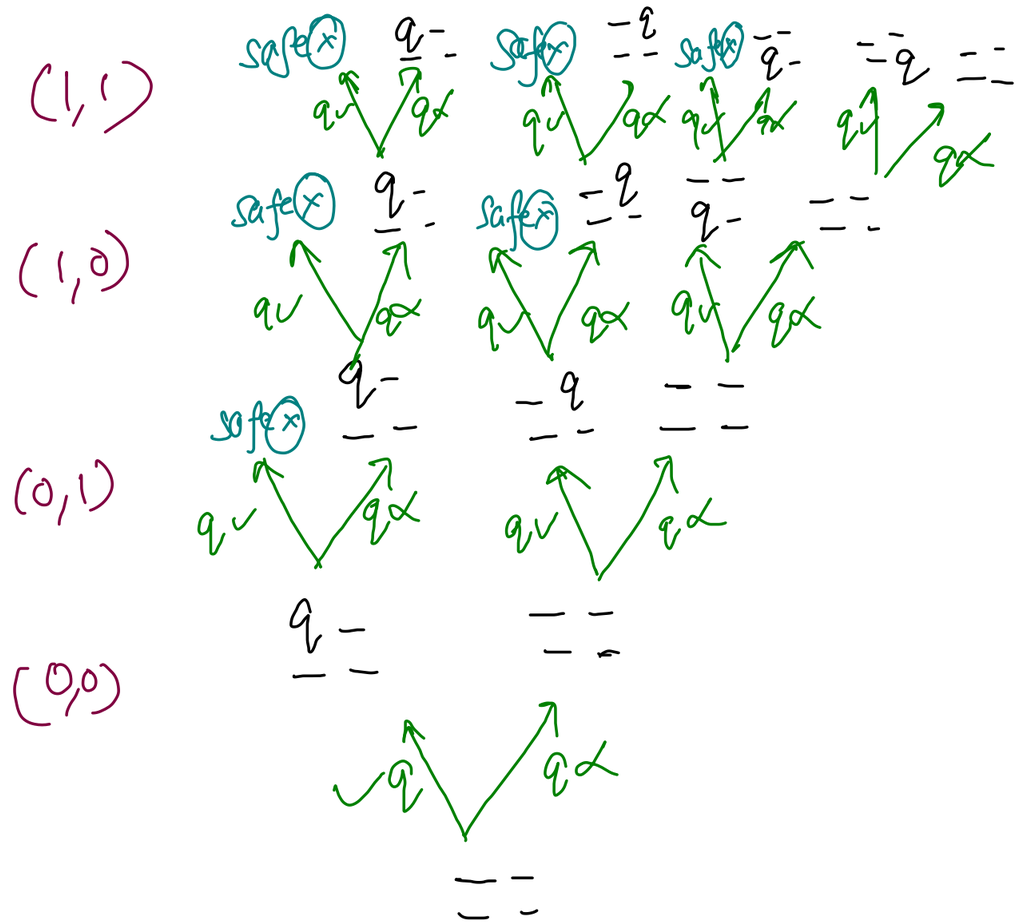


{q, q}

2^{N^2}

N = 2 ?

0 valid configurations



N = 9

$2^9 + 9^3$

$$(\text{calls})^{\text{height}} + (\text{pre} + \text{post}) * \text{height}$$

$$(2^n) + n * (n^2) = O(2^n + n^3)$$

IsQueenSafe

	c0	c1	c2	c3
r0	d_D	d_E	d_g	d_h
r1	d_C	d_D	d_f	d_g
r2	d_B	d_C	d_D	d_f
r3	d_A	d_B	d_C	d_D

① Left Diagonal

	c0	c1	c2	c3
r0	d_I	d_{II}	d_{III}	d_{IV}
r1	d_{II}	d_{III}	d_{IV}	d_V
r2	d_{III}	d_{IV}	d_V	d_{VI}
r3	d_{IV}	d_V	d_{VI}	d_{VII}

② Right Diagonal

Total left or right diagonals $= \boxed{2 * n - 1}$

	c0	c1	c2	c3
r0	r0	r0	r0	r0
r1	r1	r1	r1	r1
r2	r2	r2	r2	r2
r3	r3	r3	r3	r3

③ Row

	c0	c1	c2	c3
r0	c0	c1	c2	c3
r1	c0	c1	c2	c3
r2	c0	c1	c2	c3
r3	c0	c1	c2	c3

④ column

Total rows or columns $= \boxed{n}$

O(N) time, O(1) space

Recursion space: → O(n2)

Output: O(n2)

```java
public static boolean isQueenSafe(int r, int c, boolean[][] vis){
    // row (left)
    for(int j=0; j<c; j++){
        if(vis[r][j] == true)
            return false;
    }
    // col (up)
    for(int i=0; i<r; i++){
        if(vis[i][c] == true)
            return false;
    }

    // left diagnol (up)
    int i = r, j = c;
    while(i >= 0 && j >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j--;
    }

    // right diagnol (down)
    i = r; j = c;
    while(j < vis.length && i >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j++;
    }

    return true;
}
```

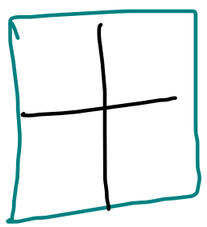
```java
public static void nqueen(int r, int c, int qpsf, boolean[][] vis) {
    if (qpsf == vis.length) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < vis.length; i++) {⟺}
        res.add(ans);
        return;
    }

    if(r == vis.length){
        return;
    }

    if (isQueenSafe(r, c, vis)) {
        vis[r][c] = true;

        if (c == vis.length - 1) {
            nqueen(r + 1, 0, qpsf + 1, vis);
        } else {
            nqueen(r, c + 1, qpsf + 1, vis);
        }

        vis[r][c] = false;
    }

    if (c == vis.length - 1) {
        nqueen(r + 1, 0, qpsf, vis);
    } else {
        nqueen(r, c + 1, qpsf, vis);
    }
}
```

N-Queen Permutation

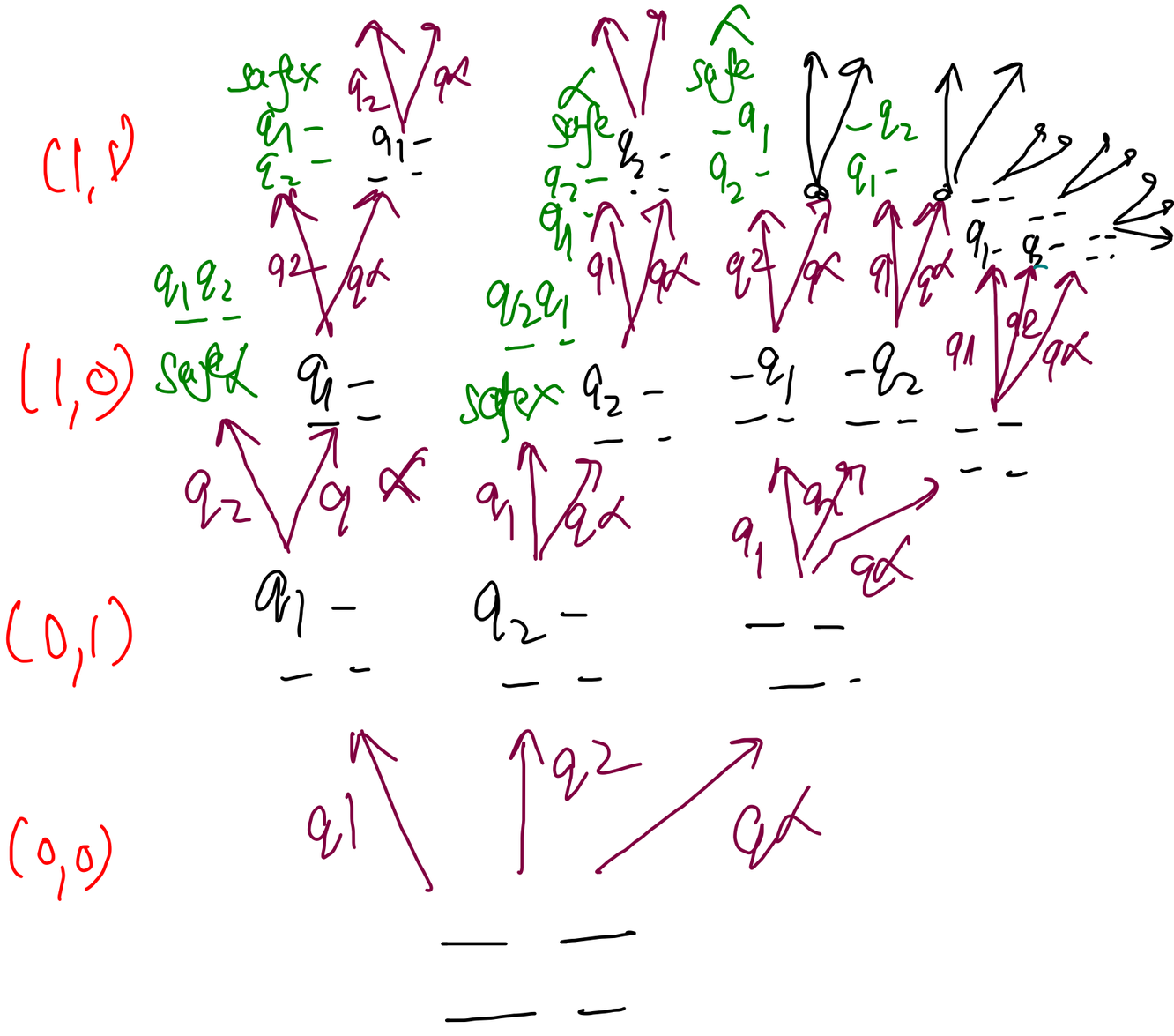
$\{q_1, q_2\}$

Bon on level

$2 * 2$ $N = 2$



{0 valid configurations}



$N = 4 \Longrightarrow$ Permutations $= 2 * 4!$

visited queens array $24 = 48$

Run on level approach { Order Different }

```java
public static void nqueens(int qpsf, int r, int c, boolean[] queenPlaced, int[][] chess) {
    int n = queenPlaced.length;
    if(qpsf == n){
        for(int i=0; i<n; i++){...}
        System.out.println();
        return;
    }

    if(r == n){ return; }

    if(c == n-1){
        nqueens(qpsf, r+1, 0, queenPlaced, chess);
    } else {
        nqueens(qpsf, r, c+1, queenPlaced, chess);
    }

    for(int i=0; i<n; i++){
        if(queenPlaced[i] == false && isQueenSafe(r, c, chess)){
            queenPlaced[i] = true;
            chess[r][c] = (i + 1);

            if(c == n-1){
                nqueens(qpsf+1, r+1, 0, queenPlaced, chess);
            } else {
                nqueens(qpsf+1, r, c+1, queenPlaced, chess);
            }

            chess[r][c] = 0;
            queenPlaced[i] = false;
        }
    }
}
```

Mute Start Video Security Participants 38 Chat New Share Pause Share Annotate Remote Co
You are screen sharing Stop Share

IsQueenSafe

	c0	c1	c2	c3
r0	d_D	d_E	d_g	d_h
r1	d_C	d_D	d_f	d_g
r2	d_B	d_C	d_D	d_f
r3	d_A	d_B	d_C	d_D

① Left Diagonal

	c0	c1	c2	c3
r0	d_I	d_{II}	d_{III}	d_{IV}
r1	d_{II}	d_{III}	d_{IV}	d_V
r2	d_{III}	d_{IV}	d_V	d_{VI}
r3	d_{IV}	d_V	d_{VI}	d_{VII}

② Right Diagonal

Total left or right diagonals $= \boxed{2 * n - 1}$

	c0	c1	c2	c3
r0	r0	r0	r0	r0
r1	r1	r1	r1	r1
r2	r2	r2	r2	r2
r3	r3	r3	r3	r3

③ Row

	c0	c1	c2	c3
r0	c0	c1	c2	c3
r1	c0	c1	c2	c3
r2	c0	c1	c2	c3
r3	c0	c1	c2	c3

④ column

Total rows or columns $= \boxed{n}$

T/F	T/F	T/F	T/F

NQueen {Combinat} {Backtracking}

	☆	#	
#			☆
☆			#
	#	☆	

(rows)

Item on level

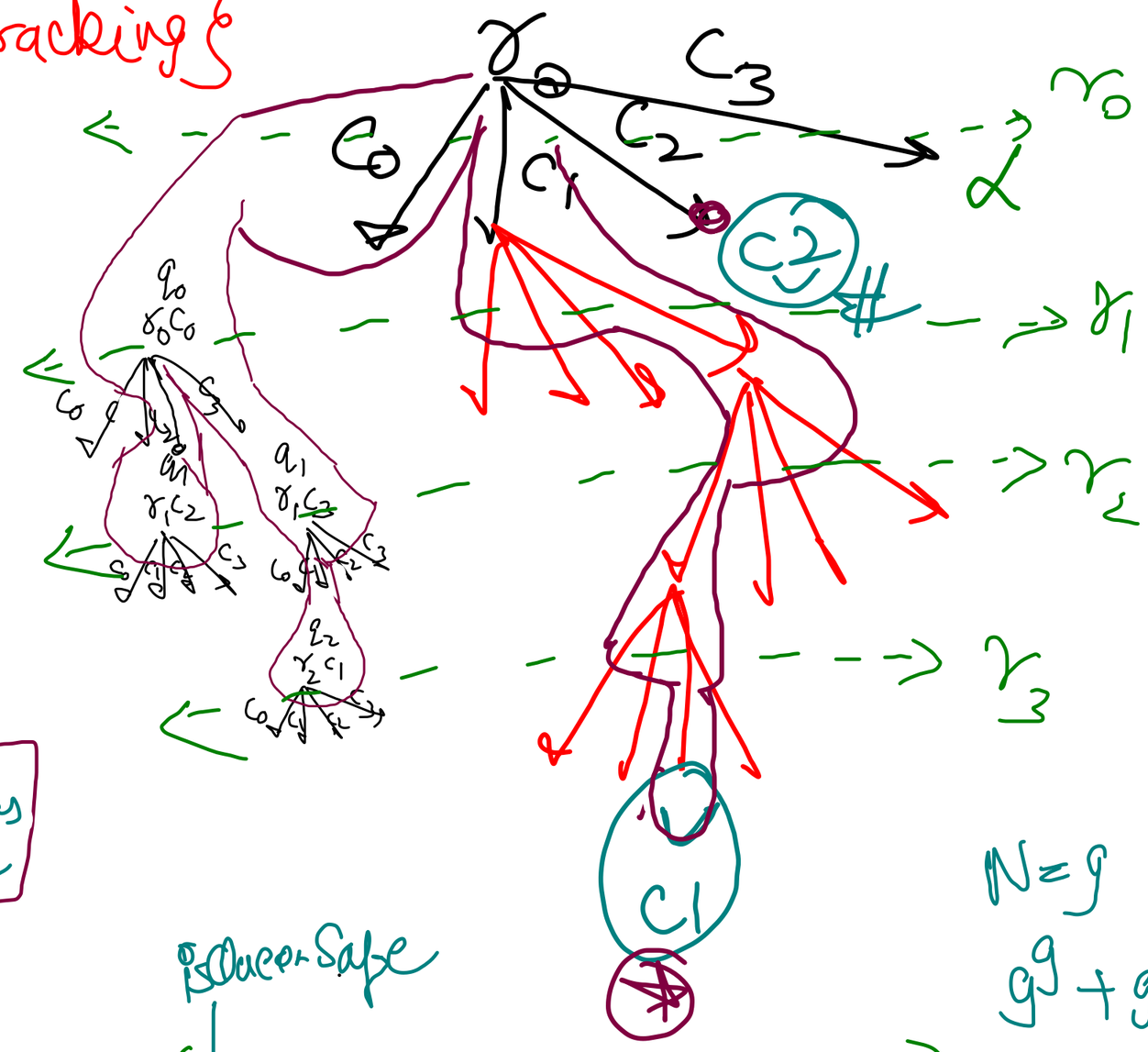
row in level

column as option

4 rows

4 columns

4 queens



N=9

$9^9 + 9^2$

isQueenSafe

$$(n)^n + (n) * n \Rightarrow O(n^n + n^2)$$

O(N) time, O(1) space

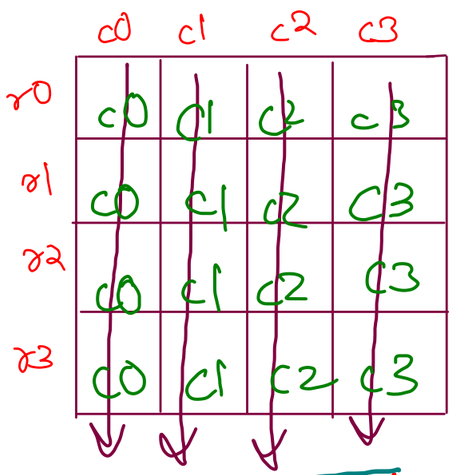
```java
public static boolean isQueenSafe(int r, int c, boolean[][] vis){
    // row (left)
    for(int j=0; j<c; j++){
        if(vis[r][j] == true)
            return false;
    }
    // col (up)
    for(int i=0; i<r; i++){
        if(vis[i][c] == true)
            return false;
    }

    // left diagnol (up)
    int i = r, j = c;
    while(i >= 0 && j >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j--;
    }

    // right diagnol (down)
    i = r; j = c;
    while(j < vis.length && i >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j++;
    }

    return true;
}
```

Recursion Call stack ? → O(n)

output : O(n²)

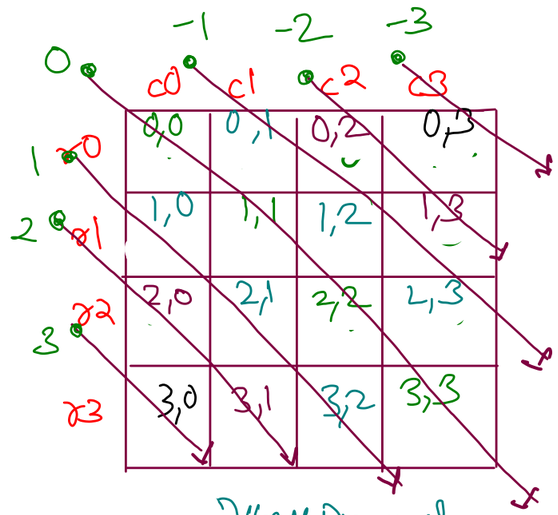
```java
public static void nqueen(int r, boolean[][] vis) {
    if (r == vis.length) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < vis.length; i++) {⟺}
        res.add(ans);
        return;
    }


    for(int c=0; c<vis.length; c++){
        if(isQueenSafe(r, c, vis)){
            vis[r][c] = true;
            nqueen(r + 1, vis);
            vis[r][c] = false;
        }
    }
}
```

③ Branch & Bound {optimize in isQueenSafe} in terms of time complexity



column {n}

1 byte



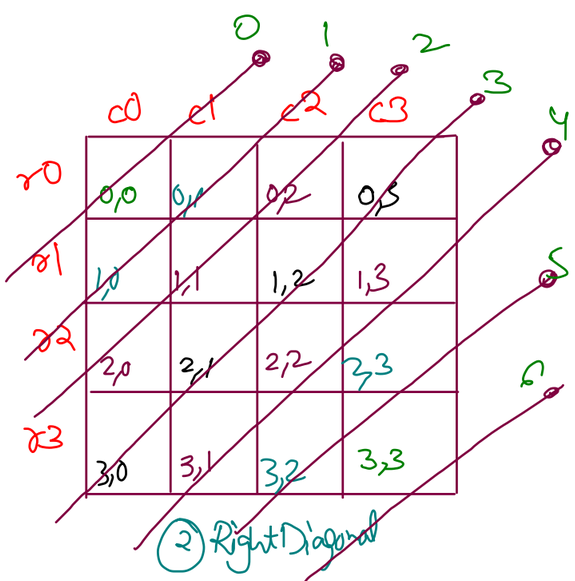
① Left Diagonal

{2n-1}

-3 -2 -1 0 1 2 3
+3 +3 +3 +3 +3 +3 +3
0 1 2 3 4 5 6

left diagnol for r,c

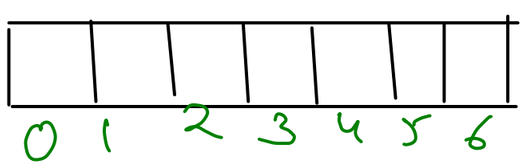
= ld[r-c+(n-1)]



② Right Diagonal

sum => i+j

rd[r+c] T F



{2n-1}

→ extra space

```java
public static boolean isQueenSafe(int r, int c, boolean[] ld, boolean[] rd, boolean[] col){
    int n = col.length;
    return ((ld[r - c + n - 1] == true) || (rd[r + c] == true) || (col[c] == true)) ? false : true;
}
```

⇒ O(1) time
O(n) space

```java
public static void nqueen(int r, boolean[][] chess, boolean[] ld, boolean[] rd, boolean[] col) {
    int n = chess.length;
    if (r == n) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            String curr = "";
            for (int j = 0; j < n; j++) {
                if (chess[i][j] == true) {
                    curr = curr + "Q";
                } else {
                    curr = curr + ".";
                }
            }
            ans.add(curr);
        }
        res.add(ans);
        return;
    }


    for(int c=0; c<n; c++){
        if(isQueenSafe(r, c, ld, rd, col)){
            chess[r][c] = ld[r - c + n - 1] = rd[r + c] = col[c] = true;
            nqueen(r + 1, chess, ld, rd, col);
            chess[r][c] = ld[r - c + n - 1] = rd[r + c] = col[c] = false;
        }
    }
}
```

O(n²) output

O(n) extra space

Recursion call stack :→ O(n)

Time Comp

$(4)^n + (k)n$

isQueenSafe

$\Rightarrow 4^n + n$

You are screen sharing Stop Share

NQueen Combinatn

Time Complexity

Bon on level :→ $2^{n^2} + n^3$: 120 ms

Backtracking :- $4^n + n^2$: 4 ms

Branch & Bound :- $4^n + n$: 3 ms

Space Complexity

Bon on level :→ $O(n^2)$ Recursion call stack

Backtracking :→ $O(n)$ Recursion " "

Branch & Bound :→ $O(N)$ extra space

$O(N)$ Recursion call stack

$O(n^2)$ output (chess)

BT Manipulat :→ $O(1)$ extra space

$O(N)$ Recursion call stack

④ Bit Manipulation

Replacing boolean visited array by bitset

boolean visited array ⇓ O(n) bytes

T/F

bitset ⇓ n bits ⇓ O(1) space

unset set

{0/1}

2	56	
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

111000

integer : 4 bytes : 32 bits

↳ ld

integer ↳ rd

integer ↳ cal

```java
public static boolean isQueenSafe(int r, int c, int n, BitSet ld, BitSet rd, BitSet col){
    return ((ld.get(r - c + n - 1) == true) || (rd.get(r + c) == true) || (col.get(c) == true)) ? false : true;
}
```

```java
public static void nqueen(int r, boolean[][] chess, BitSet ld, BitSet rd, BitSet col) {
    int n = chess.length;
    if (r == n) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) {⇔}
        res.add(ans);
        return;
    }


    for(int c=0; c<n; c++){
        if(isQueenSafe(r, c, n, ld, rd, col)){
            chess[r][c] = true;
            ld.set(r - c + n - 1, true);
            rd.set(r + c, true);
            col.set(c, true);

            nqueen(r + 1, chess, ld, rd, col);

            ld.set(r - c + n - 1, false);
            rd.set(r + c, false);
            col.set(c, false);
            chess[r][c] = false;
        }
    }
}
```

N Queen - II

using C++

Count

```
bitset<1000000> col,d1,d2;
void solve(int i,int n,int &ans)
{
    if(i == n) {ans++; return;}
    for(int j=0;j<n;j++)
    {
        if(!col[j] && !d1[i-j+n-1] && !d2[i+j])
        {
            col[j] = d1[i-j+n-1] = d2[i+j] = 1;
            solve(i+1,n,ans);
            col[j] = d1[i-j+n-1] = d2[i+j] = 0;
        }
    }
}
class Solution {
public:
    int totalNQueens(int n) {
        if(n <= 1) return n;
        int ans = 0;
        solve(0,n,ans);
        return ans;
    }
};
```

R&B - Level-2 - Lecture 8

Sudoku & Other Puzzles

Puzzles
Valid Sudoku (IB)
Sudoku Solver (IB)
Using Backtracking
Using Bit Manipulation
Crossword Puzzle
Cryptarithmetic Puzzle
Cryptarithmetic Puzzle
Max Score
Max Score

1 2 3 4

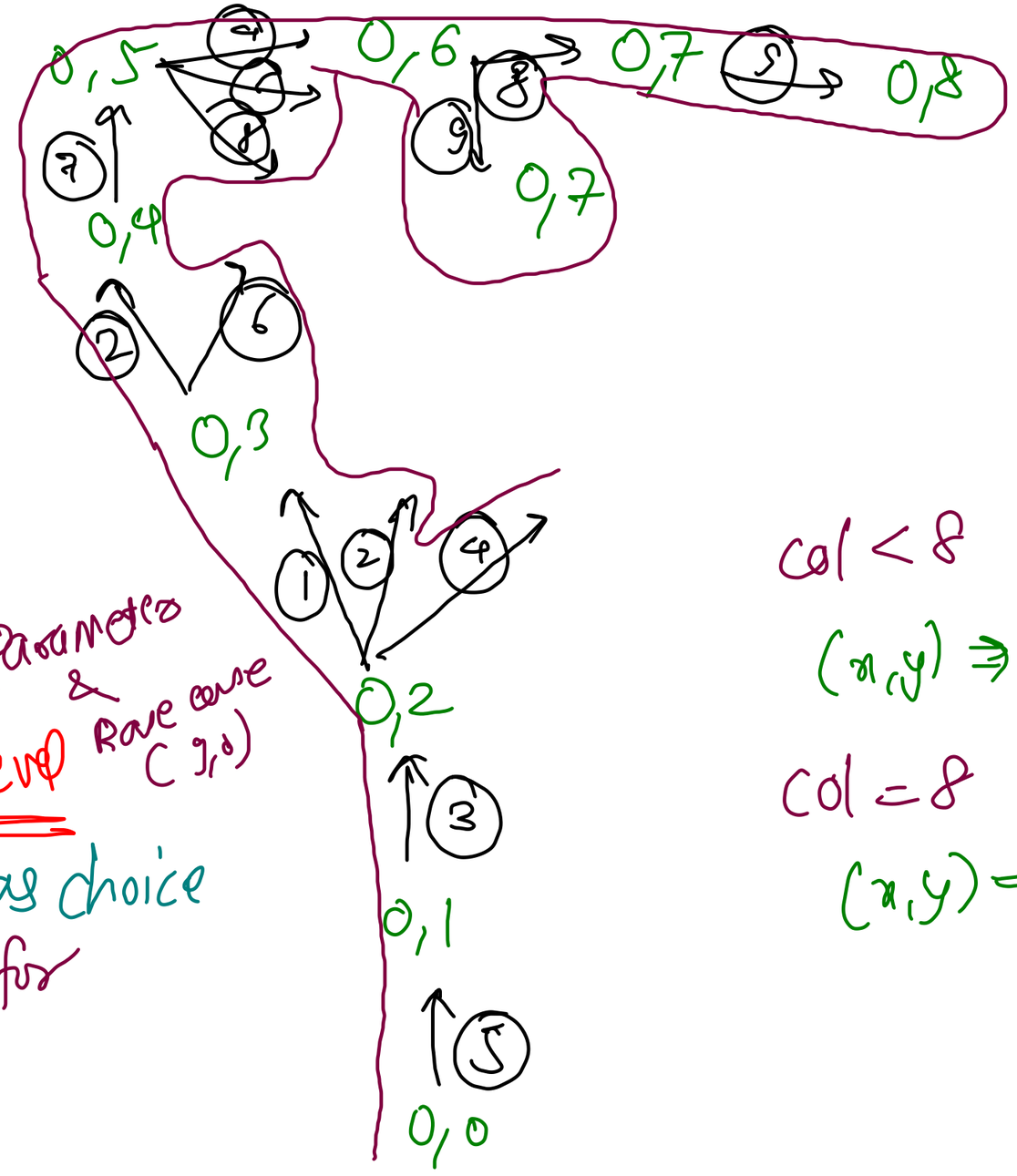
5 N Knight , Knight's Tour

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

① In each row, we should have 1-9
↳ 9 columns

② In each column, we should have 1-9
↳ 9 rows

③ In each 3*3 matrix, we should have 1-9



cell → Born on level

→ parameters & Have cone (9,0)

Items as choice
↳ for

col < 8
(x,y) ⇒ (x, y+1)

col = 8
(x,y) ⇒ (x+1, 0)

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

$\boxed{\text{top-left}}$

val

0, 1, 2 ⇒ 0 = 0*3 = (val/3)*3

3, 4, 5 ⇒ 3 = 1*3 = (val/3)*3

6, 7, 8 ⇒ 6 = 2*3 = (val/3)*3

8, 4 ⇒ 6, 3

row+0 col+0 col+1

(6,3) (6,4) (6,5)

row+1

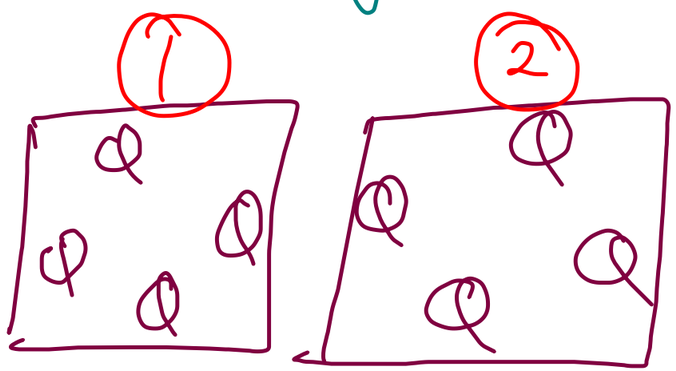
(7,3) (7,4) (7,5)

(8,3) (8,3) (8,5)

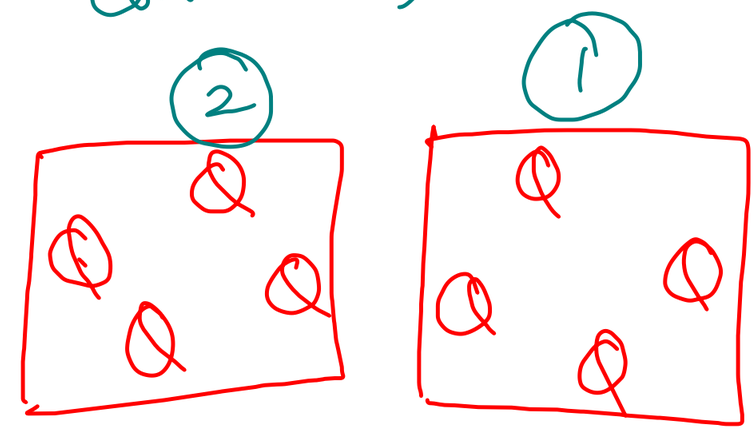
row+2

nQueen

row by row



column by column



Max Score

```
Input: words = ["dog","cat","dad","good"], letters =
["a","a","c","d","d","d","g","o","o"], score =
[1,0,9,5,0,0,3,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0]
Output: 23
```

a - ~~2~~ ~~1~~ ~~7~~ ~~1~~ 2

c - 1

d - ~~3~~ ~~2~~ ~~0~~ ~~2~~ ~~3~~ ~~1~~ ~~0~~ ~~1~~ 3

g - ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

o - ~~2~~ ~~0~~ ~~2~~ ~~1~~ ~~2~~ 1

"dog"

"dad" + "good"

"good"

"dad" + "good"

"dad"

"dog"

"dad"

~~"good"~~

23, 0, freq

cursor

```java
public int maxScore(int idx, String[] words, int[] freq, int[] score){
    if(idx == words.length) return 0;

    // yes
    int currScore = 0;
    boolean flag = true;
    for(char ch: words[idx].toCharArray()){
        freq[ch - 'a']--;
        if(freq[ch - 'a'] < 0){
            flag = false;
        }

        currScore += score[ch - 'a'];
    }

    if(flag == true){
        currScore += maxScore(idx + 1, words, freq, score);
    } else {
        currScore = 0;
    }

    // backtrack
    for(char ch: words[idx].toCharArray()){
        freq[ch - 'a']++;
    }

    // no
    return Math.max(currScore, maxScore(idx + 1, words, freq, score));
}
```

Move Zeroes to End

$\{\cancel{0}, \cancel{1}, \cancel{0}, 3, 12\}$ $\Longrightarrow$ $\{1, 3, 12, 0, 0\}$

$f(0,0) \rightarrow f(1,0) \rightarrow f(2,1) \rightarrow f(3,1) \rightarrow f(4,2) \rightarrow f(5,3)$

```java
public static void moveZeroes(int idx, int countOfNonZero, int[] nums) {
    if(idx == nums.length){
        int countOfZeros = nums.length - countOfNonZero;
        for(int i=0; i<countOfZeros; i++){
            nums[nums.length - 1 - i] = 0;
        }
        return;
    }

    if(nums[idx] != 0){
        nums[countOfNonZero] = nums[idx];
        moveZeroes(idx + 1, countOfNonZero + 1, nums);
    }
    else {
        moveZeroes(idx + 1, countOfNonZero, nums);
    }
}
```

Today's Questions

1. Crossword
2. Word Search - (I, II, III)
3. Cryptarithmetic
4. Josephus

Cryptarithmetic

1 : 1 mapping

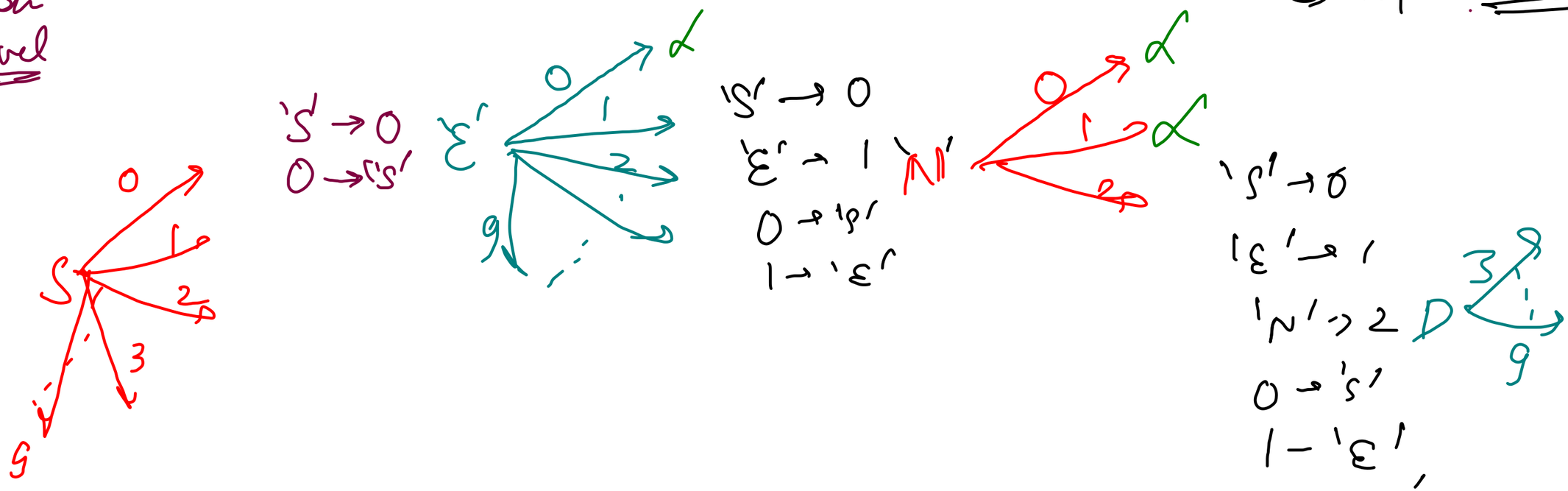
char (A-Z) → digit (0-9)

"SEND" + "MORE" = "MONEY"

unique char on level

{'S', 'E', 'N', 'D', 'M', 'O', 'R', 'Y'} base case

equivalency check



'S' → 0

'S' → 0

'E' → 1

'N' → 2

'M' → 1

'O' → 0

'R' → 9

'Y' → 2

Josephus

$k = 4$

$n = 8$: 0 1 2 3 4 5 6 7 } y
0 1 2 ~~3~~ 4 5 6 7 } x

$n = 7$: 0 1 2 3 4 5 6 } x
4 5 6 ~~7~~ 0 1 2 } y

$n = 6$: 0 1 2 ~~4~~ 5 6

$n = 5$: 5 6 0 ~~1~~ 2

$n = 4$: 2 5 6 ~~0~~

$n = 3$: ~~2~~ 5 6

$n = 2$: 5 ~~6~~

$n = 1$: 5

~~0~~ 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~

{0 1 2 3 4 5 6} x
{4 5 6 7 0 1 2} y

$n = 8$
$k = 4$

actual value — index

$$y = (x + k) \% n$$

Shifting or (E)
rotation on (x)

Meaning Exp

$$(\text{josephus}(n-1, k) + k) \% n$$

```java
public static int solution(int n, int k){
    if(n == 1) return 0; // return index
    return (solution(n - 1, k) + k) % n;
}
```

$(0+4)\%7 = 1$

0 1 2 3 4 5 6 — $n=7$, $k=4$

$(0+4)\%6 = 4$

0 1 2 3 4 5 — $n=6$, $k=4$

4 5 6 0 1 2

$(1+4)\%5 = 5\%5 = 0$

0 1 2 3 4 — $n=5$, $k=4$

1 2 4 5 6

$(1+4)\%4 = 5\%4 = 1$

0 1 2 3 — $n=4$, $k=4$

6 1 2 4

$(0+4)\%3 = 4\%3 = 1$

0 1 2 — $n=3$, $k=4$

6 1 2

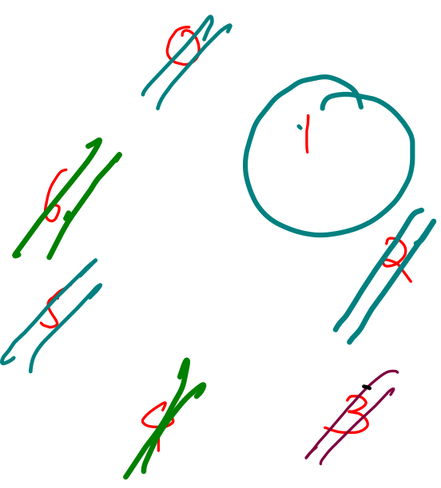
$(0+4)\%2 = 4\%2 = 0$

0 1 — $n=2$, $k=4$

1 2

0 — $n=1$, $k=4$

1



TC ⇒ $O(n)$

SC ⇒ $O(1)$ extra space

$O(n)$ Recursion

```java
public static int solution(int n, int k){
    Queue<Integer> q = new ArrayDeque<>();
    for(int i=0; i<n; i++){
        q.add(i);
    }
    int count = 0;
    while(q.size() > 1){
        int val = q.remove();
        count++;

        if(count == k){
            count = 0;
        } else {
            q.add(val);
        }
    }
    return q.peek();
}
```

constructive algorithm

Queue { Brute force }

$O(n*k) \Rightarrow$ Time

$O(n) \Rightarrow$ Queue space

Henicographical No's

$N = 999$

1

10 100 101 102 103 104 - - - 109

11 110 111 112 113 - - - - 119

12 120 121 122 - - - - 129

⋮

19 191 192 - - - - 199

2

20 200 201 202 - - - 209

21 210 211 - - - - - 219

22 220 221 - - - 229

23 230 231 - - - - 239

⋮

29 290 291 - - - 299

3

30 → 301 - - - 309

31 → 310 - - - 319

32 -

⋮

39 → 390 - - - 399

4

40, 41, 42, - - - 49

5

50, - - 59

6

7

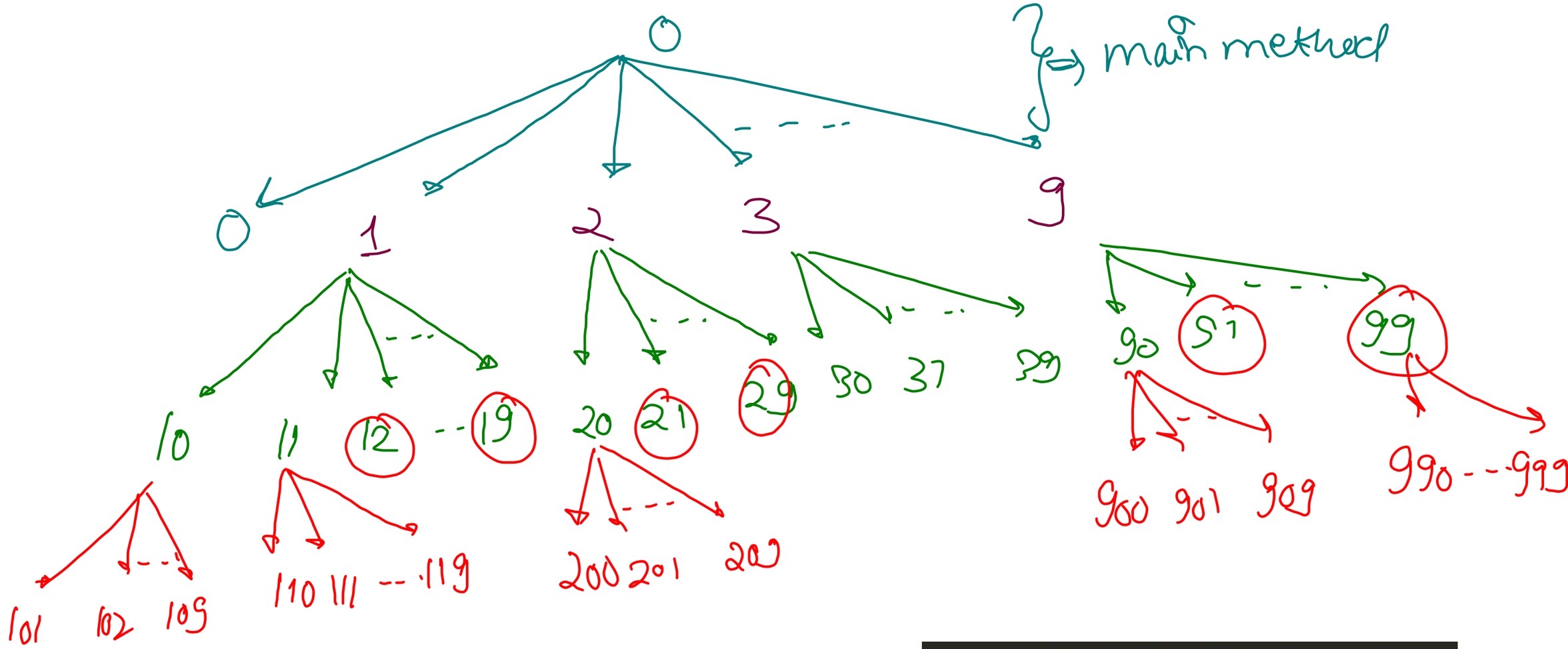
8

9

90 91 92 - - 99

900 901 - - 909

990-999




```java
public void lexicalOrder(int input, int n){
    if(input > n) return;

    res.add(input);
    for(int i=0; i<10; i++){
        lexicalOrder(input * 10 + i, n);
    }
}
```

R&B - Level 2 - Lecture 10

1. Crossword
2. Word Search - II

   Word Search - III {Homework}

3. Friends Pairings
4. Largest No in K swaps
5. Restore IP Address

More questions to practice

Paranthesis Addition Ways
Stepping Numbers
Kth Symbol in Grammar
Split into Fibonacci Sequence
Distinct Subsets Permutations

N knights

Friends Pairing

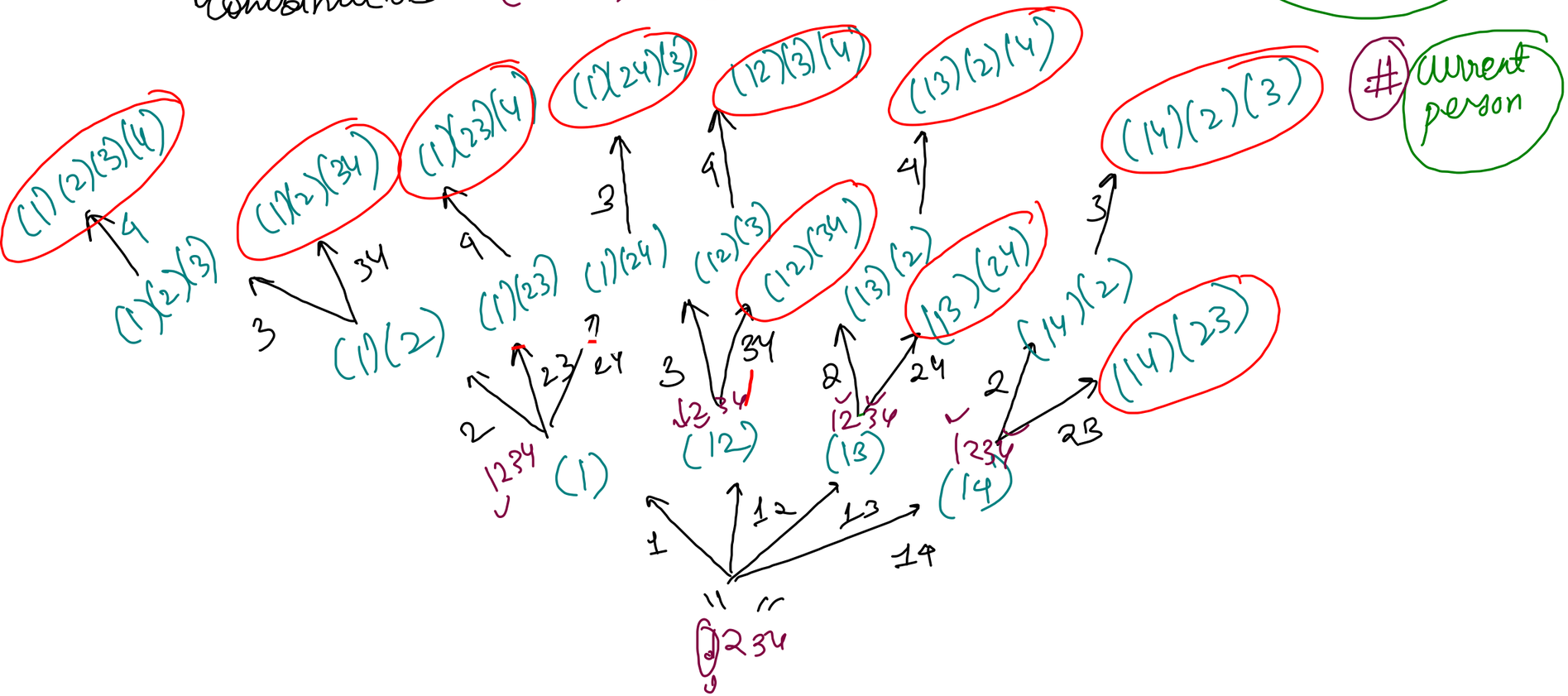
Combination

$n = 4$

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

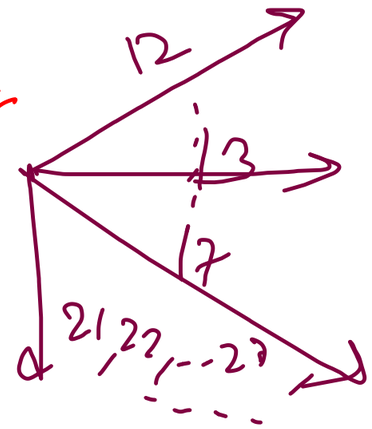
visited array

current person



Largest No in K swaps

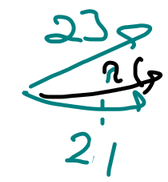
"1234567"



2134567

3214567

7234561



7634521

7654321

$k = 3$

level 1
swap = 1

level 2
swap 2

level 3
swap 3

Word Search - 11

o	a	a	n
e	t	a	e
i	h	k	r
i	f	l	v

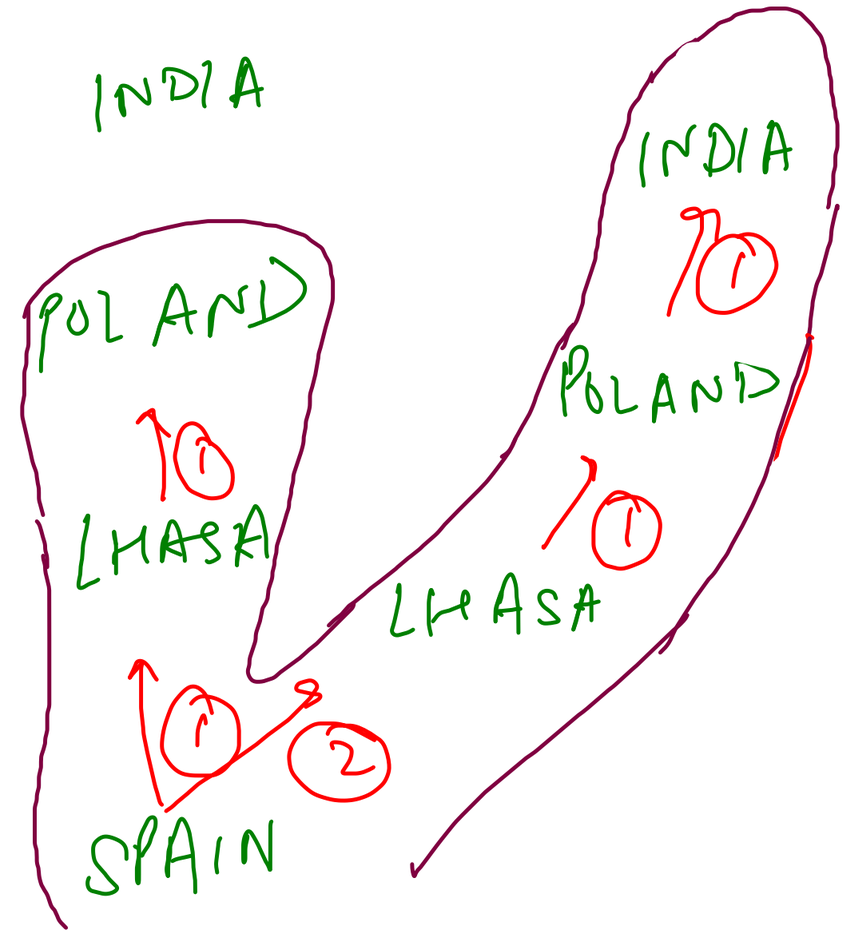
{oath, pea, eat, rain}

```java
public static void solution(int totalFriends, boolean[] friendInGroup, String groupsSofar) {
    int i = 0;
    for(int j=1; j<=totalFriends; j++){
        if(friendInGroup[j] == false){
            i = j;
            break;
        }
    }

    if(i == 0){}

    // Group of 1 Member
    friendInGroup[i] = true;
    solution(totalFriends, friendInGroup, groupsSofar + "(" + i + ") ");

    for(int j=1; j<=totalFriends; j++){
        if(friendInGroup[j] == false){
            friendInGroup[j] = true;
            solution(totalFriends, friendInGroup, groupsSofar + "(" + i + "," + j + ") ");
            friendInGroup[j] = false;
        }
    }

    friendInGroup[i] = false;
}
```

Crossword

[POLAND, LHASA, SPAIN, INDIA]:

```
++++++++++
+--L---+++
+++-++++++
+++-++++++
+++------++
+++-++-+++
++++++-+++
++++++-+++
++++++-+++
++++++++++
```

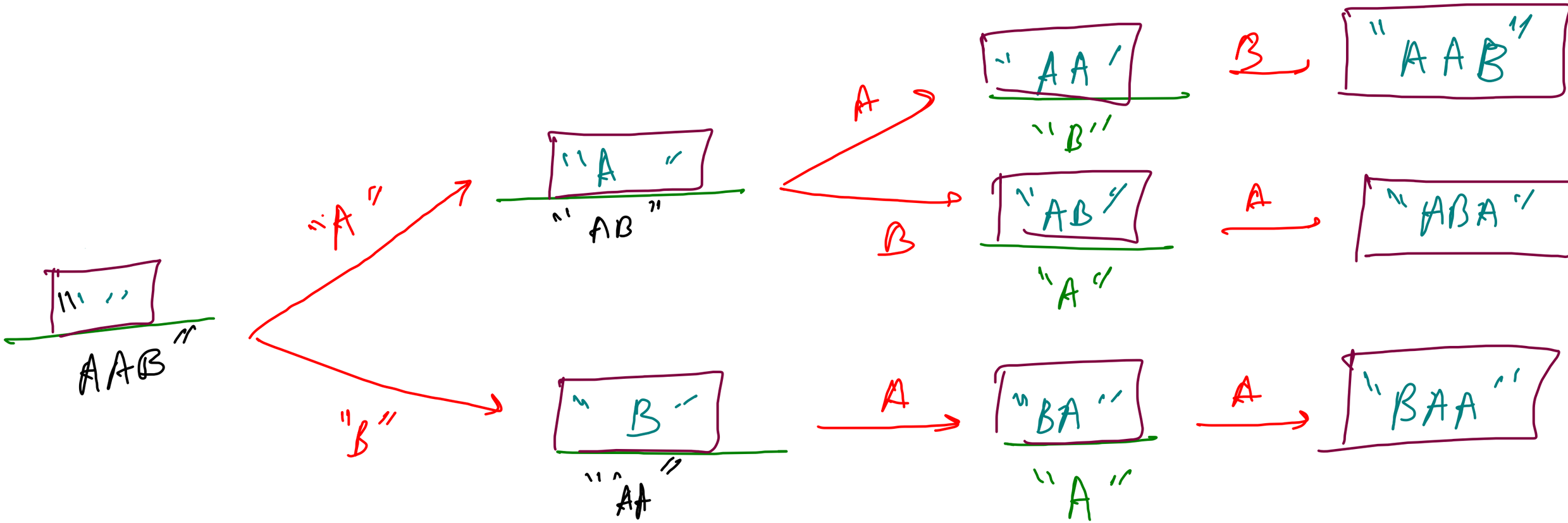


1. Completely fill + out of bound

2. Backtrack only those which are filled by you!!

Subsets + Permutations

{0 1 0 7 9} Letter Tile Possibilities



Box on level
Item as
option

Tentative Lecture Plan {DSA - Level 2}

1. Recursion & Backtracking → Complete (1)
2. Searching & Sorting → Complete (10)
3. Arrays & Strings
   - → Greedy Algo → complete (4)
   - → 1D & 2D arrays
   - → Strings

   } → 7-8 lectures {2 weeks}
4. HashMap & Heap
   - → Two pointer / Sliding window → Complete (5)
   - → Hashmap
   - → Heap

   } 5 lectures {1 week}

Binary Tree & BST → 10 lectures (2 weeks)

Dynamic Programming → 10-12 lectures (3 weeks)

Linked list → 3 lectures

Stack & Queue → 3 lectures

{1.5 weeks}

Graphs → 7-8 lectures {2 weeks}

Tries → 2 lectures

Bit Manipulation → 3 lectures

{2 weeks}

Level 3

- AVL Trees
- Number Theory & Mathematics
- Range Query
- Advanced DP { DP on Grid, DP on tree, Digit DP, etc }
- Graphs { Euler tree, Network flow, etc }
- Geometry
- String Algorithms

System Design (15th dec)

- OOPS (Java)
- LLD
- HLD