Análisis de Algoritmos

Ejercicios 11:

"Diseño de soluciones mediante programación voraz (Greedy)"

Nombre: Luis Fernando Ramírez Cotonieto

Fecha de entrega:21 de Junio del 2021

Grupo:3CM13

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

Problema 03: 12405 - Scarecrow

Taso owns a very long field. He plans to grow different types of crops in the upcoming growing season. The area, however, is full of crows and Taso fears that they might feed on most of the crops. For this reason, he has decided to place some scarecrows at different locations of the field.

The field can be modeled as a $1 \times N$ grid. Some parts of the field are infertile and that means you cannot grow any crops on them. A scarecrow, when placed on a spot, covers the cell to its immediate left and right along with the cell it is on.

Given the description of the field, what is the minimum number of scarecrows that needs to be placed so that all the useful section of the field is covered? Useful section refers to cells where crops can be grown.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a line containing an integer N ($0 < N < 100$). The next line contains N characters that describe the field. A dot (.) indicates a crop-growing spot and a hash (#) indicates an infertile region.

Output

For each case, output the case number first followed by the number of scarecrows that need to be placed.

Aceptación de juez:

#	Problem	Verdict	Language	Run Time	Submission Date
26504708	12405 Scarecrow	Accepted	C++11	0.000	2021-06-22 03:32:24

Análisis de complejidad:

(n-1)

Explicación:

Cada espantapájaros afecta a su propia estructura y a ambos lados de la estructura, se pregunta al menos cuántos espantapájaros e indica el número mínimo de espantapájaros requerido para el primer campo de trigo I.

Código:

```cpp
#include <cstdio>
using namespace std;

int main()
{



    int t;
    scanf("%d", &t);

    for(int tc = 1; tc <= t; ++tc) {
        int n;
        char str[100+3];

        scanf("%d\n%s", &n, str);

        int scarecrow = 0;

        for(int i = 0; i < n; )
            (str[i] == '.')? ++scarecrow, i += 3 : ++i;

        printf("Case %d: %d\n", tc, scarecrow);
    }

    return 0;
}
```

Problema 05: 11631 - Dark roads

Economic times these days are tough, even in Byteland. To reduce the operating costs, the government of Byteland has decided to optimize the road lighting. Till now every road was illuminated all night long, which costs 1 Bytelandian Dollar per meter and day. To save money, they decided to no longer illuminate every road, but to switch off the road lighting of some streets. To make sure that the inhabitants of Byteland still feel safe, they want to optimize the lighting in such a way, that after darkening some streets at night, there will still be at least one illuminated path from every junction in Byteland to every other junction.

What is the maximum daily amount of money the government of Byteland can save, without making their inhabitants feel unsafe?

Input

The input file contains several test cases. Each test case starts with two numbers m and n, the number of junctions in Byteland and the number of roads in Byteland, respectively. Input is terminated by $m = n = 0$. Otherwise, $1 \leq m \leq 200000$ and $m - 1 \leq n \leq 200000$. Then follow n integer triples x, y, z specifying that there will be a bidirectional road between x and y with length z meters ($0 \leq x, y < m$ and $x \neq y$). The graph specified by each test case is connected. The total length of all roads in each test case is less than 2^{31}.

Output

For each test case print one line containing the maximum daily amount the government can save.

Aceptación de juez:

#	Problem	Verdict	Language	Run Time	Submission Date
26504740	11631 Dark roads	Accepted	C++11	0.240	2021-06-22 03:58:05

Análisis de complejidad:

O (a log n)

Explicación:

El clásico problema MST del árbol de expansión mínima se resuelve mediante el algoritmo Kruskal, que requiere el uso de la búsqueda de uniones.
En el programa, la representación del gráfico es relativamente clásica, y es relativamente simple almacenar el gráfico con un vector. También puede utilizar una matriz de listas de bordes para representar el gráfico.

```cpp
#include <cstdio>
#include <vector>
#include <bitset>
#include <map>
#include <utility>
#include <string>
#include <queue>
using namespace std;

typedef pair<int, int> ii;
typedef vector<int> vi;
typedef vector<ii> vii;

int m, n, totalWeight;
priority_queue<ii> pq;
vector<bool> taken;
vector<vii> AdjList;

void primproc(int u) {
  taken[u] = true;
  for (int i = 0; i < (int) AdjList[u].size(); i++) {
    ii v = AdjList[u][i];
    if (!taken[v.first])
      pq.push(ii(-v.second, -v.first));
  }
}

int main() {
  int u, v, w;

  while (scanf("%d %d", &m, &n), m || n) {
    totalWeight = 0;
    AdjList.assign(m, vii());

    for (int i = 0; i < n; i++) {
      scanf("%d %d %d", &u, &v, &w);
      AdjList[u].push_back(ii(v, w));
```

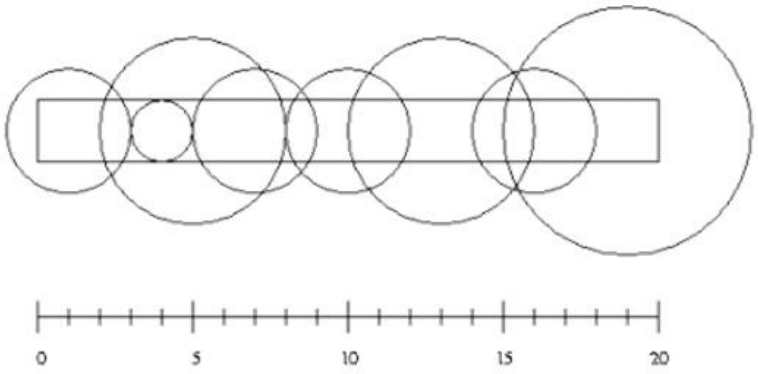
```
        AdjList[v].push_back(ii(u, w));
        totalWeight += w;
    }

    taken.assign(m, false);
    primproc(0);
    int mstCost = 0;
    while (!pq.empty()) {
        ii front = pq.top();
        pq.pop();
        u = -front.second, w = -front.first;
        if (!taken[u]) {
            mstCost += w;
            primproc(u);
        }
    }
    printf("%d\n", totalWeight - mstCost);
  }
  return 0;
}
```

Problema 05: 10382 - Watering Grass

n sprinklers are installed in a horizontal strip of grass l meters long and w meters wide. Each sprinkler is installed at the horizontal center line of the strip. For each sprinkler we are given its position as the distance from the left end of the center line and its radius of operation.

What is the minimum number of sprinklers to turn on in order to water the entire strip of grass?



Input

Input consists of a number of cases. The first line for each case contains integer numbers n, l and w with $n \leq 10000$. The next n lines contain two integers giving the position of a sprinkler and its radius of operation. (The picture above illustrates the first case from the sample input.)

Output

For each test case output the minimum number of sprinklers needed to water the entire strip of grass. If it is impossible to water the entire strip output '`-1`'.

Aceptación de juez:

#	Problem	Verdict	Language	Run Time	Submission Date
26504760	10382 Watering Grass	Accepted	C++11	0.000	2021-06-22 04:13:59

Explicación:

En primer lugar, esta pregunta es un pequeño hoyo, ha sido un problema, definiré todo el recuento como doble tipo antes. (Escuche a los jefes decir con un error de precisión, incluso si el yeso no funciona). El primer punto, el radio debe ser mayor que el radio dado del césped, de lo contrario continúe.
En segundo lugar, la longitud de sus requisitos reales cubrió completamente el césped. es un próximo intervalo regular problemas de cobertura.

Análisis de complejidad:

O (a log n)

```
#include <bits/stdc++.h>
#define REP(r,x,y) for(register int r=(x); r<(y); r++)
#ifdef sahdsg
#define DBG(...) printf(__VA_ARGS__)
#else
#define DBG(...) (void)0
#endif
#define EPS 0
using namespace std;
struct node{
    double l,r;
};
struct cmp {
    inline bool operator () (const node&l, const node&r) {
        if(fabs(l.l-r.l)>EPS) return l.l>r.l;
        return l.r<r.r;
    }
};
priority_queue<node, vector<node>, cmp> q;
int main() {
    #ifdef sahdsg
    freopen("in.txt", "r", stdin);
    #endif
    int n,l;
    double w;
    while(~scanf("%d%d%lf", &n, &l, &w)) {
        while(!q.empty())q.pop();
        w/=2;
        REP(i,0,n) {
            double p,r;
            scanf("%lf%lf", &p, &r);

            double ll=sqrt((double)(r*r-w*w));
            if(!isnan(ll))
            q.push((node){p-ll,p+ll});
        }
        double fi=0;
```

```
    int ans=0;
    while(!q.empty()) {
        node t=q.top(); q.pop();
        while(t.l-fi<-EPS) {
            if(t.r>=fi)
                q.push((node){fi,t.r});
            t=q.top(); q.pop();
        }
        if(t.l>fi+EPS) break;
        fi=t.r;
        ans++;
        if(fi>=l-EPS) break;
    }
    DBG("!%lf\t", fi);
    if(fi<l-EPS) puts("-1");
    else printf("%d\n", ans);
}
return 0;
```