#6Companies30days & #Revise-WithArsh @ArshGoyal

Challange Company 1 : Microsoft

Q1. You are given an array of strings token that represent an arithemetic expression in a Reverse Polish Notation.

Ex 1: Input : ~~kuch bhi~~ tokens = ["2","1","+","3","*"]
output : 9 Explanation : ((2+1) * 3) = 9

Reverse Polish Notation or simply Postfix Notation

Now to Convert String to Integer

=> String str
stringstream ss(str);
int data
ss >> data
print(data);

=> str = "2" => data = 2

Code: C++

```
class Solution {
public:
    int evalRPN(vector<string> & tokens){
        stack <int> st;
        for(auto x : tokens){
            if(x=="+"||x=="-"||x=="/"||x=="*"){
                int b = st.top(); st.pop();
                int a = st.top() ; st.pop();
                if(x=="+")
                    st.push(a+b);
                if(x=="-")
                    st.push(a-b);
                if(x=="/")
                    st.push(a/b);
                if(x=="*")
                    st.push(a*b);
            }
            else{
                stringstream ss(x);
                int data;
                ss >> data;
                st.push(data);
            }
        } return st.top();
    }
};
```

2. Combination Sum with a twist [Combination Sum III]

Q. Find all valid combinations of k numbers that sum upto n such that the following conditions are true:

- Only no. 1 through 9 are used.
- Each no. are is used atmost once.

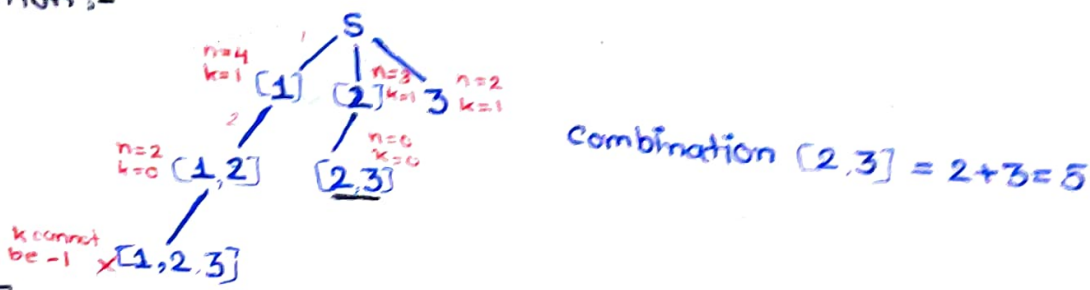
Return a list of all possible combinations. The list must not contain same cond^n twice, & combinations may be return-ed in any order.

Ex: Input: [1|2|3] , k = 2 , n = 5

Subsets: [1],[2],[3],[1,2],[1,3],[2,3]

Output: [2,3]

Explanation:-



Code: C++

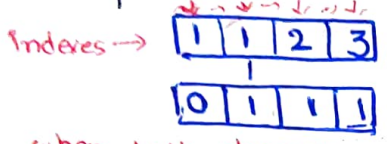
```
class Solution {
public:
    void help(vector<vector<int>> &ans, vector<int>cur, int n, int k, int start){
        if(n==0 and k==0){
            ans.push_back(cur); return;
        }
        if(k==0){return;}
        for(int i=start; i<=9; i++){
            cur.push_back(i);
            help(ans, cur, n-i, k-1, i+1);
            cur.pop_back();
        }
        return;
    }
    vector<vector<int>> combinationSum3(int k, int n){
        vector<vector<int>> ans;
        vector<int> cur;
        help(ans, cur, n, k, 1);
        return ans;
    }
};
```

3. Bulls & Cows : 299. Bulls and Cows (Leetcode)

Q. Given the secret no. & friend guess, return hint for guess.
Format of hint = "xAyB" where x = no. of bulls, y = no. of cows.
Note: Both secret & guess may contain duplicate digits.

Ex1: Input : secret = "1807", guess = "7810"
Output: "1A3B" => 1807 / 7810

Ex2: Input : secret = 1123, guess = "0111" | Output : "1A1B"
Explanation:

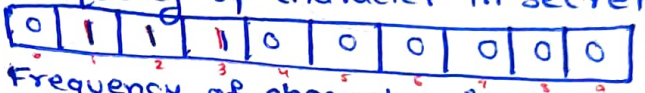


when both characters are equal, increment bulls.

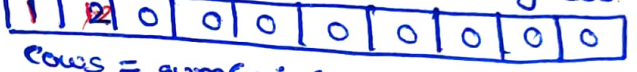
∴ 1A1B

Bulls = 0 => 1 (as index 1 of both are same)

Frequency of character in secret



Frequency of character in guess



Cows = sum(min(guess[i], secret[i]))
= 0 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 1

Code: C++

```
class Solution {
public:
    string getHint(string secret, string guess) {
        int bulls = 0;
        int cows = 0;
        vector<int> s(10, 0);
        vector<int> g(10, 0);

        for(int i=0; i<secret.length(); i++) {
            if(secret[i] == guess[i]) {
                bulls++;
            }
            else {
                s[secret[i]-'0']++;
                g[guess[i]-'0']++;
            }
        }
        for(int i=0; i<10; i++) {
            cows += min(s[i], g[i]);
        }
        string ans = "";
        ans += to_string(bulls); ans += 'A';
        ans += to_string(cows); ans += 'B';
        return ans;
    }
};
```

4. You are given an integer array nums of length n. Return maximum length of Rotation function.

Ex : Input : nums = [4,3,2,6]

Output : 26

Explanation :

[0,1,2,3]

[4,3,2,6][4,3,2,6]

F(1) = 0×4 + 1×3 + 2×2 + 3×6

Now pivot moves backwards

F(2) = 1×4 + 2×3 + 3×2 + 0×6

or

f(2) = 0×4 + 4 + 1×3 + 3 + 2×2 + 2 + 3×6 − 3×6

= f(1) + 4 + 3 + 2 − 3×6

and for further, like we moves our pivot backward by one

= f(1) + 4 + 3 + 2 + 6 − 4×6

Here we can derive that :

= f(1) + sum(arr) − len(arr) × arr[pivot]

from this constant operation

ans = 26

Hence Complexity = O(n)

Code :

```
class Solution(object):
    def maxRotateFunction(self, nums : List[int])
→ int :
        s = sum(nums)
        d = sum(elem * idx for idx, elem in
        enumerate(nums))
        sol = d
        for pivot in range(len(nums)-1, -1, -1)
            d += s - len(nums) * nums[pivot]
            sol = max(d, sol)
        return sol
```

5. Largest Divisible Subset

Q. Given a set of distinct positive integers nums, return the largest subset answer such that every pair (ans[i], ans[j]) of elements in this subset satisfies:

- ans[i] % ans[j] == 0 , or
- ans[j] % ans[i] == 0

If there are multiple sol.ns, return any of them.

A = 2 B = 4 C = 8 C % B = 0 B % A = 0 ∴ C % A = 0

Ex:

nums[]	1	4	5	8	12	9

Sorting nums

nums[]	1	4	5	8	9	12

<j ← i

count[]	1	1	1	1	1	1

using dp

count[]	1	2	2	3	2	3

→ MAX

4 % 1 = 0 count[i] = max(count[i], count[j] + 1)

= Max(1, 1+1) = 2

5 % 4 ≠ 0

5 % 1 = 0 count[2] = max(count[2], count[0] + 1)

= Max(1, 1+1) = 2.

Similarly...

maxIndex = 3

currentCount = count(maxIndex) = 3

ans subset is {8 4 1}

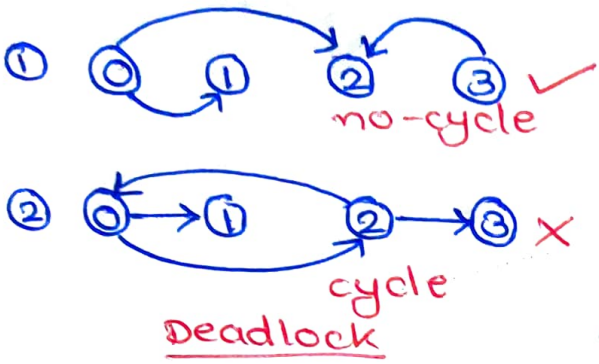
7. Course Schedule

∴ n course 0 - (n-1)

⇒ pre-requisite [i] = [ai, bi]

ai (2nd) ⟶ bi (1st)

Course Schedule ⟶ Topological Cycle Sort



Deadlock

no-cycle True ✓

if cycle False ✗

10. Shortest Unsorted Continuous Subarray

Naive approach : Ex:-

2	6	4	8	10	9	15

after sorting ⇓

2	4	6	8	9	10	15

⟵ 5 ⟶

Complexity :

Time : (O(NlogN)) → As we are doing sort

Space : (O(N)) → As we are using an extra array

Better approach : Ex:-

2	6	4	8	10	9	15

min	MAX-VALUE 4
max	MIN-VALUE 10

① Initialise min & max.

② Find first value, which is decreasing in order. (min)
Similarly we will find max. which is increasing in order.

③ we will find that this particular part is unsorted.

2	6	4	8	10	9	15

decreasing ⟶ increasing

⟵ unsorted ⟶

④ ans would be →

⑤

2	6	4	8	10	9	15

⟵ unsorted ⟶

Length = right - left + 1 = 5

Find index in left for a no. greater than the min found

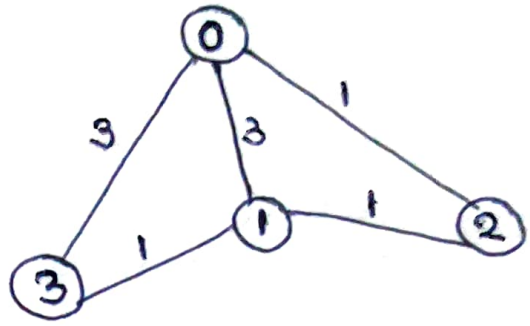
Find index in right for a no. lesser than the max found

Time : O(N)
Space : O(1)

11. No. of ways to arrive at a Destination

* Dijkstra Algorithm

Example:



1. cur = 0, cost = 0, ways : {1,0,0,0}
   dist : {0, max, max, max}
   for loop:
   nbr = 3 :
   pq : (3,3) ways : {1,0,0,1}
   dist : {3, max, max, 3}
   nbr = 1 :
   pq : (3,3) (3,1), ways : {1,1,0,1}
   dist : {0, 3, max, 3}
   nbr = 2 :
   pq : (1,2) (3,3) (3,1), ways : {1,1,1,1}
   dist : {0, 3, 1, 3}

2. cur : 2, cost = 1, ways : {1,1,1,1}
   dist : {0,3,1,3}
   for loop:
   nbr = 1
   pq : (2,1), (3,3), (3,1)
   ways : {1,1,1,1}
   dist : {0,2,1,3}
   nbr = 0
   terminated.

3. cur : 1, cost : 2, ways : {1,1,1,1}
   dist : {0,2,1,3}
   for loop:
   nbr = 3
   pq = (3,3), (3,1)
   ways = {1,1,1,2}
   dist = {0,2,1,3}
   nbr = 0 :
   terminated
   nbr = 2 :
   terminated

4. cur : 3, cost : 3, ways :
   dist : {0,2,1,3} {1,1,1,2}
   for loop:
   nbr = 1 :
   terminated
   nbr = 0 :
   terminated
   pq = (3,1)

5. cur : 1, cost : 3, ways : {0,2,1,3}
   for loop :
   nbr = 3 :
   terminated
   nbr = 0 :
   terminated
   nbr = 2 :
   terminated

Answer = 2

12. Longest Happy Prefix

KMP : Ex: s = cdacdacdc p = cdacdc m, n → O(m*n) → O(m+n)

DP : LPS = [0 0 0 0 0 0] ⇒ LPS = [0 0 0 1 2 1]

s = O(m), LPS = O(n)

∴ O(m+n) ans = 1

Ex: ababab.

LPS = [0 0 1 2 3 4]

ans = abab.

13. Airplance Seat Assignment Probability

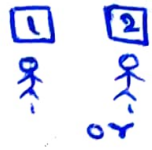
Ex: Input: n = 1 Output: 1.0000.

Seat 1 ↓ correct person ans = 1

Ex2: n = 2

Output: 0.5000 ⇒ 2 possibilities.

wrong person.

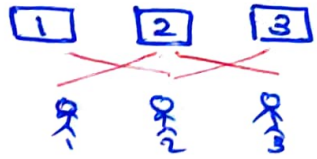


Correct Person.

$\frac{1}{2} = 0.5$

Ex 3: n = 3

Correct person $\frac{1}{3}$



wrong person wrong seat $\frac{1}{3} \times \frac{1}{2}$

$= \frac{1}{2} = 0.5$.

⋮

Ex 4 : Mathematical Induction

for n people n seat correct or wrong

Formula : $\frac{1}{n} + \frac{n-1}{n+1} \times \frac{1}{2} = \frac{2(n+1)}{4(n+1)} = \frac{1}{2} = 0.5$

($\frac{1}{n}$: right, $\frac{n-1}{n+1}$: wrong)

So, for more than 1, ans = 0.5

4. Min. Deletions to Make Array Divisible

1. we need to find gcd.
2. So the smallest no. in nums should be divisible by gcd. of all the elements of arr. nums Divide.