GeeksMan
Recursion
Lesson 3

How to draw a recursive tree

To solve a particular problem , that you know can be divided into further subproblems , can easily be solved by first drawing its recursive tree .

Lets understand how to make a recursive tree with the help of an example :

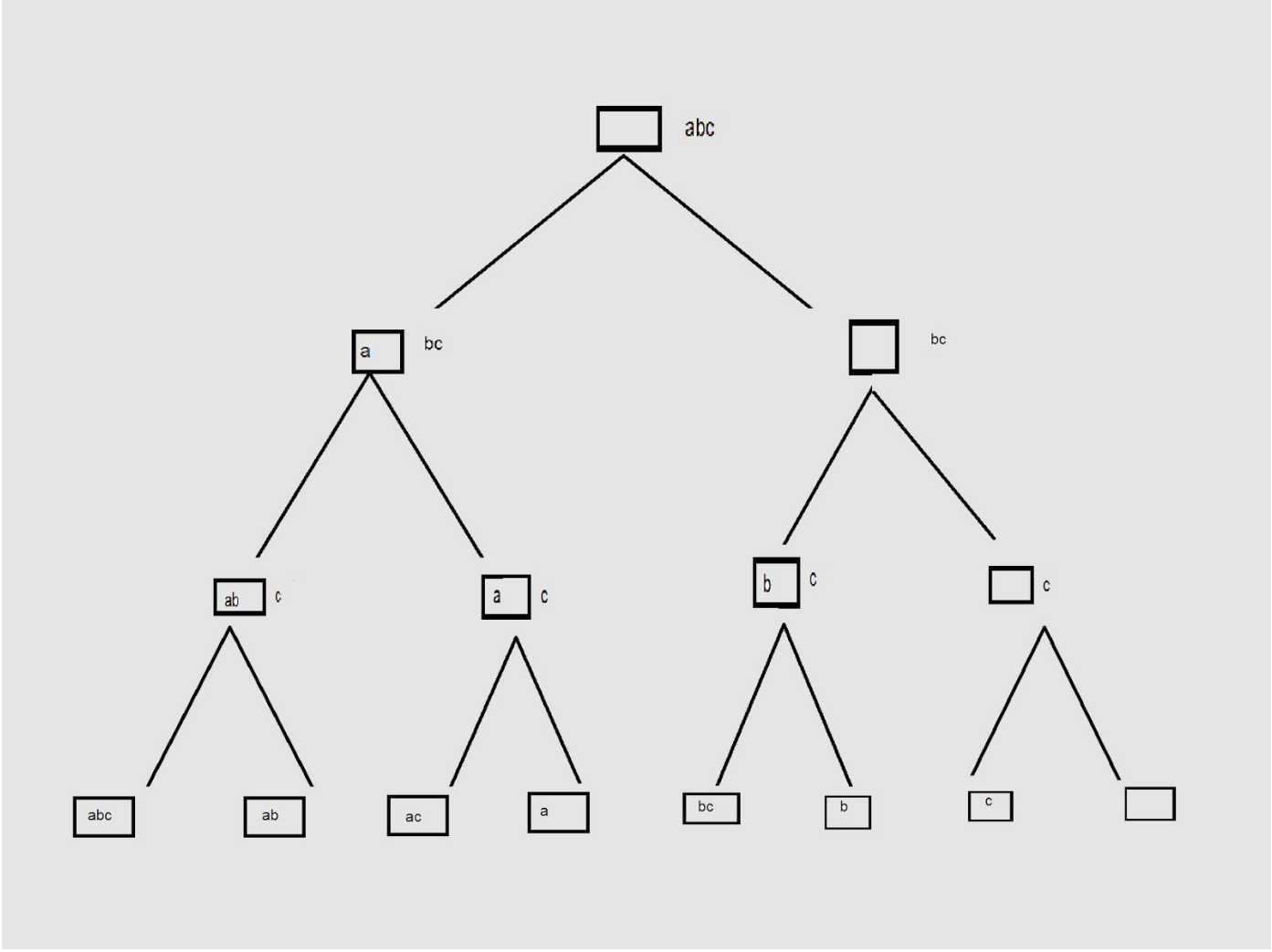
Print all the contiguous substrings of a given string.

Input : abc

Output : “a” , “b” , “c” , “ab” , “bc” , “abc” , “null” .

Solution :

Consider the following recursive tree

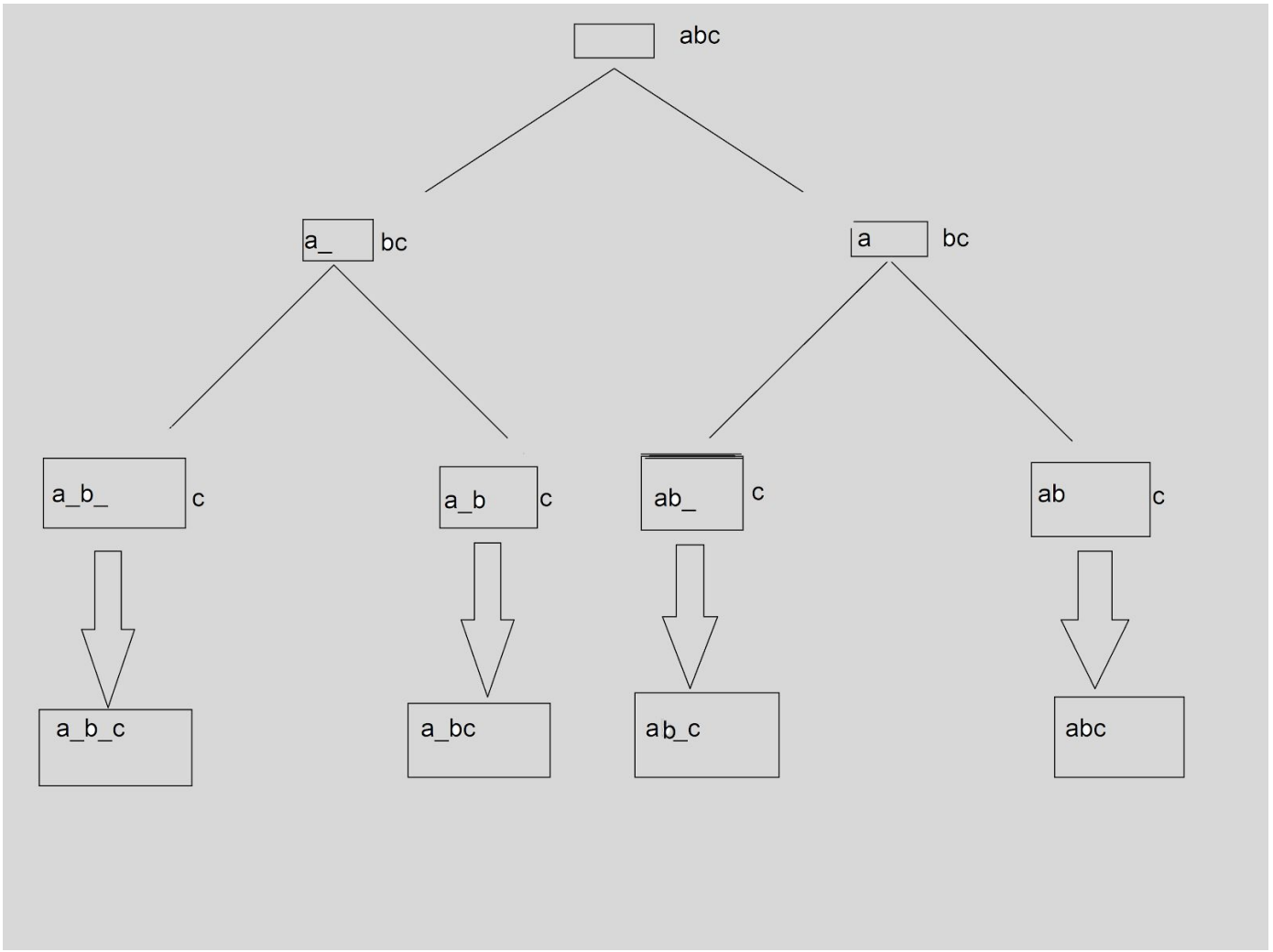


```cpp
#include <iostream>
#include<bits/stdc++.h>
using namespace std;
void solve(string ip,string op)
{
    if(ip.length()==0)
    {
        cout<<op<<" ";
        return;
    }
    string op1=op;
    string op2=op;
    op2.push_back(ip[0]);
    ip.erase(ip.begin());
    solve(ip,op2);
    solve(ip,op1);
    return;
}
int main() {
    string ip;
    cin>>ip;
    string op="";
    solve(ip,op);
    return;
}
```

Print all possible strings

Given a string **str** your task is to complete the function **spaceString** which takes only one argument the string **str** and finds all possible strings that can be made by placing spaces (zero or one) in between them.

For eg . for the string abc all valid strings will be abc , ab c , a bc , a b c.



Permutations of a given string

Given a string S. The task is to print all permutations of a given string.

Input:

The first line of input contains an integer **T,** denoting the number of test cases. Each test case contains a single string **S** in capital letter.

Output:

For each test case, print all permutations of a given string **S** with single space and all permutations should be in lexicographically increasing order.

Constraints:

$1 \leq T \leq 10$

$1 \leq$ size of string ≤ 5

Example:

Input:

2

ABC

ABSG

Output:

ABC ACB BAC BCA CAB CBA

ABGS ABSG AGBS AGSB ASBG ASGB BAGS BASG BGAS BGSA BSAG BSGA GABS GASB GBAS GBSA GSAB GSBA SABG SAGB SBAG SBGA SGAB SGBA

Explanation:

Testcase 1: Given string ABC has permutations in 6 forms as ABC, ACB, BAC, BCA, CAB and CBA .

```
#include<bits/stdc++.h>
using namespace std;
void perm(string ip,string op)
{
    if(ip.length()==0)
    {
        cout<<op<<" ";
        return;
    }
    for(int i=0;i<ip.length();i++)
    {
        string ip1=ip;
        string op1=op;
        op1=op1+ip[i];
        ip1.erase(ip1.begin()+i);
        perm(ip1,op1);
    }
}
void solve()
{
string ip;
cin>>ip;
sort(ip.begin(),ip.end());
string op="";
perm(ip,op);
}
int main()
{
     int t;
```

```
30.  cin>>t;
31.  while(t--)
32.  {
33.       solve();
34.      cout<<endl;
35.  }
36.}
```

Generate all valid parentheses

```
1. #include <iostream>
2. #include<bits/stdc++.h>
3. using namespace std;
4. void find_bal_par(int open,int close,string op,vector<string> &v)
5. {
6.    if(open==0&&close==0)
7.    {
8.       v.push_back(op);
9.       return;
10.   }
11.   if(open!=0)
12.   {
13.      string op1 = op;
14.      op1.push_back('(');
15.      find_bal_par(open-1,close,op1,v);
16.   }
17.   if(close>open)
18.   {
19.      string op2=op;
20.      op2.push_back(')');
21.      find_bal_par(open,close-1,op2,v);
22.   }
23.}
24.void find_p(int n)
25.{
26.   vector<string> v;
27.   int open=n,close=n;
28.   string op="";
29.   find_bal_par(open,close,op,v);
```

```
    for(auto i:v)
    {
        cout<<i<<endl;
    }
}
int main() {
    int n;
    cin>>n;
    find_p(n);
    return 0;
}
```

Subset sum

Given a list(Arr) of N integers, print sums of all subsets in it. Output should be printed in increasing order of sums.

Example 1:

Input:
N = 2
Arr = [2, 3]
Output:
0 2 3 5
Explanation:
When no elements are 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.

Example 2:

Input:
N = 3
Arr = [5, 2, 1]
Output:
0 1 2 3 5 6 7 8

Expected Time Complexity: O(2^N)

Expected Auxiliary Space: O(N)

Constraints:

1 <= N <= 15

0 <= Arr[i] <= 10000

```cpp
#include<bits/stdc++.h>
#include <iostream>
using namespace std;
void solve(int a[] , int start , int end , int sum , vector<int>&v)
{
    if(start > end)
    {
        v.push_back(sum);
        return;
    }
    solve(a , start + 1 , end , sum , v);
    solve(a , start + 1 , end , sum + a[start] , v);
}
int main()
{
    int t;
    cin >> t;
    while(t--)
    {
        int n;
        cin >> n;
        int a[n];
        vector<int>v;
        for(int i = 0;i < n;i++)
            cin >> a[i];
            int start = 0 ;
            int end = n-1;
            int sum = 0;
        solve(a , start , end , sum , v);
        sort(v.begin() , v.end());
```

```
31.      for(int i = 0 ; i < v.size() ; i++)
32.          cout << v[i] << " ";
33.      cout << endl;
34.  }
35.       return 0;
36.}
```

Questions for Practice:

1) Print all subsets:

Input : abc

Output : epsilon c b bc a ac ab abc

2) Print all unique subsets :

Input : aab

Output : Null b aa aab

3) Permutations with case change :

Input : abc

Output : abc abC aBc aBC Abc AbC ABc ABC

4) Permutations with letter change :

Input : a1B2

Output : a1B2 a1b2 A1B2 A1b2

5) Print n bit binary no. having more 1 than 0 :

Given a positive integer n, print all n-bit binary numbers having more 1's than 0's for any prefix of the number.

Input : n = 2

Output : 11 10

Input : n = 4

Output : 1111 1110 1101 1100 1011 1010