



## Tarea 08

---

Algoritmos de codificación voraz

**Alumno:**

López Manríquez Ángel

(2017630941)



# Índice

<b>1. Minimal coverage</b>	<b>2</b>
1.1. Descripción . . . . .	2
1.2. Código . . . . .	2
1.3. Explicación . . . . .	3
1.4. Validación del juez . . . . .	3
<b>2. Watering grass</b>	<b>3</b>
2.1. Descripción . . . . .	3
2.2. Código . . . . .	4
2.3. Explicación . . . . .	5
2.4. Validación del juez . . . . .	5
<b>3. Scarecrows</b>	<b>5</b>
3.1. Descripción . . . . .	5
3.2. Input . . . . .	5
3.3. Código . . . . .	5
3.4. Explicación . . . . .	6
3.5. Validación del juez . . . . .	6

## 1. Minimal coverage

### 1.1. Descripción

Dados varios segmentos de linea (en el eje  $x$ ) con coordenadas  $[L_i, R_i]$ . Tienes que escoger la minima cantidad de ellas, tal que ellas deberian completar correctamente el segmento  $[0, M]$ .

### 1.2. Código

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int t;
6     cin >> t;
7     while(t--){
8         int m, x, y;
9         cin >> m;
10        vector<pair<int, int>> segments, ans;
11        while(cin >> x >> y){
12            if(x == 0 && y == 0) break;
13            segments.push_back(make_pair(x, y));
14        }
15        sort(segments.begin(), segments.end());
16        int pos = 0;
17        bool found = false, change = true;
18        int i = 0;
19        while(change){
20            change = false;
21            int rightMost = pos;
22            int nuevo;
23            while(i < segments.size() && segments[i].first <= pos){
24                if(segments[i].second > rightMost){
25                    rightMost = segments[i].second;
26                    nuevo = i;
27                    change = true;
28                }
29                i++;
30            }
31            pos = rightMost;
32            if(change){
33                ans.push_back(segments[nuevo]);
34            }
35            if(pos >= m){
36                found = true;
37                break;
38            }
39        }
40        if(!found){
41            ans = vector<pair<int, int>>();
42        }
43        sort(ans.begin(), ans.end());

```

```

44     cout << ans.size() << "\n";
45     for(auto it = ans.begin(); it != ans.end(); ++it){
46         cout << it->first << " " << it->second << "\n";
47     }
48     if(t > 0) cout << "\n";
49 }
50 return 0;
51 }

```

### 1.3. Explicacion

Se procede a ordenar los segmentos dados con base en su primer elemento  $L$ , se crean tres variables, una entera para guardar la posicion mas cercana a  $\infty$ , dos variables booleanas para determinar si hay un nuevo segmento encontrado y otra para usarse en la condicion del ciclo. Al principio, hay cambios para iniciar en el lazo donde preguntamos por los segmentos de recta que mejor se acomoden a nuestra condicion y se deja de ejecutar tan pronto y la posicion llegue a su limite o no haya cambio.

Para cada caso, se tiene una complejidad  $O(n \lg n + n \lg n + n) = O(n \lg n)$ , ya que tenemos dos ordenamientos de la biblioteca estandar y un bucle lineal.

### 1.4. Validacion del juez

#### My Submissions

#	Problem	Verdict	Language	Run Time	Submission Date
22344629	10020 Minimal coverage	Accepted	C++11	0.070	2018-11-24 01:30:20
22344620	10020 Minimal coverage	Accepted	C++11	0.070	2018-11-24 01:25:50
19514898	352 The Seasonal War	Accepted	C++11	0.000	2017-06-12 22:52:28

## 2. Watering grass

### 2.1. Descripción

$n$  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?

$n$  aspersores estan instalados en una tira de hierba con  $l$  metros de largo y  $w$  metros de amplio. Cada aspersor es instalado en el centro de la linea horizontal de la tira. Para cada aspersor se nos da su posicion como la distancia desde la izquierda y su radio de operacion.

¿Cual es el minimo numero de aspersores para prender con el fin de regar toda la tira de hierba?

## 2.2. Código

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int n, l;
6     double w;
7     while(cin >> n >> l >> w){
8         vector<pair<double, double>> segments;
9         for(int i = 0; i < n; i++){
10             double c, r;
11             cin >> c >> r;
12             if(r > w / 2){
13                 double dist = sqrt(r * r - w * w / 4);
14                 segments.push_back(make_pair(c - dist, c + dist));
15             }
16         }
17         sort(segments.begin(), segments.end());
18         double pos = 0;
19         bool found = false, change = true;
20         int i = 0;
21         int ans = 0;
22         while(change){
23             change = false;
24             double rightMost = pos;
25             while(i < segments.size() && segments[i].first <= pos){
26                 if(segments[i].second > rightMost){
27                     rightMost = segments[i].second;
28                     change = true;
29                 }
30                 i++;
31             }
32             pos = rightMost;
33             if(change){
34                 ans++;
35             }
36             if(pos >= l){
37                 found = true;
38                 break;
39             }
40         }
41         if(found){
42             cout << ans << "\n";
43         }else{
44             cout << "-1\n";
45         }
46     }
47     return 0;
48 }
```

### 2.3. Explicacion

La solucion del problema es similar a la anterior, con la diferencia que este problema esta en  $\mathbb{R}^2$ . Para resolver este subproblema preguntamos si  $r$  es mayor a  $w/2$ , de serlo guardamos el segmento  $[L, R]$  tal que  $L = c - d, R = c + d$  donde  $c$  es el centro y  $d = \sqrt{r^2 - w^2/4}$ . A partir de aqui se resuelve analogamente al problema anterior.

Su complejidad  $O$  es la misma a la anterior mas sin embargo en este hay menos comparaciones a lo mucho por hacer en el bucle dado que filtramos los datos al principio.

### 2.4. Validacion del juez

#### My Submissions

#	Problem	Verdict	Language	Run Time	Submission Date
22344813	10382 Watering Grass	Accepted	C++11	0.010	2018-11-24 02:52:52
22344629	10020 Minimal coverage	Accepted	C++11	0.070	2018-11-24 01:30:20
22344620	10020 Minimal coverage	Accepted	C++11	0.070	2018-11-24 01:25:50
19514898	352 The Seasonal War	Accepted	C++11	0.000	2017-06-12 22:52:28

## 3. Scarecrows

### 3.1. Descripción

Taso posee un campo muy largo. Él planea cultivar diferentes tipos de cultivos en la próxima temporada de crecimiento. El área, sin embargo, está llena de cuervos y Taso teme que puedan alimentarse de la mayoría de los cultivos. Por esta razón, ha decidido colocar algunos espantapájaros en diferentes lugares del campo. El campo se puede modelar como una cuadrícula  $1 \times N$ . Algunas partes del campo son infériles y eso significa que no puedes cultivar ningún cultivo en ellas. Un espantapájaros, cuando se coloca en un lugar, cubre la celda a su izquierda y derecha inmediatas junto con la celda en la que se encuentra. Dada la descripción del campo, ¿cuál es el número mínimo de espantapájaros que se debe colocar para cubrir toda la sección útil del campo? Sección útil se refiere a las células donde los cultivos pueden crecer.

### 3.2. Input

Un punto (.) Indica un punto de cultivo y un hash (#) indica una región infértil.

### 3.3. Código

```

1 #include <bits/stdc++.h>
2 using namespace std;
```

```

3
4 int main(){
5     ios_base::sync_with_stdio(0);
6     int t, c = 1;
7     cin >> t;
8     while(t--){
9         int n;
10        cin >> n;
11        vector<char> piso(n + 2);
12        for(int i = 0; i < n; i++){
13            cin >> piso[i];
14        }
15        piso[n] = piso[n + 1] = '#';
16        int ans = 0;
17        bool between = false;
18        int i = 0;
19        while(i < n){
20            if(piso[i] == '.'){
21                if( (piso[i + 1] == '.' && piso[i + 2] == '.') ||
22                    (piso[i + 1] == '#' && piso[i + 2] == '.') ||
23                    (piso[i + 1] == '#' && piso[i + 2] == '#') ||
24                    (piso[i + 1] == '.' && piso[i + 2] == '#') ) {
25                    ans++;
26                    i += 3;
27                }
28            }else{
29                i++;
30            }
31        }
32        cout << "Case " << (c++) << ":" << ans << "\n";
33    }
34    return 0;
35 }
```

### 3.4. Explicacion

El problema es relativamente facil, ya que estamos en una dimension, basicamente con un bucle for lineal recorremos el “campo”, si nos encontramos un punto de cultivo preguntamos si en las siguientes dos posiciones hay al menos un punto de cultivo, si lo hay, incrementamos el numero de soluciones posibles en una unidad y avanzamos tres posiciones con el fin de no contar soluciones repetidas.

La complejidad es  $O(n)$  ya que el bucle es lineal.

### 3.5. Validacion del juez

#### My Submissions

#	Problem	Verdict	Language	Run Time	Submission Date
22344955	12405 Scarecrow	Accepted	C++11	0.000	2018-11-24 03:38:45
22344813	10382 Watering Grass	Accepted	C++11	0.010	2018-11-24 02:52:52
22344629	10020 Minimal coverage	Accepted	C++11	0.070	2018-11-24 01:30:20