Medium

26 December 2022 15:58

que 1 :- Generate all binary strings :-

Generate all binary strings

Medium Accuracy: 68.1% Submissions: 238+ Points: 4

Land your Dream Job with Mega Job-a-thon. Register Now!

Given an integer **N** , generate all binary strings of size N which do not contain consecutive 1s.

A binary string is that string which contains only 0 and 1.

Example 1:

Input:
N = 3
Output:
000 , 001 , 010 , 100 , 101
Explanation:
None of the above strings contain consecutive 1s. "110" is not an answer as it has '1's occuring consecutively.

Screen clipping taken: 26-12-2022 16:51

```
//User function Template for Java

class Solution {
    private static void getList(int ind , String str, int n , List<String>ans){
        if(ind ==n){
            ans.add(str);
            return;
        }
        getList(ind +1, str +'0' , n, ans);

        if(ind >0 && str.charAt(ind-1) == '1'){

            return;
        }
        getList(ind +1, str + '1' , n, ans);
    }
  public static List<String> generateBinaryStrings(int n) {
    // code here
    List<String>ans = new ArrayList<>();
    getList(0, "", n, ans);

    return ans;

  }
}
```

Screen clipping taken: 26-12-2022 17:55

que 2 :-

Call stack

O=2 C=3 "("

① Observation :- Well formed parenthesis means- each "(" has a corresponding ")" so "( )" valid but)(not valid-

② Observation:- no of "(" >= ")" = n. and the total length of generated string will be n*2.

At each point in our string, we are faced with 3 choices.

① If the length of string = n*2 → if yes then show it as result.

② Is the no. of "(" less than n → if yes then add another "("

③ Is the no of ")" less than n and less than "(". If yes then add another ")"

if none of the condition Match then poped

if (str.length == 2*n) // → Add to our result.

If (open<n) → str+"(" and recursively call the fn.

If (close<open) → str+")" and ——:

#

```
35
36  //User function Template for Java
37
38  class Solution {
39
40      public void backtracking(  List<String>ans, int n , String curr, int open , int close){
41          if(curr . length() == 2 * n){
42              ans.add(curr);
43              return;
44          }
45          if(open < n){
46              backtracking(ans , n , curr +'(' , open +1, close);
47          }
48          if(close < open){
49              backtracking(ans, n, curr + ')' , open , close +1);
50          }
51      }
52      public List<String> AllParenthesis(int n)
53      {
54          // Write your code here
55          List<String>ans = new ArrayList<String>();
56          backtracking(ans, n, "", 0 , 0);
57          return ans;
58      }
59  }
```

Screen clipping taken: 26-12-2022 19:40

que:- 3. Power set:-

Power Set

Easy Accuracy: 43.3% Submissions: 38K+ Points: 2

Land your Dream Job with Mega Job-a-thon. Register Now!

Given a string S, Find all the possible subsequences of the String in lexicographically-sorted order.

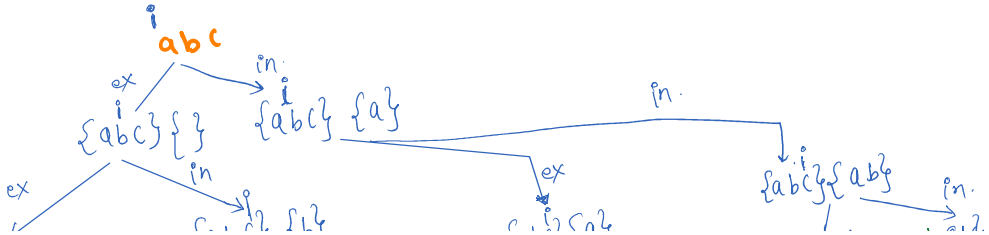
Example 1:

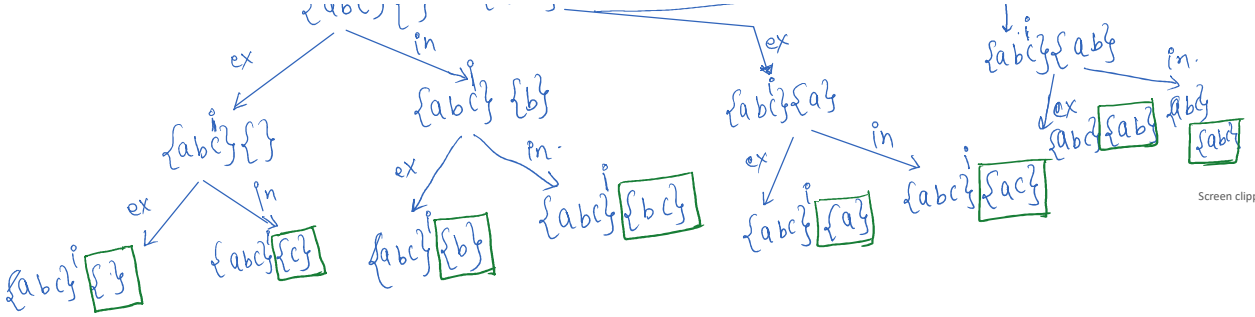
Input : str = "abc"
Output: a ab abc ac b bc c
Explanation : There are 7 subsequences that can be formed from abc.

Example 2:

Input: str = "aa"
Output: a a aa
Explanation : There are 3 subsequences that can be formed from aa.

Screen clipping taken: 26-12-2022 19:59





Screen clipping taken: 26-12-2022 21:37

```java
//User function Template for Java

class Solution
{
    public static void allpossible( List<String>ans  , String str , String s , int ind){
        if(ind >= s.length()){
            if(str.length() !=0)
                ans.add(str);
                return;
        }
        allpossible(ans, str + s.charAt(ind), s,ind +1);
        allpossible(ans,  str,s, ind +1 );


        return;
    }
    public List<String> AllPossibleStrings(String s)
    {
        // Code here
        List<String>ans  = new ArrayList<>();
        int ind = 0;
        allpossible(ans, "" , s,ind);
        Collections.sort(ans);
        return ans;
    }
}
```

que 39:- Combination Sum - I :-

39. Combination Sum

Medium 14.5K 290

Companies

Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all* ***unique combinations*** *of* `candidates` *where the chosen numbers sum to* `target`. You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

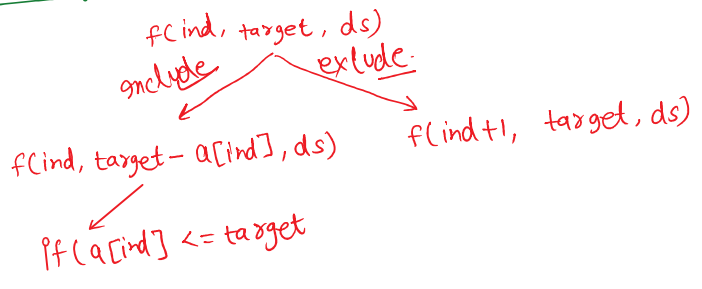
The test cases are generated such that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

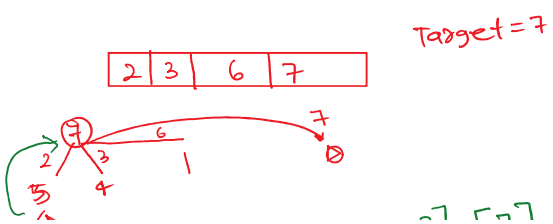
Example 1:

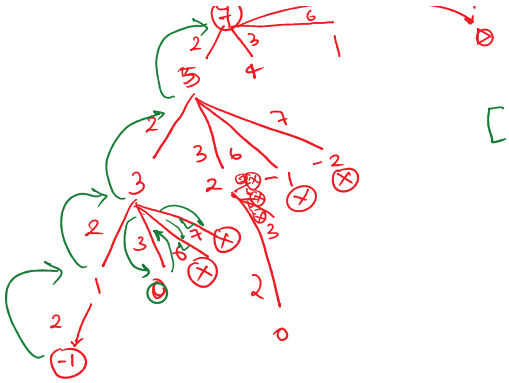
```
Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple
times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
```

Screen clipping taken: 26-12-2022 22:13

DRY Run the Problem :-

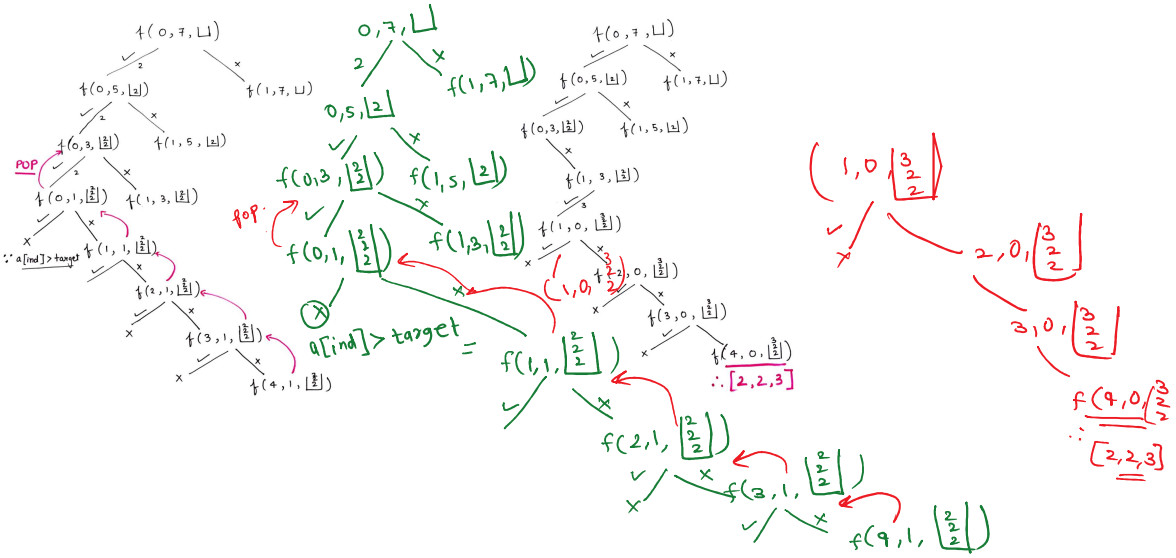






[2,2,3], [7],

case 1°:-



```
class Solution {
    List<List<Integer>>ans  = new ArrayList<>();

    void solve(int[] candidates, int target, int ind, int n , List<Integer>list){
        if( ind >= n || target < 0){
            return;
        }
        if(target ==0){
            ans.add(new ArrayList<>(list));
            return;
        }
        list.add(candidates[ind]);
        solve(candidates, target - candidates[ind] , ind , n , list);  //include
        list.remove(list.size() -1);
        solve(candidates, target, ind +1, n, list);  //exclude

    }
    public List<List<Integer>> combinationSum(int[] candidates, int target) {
        solve(candidates, target, 0, candidates.length , new ArrayList<>());
        return ans;
    }
}
```

Screen clipping taken: 27-12-2022 10:06

que 2°:- Combination sum 2°

40. Combination Sum II

Medium 7.5K 183

Companies

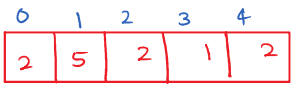
Given a collection of candidate numbers (`candidates`) and a target number (`target`), find all unique combinations in `candidates` where the candidate numbers sum to `target`.

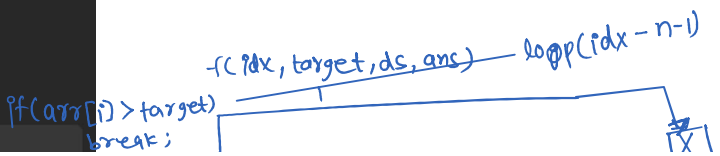
Each number in `candidates` may only be used **once** in the combination.

Note: The solution set must not contain duplicate combinations.

Example 1:

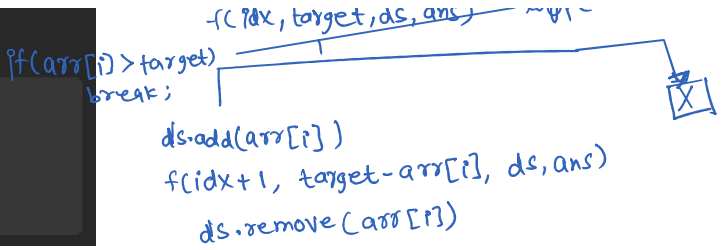
Input: candidates = [10,1,2,7,6,1,5], target = 8





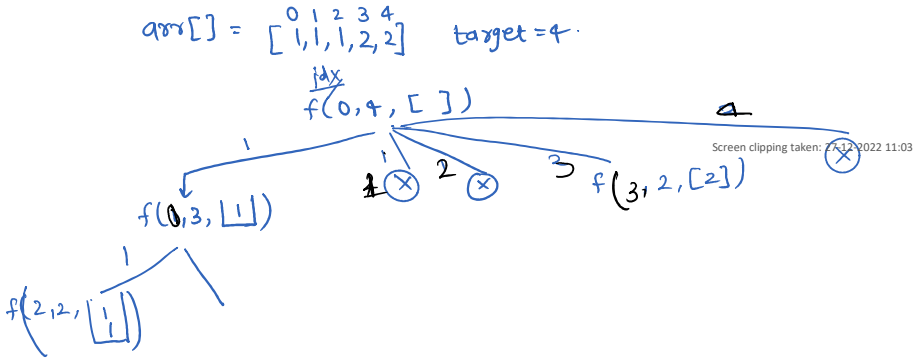
Example 1:

```
Input: candidates = [10,1,2,7,6,1,5], target = 8
Output:
[
[1,1,6],
[1,2,5],
[1,7],
[2,6]
]
```



Base case

```
if(target == 0)
    ans.add(ds);
```




```java
class Solution {
    public void findcombination(int ind , int[] candidates, int target, List<List<Integer>>ans,
ArrayList<Integer> list){
        if(target ==0){
            ans.add(new ArrayList<>(list));
            return;
        }
        for(int i=ind ;i<candidates.length;i++){
            if(i > ind && candidates[i-1] == candidates[i] ){
                continue;
            }
            if(candidates[i] > target){
                break;
            }
            list.add(candidates[i]);
            findcombination(i +1, candidates, target - candidates[i] , ans , list);
            list.remove(list.size() -1);
        }
    }
    public List<List<Integer>> combinationSum2(int[] candidates, int target) {
        List<List<Integer>>ans = new ArrayList<>();
        Arrays.sort(candidates);
        findcombination(0, candidates, target, ans, new ArrayList<>());
        return ans;
    }
}
```

que3:- Combination sum-3

Screen clipping taken: 27-12-2022 11:35

216. Combination Sum III

Medium 4.5K 92

Companies

Find all valid combinations of `k` numbers that sum up to `n` such that the following conditions are true:

- Only numbers `1` through `9` are used.
- Each number is used **at most once**.

Return *a list of all possible valid combinations*. The list must not contain the same combination twice, and the combinations may be returned in any order.

Example 1:

```
Input: k = 3, n = 7
Output: [[1,2,4]]
Explanation:
1 + 2 + 4 = 7
There are no other valid combinations.
```

Example 2:

```
Input: k = 3, n = 9
Output: [[1,2,6],[1,3,5],[2,3,4]]
```

```java
class Solution {
    public static void solve(List<List<Integer>>ans , int start , int k, int n , ArrayList<Integer>list){
        if(n < 0) return;
        if(k==0 && n==0){
            ans.add(new ArrayList<>(list));
            return;
        }
        for(int i=start ;i<=9;i++){
            list.add(i);
            solve(ans, i+1, k-1, n-i, list);
            list.remove(list.size() -1);
        }
    }
    public List<List<Integer>> combinationSum3(int k, int n) {
        List<List<Integer>>ans = new ArrayList<>();

        solve(ans, 1, k , n, new ArrayList<>());

        return ans;
    }
}
```

Screen clipping taken: 27-12-2022 11:35

que:- Subset sum°-

Subset Sums

Basic Accuracy: 72.55% Submissions: 53K+ Points: 1

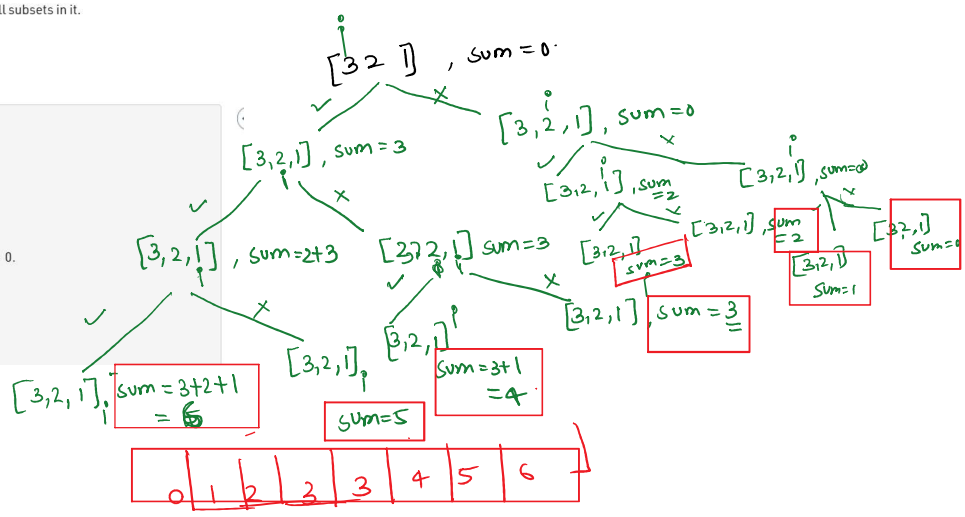
Land your Dream Job with Mega Job-a-thon. Register Now!

Given a list **arr** of **N** integers, print sums of all subsets in it.

Example 1:

Input:
N = 2
arr[] = {2, 3}
Output:
0 2 3 5
Explanation:
When no elements is taken then Sum = 0.
When only 2 is taken then Sum = 2.
When only 3 is taken then Sum = 3.
When element 2 and 3 are taken then
Sum = 2+3 = 5.

Screen clipping taken: 27-12-2022 11:37



Time complexity $O(2^n) + O(2^n \log(2^n))$

SC°- $O(2^n n)$

```java
//User function Template for Java//User function Template for Java
class Solution{
    public static void solve(ArrayList<Integer>arr, int n , ArrayList<Integer>ans, int ind , int sum){
        if(ind == n){
            ans.add(sum);
            return;
        }
        solve(arr, n , ans , ind +1, sum + arr.get(ind));    //pick the element

        solve(arr, n, ans, ind +1, sum );
    }
    ArrayList<Integer> subsetSums(ArrayList<Integer> arr, int N){
        // code here
        ArrayList<Integer>ans = new ArrayList<>();
         solve(arr, N, ans, 0, 0);
         Collections.sort(ans);
         return ans;
    }
}
```

Screen clipping taken: 27-12-2022 11:53

que:- Subset sum - II°-

90. Subsets II

Medium 7.2K 201 Companies

90. Subsets II

Medium 7.2K 201

Companies

Given an integer array `nums` that may contain duplicates, return *all possible subsets (the power set)*.

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

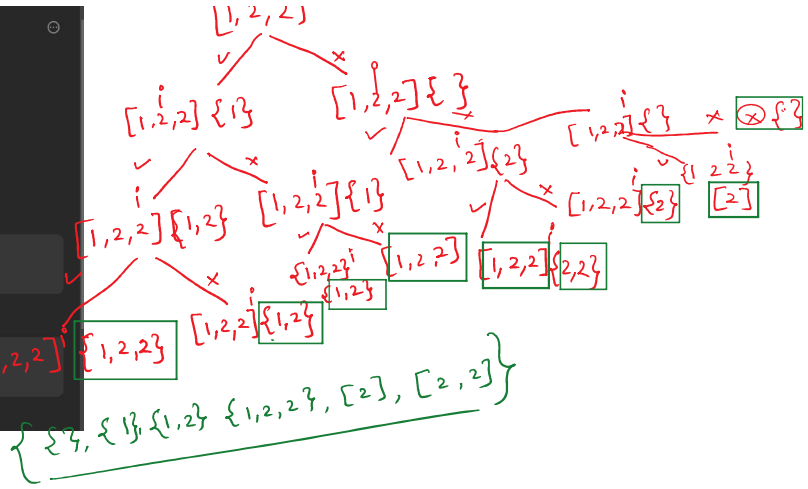
Screen clipping taken: 27-12-2022 12:14

Example 1:

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

Example 2:

```
Input: nums = [0]
Output: [[],[0]]
```



Screen clipping taken: 27-12-2022 11:54

```java
class Solution {
    public static void solve(int ind , int[] nums, ArrayList<Integer>list, List<List<Integer>>ans){
        ans.add(new ArrayList<>(list));
        for(int i= ind ;i<nums.length;i++){
            if(i != ind && nums[i] == nums[i-1])  continue;
            list.add(nums[i]);
            solve(i+1, nums, list, ans);
            list.remove(list.size() -1);
        }
    }
    public List<List<Integer>> subsetsWithDup(int[] nums) {
        Arrays.sort(nums);
        List<List<Integer>>ans = new ArrayList<>();
        solve(0 , nums, new ArrayList<>(), ans );
        return ans;
    }
}
```

TC:- $k \times 2^n$.

SC:- $k \times 2^n$.

que:- letter combination of a phone number

17. Letter Combinations of a Phone Number

Medium 13.6K 790

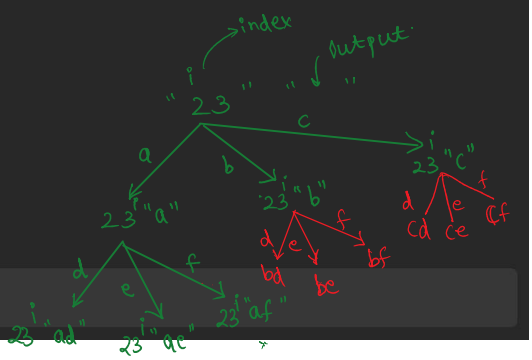
Companies

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

```
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
```



Screen clipping taken: 27-12-2022 12:16

```java
class Solution {
    private String[] keyword = new String[]{" ", "" , "abc" , "def" , "ghi" , "jkl", "mno" ,"pqrs", "tuv", "wxyz"};
    public void solve(String digits, int ind, String str, List<String>ans){
        if(digits.length() == ind){
            ans.add(str);
            return;
        }
        String s = keyword[digits.charAt(ind) -'0'];

        for(char ch : s.toCharArray()){
            solve(digits, ind +1, str + ch , ans);
        }
    }
    public List<String> letterCombinations(String digits) {
        List<String>ans = new ArrayList<>();
        if(digits.length() == 0){
            return ans;
        }
        solve(digits, 0, "" , ans);
        return ans;
    }
}
```