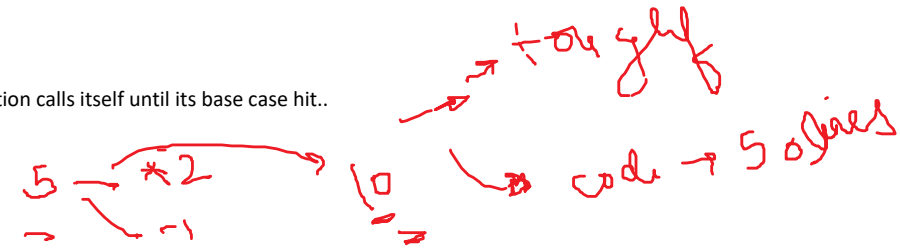
Recursion 1

11 July 2022 08:34

What is Recursion ?

A function calls itself until its base case hit..



Why we use Recursion ?

code redundancy and code is fully optimized

Recursion is fully done by stack in backend

What is PMI?

Recursion Follows The principle of Mathematical Induction

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

1.) Lets us suppose this equation holdes for n = k

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

2.) Lets us suppose the equation also holdes for n = k + 1

$$\sum_{i=1}^{k+1} i = \frac{(k+1)(k+2)}{2}$$

$$\sum_{i=1}^{k+1} i = 1 + 2 + 3 + \dots + k + (k+1)$$

$$= \sum_{i=1}^{k} i + (k+1)$$

k = 1,

$$\frac{(1+1)(1+2)}{2} = \frac{k(k+1)}{2} + (k+1)$$

$$= \frac{2 \times 3}{2} = 3 \qquad = \frac{(k+1)(k+2)}{2}$$

3.) k = 1

$$\sum_{i=1}^{k} = \frac{k(k+1)}{2} \quad \cdot k = 2 \quad \cdot \quad \frac{2*3}{2} = 3$$

$\sum_{i=1}^{k} = \frac{k(k+1)}{2}$

Exp, faith, link.

(1) Print Decreasing Order

5 → 5, 4, 3, 2, 1.

4 → 4, 3, 2, 1

Ex: p(5) = 5, 4, 3, 2, 1

faith p(4) = 4, 3, 2, 1

link: p(5) = 5; p(4)

4, 3, 2, 1

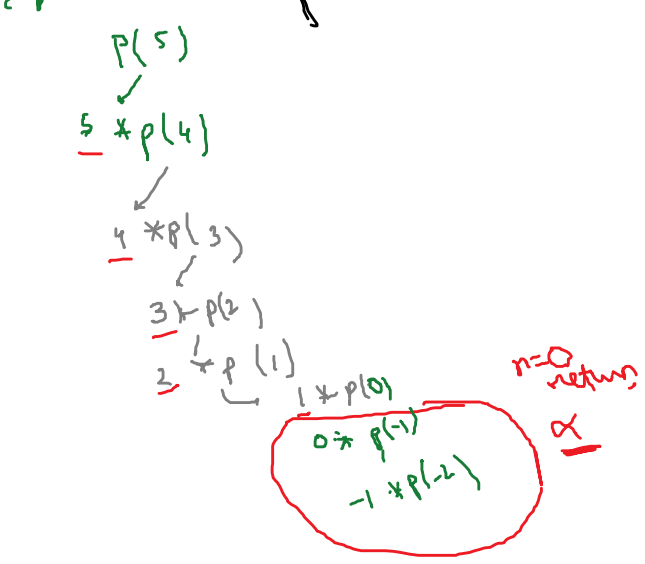
6: pd(n) =

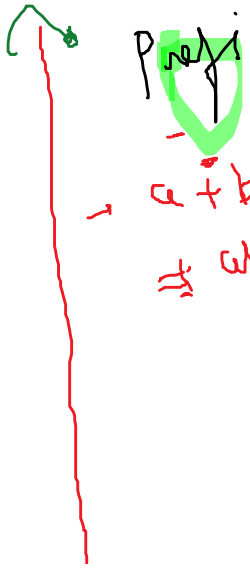
Ex: p(6) = 6, 5, 4, 3, 2, 1

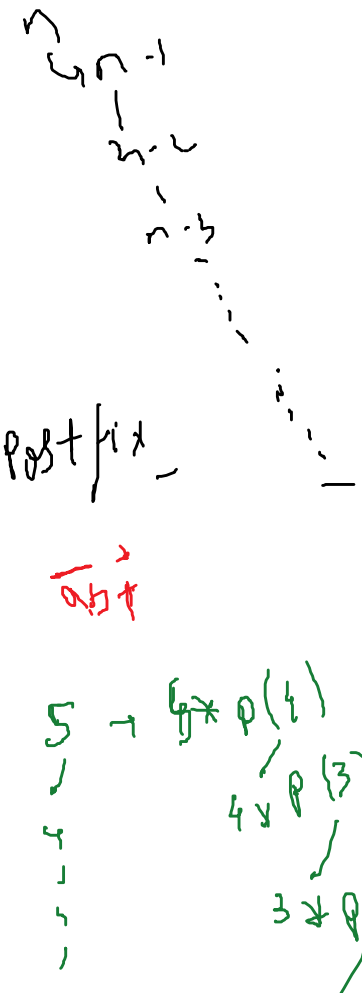
faith p(5) = 5, 4, 3, 2, 1

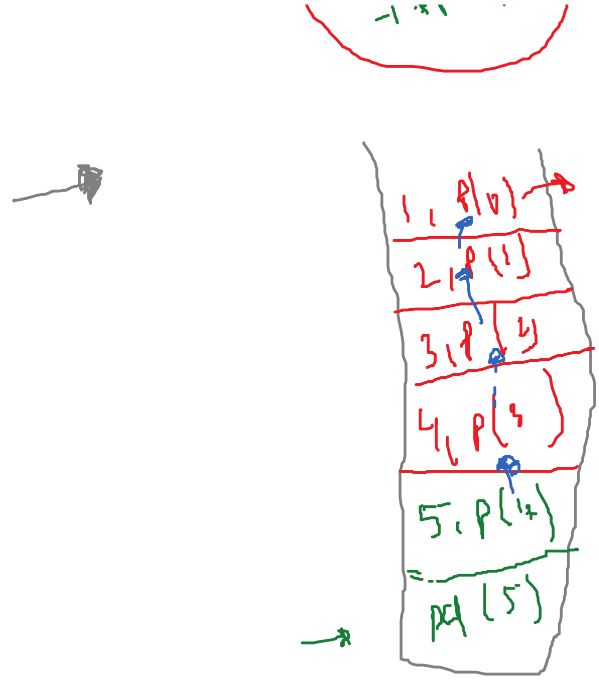
link: p(6) = 6, p(5) pd(n-1)

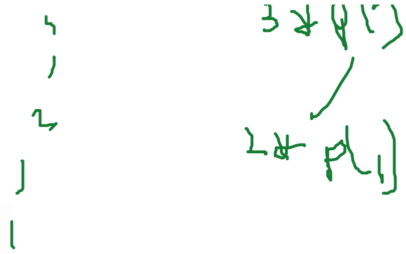
pd(n) n → 5, 4, 3, 2, 1







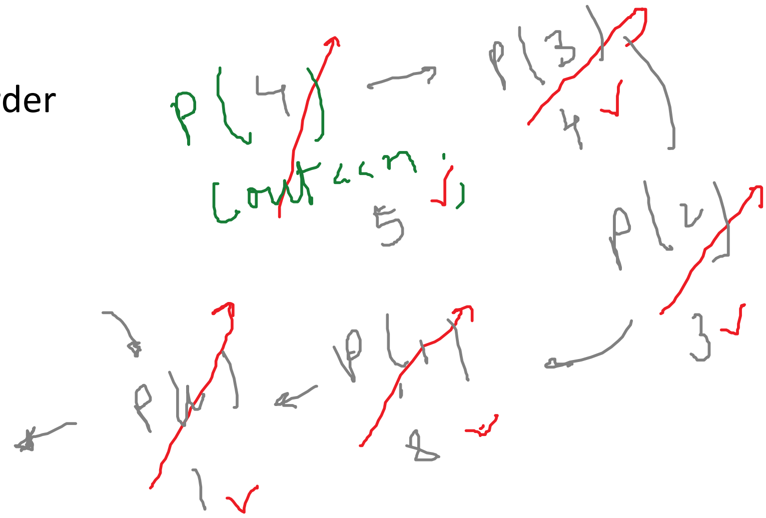




```
oid printDecreasing(int n){
    if(n == 0) return ;
    cout<< n << endl;  // 5
    printDecreasing(n - 1); // p(4);

nt main(){
    int n;
    cin >> n;
    printDecreasing(n);
```

② Print Increasing Order

5: 1, 2, 3, 4, 5



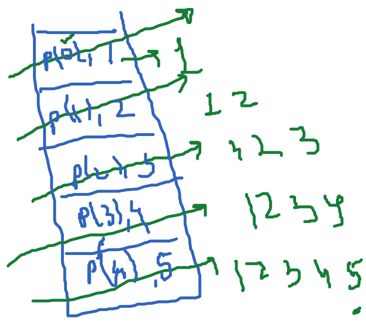
Ex: p(5): 1, 2, 3, 4, 5

faith: p(4): 1, 2, 3, 4

p(3): 1, 2, 3

work: p(5) = 1, 2, 3, 4, 5

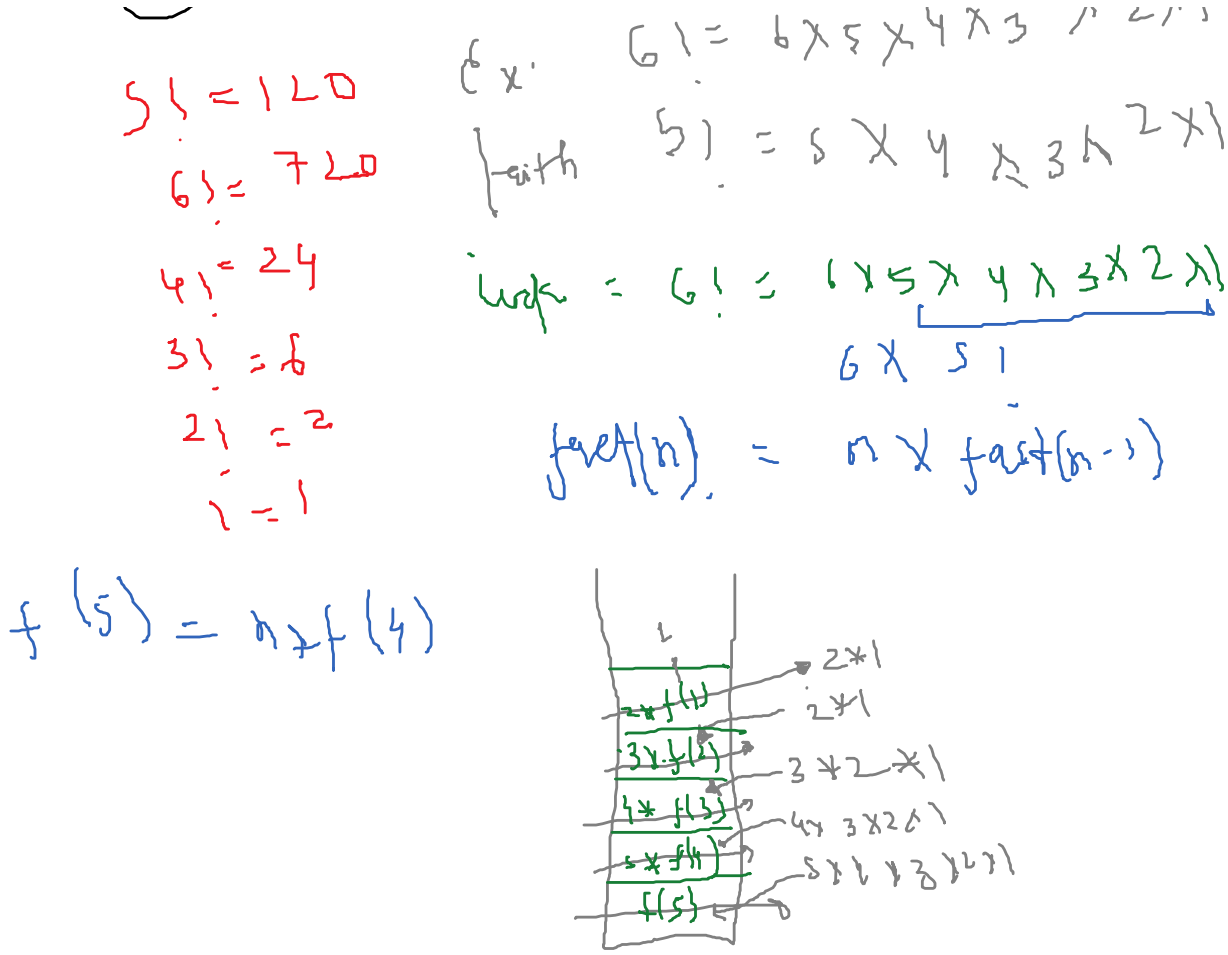
p(4) . 5



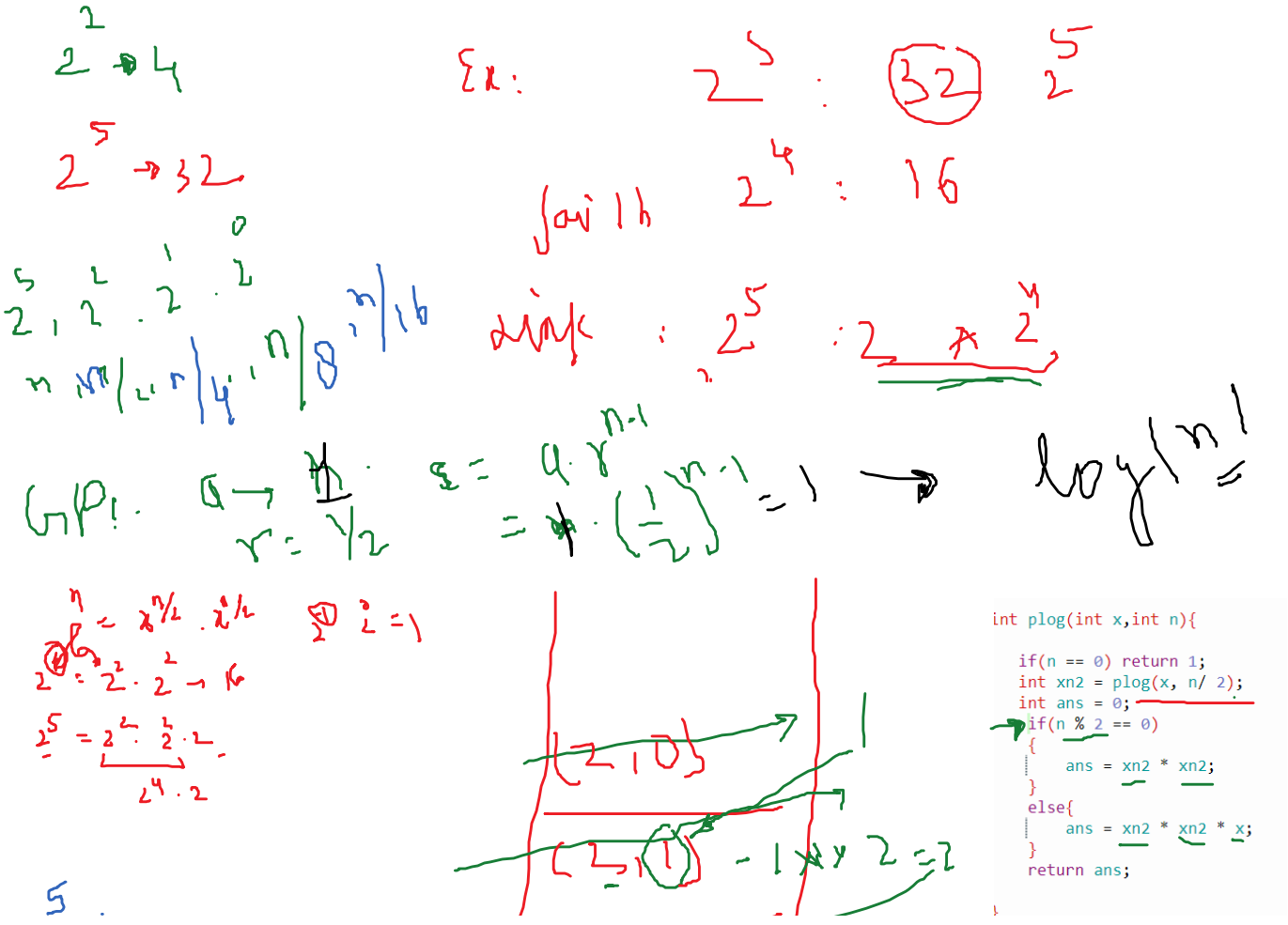
③ → Factorial of a Number

5! = 120

Ex: 6! = 6 x 5 x 4 x 3 x 2 x 1

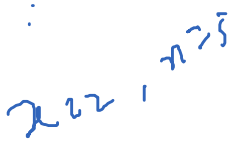


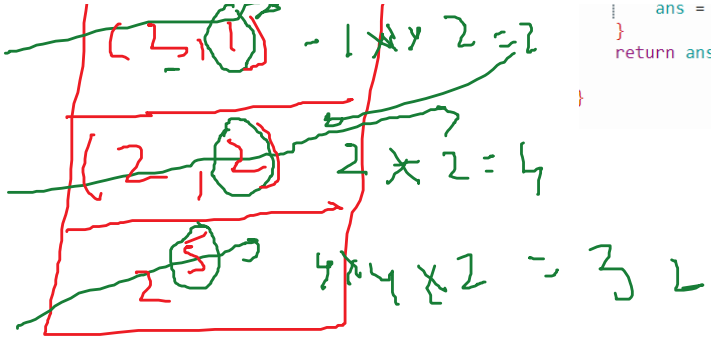
Power Lograthemic



```
int plog(int x,int n){

   if(n == 0) return 1;
   int xn2 = plog(x, n/ 2);
   int ans = 0;
   if(n % 2 == 0)
   {
       ans = xn2 * xn2;
   }
   else{
       ans = xn2 * xn2 * x;
   }
   return ans;
}
```

2^5 :



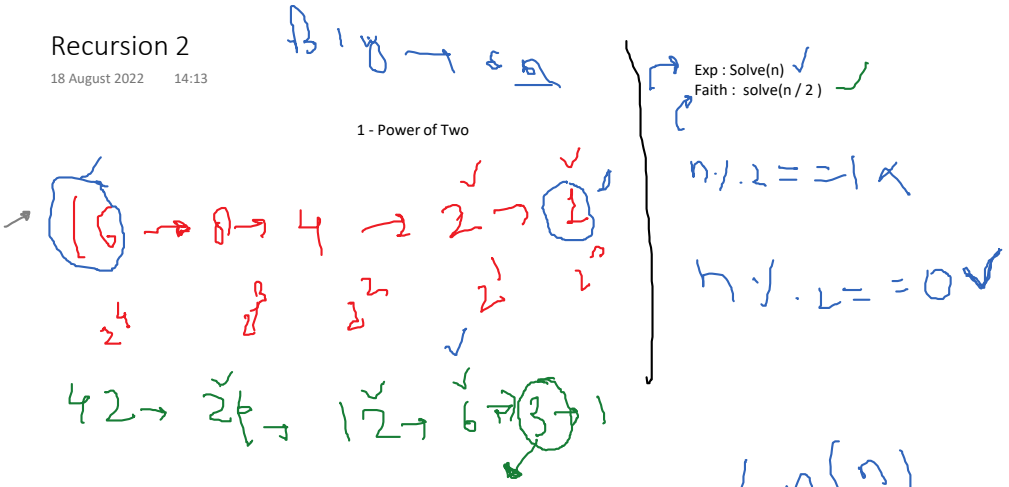



```
    ans = xn2 * xn2 * x;
}
return ans;
```

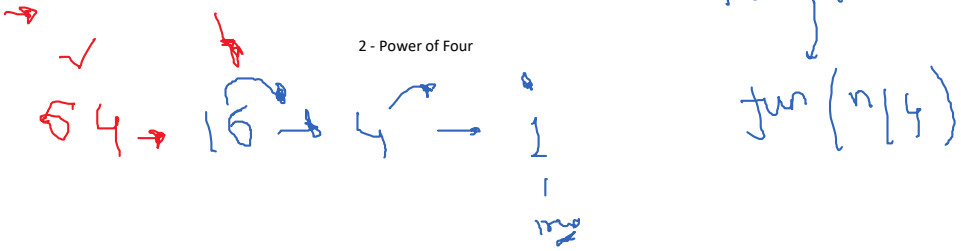
Recursion 2

18 August 2022 14:13

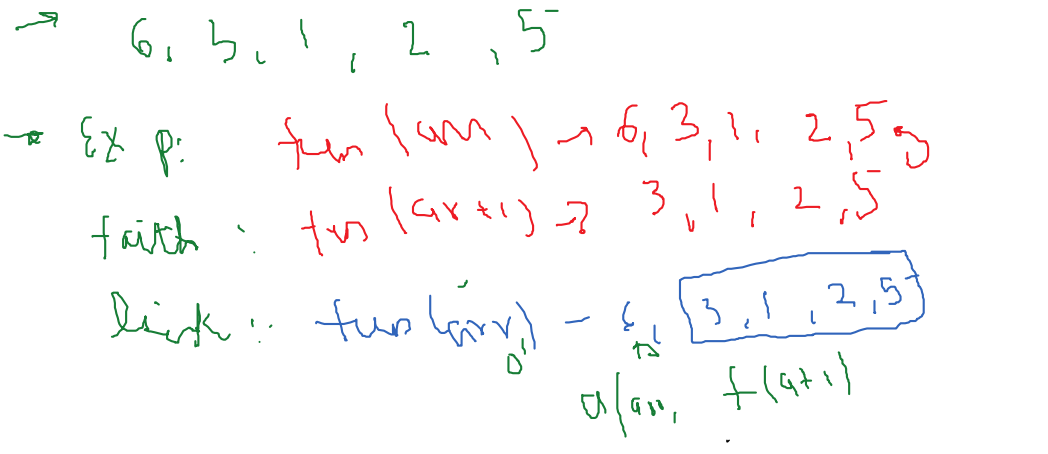
1 - Power of Two



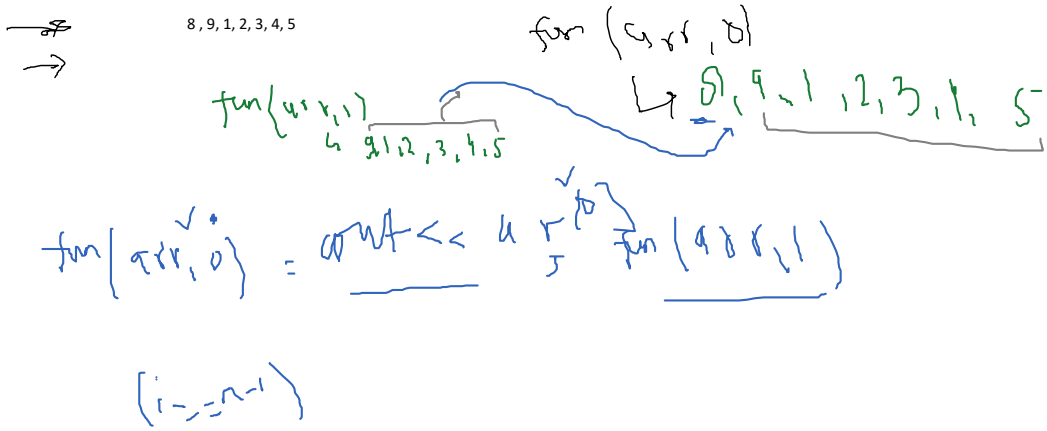
2 - Power of Four



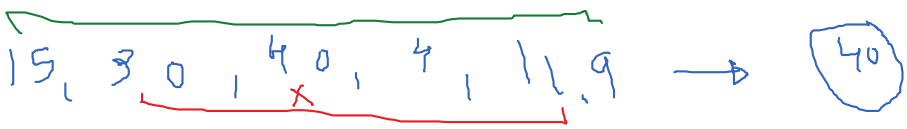
3- Display an array

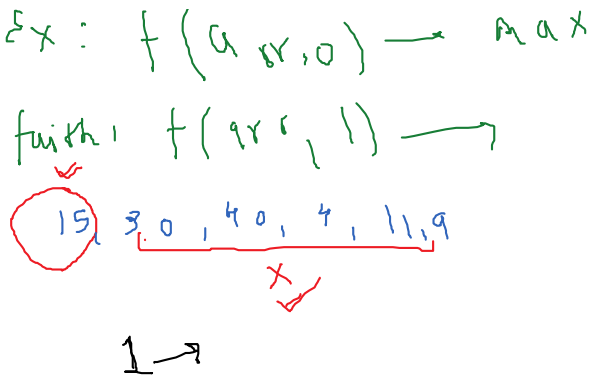


8 , 9, 1, 2, 3, 4, 5

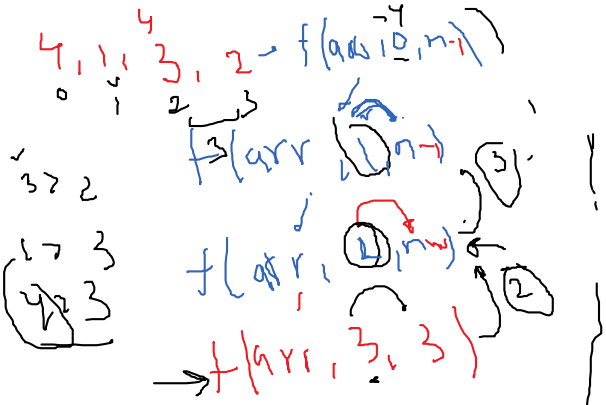


3 - Max of an array

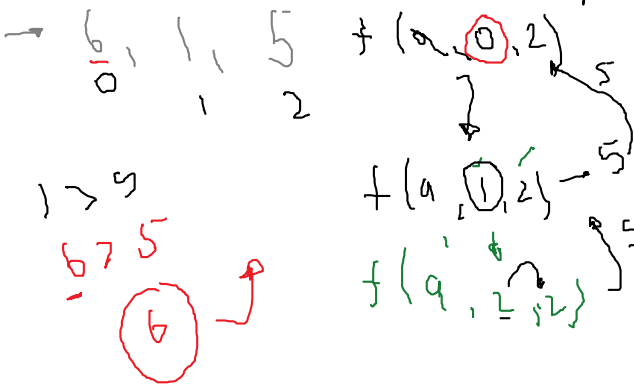




```
Solve(arr , 0, n)
{
If(idx == n - 1)
{
        Cout << arr[idx] << endl;
        Return ;
}
        Int x = solve(arr , idx + 1 , n);
        If(arr[idx] > x)
        {
                Return arr[idx];

                Screen clipping taken: 18-08-2022 15:23

        }
        Else
                Return x;
}
```

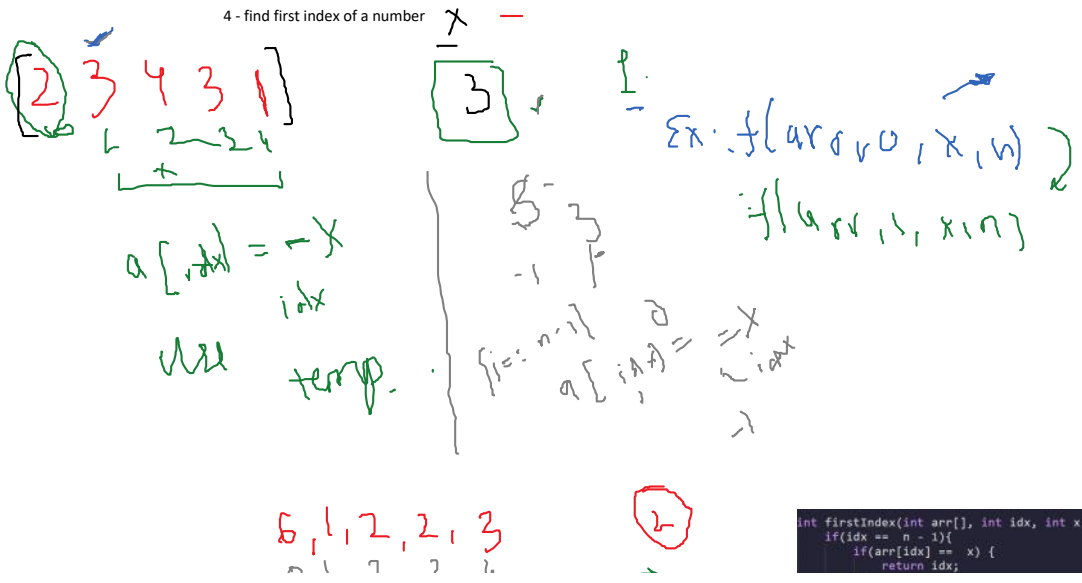


```
int max(int arr[], int idx, int n){
    if(idx ==  n - 1){
        return arr[idx] ;
    }

    int temp = max(arr , idx + 1 , n) ;
    if(arr[idx] > temp)
    {
        return arr[idx];
    }
    else{
        return temp;
    }
}
```

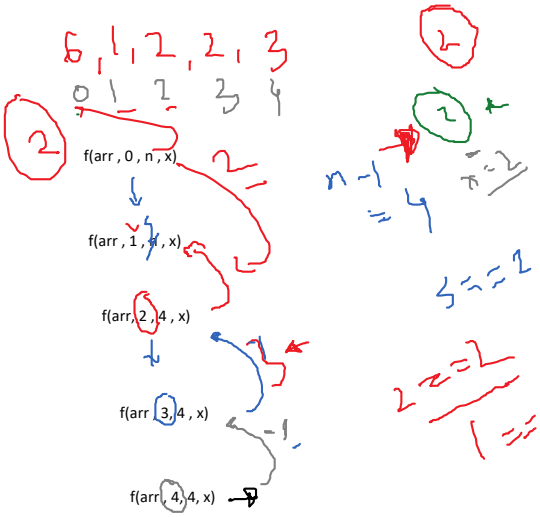


```
int max(int arr[], int idx, int n){
    if(idx ==  n - 1){
        return arr[idx] ;
    }

    int temp = max(arr , idx + 1 , n) ;
    if(arr[idx] > temp)
    {
        return arr[idx];
    }
    else{
        return temp;
    }
}
```

4 - find first index of a number x

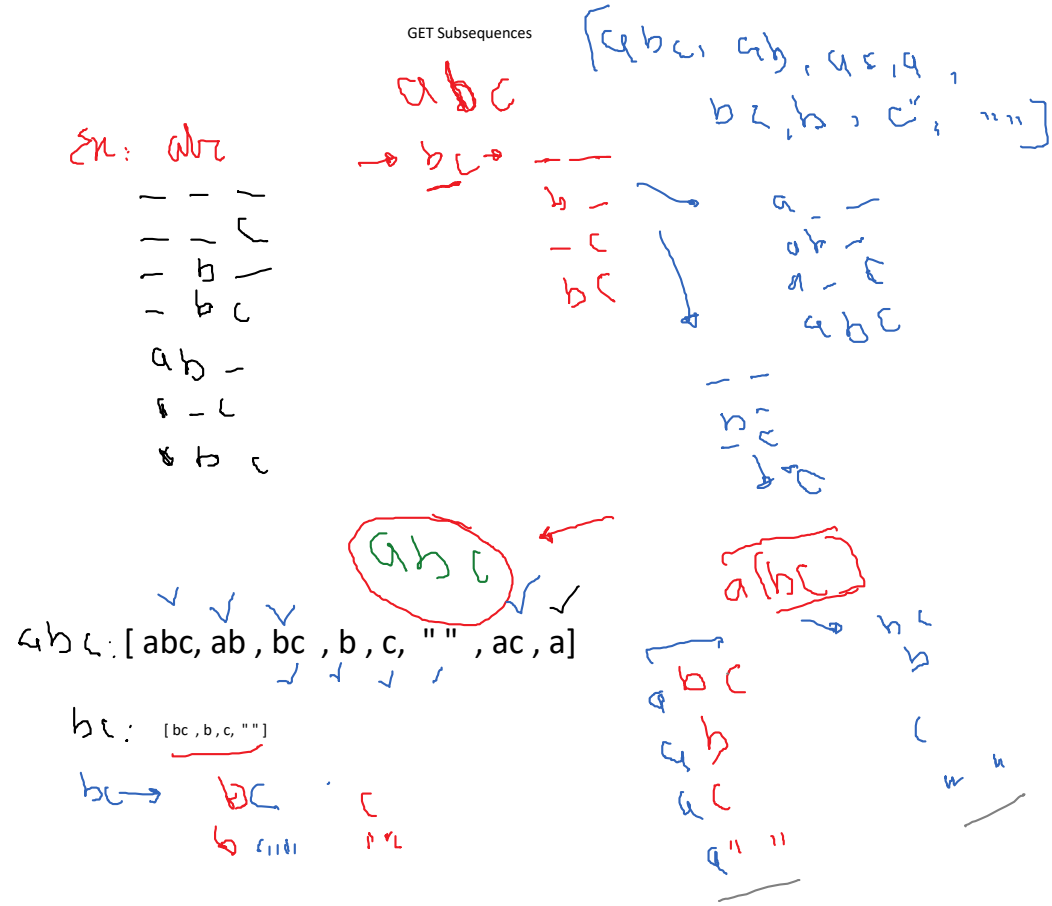


```
int firstIndex(int arr[], int idx, int x, int n){
    if(idx ==  n - 1){
        if(arr[idx] ==  x) {
            return idx;
```



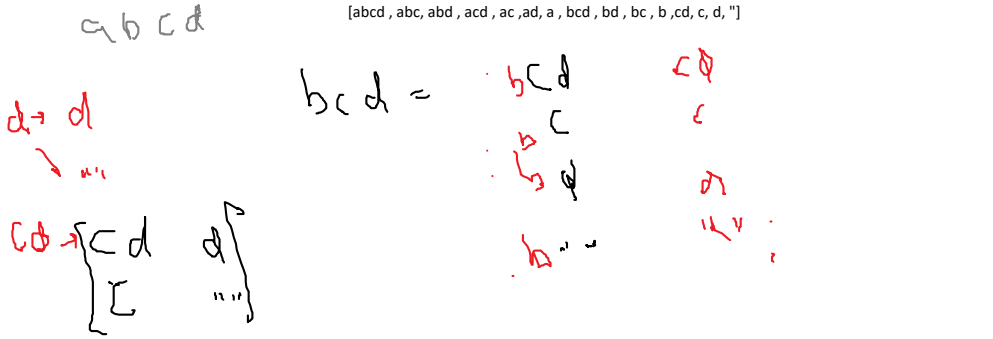

```
int firstIndex(int arr[], int idx, int x, int n){
    if(idx ==  n - 1){
        if(arr[idx] ==  x) {
            return idx;
        }
        else{
            return -1;
        }
    }

    // faith ;
    int temp = firstIndex(arr , idx + 1 , x,  n);
    if(arr[idx] == x) {
        return idx;
    }else{
        return temp;
    }
}
```

GET Subsequences



abc: [abc, ab , bc , b , c, "" , ac , a]

bc: [bc , b , c, ""]



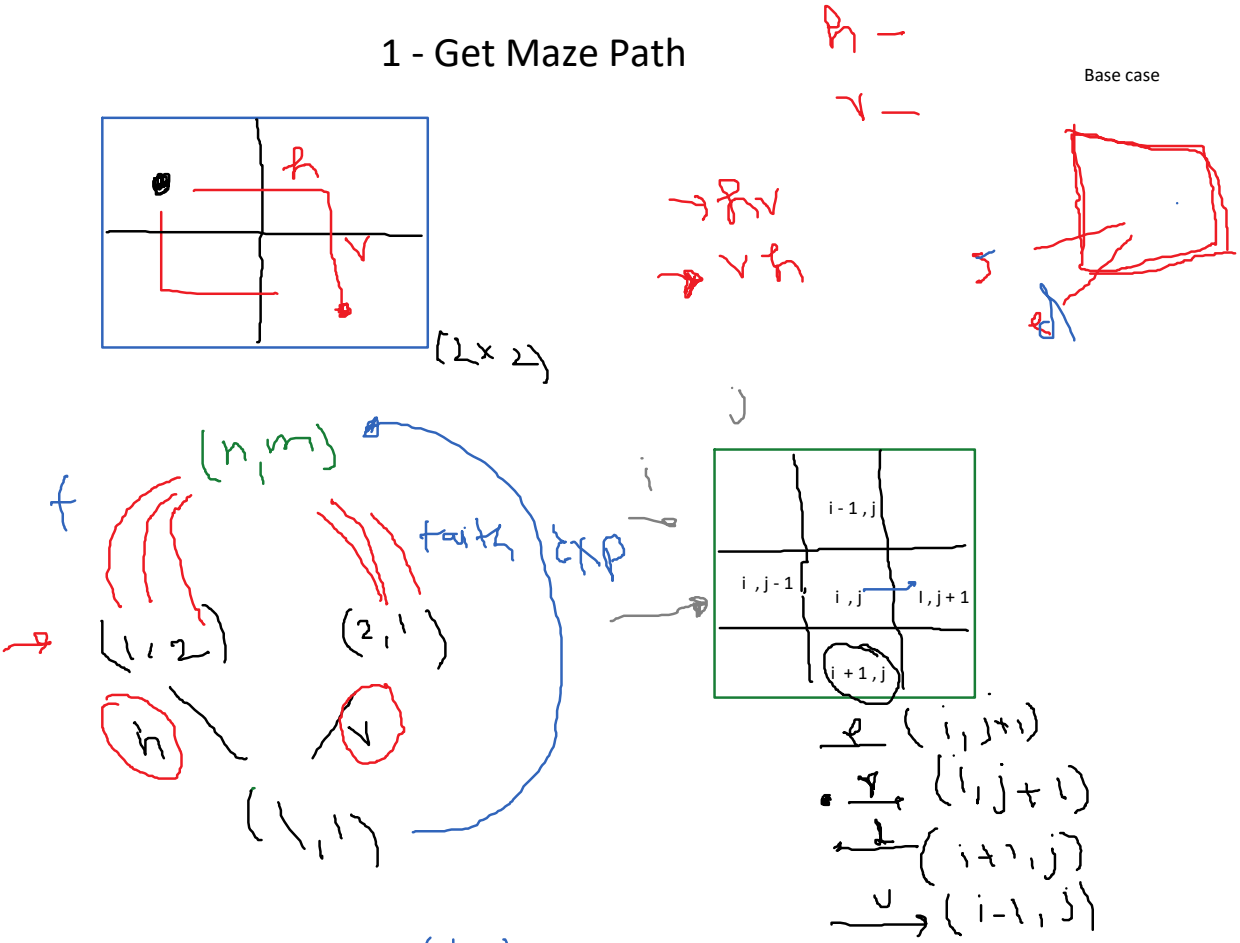
abcd

[abcd , abc, abd , acd , ac ,ad, a , bcd , bd , bc , b ,cd, c, d, ""]

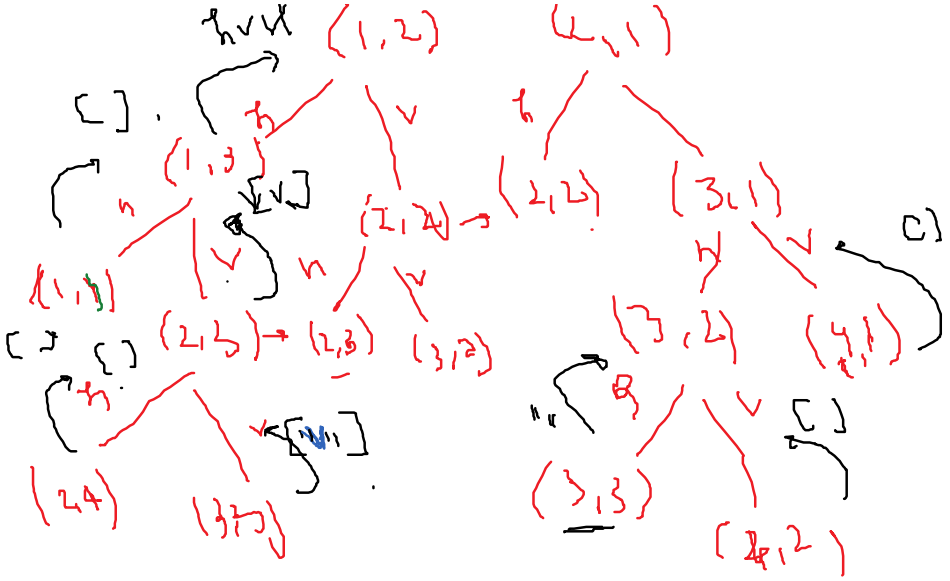
Recursion 3

19 August 2022 14:30

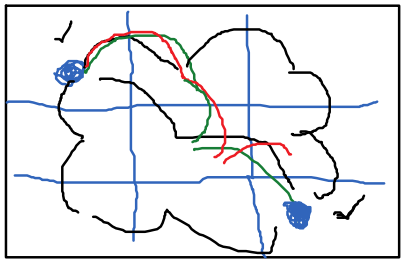
1 - Get Maze Path

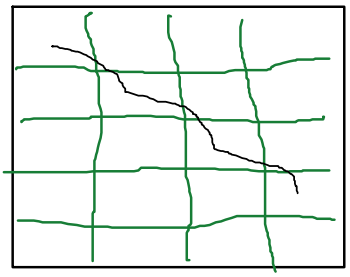


Sr > dr

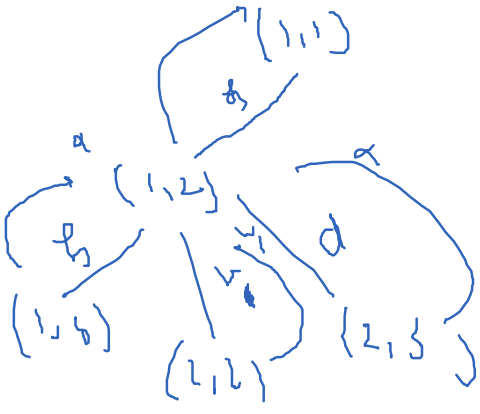


Get Maze Path With Jumps

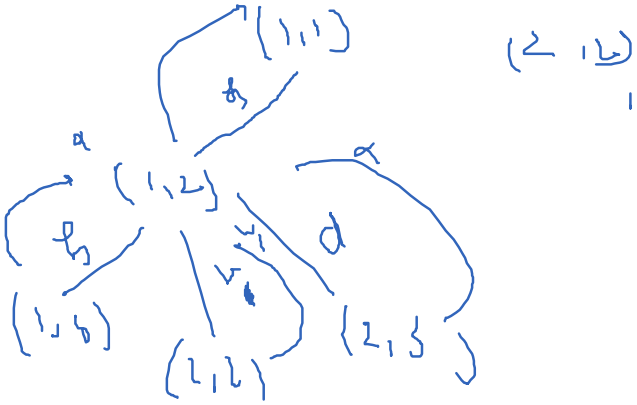




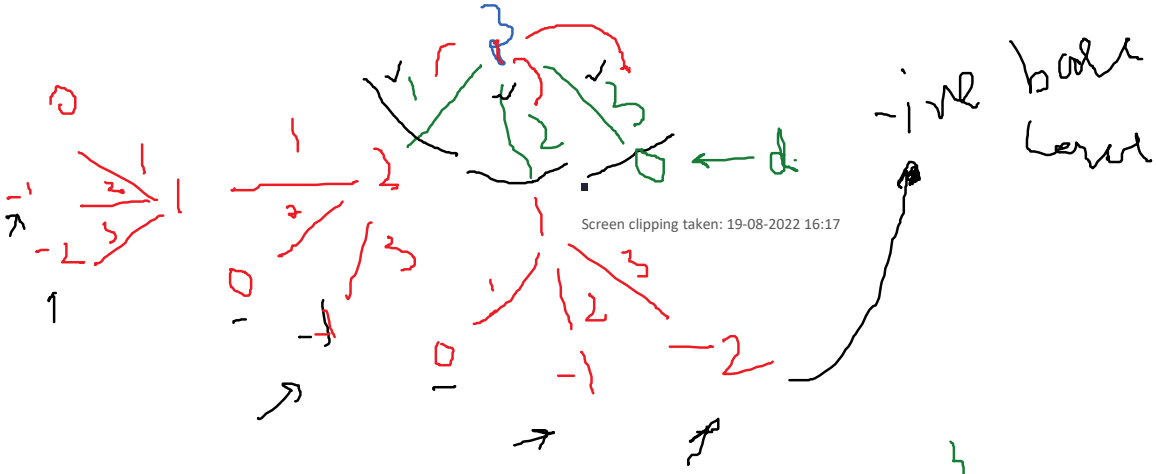
```
for(int i = 1;  i<= dr - sr  ; i++)
{
    vector<string> hpath = get_maze_paths(sr , sc + i , dr, dc); //horizontal
    for(auto x : hpath)
    {
        string idx = to_string(i) ;
        ans.push_back("h" + idx + x) ;
    }
}

// for vertical calls
for(int i = 1;  i<= dc - sc  ; i++)
{
    vector<string> vpath = get_maze_paths(sr + i , sc, dr, dc ); // vertical
    for(auto x : vpath)
    {
        string idx = to_string(i) ;
        ans.push_back("v" + idx + x) ;
    }
}

// for diagonal calls;
```



```cpp
for(int i = 1;  i<= dr - sr  ; i++)
{
    vector<string> hpath = get_maze_paths(sr , sc + i , dr, dc); //horizontal
    for(auto x : hpath)
    {
        string idx = to_string(i) ;
        ans.push_back("h" + idx + x) ;
    }
}

// for vertical calls
for(int i = 1;  i<= dc - sc  ; i++)
{
    vector<string> vpath = get_maze_paths(sr + i , sc, dr, dc ); // vertical
    for(auto x : vpath)
    {
        string idx = to_string(i) ;
        ans.push_back("v" + idx + x) ;
    }
}

// for diagonal calls;
// int mini = min(dc-sc , dr- sr);
for(int i = 1;  i<= dc - sc && i<= dr - sr ; i++)
{
    vector<string> dpath = get_maze_paths(sr + i, sc + i, dr, dc); // diagnoal
    for(auto x : dpath)
    {
        string idx = to_string(i) ;
        ans.push_back("d" + idx + x) ;
    }
}
return ans ;
```



Get Stair Paths

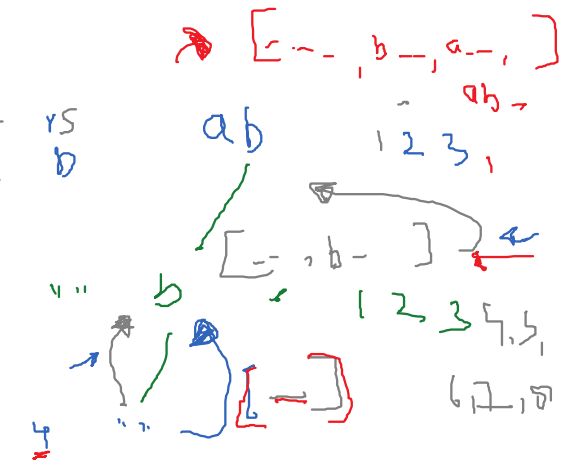


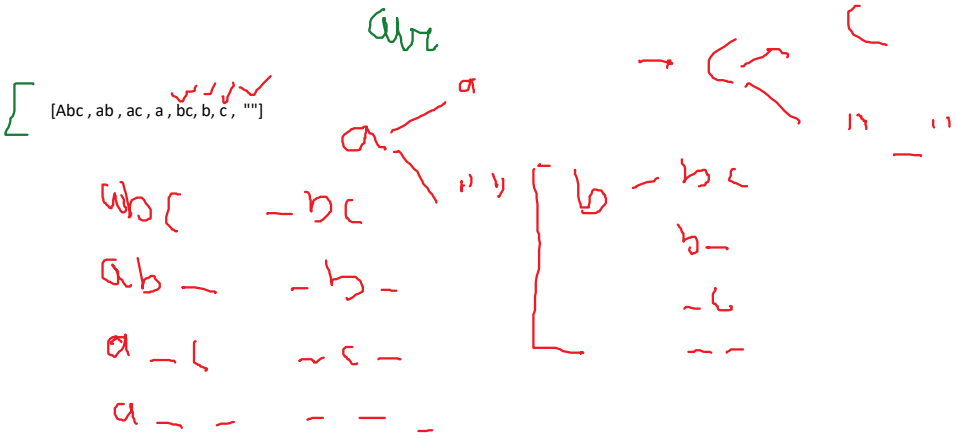

```cpp
vector<string> get_stair_paths(int n)
{
    if(n < 0){
        return {};
    }
    else if( n == 0) {
        return {""};
    }

    vector<string> one = get_stair_paths(n - 1);
    vector<string> two = get_stair_paths(n - 2);
    vector<string> three = get_stair_paths(n - 3);

    vector<string> ans ;
    for(string x : one)
    {
        ans.push_back("1" + x);
    }

    for(string x : two)
    {
        ans.push_back("2" + x);
    }

    for(string x : three)
    {
        ans.push_back("3" + x);
    }
    return ans ;
}
```

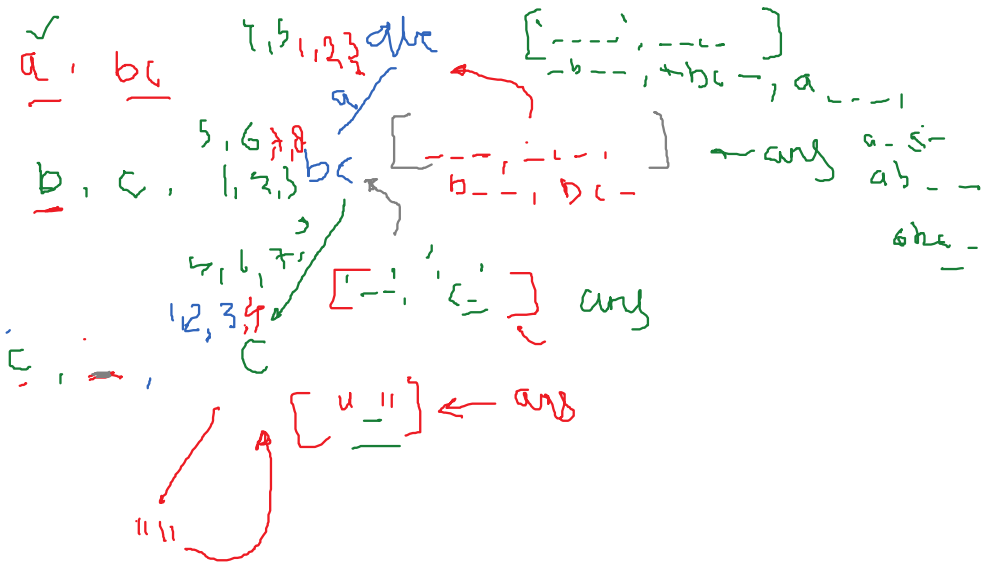
Recursion 4

21 August 2022 14:05

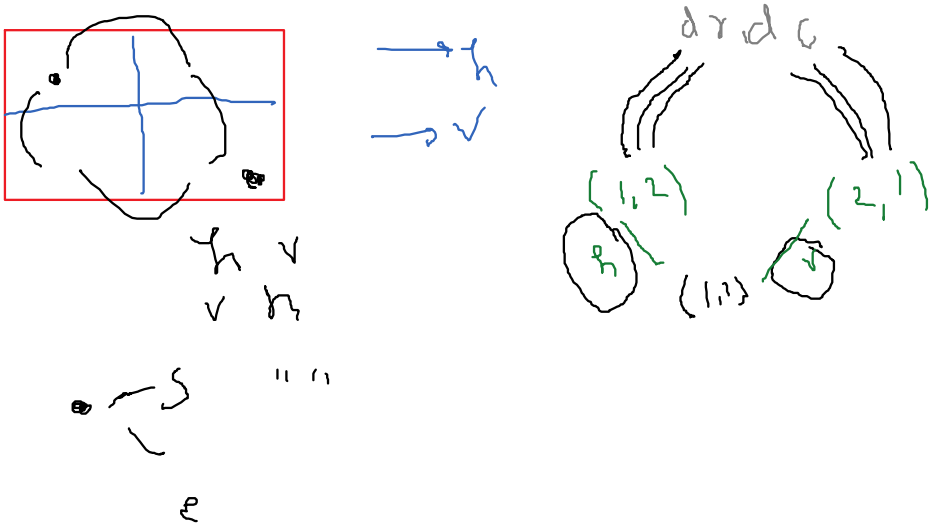
```
vector<string> gss(string s)
{
    if(s.size() == 0)
    {
        vector<string> ans;
        ans.push_back("");
        return ans;
    }
    //taking first char;
    char ch = s[0] ;
    string rest = s.substr(1);
    vector<string> ans = gss(rest);
    vector<string> temp;
    for(string ss : ans)
    {
        temp.push_back("" + ss);
    }
    for(string ss : ans)
    {
        temp.push_back(ch + ss);
    }
    return temp;
}
```



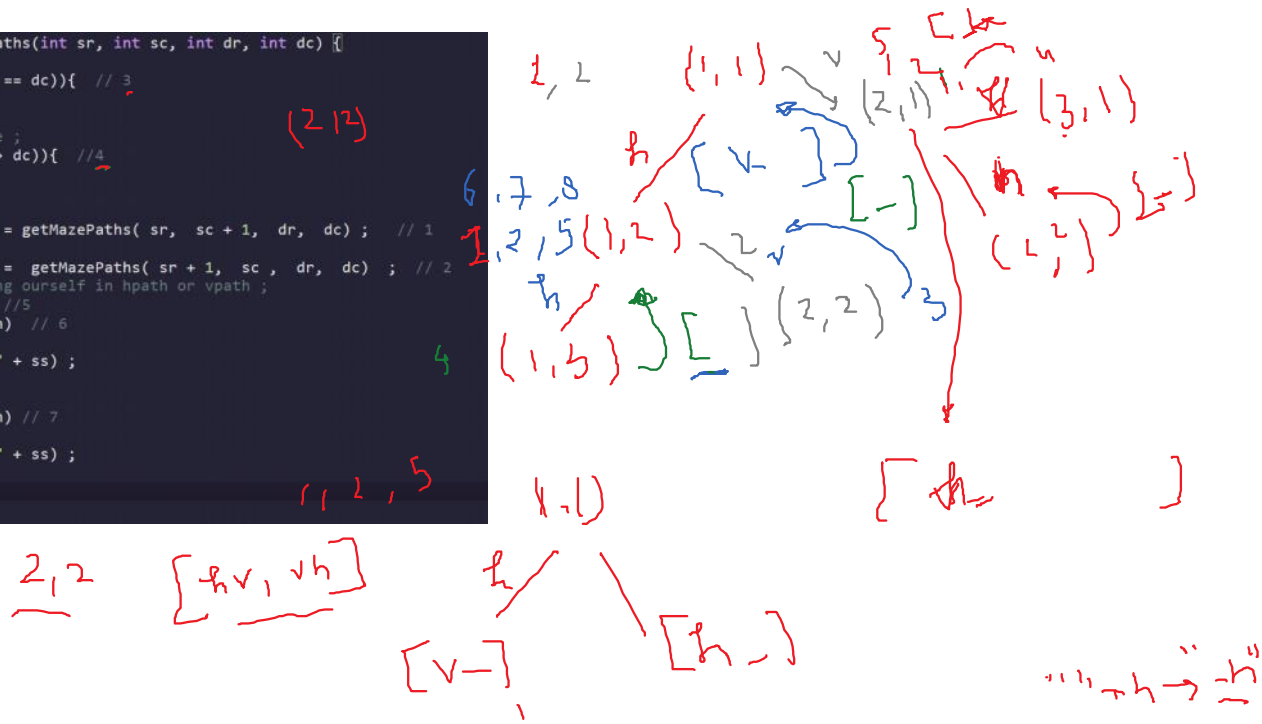




Get maze path



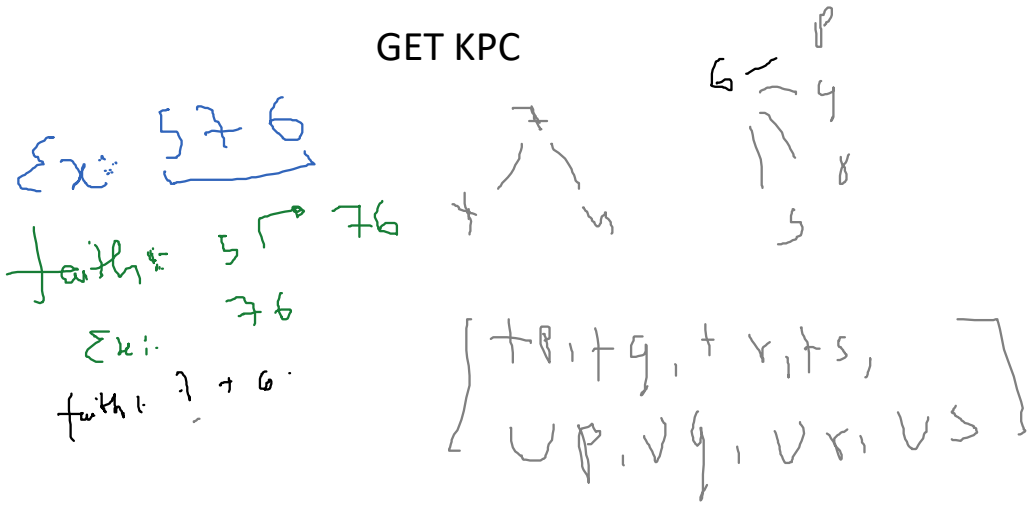
```
vector <string> getMazePaths(int sr, int sc, int dr, int dc) {
    // base case;
    if((sr == dr) && (sc == dc)){  // 3
        return {""};
    }
    // negative base case ;
    if((sr > dr) || (sc > dc)){  //4
        return {};
    }
    // horizontal
    vector<string> hpath = getMazePaths( sr,  sc + 1,  dr,  dc) ;   // 1
    // verticle;
    vector<string> vpath =  getMazePaths( sr + 1,  sc ,  dr,  dc)  ;  // 2
    // faith we are adding ourself in hpath or vpath ;
    vector<string> ans;  //5
    for(string ss : hpath)  // 6
    {
        ans.push_back("h" + ss) ;
    }

    for(string ss : vpath) // 7
    {
        ans.push_back("v" + ss) ;
    }
    return ans; // 8
}
```



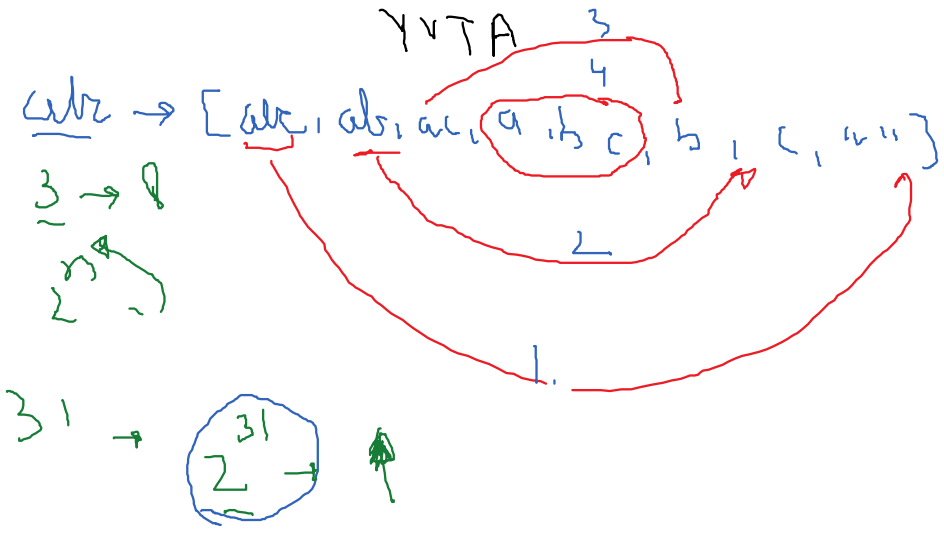
```
vector<string> get_stair_paths(int n)
{
    if(n < 0){
        return {};
    }
    else if( n == 0) {
        return {""};
    }

    vector<string> one = get_stair_paths(n - 1);
    vector<string> two = get_stair_paths(n - 2);
    vector<string> three = get_stair_paths(n - 3);

    vector<string> ans ;
    for(string x : one)
    {
        ans.push_back("1" + x);
    }

    for(string x : two)
    {
        ans.push_back("2" + x);
    }

    for(string x : three)
```

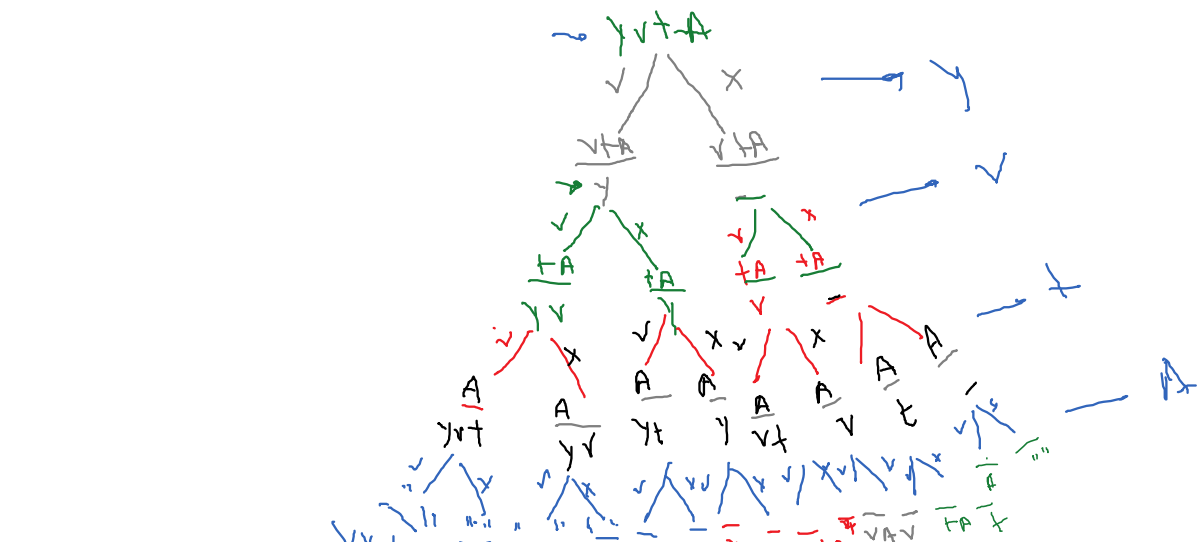
```
for(string x : two)
{
    ans.push_back("2" + x);
}

for(string x : three)
{
    ans.push_back("3" + x);
}
return ans ;
}
```

GET KPC



Print subsequence

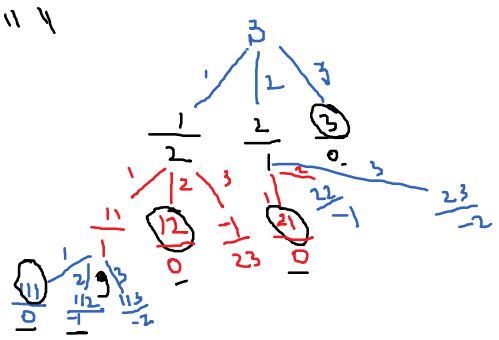




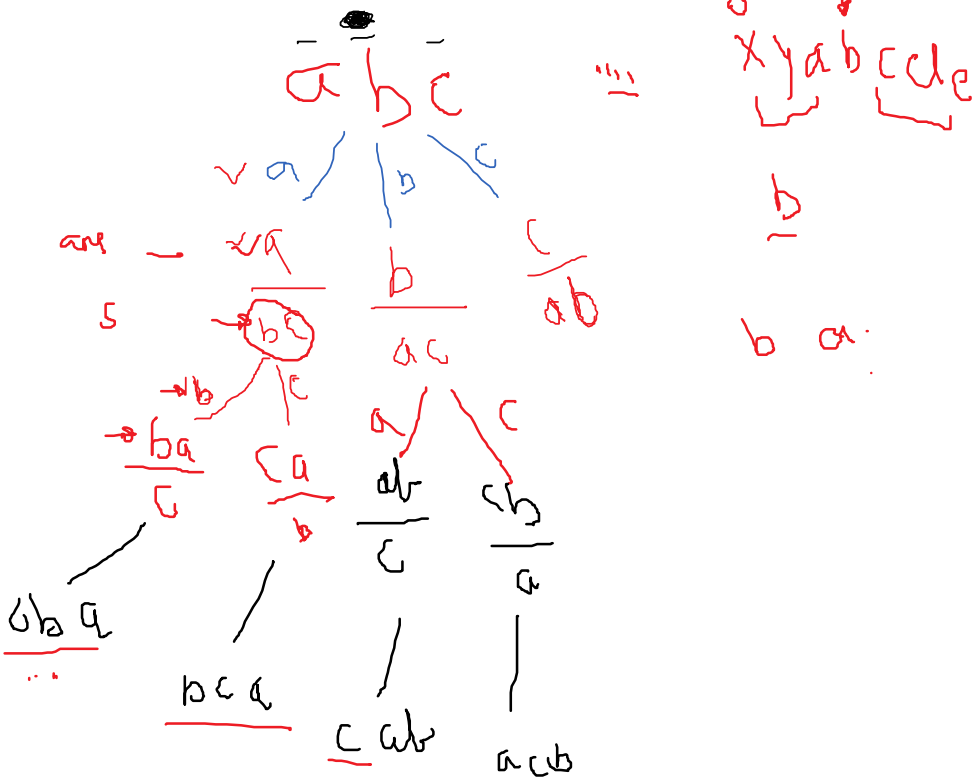
Recursion 5

23 August 2022 14:15

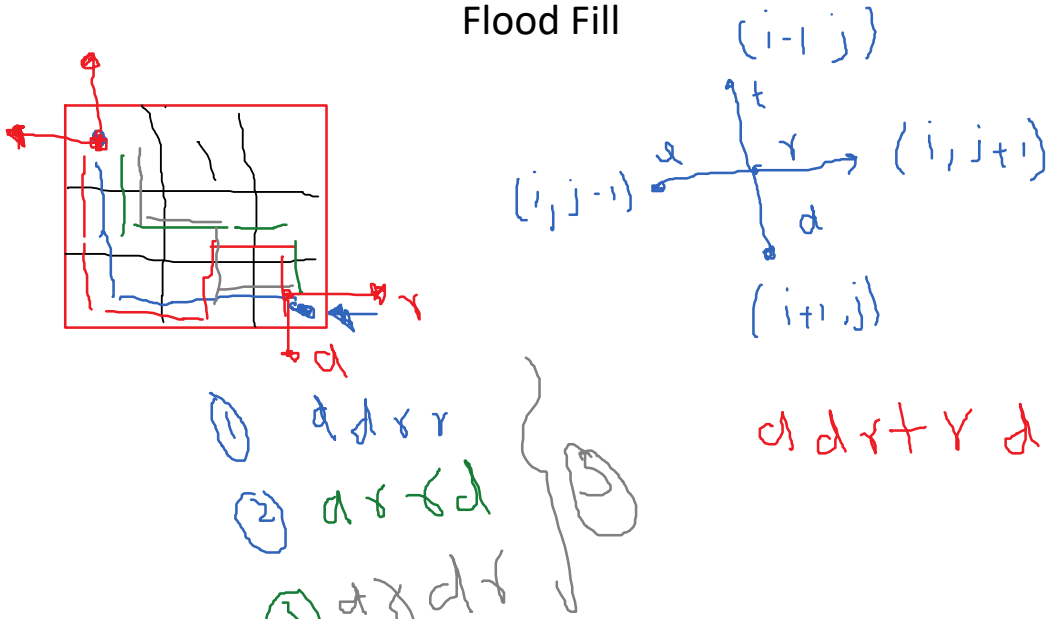
PRINT STAIRS PATH



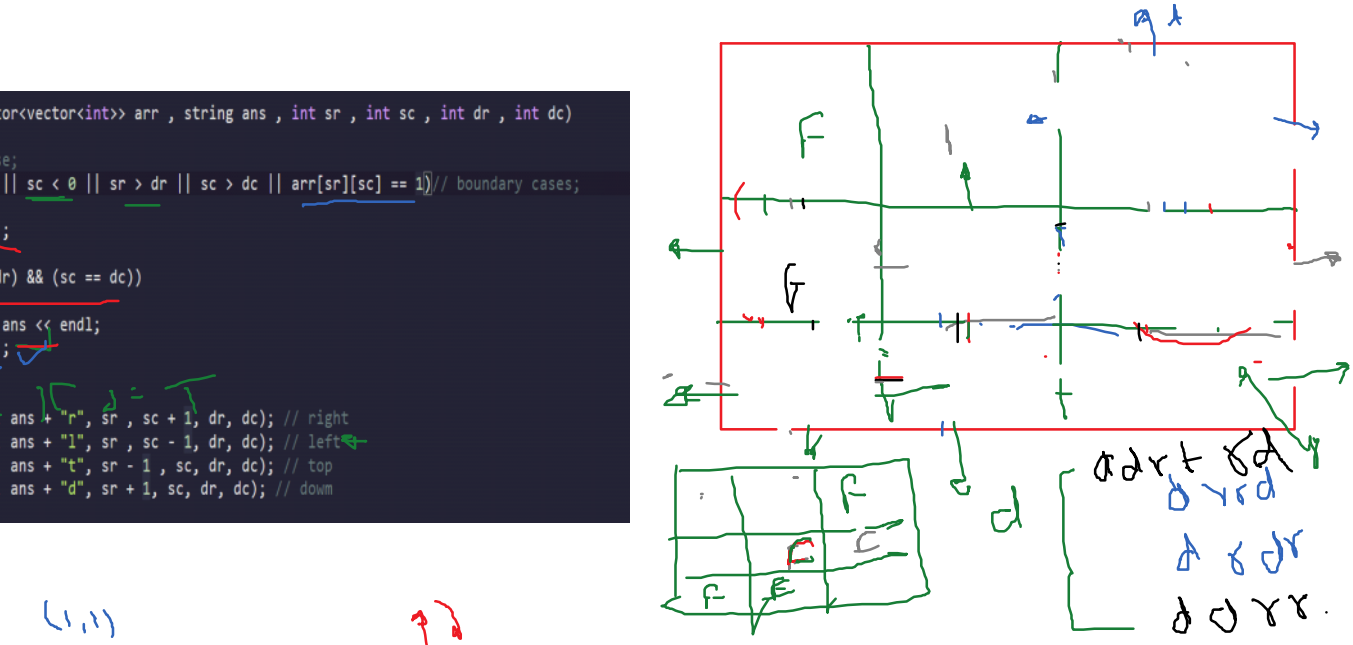
Print Permutations

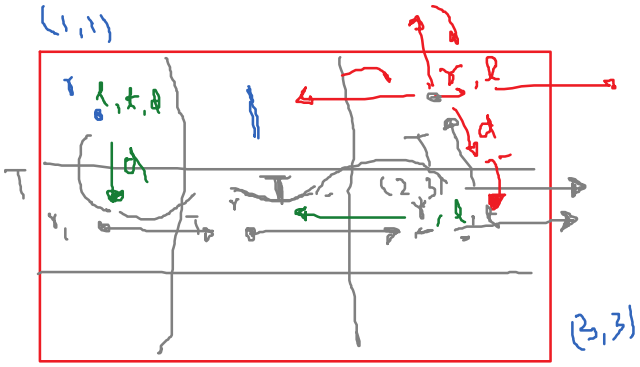


Flood Fill

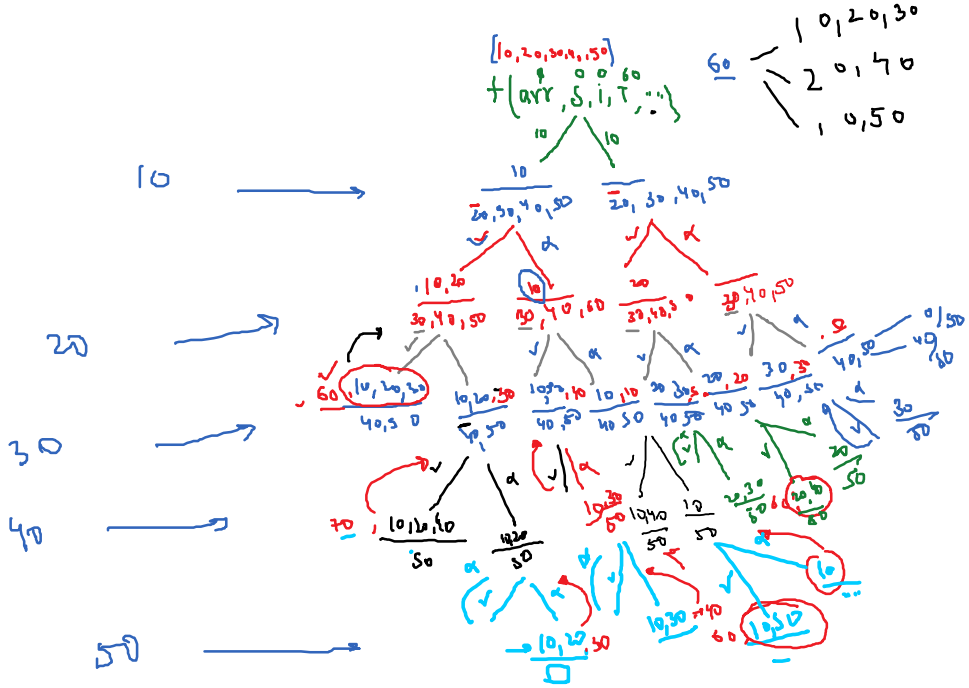


```
void solve(vector<vector<int>> arr , string ans , int sr , int sc , int dr , int dc)
{
    // base case;
    if(sr < 0  || sc < 0 || sr > dr || sc > dc || arr[sr][sc] == 1)// boundary cases;
    {
        return ;
    }
    if((sr == dr) && (sc == dc))
    {
        cout<< ans << endl;
        return ;
    }
    solve(arr , ans + "r", sr , sc + 1, dr, dc); // right
    solve(arr , ans + "l", sr , sc - 1, dr, dc); // left
    solve(arr , ans + "t", sr - 1 , sc, dr, dc); // top
    solve(arr , ans + "d", sr + 1, sc, dr, dc); // dowm
}
```





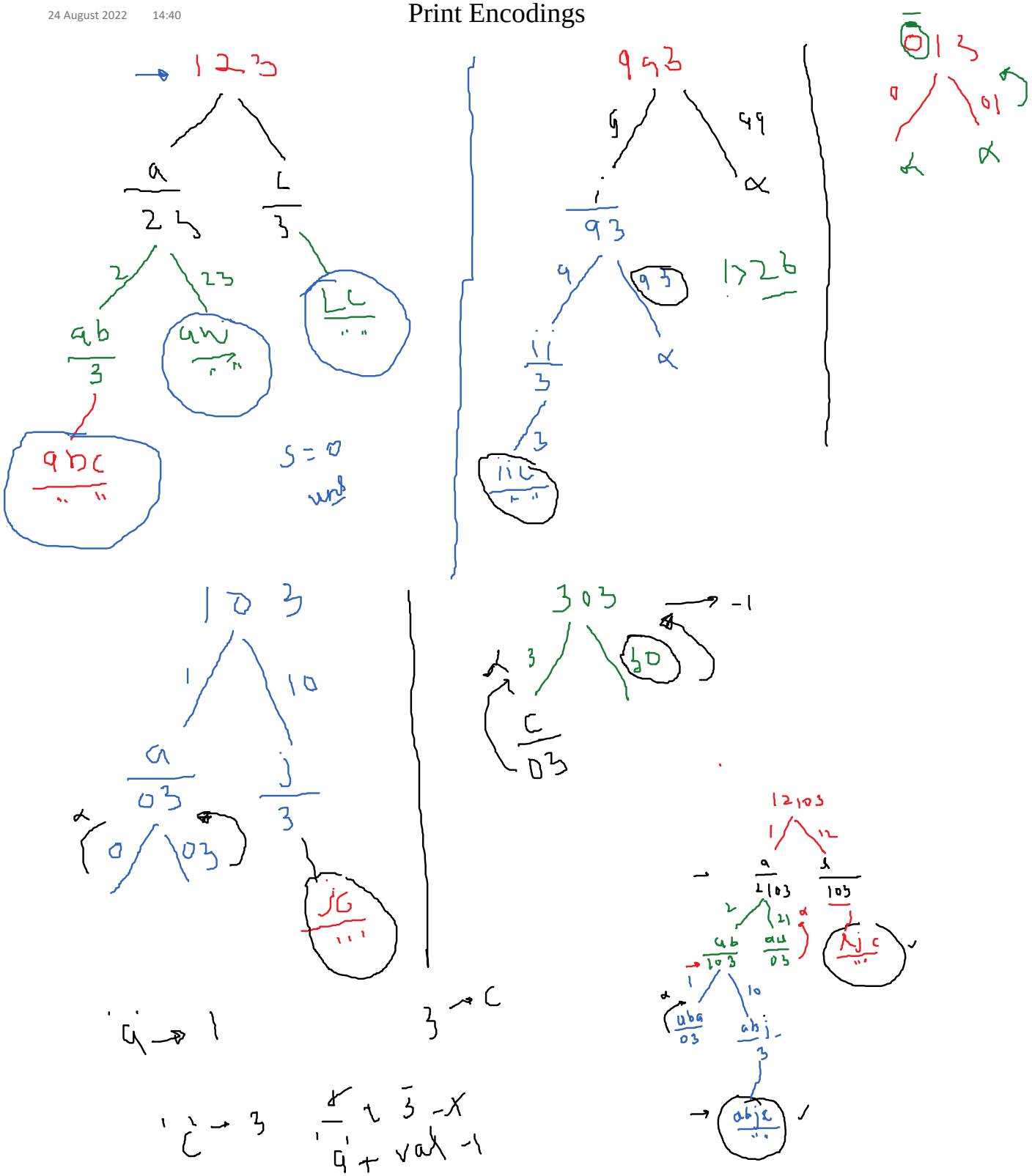
Target sum Subset



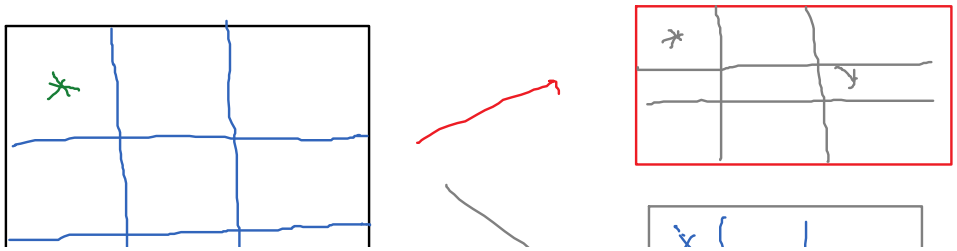
Recursion 6

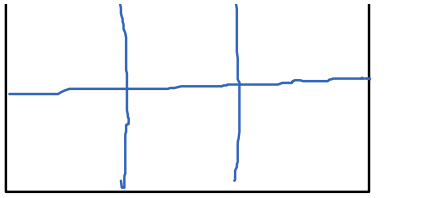
24 August 2022 14:40

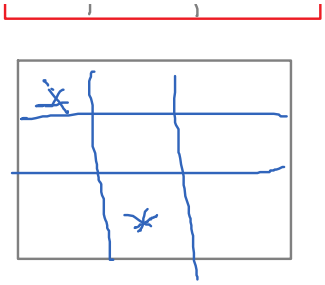
Print Encodings

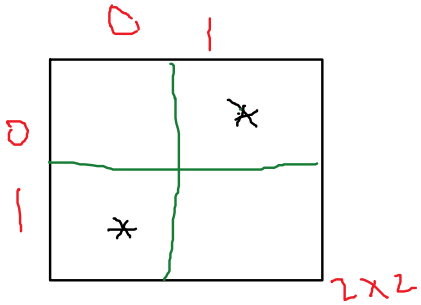


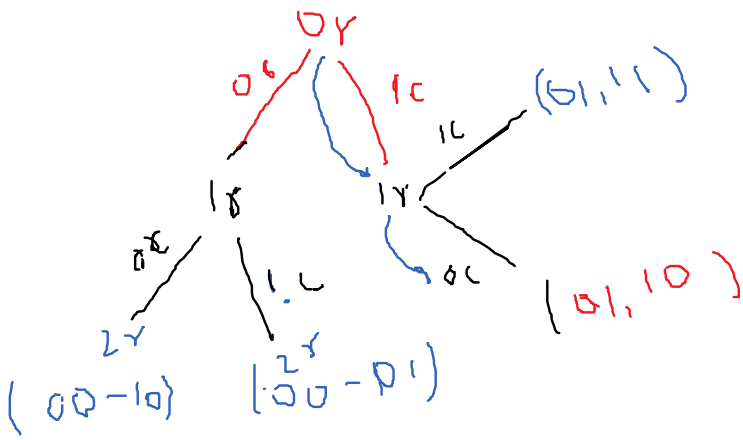
N queen Problem

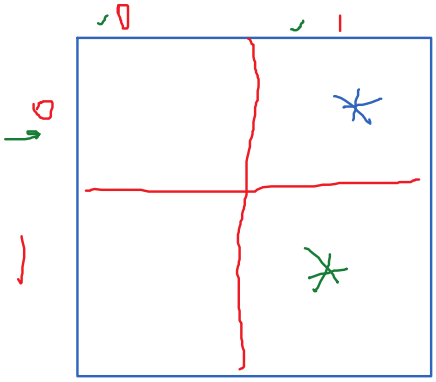


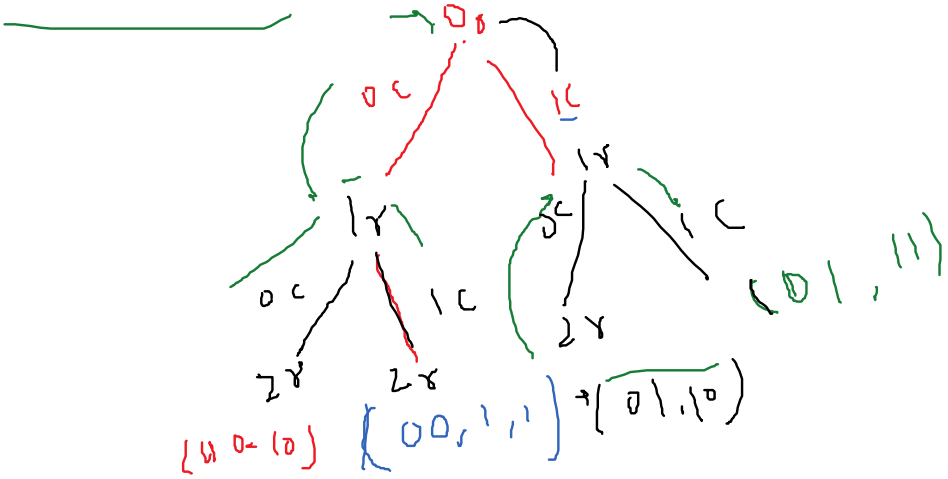




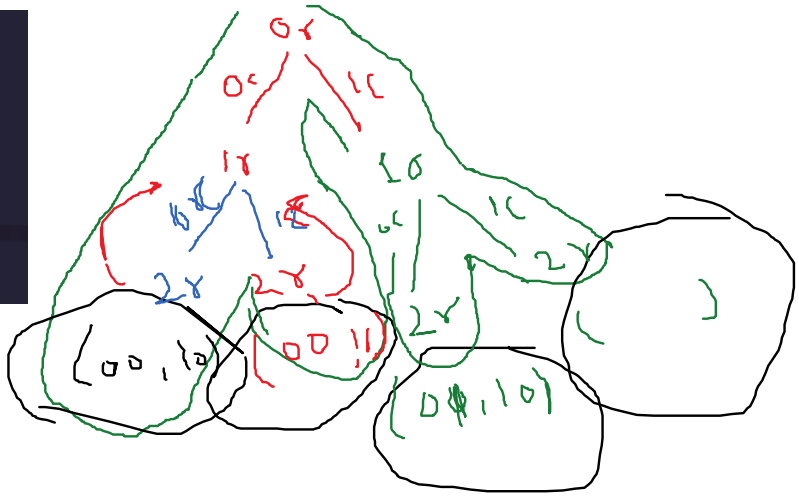


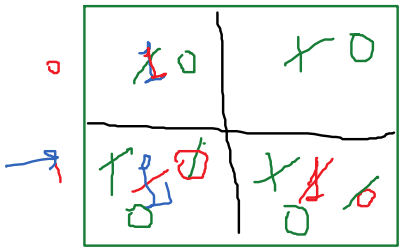


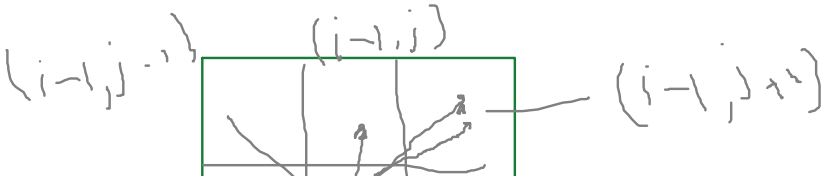


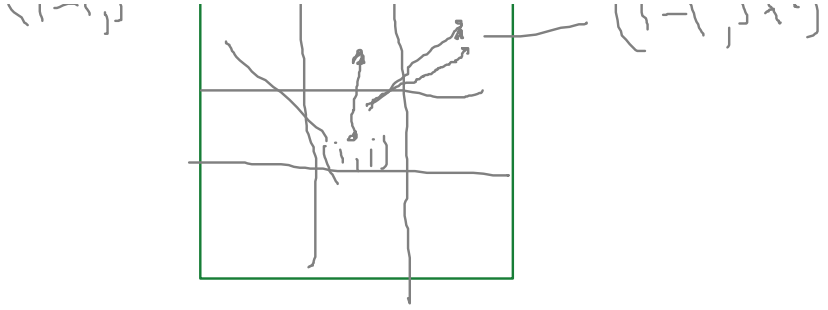


```
void printNQueens(vector<vector<int>> chess,string ans ,int row)
{
  if(row == chess.size())
  {
    cout<< ans << ". "<< end;
    return ;
  }

  // loop through the col;
  for(int col = 0 ; col < chess.size() ; col ++)
  {
    chess[row][col] = 1;
    printNQueens(chess , ans + to_string(row) + "-" + to_string(col)+ "," , row + 1);
    chess[row][col] = 0;
  }
```









Subset 1

78. Subsets

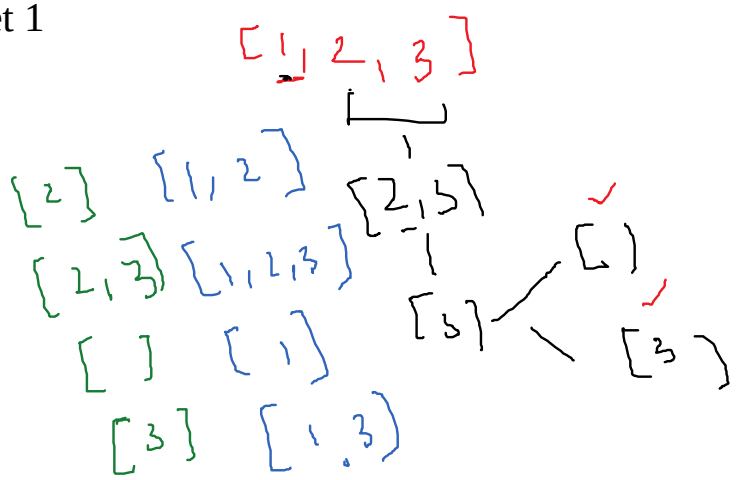
Medium 11646 169 Add to List Share

Given an integer array `nums` of **unique** elements, return *all possible subsets (the power set)*.

The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]
```

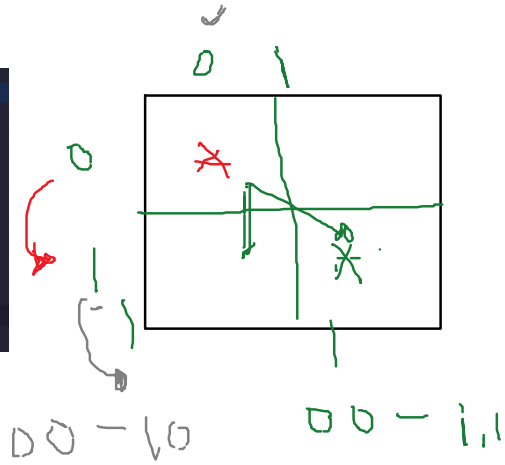


Recursion 7

01 September 2022 14:06

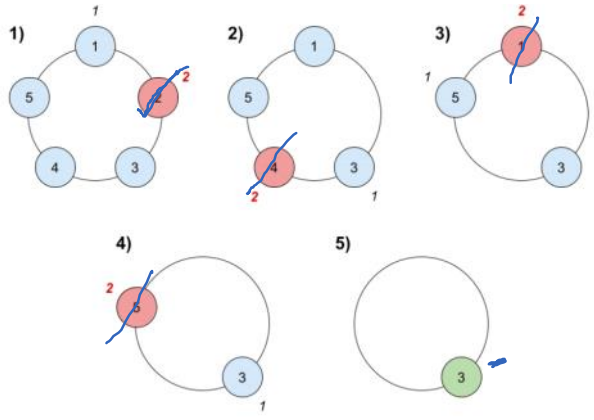
N Queen Problem

```
{
  if (row == chess.size()) {···
  }
  //loop through columns
  for (int col = 0; col < chess.size(); col++) {
    if (isQueenSafe(row, col, chess) == true)
    {
      chess[row][col] = 1;
      printNQueens(chess, qsf + to_string(row) + "-" + to_string(col) + ", ", row + 1);
      chess[row][col] = 0;
    }
  }

}
```

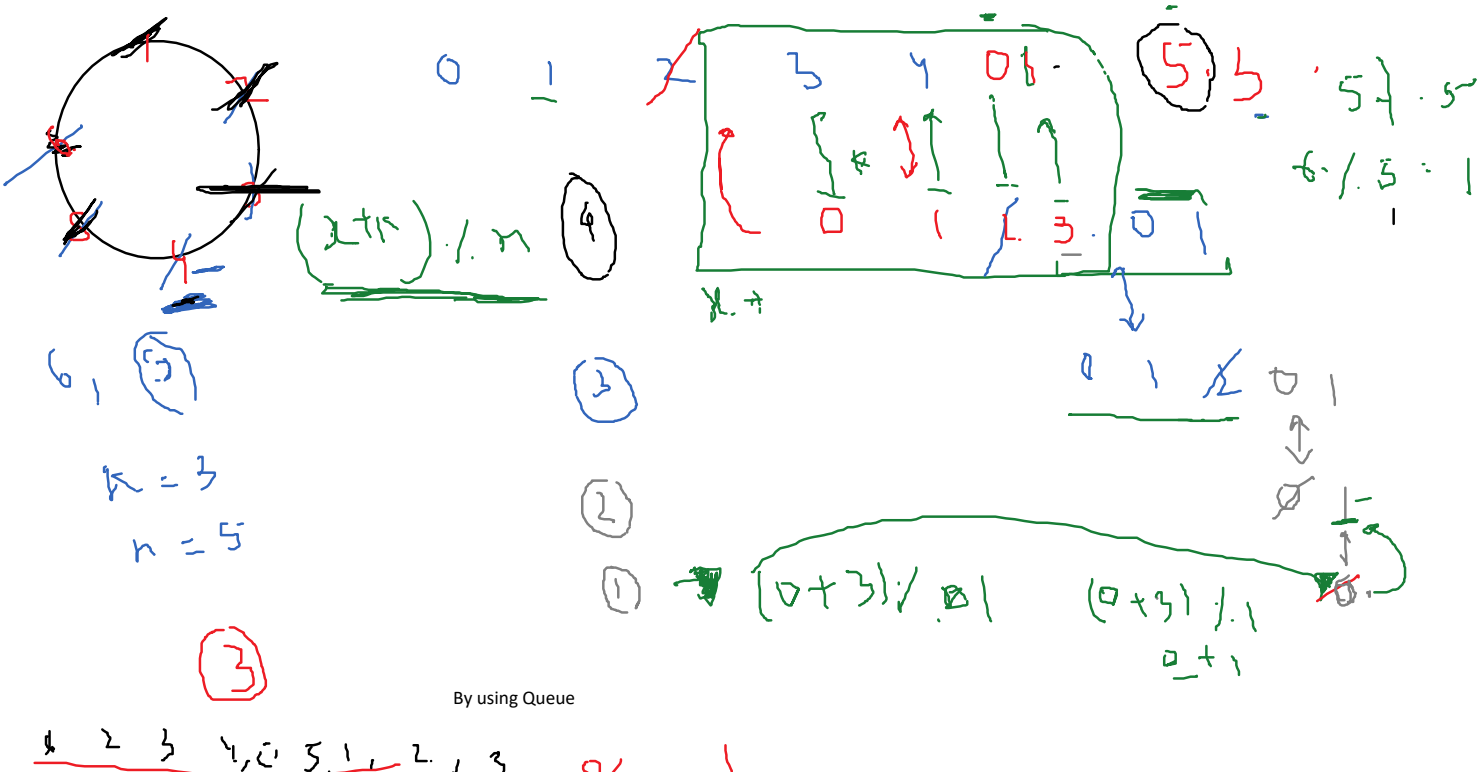


Joseph Problem

Game of Death

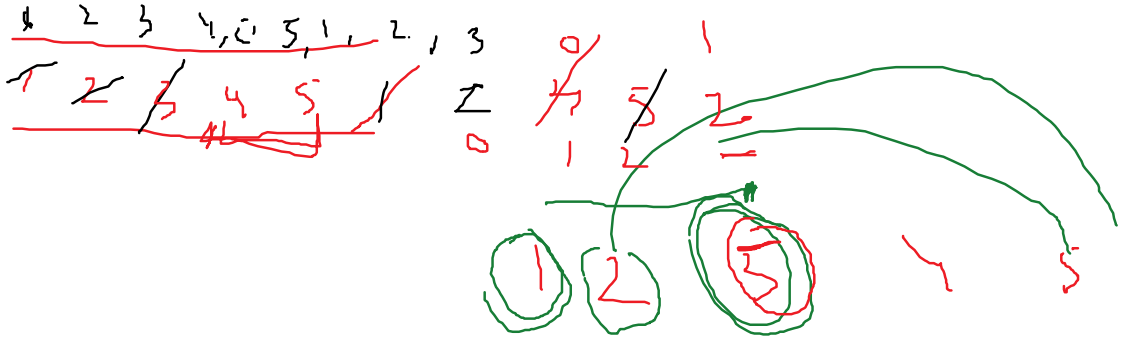


Input: n = 5, k = 2

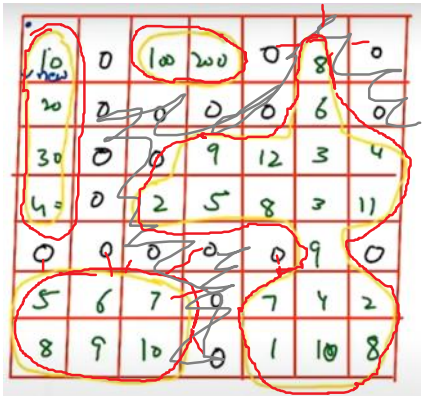


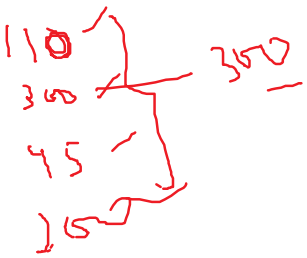
By using Queue

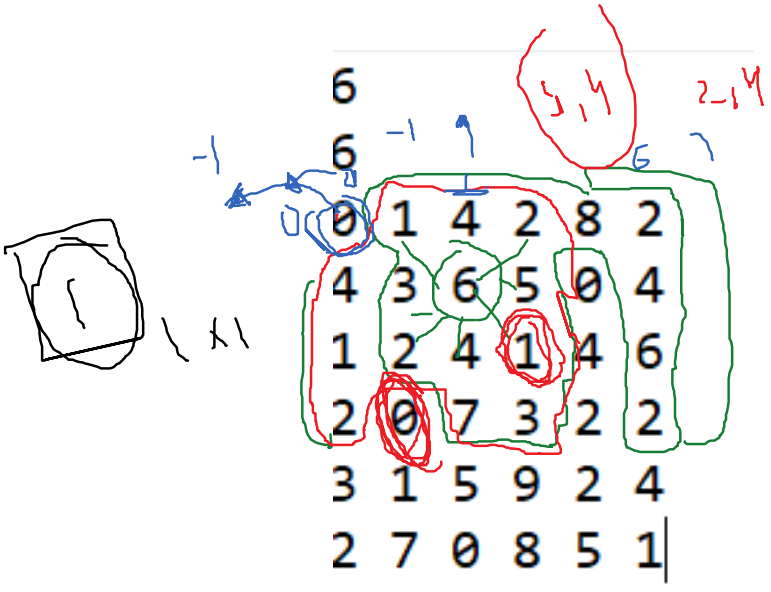
By using Queue

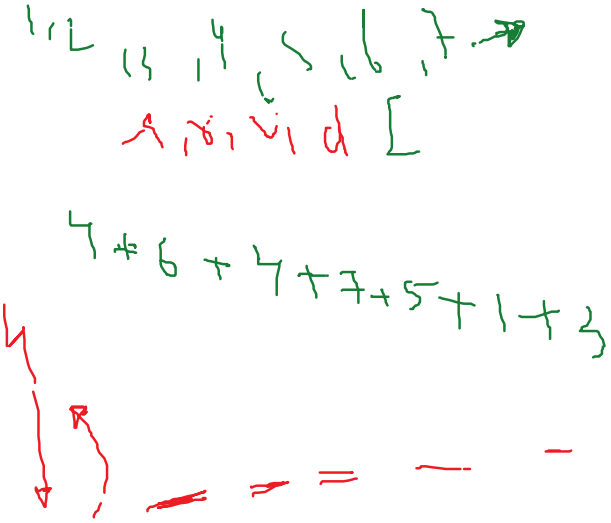


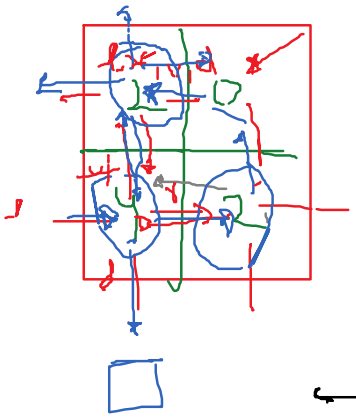
Gold Mine 2

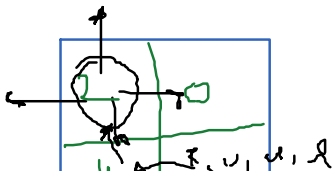


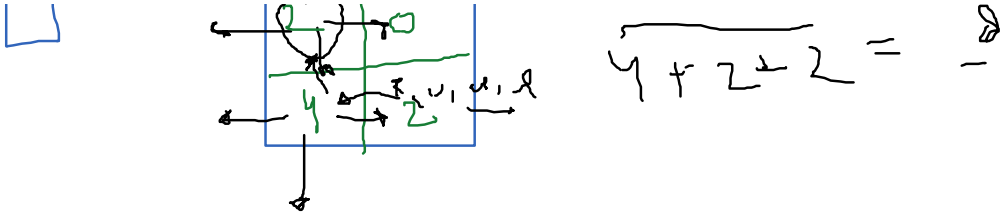


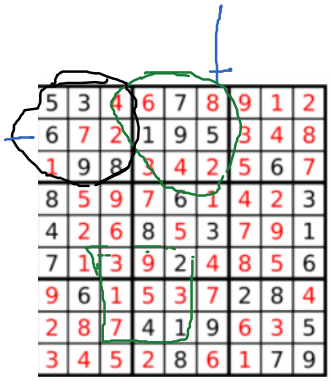




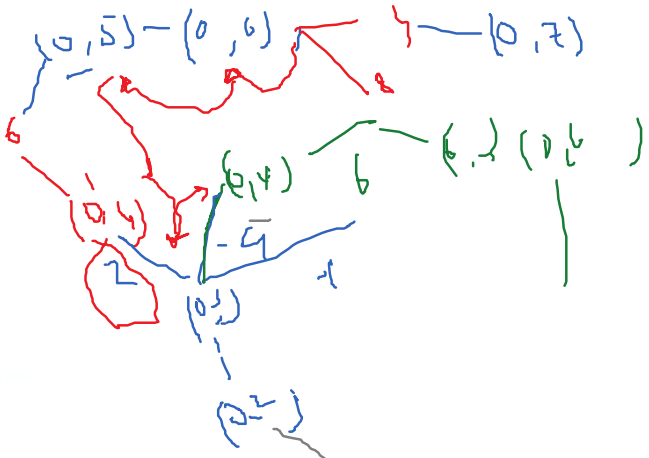


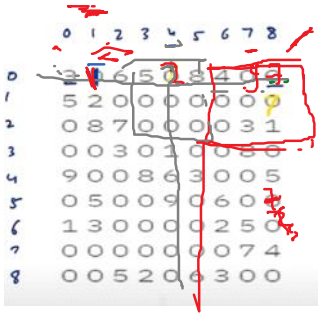






Suduko Solver

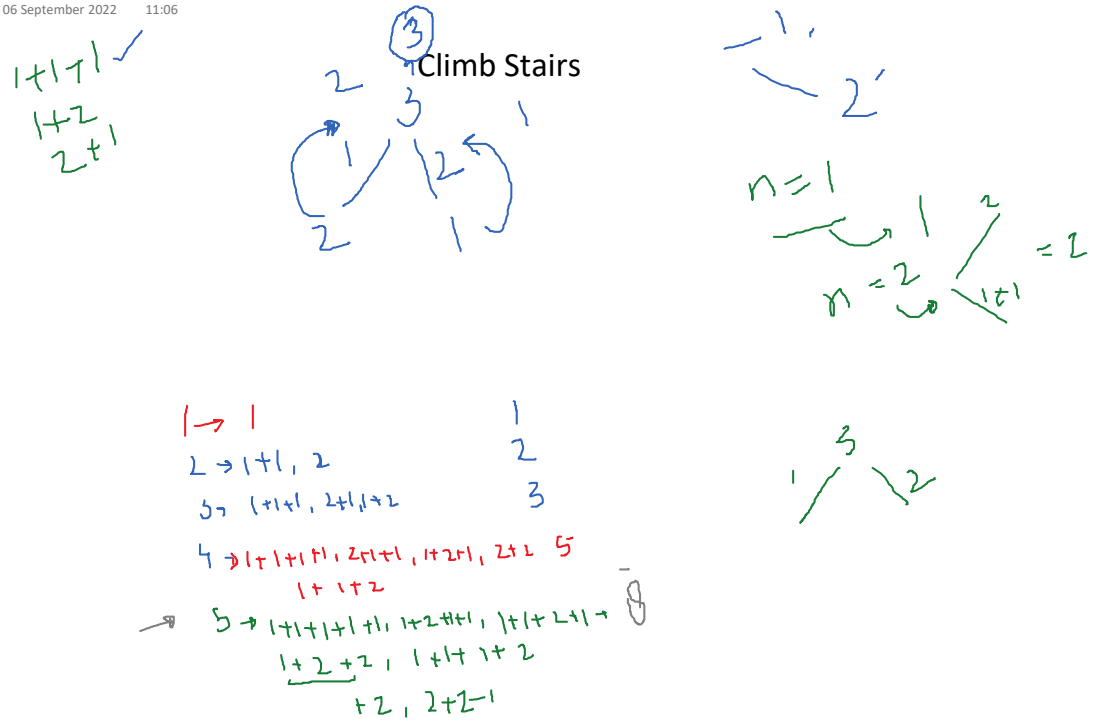




Recursion 8

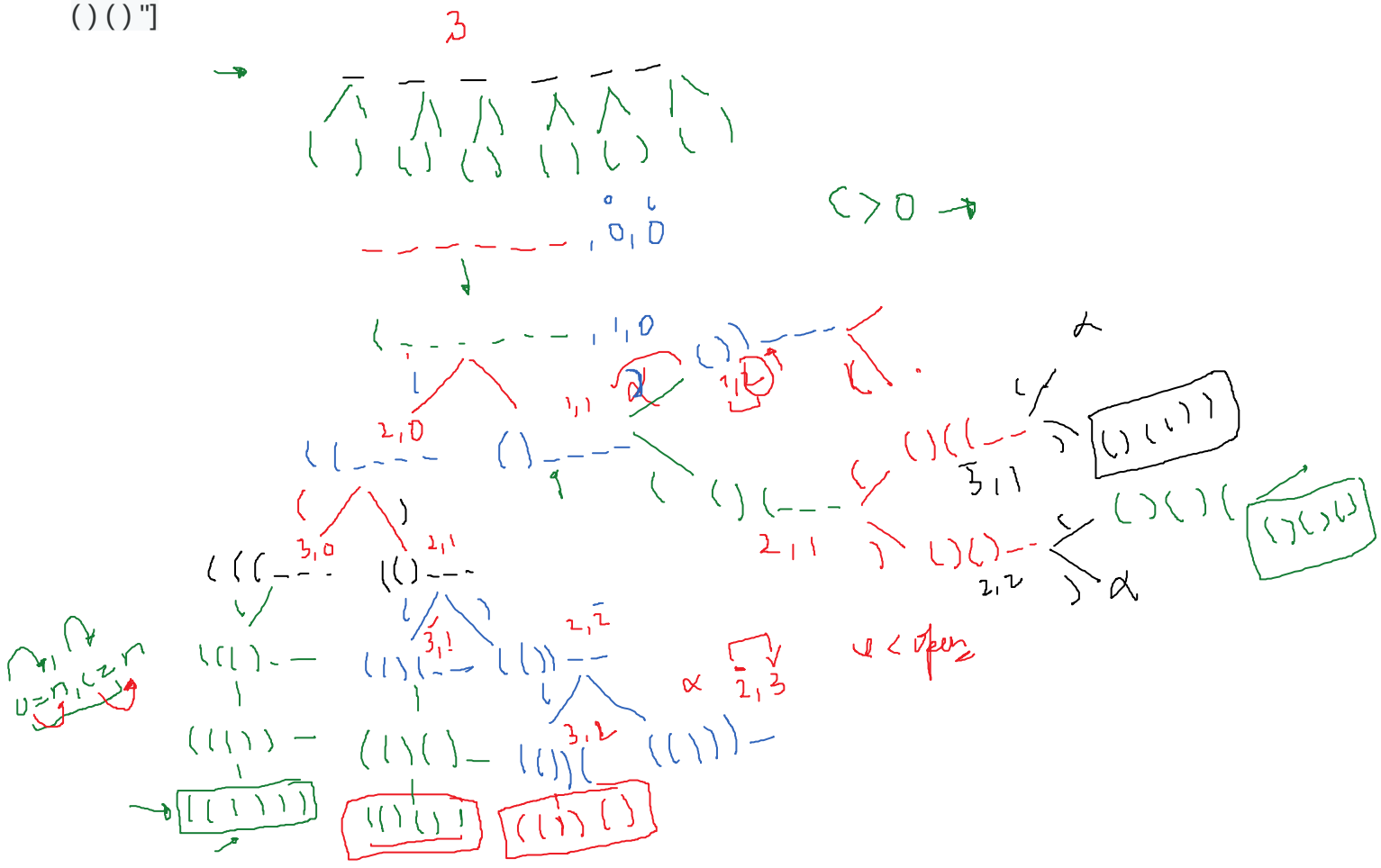
06 September 2022 11:06

Climb Stairs

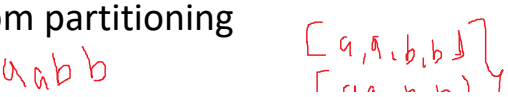


Generate Parenthese

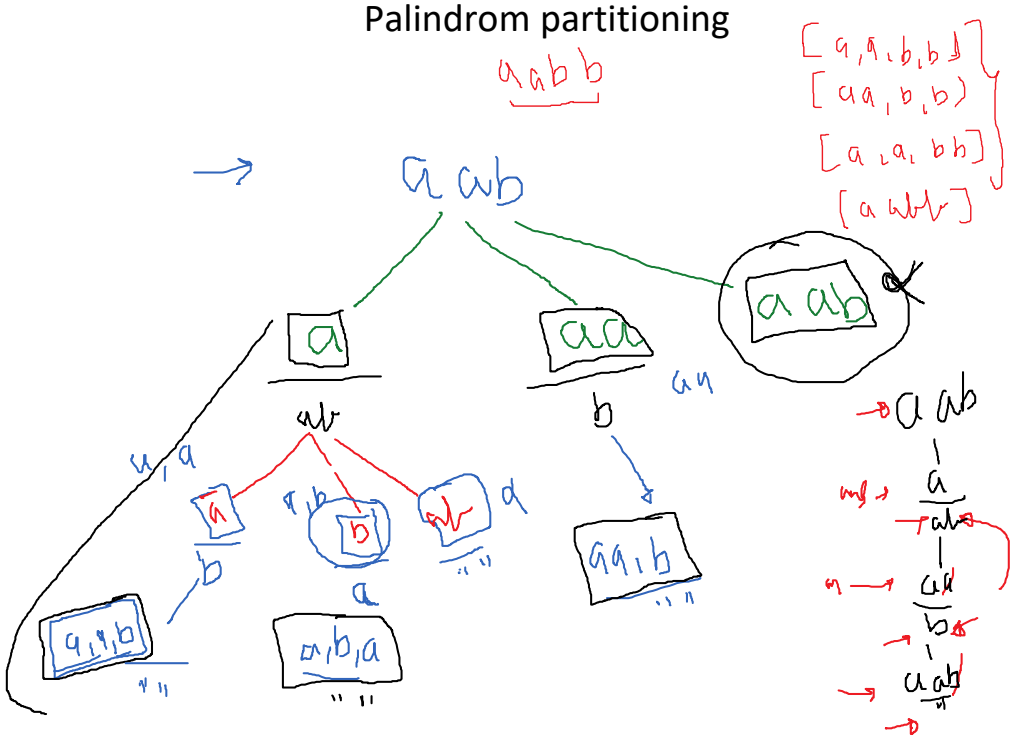
Input: n = 3
Output: ["((( ))) ", "(( ) ( )) ","(( )) ( ) ","( ) (( ))","( ) ( ) ( ) "]



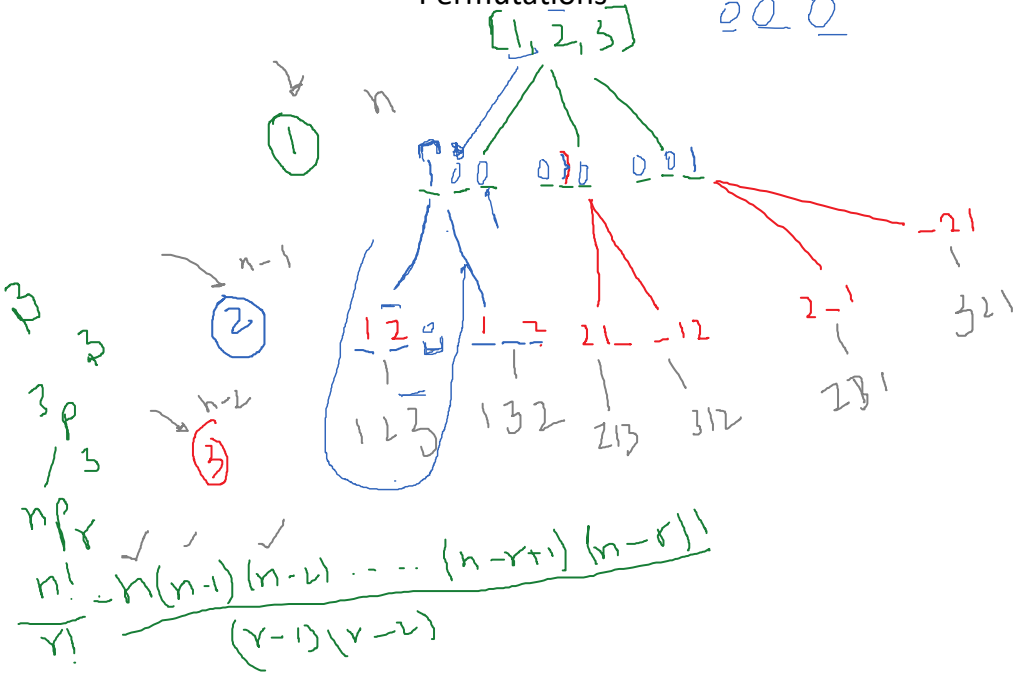
Palindrom partitioning



Palindrom partitioning



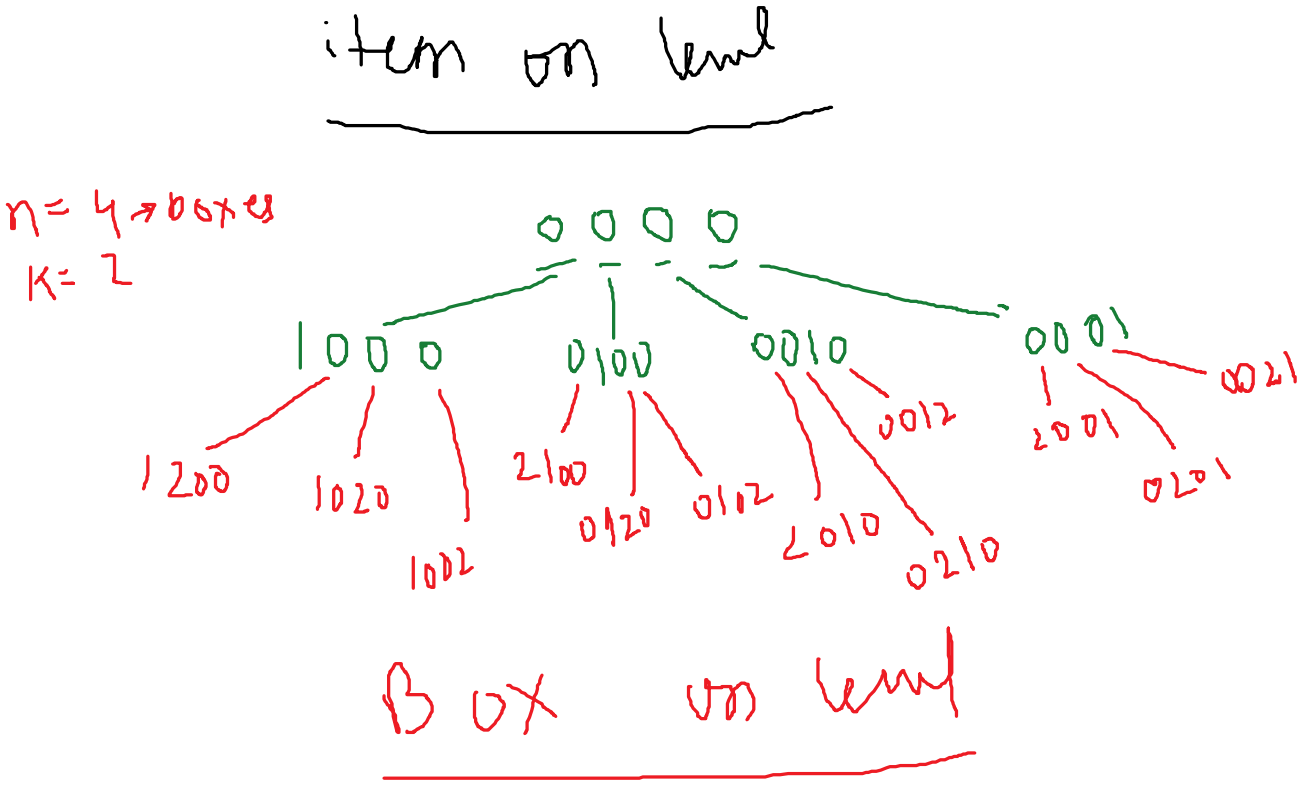
Permutations

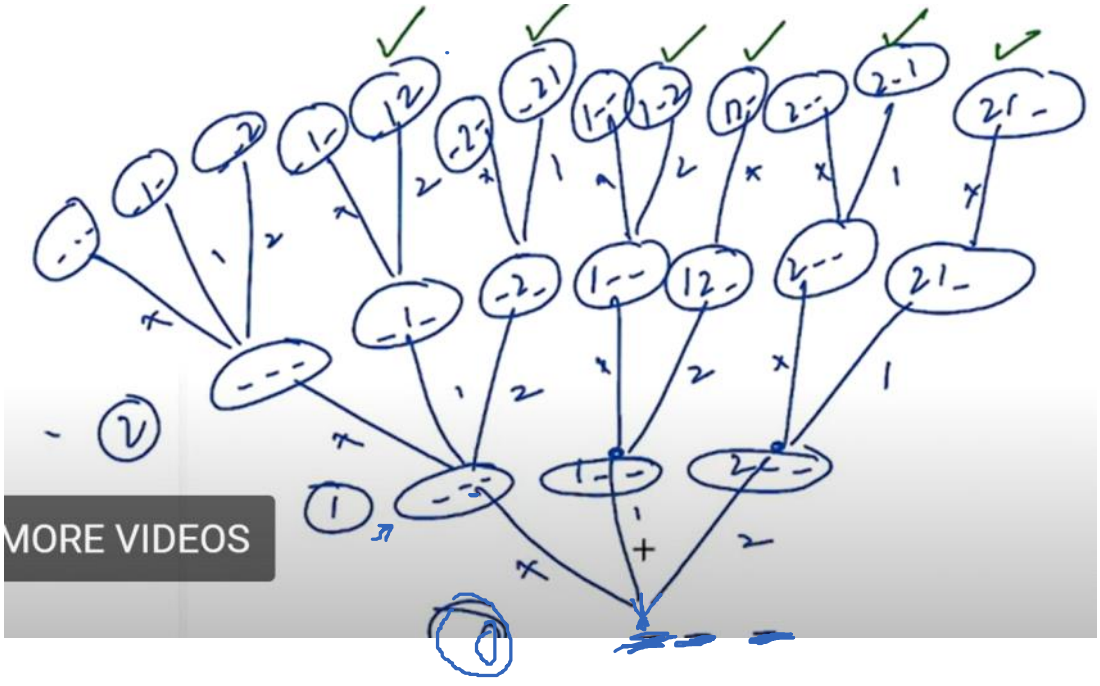


Recursion 9

07 September 2022 14:19

Permutation 1

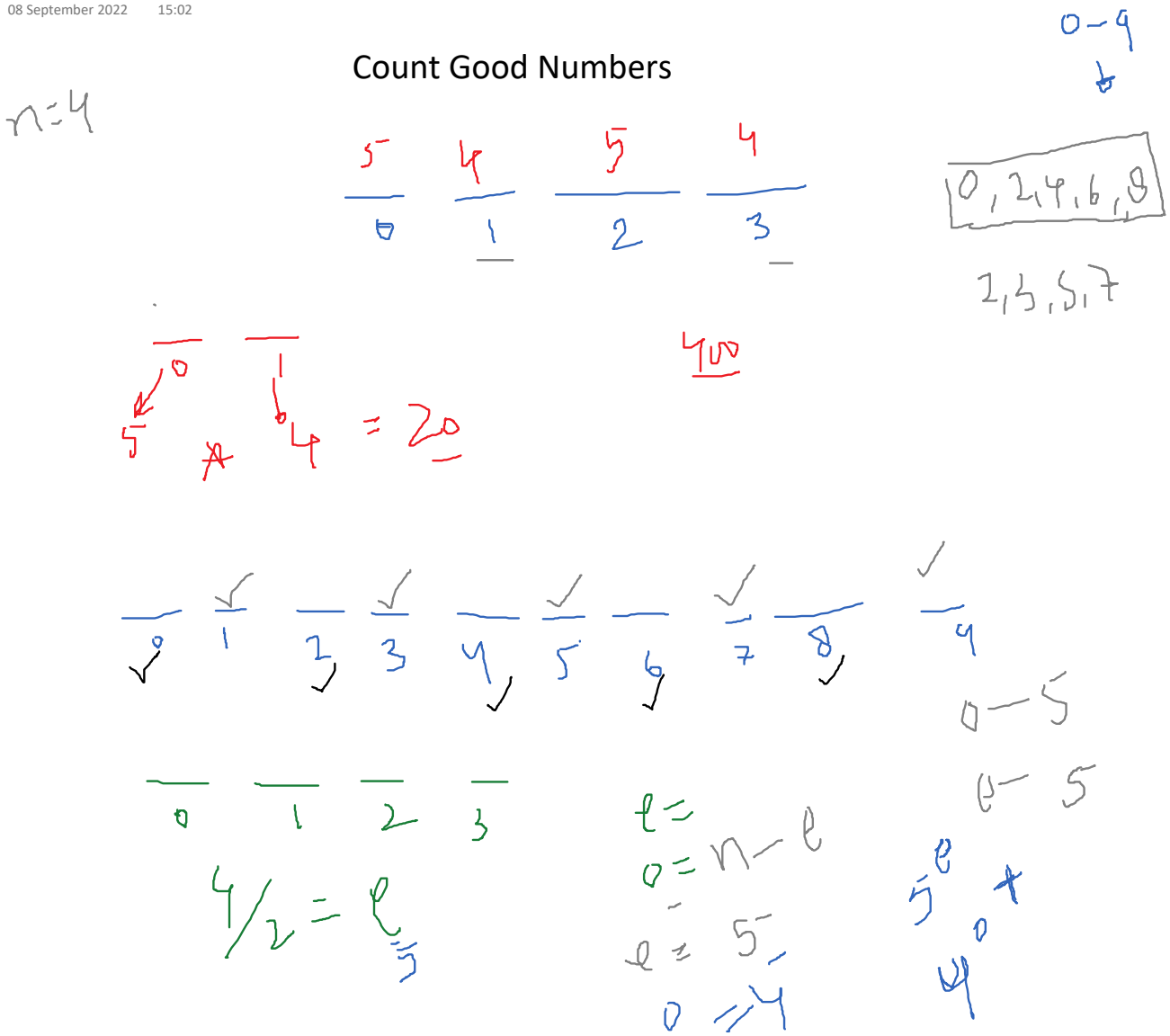




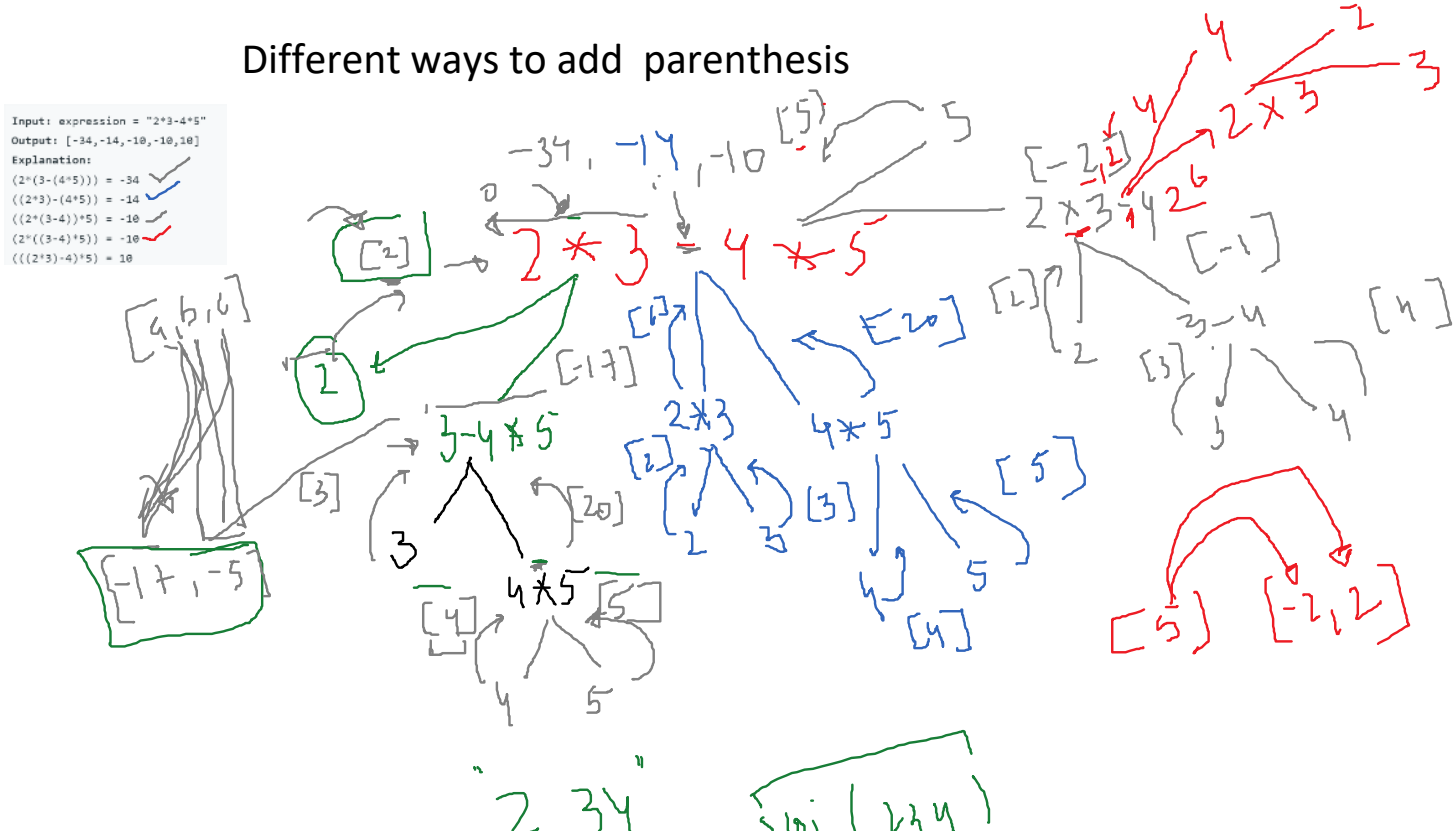
Recursion 10

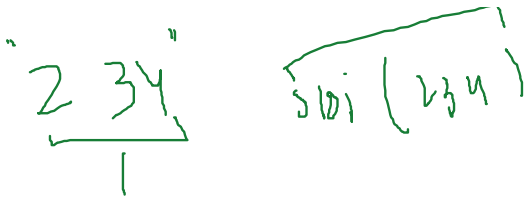
08 September 2022 15:02

Count Good Numbers



Different ways to add parenthesis




"234"
sbi (234)