The Bug

1. Given N collection of stones each numbered from 0 to n-1. We have to find number of set of three stones (i,j,k) such that (i<j<k) whose diameter sum (stone[i]+stone[j]+stone[k]) should be smaller than the given value sum.

Test Case 1:

5
12
5 1 3 4 7
correct output:
4

Test Case 2:

18
86
30 8 23 6 10 9 31 7 19 20 1 33 21 27 28 3 25 26
correct output*:*
796

Test Case 3:

5 3
-3 0 1 -2 4
correct output:
8

Bug Code:

```
package THE_BUG;

import java.util.Arrays;
import java.util.Scanner;

class Subsets{
    long countTriplets(long arr[], int n,int sum)
    {
        Arrays.sort(arr);
        int i=0;
//            int j=1;
        int count=0;
        while(i<(n-1))
        {
            int j = i+1;
            int k=i+2;
            while(j<n && k<n)
            {
                if( (arr[i]+arr[j]+arr[k])<sum)
                {
                    count++;
                    k++;
```

```
                }
                else{
                    j++;
                }
            }
            i++;
        }
        return count;
    }
}
public class subsets_Bug {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        int sum = s.nextInt();
        long []arr = new long[n];
        for(int i=0;i<n;i++)
        {
            arr[i] = s.nextLong();
        }
        Subsets obj = new Subsets();
        System.out.println(obj.countTriplets(arr,n,sum));

    }
}
```

Correct Code:

```
package THE_BUG;

import java.util.Arrays;
import java.util.Scanner;
class Subset_class{
    long countTriplets(long arr[], int n,int sum)
    {
        Arrays.sort(arr);
        int i=0;
//            int j=1;
        int count=0;
        while(i<(n-2))
        {
            int j = i+1;
            int k=arr.length-1;
            while(j<k)
            {
                if((arr[i]+arr[j]+arr[k])<sum)
                {
                    count+=k-j;
                    j++;
                }
                else{
                    k--;
                }
            }
            i++;
        }
        return count;
    }
}


public class subset {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
```

```
        int n = s.nextInt();
        int sum = s.nextInt();
        long []arr = new long[n];
        for(int i=0;i<n;i++)
        {
            arr[i] = s.nextLong();
        }
        Subset_class obj = new Subset_class();
        System.out.println(obj.countTriplets(arr,n,sum));

    }
}
```

2. An ABC Car Assembly line has n stations and p fuel pipes. Every station has a maximum of one pipe entering it and a maximum of one pipe exiting it. ABC must install pairs of Fuel tanks and fuel outlets in the assembly line in such a manner so that A tank is given to every station with one leaving pipe but no receiving pipe. And an outlet should be installed in every station with only one receiving pipe and no leaving pipe.

The Assembly Line Officials presented a network of pipes with three input values: ai, bi, di, which denotes a pipe of dimension di from house ai to house bi. Find a more efficient method of constructing this pipe network. Pipe diameters should be kept to a minimum wherever possible.

Note: The format of the generated output will be as follows. The first line will contain the letter t, which represents the total number of pairs of tanks and outlets that have been placed. The next t lines each have three integers: the tank's station number, the outlet's station number, and the pipe's minimum diameter between them.

Test Case:
17 11
6 5 2
4 7 2
14 16 2
17 12 4
3 2 8
15 13 2
16 6 8
5 17 8
7 1 9
11 4 9
12 8 2
Correct Output:
4
3 2 8
11 1 2
14 8 2

```
15 13 2
```

Test case

```
14 8
7 9 10
10 2 9
5 8 7
9 3 1
3 10 6
14 5 2
4 7 2
1 4 8
```

Correct Output

```
2
1 2 1
14 8 2
```

Bug Code

```
package THE_BUG;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.*;

class StationEdge {
    int destination;
    int weight;
    StationEdge(int x, int y)
    {
        this.destination=x;
        this.weight=y;
    }
}
class StationGraph{
    int min = Integer.MAX_VALUE;
    boolean []vis;
    ArrayList<Integer> temp = new ArrayList<>();
    int v;
    LinkedList<StationEdge>[]adj;
    StationGraph(int V){
        this.v=V;
        vis = new boolean[V];
        adj = new LinkedList[v];

    }
    void addEdge(int source,int destination,int weight) {
        adj[source].add(new StationEdge(destination, weight));
        adj[destination].add(new StationEdge(source,weight));
    }
    int dfs(int current)
    {
        vis[current] = true;
        if(adj[current].size()==0)
```

```
        {
            return current;
        }
        if(adj[current].get(0).weight<min)
            min = adj[current].get(0).weight;
        return dfs(adj[current].get(0).destination);

    }
}
class AssemblyLine{
    ArrayList<ArrayList<Integer>> solve(int n, int p, ArrayList<Integer> a ,
ArrayList<Integer> b , ArrayList<Integer> d)
    {
        int []out = new int[21];
        int []in = new int[21];
        int []dia = new int[101];
        for(int i=0;i<b.size();i++)
        {
            out[a.get(i)] = b.get(i);
            in[b.get(i)] = a.get(i);
            dia[a.get(i)]=d.get(i);
        }

        StationGraph obj = new StationGraph(n);
        for (int i=0;i<p;i++)
        {
            obj.addEdge(a.get(i),b.get(i),d.get(i));
        }
        ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
        ArrayList<Integer> al = new ArrayList<Integer>();;
        for(int i=1;i<n;i++) {
            if(in[i]==0) {
                int ret = obj.dfs(i);
                al.add(i);
                al.add(ret);
                al.add(obj.min);
                ans.add(al);
                al.clear();
            }
        }
        return ans;
    }
}

public class Assembly_line_bug {
    public static void main(String[] args){
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        int p = s.nextInt();
        ArrayList<Integer> a = new ArrayList<>();
        ArrayList<Integer> b = new ArrayList<>();
        ArrayList<Integer> d = new ArrayList<>();
        for (int i = 0; i < p; i++) {
            a.add(s.nextInt());
            b.add(s.nextInt());
            d.add(s.nextInt());
        }
        AssemblyLine obj = new AssemblyLine();
        ArrayList<ArrayList<Integer>> ans;
        ans = obj.solve(n,p,a,b,d);
        System.out.println(ans.size());
        for(int i=0;i<ans.size();i++) {
            System.out.println(ans.get(i));
        }
    }
}
```

Correct Code:

```
package THE_BUG;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Scanner;

class Edge{
    int destination;
    int weight;
    Edge(int x,int y)
    {
        this.destination=x;
        this.weight=y;
    }
}
class Graph{
    int min = Integer.MAX_VALUE;
    boolean []vis;
    ArrayList<Integer> temp = new ArrayList<>();
    int v;
    LinkedList<Edge>[]adj;
    Graph(int V){
        this.v=V;
        vis = new boolean[V];
        adj = new LinkedList[v];
        for(int i=0;i<v;i++)
        {
            adj[i] = new LinkedList<>();
        }
    }
    void addEdge(int source,int destination,int weight) {
        adj[source].add(new Edge(destination, weight));
//        adj[destination].add(new Edge(source,weight));
    }
    int dfs(int current)
    {
        vis[current] = true;
        if(adj[current].size()==0)
        {
            return current;
        }
        if(adj[current].get(0).weight<min)
            min = adj[current].get(0).weight;
        return dfs(adj[current].get(0).destination);

    }
}
class Solution123{
    ArrayList<ArrayList<Integer>> solve(int n, int p, ArrayList<Integer> a ,
ArrayList<Integer> b , ArrayList<Integer> d)
    {
        int []out = new int[21];
        int []in = new int[21];
        int []dia = new int[101];
        for(int i=0;i<b.size();i++)
        {
            out[a.get(i)] = b.get(i);
            in[b.get(i)] = a.get(i);
            dia[a.get(i)]=d.get(i);
        }

        Graph obj = new Graph(n+1);
```

```
        for (int i=0;i<p;i++)
        {
            obj.addEdge(a.get(i),b.get(i),d.get(i));
        }
        ArrayList<ArrayList<Integer>> ans = new ArrayList<ArrayList<Integer>>();
        ArrayList<Integer> al;
        for(int i=1;i<=n;i++) {
            if(in[i]==0 && out[i]>0) {
                int ret = obj.dfs(i);
                al = new ArrayList<Integer>();
                al.add(i);
                al.add(ret);
                al.add(obj.min);
                ans.add(al);
                obj.min = Integer.MAX_VALUE;

            }
        }


        return ans;
    }
}

public class Assembly_line {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        int p = s.nextInt();
        ArrayList<Integer> a = new ArrayList<>();
        ArrayList<Integer> b = new ArrayList<>();
        ArrayList<Integer> d = new ArrayList<>();
        for (int i = 0; i < p; i++) {
            a.add(s.nextInt());
            b.add(s.nextInt());
            d.add(s.nextInt());
        }
        Solution123 obj = new Solution123();
        ArrayList<ArrayList<Integer>> ans;
        ans = obj.solve(n,p,a,b,d);
        System.out.println(ans.size());
        for(int i=0;i<ans.size();i++) {
            System.out.println(ans.get(i));
        }
    }
}
```

3. Due to some programming mistake Jarvis introduces its own language. Now Stark has to find the correct order of characters in this language so that he could identify where the problem has occurred. He took N random words from Jarvis Dictionary to determine the correct sequence of characters.
   Help Stark to find the correct sequence of characters present in Jarvis Language.

   ## Input:
   N=4
   Lang={"bcc", "acd", "ada", "cca"}
   Correct order is: "bacd"

Bug Code:

```
package THE_BUG;


import java.util.*;

class Solutions34
{
    private void buildGraph(HashMap<Character,Set<Character>> map, String
[]words,int []inDeg)
    {
        for(String word: words)
            for(Character ch: word.toCharArray())
                map.putIfAbsent(ch, new HashSet<>());

        for(int i=1;i< words.length;i++)
        {
            String first = words[i-1];
            String second = words[i];
            int len = Math.min(first.length(),second.length());
            for(int j=0;j<len;j++)
            {
                char out = second.charAt(j);
                char in = first.charAt(j);
                    if(!map.get(out).contains(in))
                    {
                        map.get(out).add(in);
                        inDeg[in-'a']++; // e.g.: for inDeg of b: 'b' - 'a' = 1
                    }

            }
        }

    }
    private String bfs(HashMap<Character,Set<Character>> map, int []inDegree)
    {
        Queue<Character> q = new LinkedList<>();
        StringBuilder str = new StringBuilder();
        for(char temp:map.keySet())
        {
            int temp2 =  (temp-'a');
            if(inDegree[temp2]==0)
            {
                q.add((char)temp2);
            }
        }
        while (!q.isEmpty())
        {
            char temp = q.poll();
            str.append(q.peek());
            for(char ch:map.get(temp))
            {
                inDegree[ch]--;
                if(inDegree[ch]==0)
                    q.add(ch);
            }
        }
        return str.toString();
    }
    public String findOrder(String [] dict, int N)
    {
        int []inDeg = new int[128];
        HashMap<Character, Set<Character>> map = new HashMap<>();
        //build Graph
        buildGraph(map,dict,inDeg);
```

```
        //topological sort
        return bfs(map,inDeg);
    }
}
public class Dictionary_bug {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int N = s.nextInt();

        String []str = new String[N];
        for(int i=0;i<N;i++)
        {
            str[i] = s.next();
        }
        Solutions34 obj = new Solutions34();
        System.out.println(obj.findOrder(str,N));
    }
}
```

Correct Code:

```
package THE_BUG;


import java.util.*;

class Solutions345
{
    private void buildGraph(HashMap<Character,Set<Character>> map, String
[]words,int []inDeg)
    {
        for(String word: words)
            for(Character ch: word.toCharArray())
                map.putIfAbsent(ch, new HashSet<>());

        for(int i=1;i< words.length;i++)
        {
            String first = words[i-1];
            String second = words[i];
            int len = Math.min(first.length(),second.length());
            for(int j=0;j<len;j++)
            {
                char out = first.charAt(j);
                char in = second.charAt(j);
                if(out!=in)
                {
                    if(!map.get(out).contains(in))
                    {
                        map.get(out).add(in);
                        inDeg[in-'a']++; // e.g.: for inDeg of b: 'b' - 'a' = 1
                    }
                    break; // we only need first mismatching character
                }
            }
        }

    }
    private String bfs(HashMap<Character,Set<Character>> map, int []inDegree)
    {
        Queue<Character> q = new LinkedList<>();
        StringBuilder str = new StringBuilder();
        for(char temp:map.keySet())
        {
            if(inDegree[temp-'a']==0)
            {
                q.add(temp);
```

```
            }
        }
        while (!q.isEmpty())
        {
            char temp = q.poll();
            str.append(temp);
            for(char ch:map.get(temp))
            {
                inDegree[ch-'a']--;
                if(inDegree[ch-'a']==0)
                    q.add(ch);
            }
        }
        return str.toString();
    }
    public String findOrder(String [] dict, int N)
    {
        int []inDeg = new int[26];
        HashMap<Character, Set<Character>> map = new HashMap<>();
        //build Graph
        buildGraph(map,dict,inDeg);

        return bfs(map,inDeg);
    }
}
public class Dictionary {
    public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int N = s.nextInt();
    String []str = new String[N];
    for(int i=0;i<N;i++)
    {
        str[i] = s.next();
    }
    Solutions345 obj = new Solutions345();
        System.out.println(obj.findOrder(str,N));
    }
}
```