4. Recursion 2.

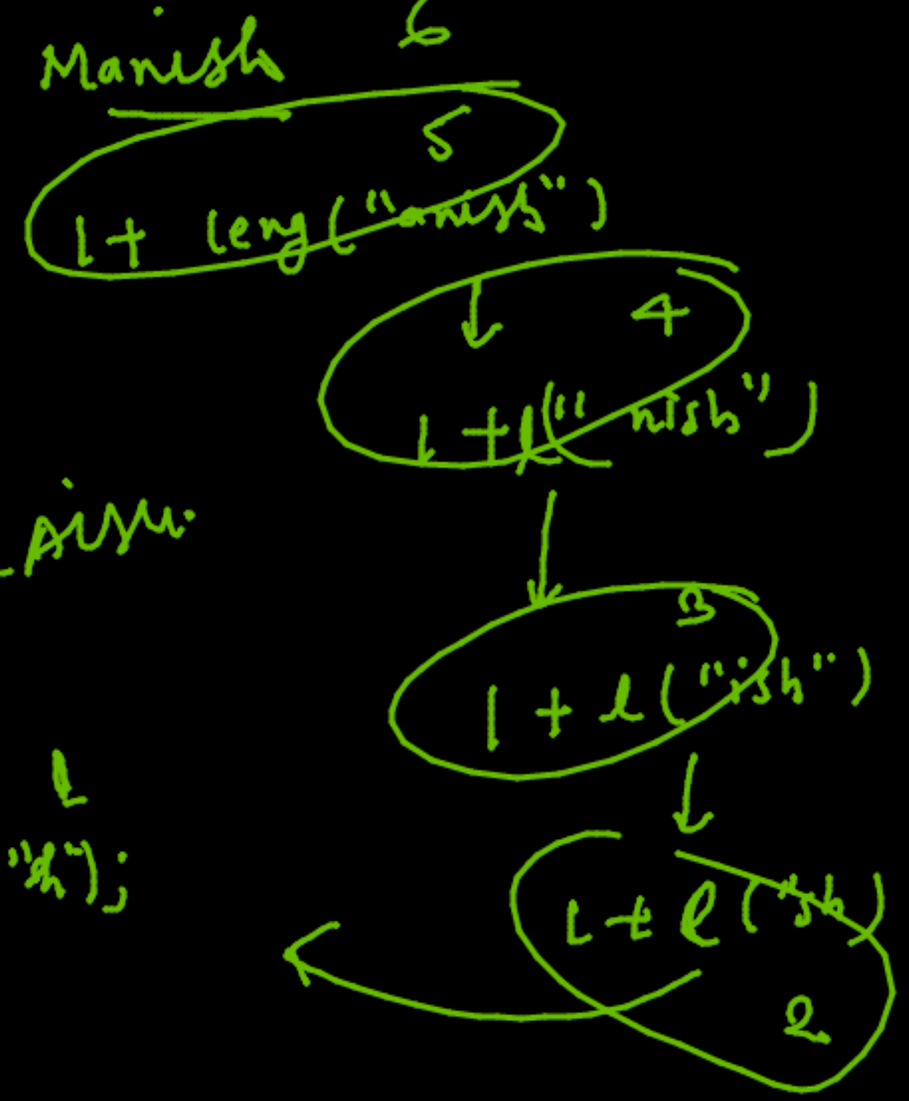
Important question

(1) check AB
(2) merge sort
(3) quick sort
(4) All permutation * (backtrack)
(5) subset of array (backtrack)
(6) print subset sum to K
(7) print all code - string (Best)
(8) print All keypad combination

length of string

```
int length(char input[]) {
    if(input[0] == '\0')
        return 0;
    return 1 + length(input + 1);
}
```

eg. Manish



Kuch aisa.

Remove duplicate

eg. aabccde
→ abcde

```
void fun(char input[])
{
    if (input[0] == '\0') return;
    if (input[0] == input[1]) {
        int i=0, n=strlen(input);
        for (; i<n-1; i++)
            input[i] = input[i+1];
        input[i] = '\0';
        fun(input);
    }
    else {
        fun(input+1);
    }
}
```

shifting

i	n	
0		aape
1		aape
2		apple
3		apee

0 < 4
1 < 4
2 < 3
2 < 2
2 < 3

Check ab

Cond"

(i) Start with a

(ii) a followed by a or bb or nothing

(iii) bb followed by a

eg. abb
→ true

abbaabb
→ true

aba
→ false

abbaabb

Staircase

Code

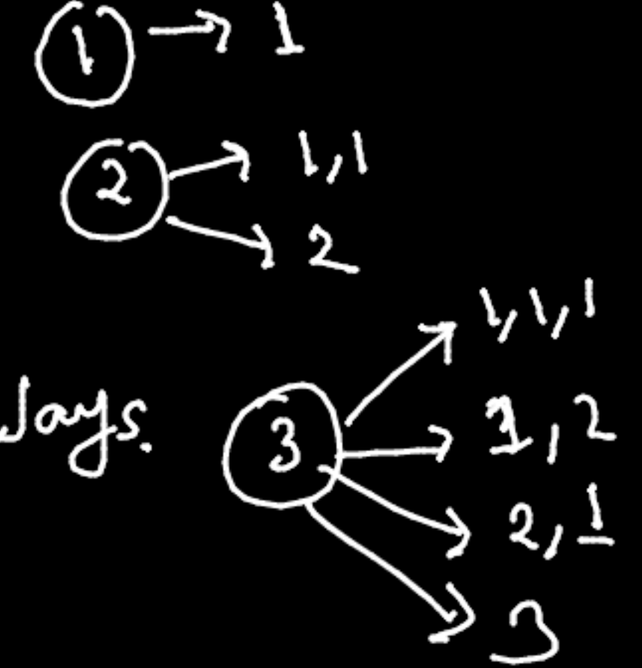
1 or 2 or 3 steps.

```
int ways(int n) {
    if (n <= 2)
        return n;
    if (n == 3)
        return 4;
    return f(n-1) + f(n-2) + f(n-3);
}
```

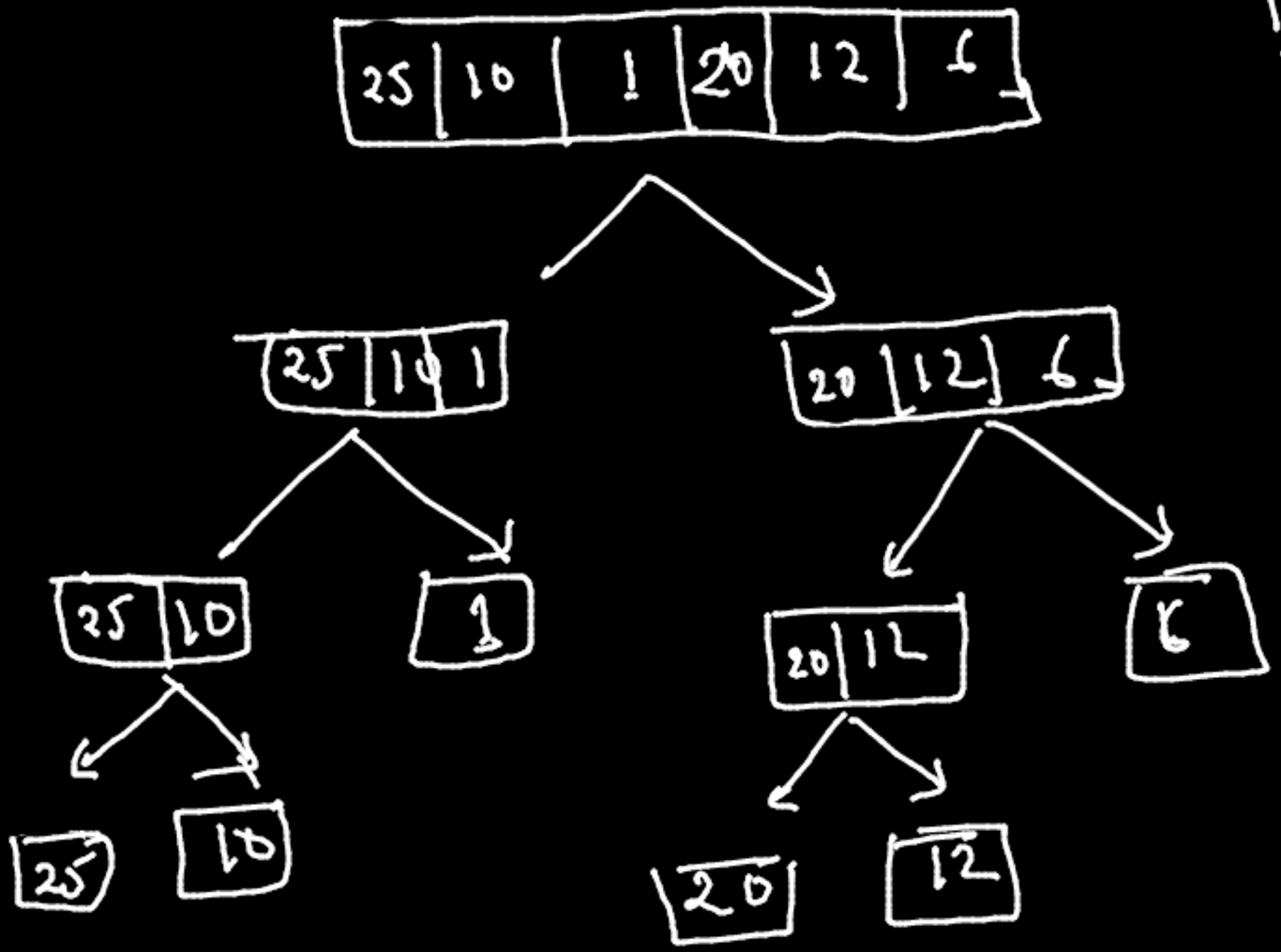
n = 4

1, 1, 1, 1
1, 2, 1
1, 3
1, 1, 2
2, 2
2, 1, 1
3, 1

7 Ways.



Merge sort



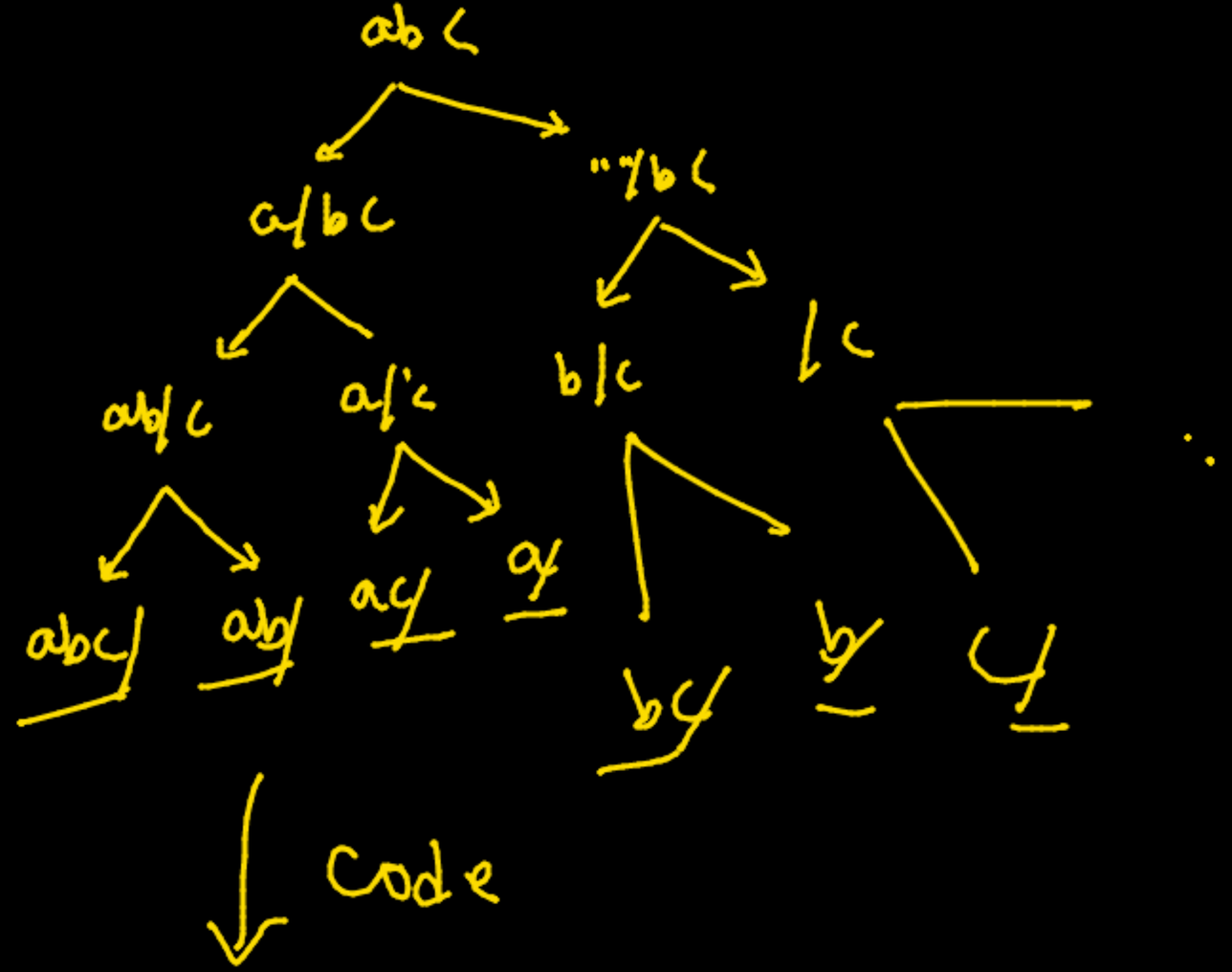
Subsequence → 2^n e.g. $2^3 = 8$

abc

subsequence

- ""
- a
- b
- c
- ab
- ac
- bc
- abc

2^n



```
void subsequence(string input, string output)
{
    if (input.empty()){
        cout << output <<endl;
        return;
    }
    //including that character in output and moving one step further
pick subsequence(input.substr(1), output + input[0]);
    // not including that character in output and moving one step further)
Dont pick subsequence(input.substr(1), output);

}

main() {
    subsequence("abc","");
}
```

Subsequence (Another soln)

abc

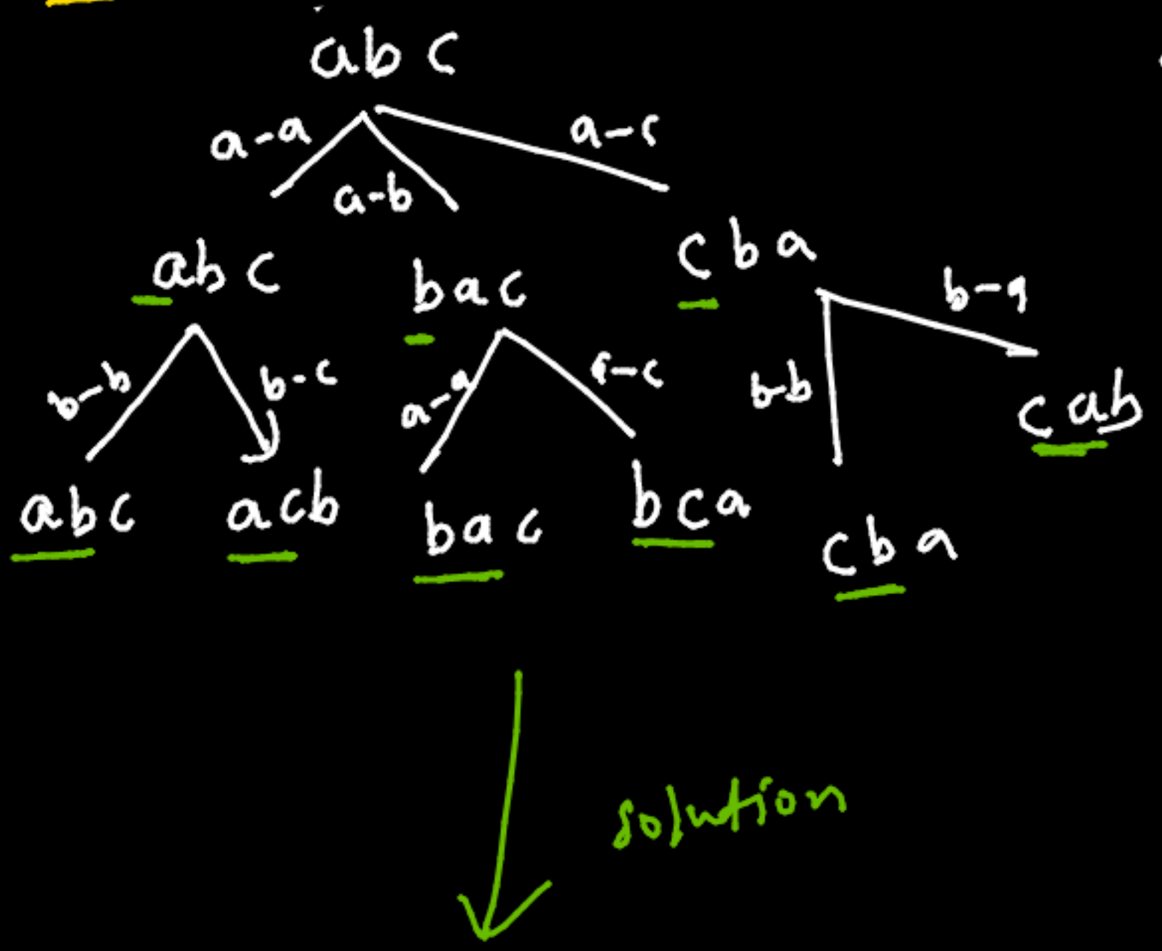
```
int subseq( string input, string output[]){
    if (input.size()==0){
        output[0]="";
        return 1;
    }
    int smalloutputsize= subseq(input.substr(1),output);
    for(int i=0; i<smalloutputsize; i++)
        output[i+smalloutputsize]= input[0]+output[i];

    return 2* smalloutputsize;
}
```

a
ac
ab
abc

Return permutation → n!

eg. abc

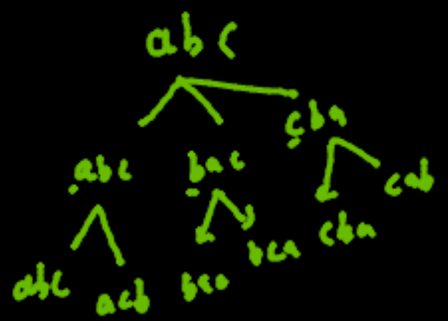
→ abc
→ acb
→ bac
→ bca
→ cab
→ cba

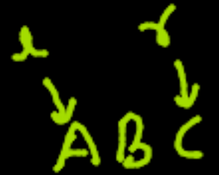


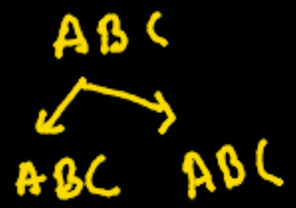
a-b (a swap b)

fix

solution







l	r	i
~~0~~	2	0
~~1~~		1
2		2

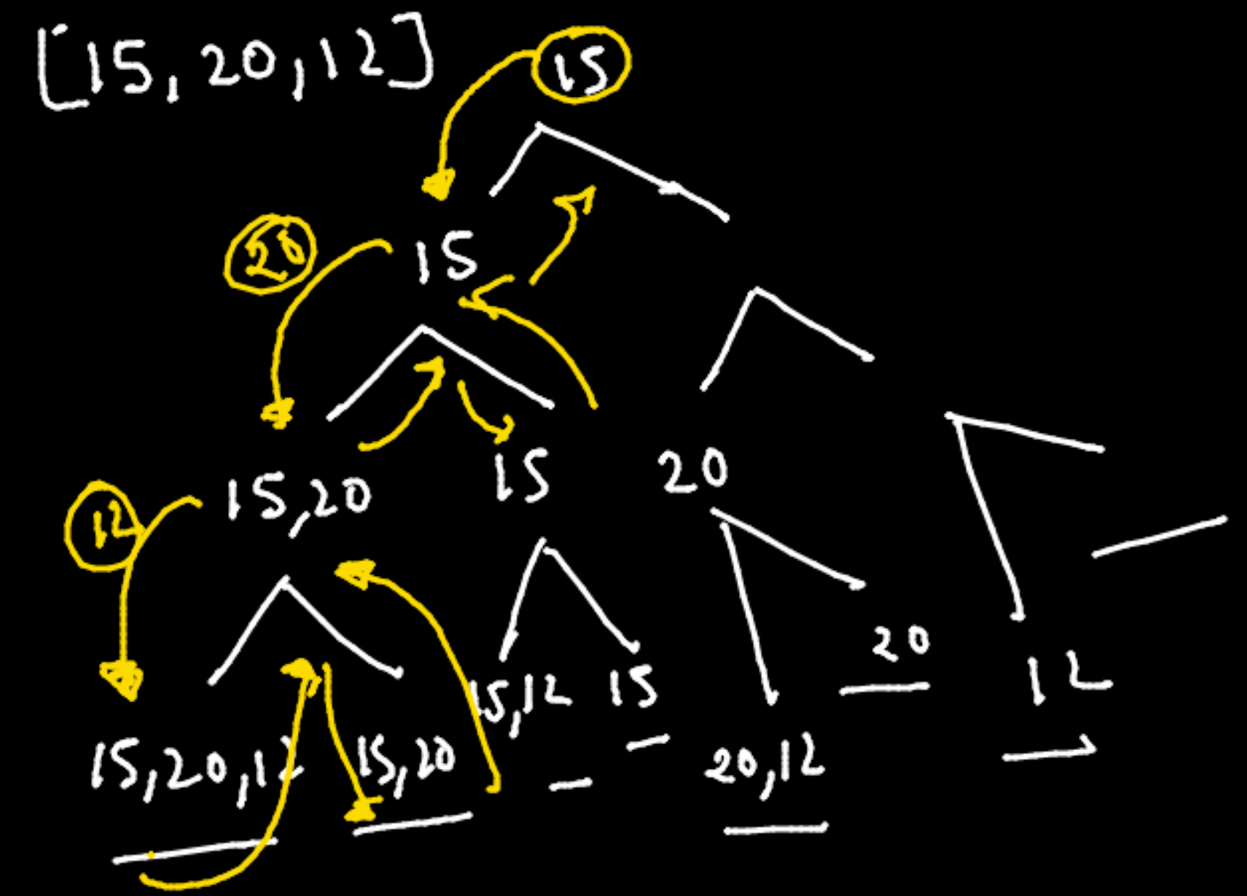
```
void permutation (string &s, int l, int r) {
    if (l==r) {
        cout<<s;
    }
    else {
        for (int i=l; i<=r; i++){
            swap (s[l], s[i]);
            permutation (s, l+1, r)
            swap (s[l], s[i]);
        }
    }
```

ABC

Subset of array → 2^n ← length of array.

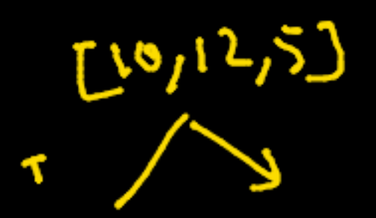
eg. [15, 20, 12]

15
20
12
15 20
15 12
20 12
15 20 12
empty



code ↓

```
void printSubset(int arr[], int i, int n, vector<int>v){
    if (i == n){
        for (auto x : v)
            cout << x << " ";
        cout << endl;
        return
    }
```

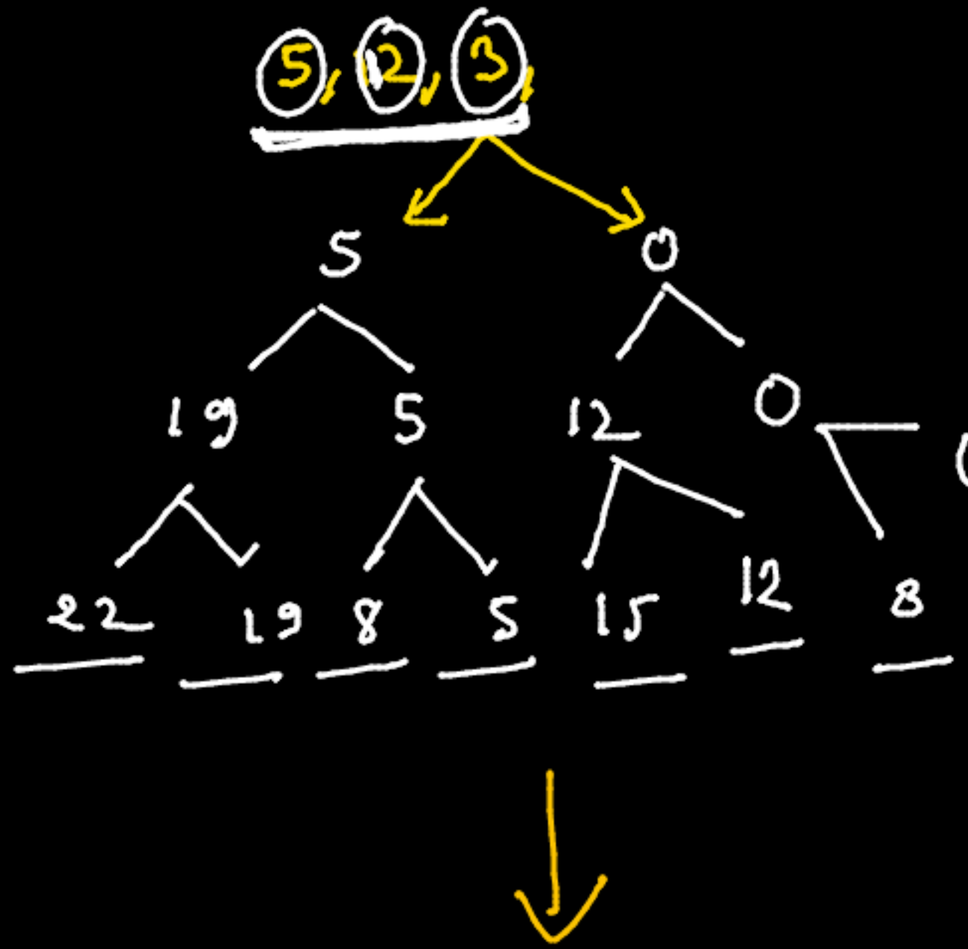



```
    v.push_back(arr[i]);
    printSubset(arr, i+1, n, v)
    v.pop_back();
    printSubset(arr, i+1, n, v);
```

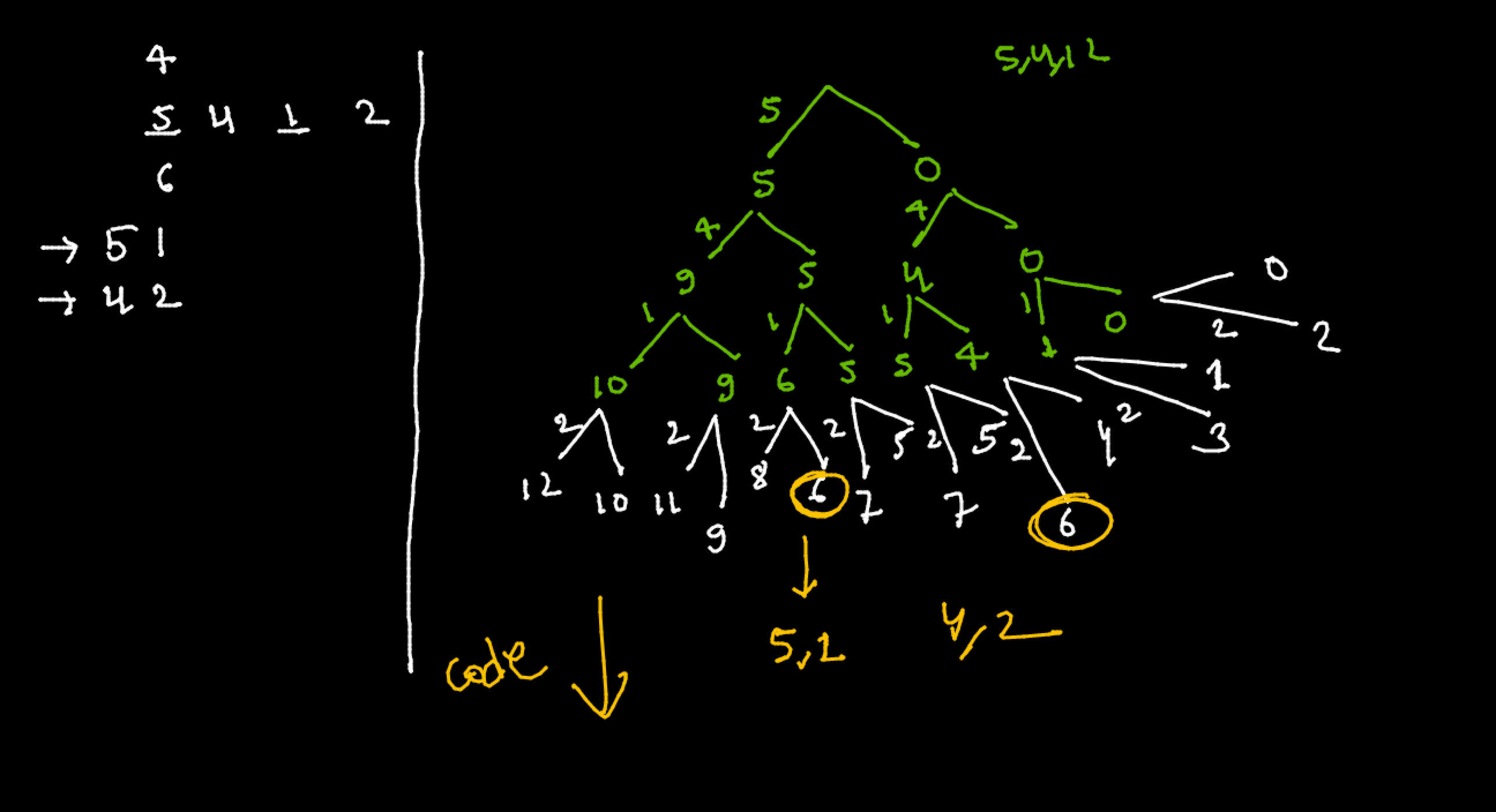
Including element then calling recursive function to print all subset then pop that particular element

Not including an element.

Return Subset sum To K



Combination.
Taking all possible subset.


4
5 4 1 2
6
→ 5 1
→ 4 2
5,4,1,2
code
5,1
4,2

```
void printSubsetTok(int arr[], int i, int n, vector<int> v, target) {
    if (i >= n) {
        if (target == 0)
        {
            for (auto x : v)
                cout << x << " ";
            cout << endl;
        }
        return;
    }
    target -= arr[i];
    v.push_back(arr[i]);
    printsubsettok(arr, i+1, n, v, target)
```

Including that element & target push that element and move forward.

```
//Backtract
target += arr[i];
v.pop_back();
printsebsetTok(arr, i+1, n, v, target)
```

Not including that element

Quick Sort

right
place
me dulo


```
void qs(int arr[], int l, int r)
    if (l >= r)
        return;
    c = partation(arr, l, r)
    qs(arr, l, c-1);
    qs(arr, c-1, r);
```

```
int partition (int arr[], int start, int end) {
    int pivot = arr[start]
    int countsmaller = 0
    for (int i = start ; i <= end ; i++)
    {
        if (pivot >= arr[i])
            countsmaller++;
    }

    int pivotIndex = start + countsmaller;
    swap( arr[start], arr[pivotIndex]);

    int i = start, j = end
    while (i < pivotIndex && j > pivotIndex)
    {
        if (pivot >= arr[i])
            i++;
        else if (pivot < arr[j])
            j--;
        else {
            swap(arr[i], arr[j]);
            i++; j--;
        }
    }
    return pivotIndex;
}
```

1 a
2 b
3 c
4 d
5 e
6 f
7 g
8 h
9 i
10 j
11 k
12 l
13 m
14 n
15 o

16 p
17 q
18 r
19 s
20 t
21 u
22 v
23 w
24 x
25 y
26 z

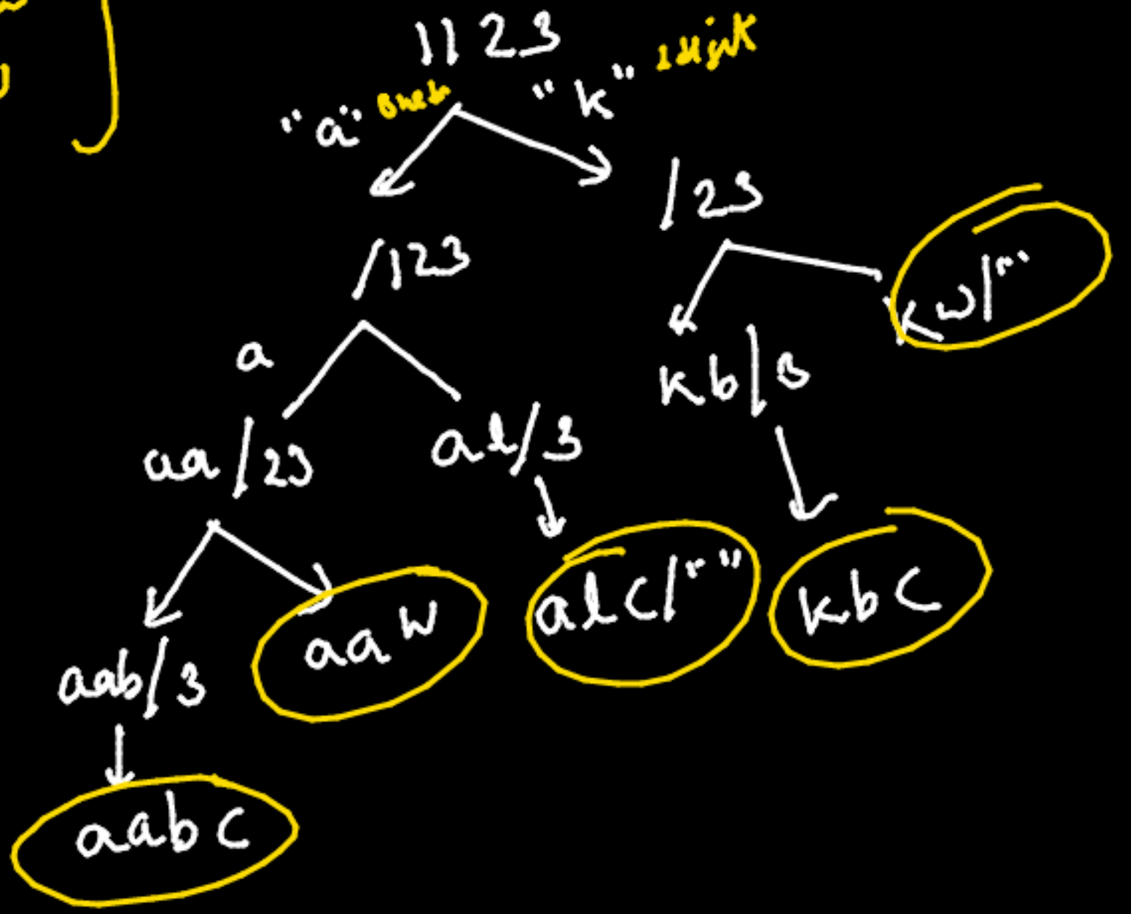
Print all Codes

"1123"

- aabc
- alc
- aaw
- kw
- kbc

} 5 outcomes

Choices → to chose 1 digits
① chose 2 digits



```
void printAll ( string input, string output ) {
    if (input.empty ()) {
        cout << output << endl;
        return;
    }

    char c1 = input[0] - '0' + 'a' - 1;   → first 1digit to char
    int b = stoi ( input.substr (0,2));   → first two digit
    char c2 = b - 1 + 'a';                → first 2 digit into char.

    printAll (input.substr(1), output + c1);   → for 1 digit ex (1:)
    if ( b>=10 && b<=26)
    {   printAll (input.substr (2), output+c2);   → for two digit ex:("11")
    }
}
```

input = "cubc"
output = "";