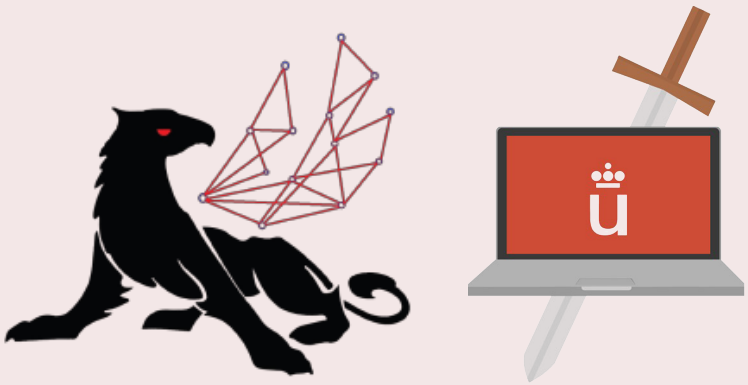


Concurso GRAFOS I 2023

Estadísticas y Soluciones

URJC Escuela Técnica Superior de Ingeniería Informática

Clasificación de los problemas

Problema	Categoría
A - Haya paz	Ad hoc, Strings
B - CTF	BFS. Pensar. (**Dijkstra????**)
C - BENJAMINO	BFS por niveles
D - Fiesta	Componentes conexas
E - Tareas	Orden topológico

Estadísticas

Problema	# casos de prueba	Espacio en disco
A - Haya paz	8	595 B
B - CTF	10	45,94 MB
C - BENJAMINO	3	116 B
D - Fiesta	11	7,58 MB
E - Tareas	7	1,5 MB
- **Total**	39	55 MB (+-)

Estadísticas*

Problema	Primer equipo en resolverlo	Tiempo
A - Haya paz	m.zucchi	1
B - CTF	-	-
C - BENJAMINO	e.gomezf	30
D - Fiesta	i.penedo	19
E - Tareas	-	-

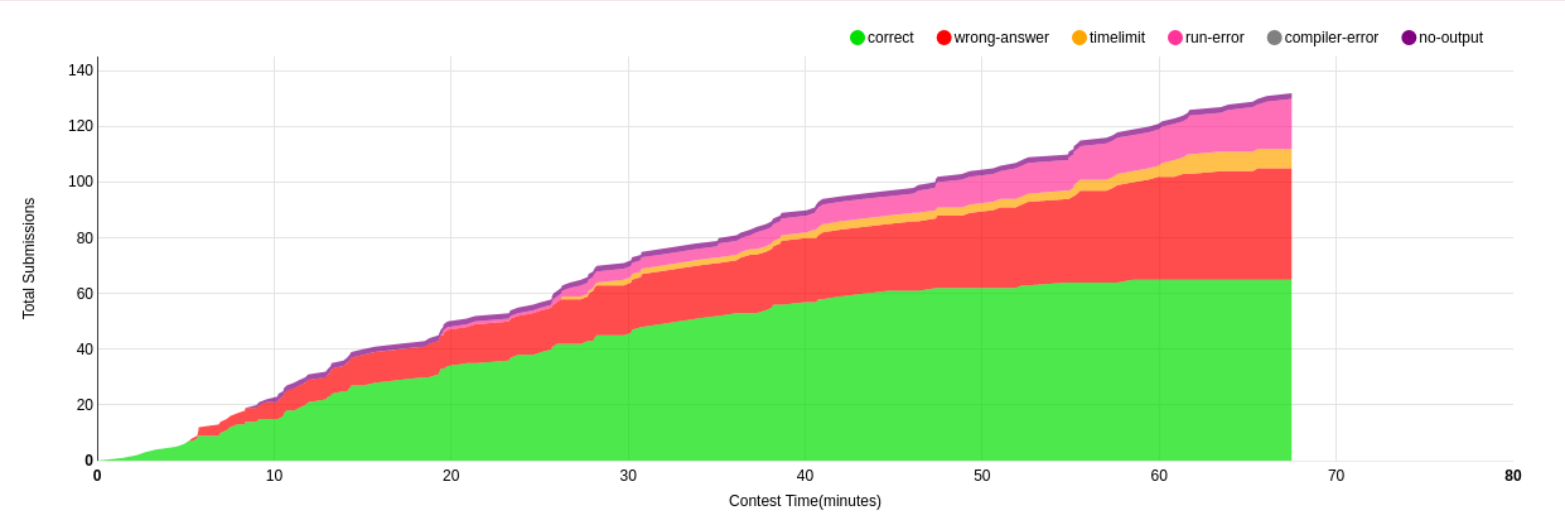
*Antes de congelar el marcador.

Estadísticas*

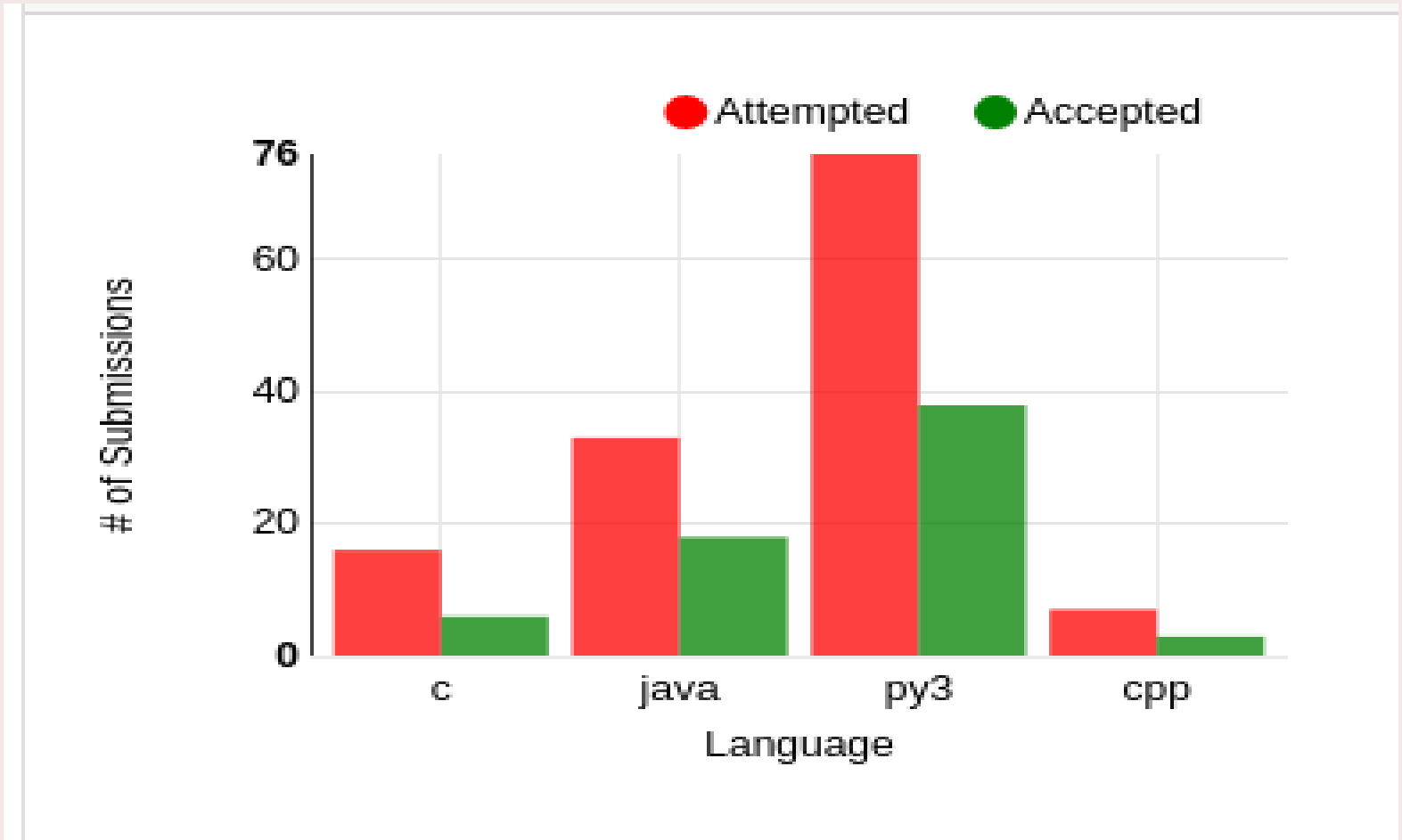
Problema	Envíos	Válidos	% éxito
A - Haya paz	74	54	73 %
B - CTF	3	0	0 %
C - BENJAMINO	19	3	16 %
D - Fiesta	26	8	31 %
E - Tareas	8	0	0 %

*Antes de congelar el marcador.

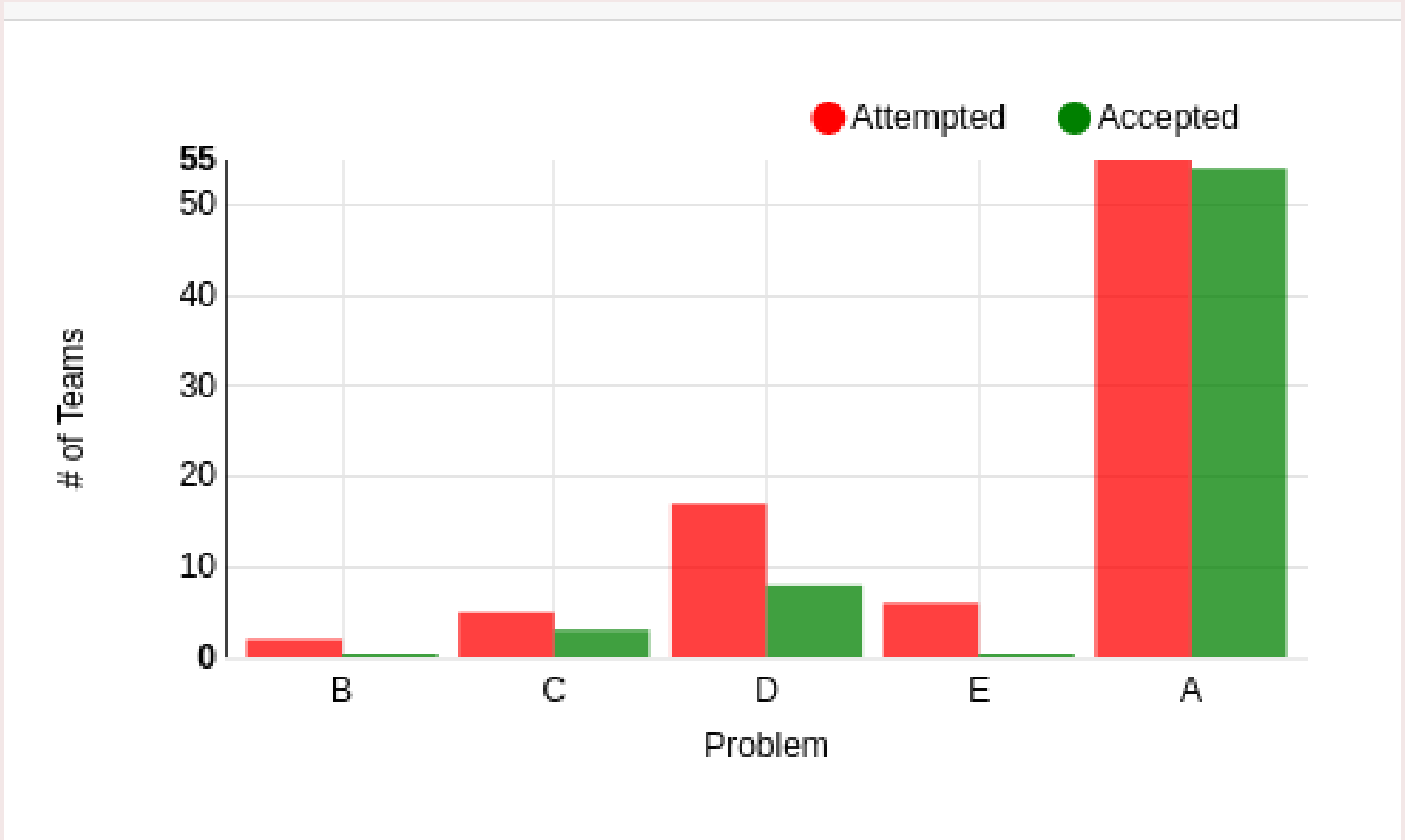
Estadísticas varias



Estadísticas varias



Estadísticas varias



A. Haya paz

Envíos	Válidos	% éxito
74	54	73 %

A. Haya paz

Dadas N palabras, queremos saber si al menos una es "grafo".

A. Haya paz

```
seHabla = false

para cada palabra de las N:
    seHabla = seHabla O palabra.igual("grafo")

si(seHabla): imprimir("SI")
si no: imprimir("NO")
```

B. CTF

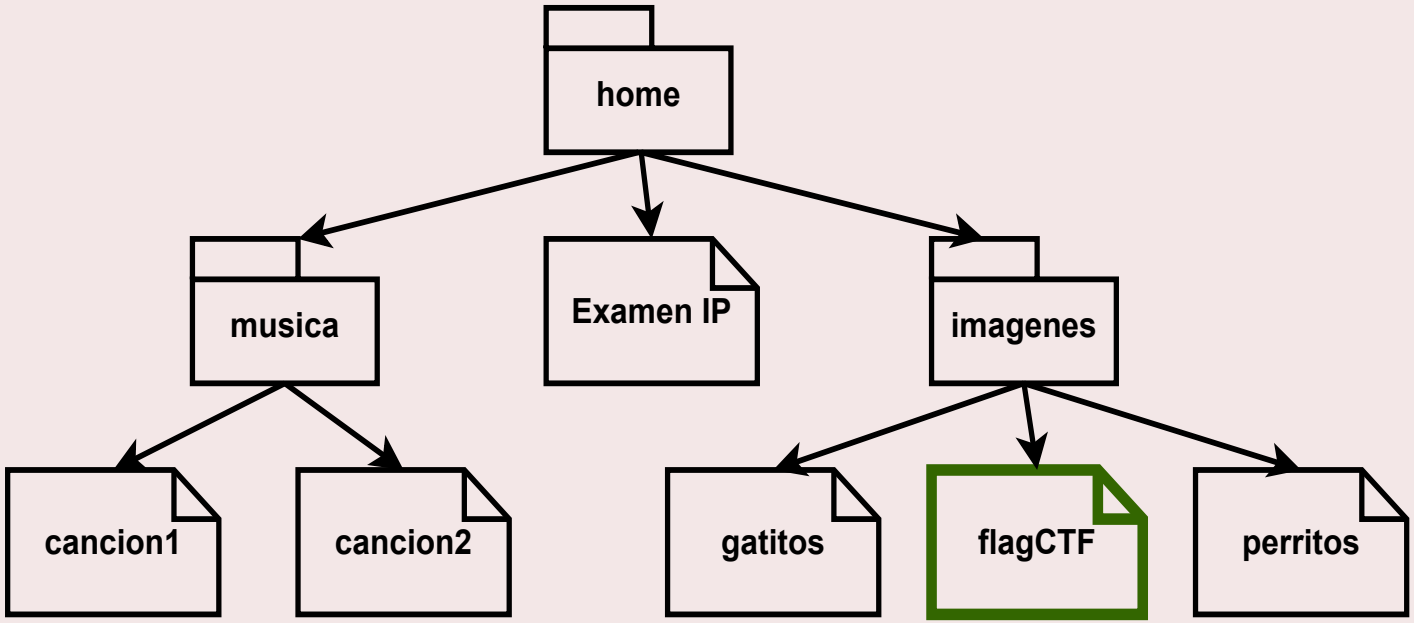
Envíos	Válidos	% éxito
3	0	0 %

B. CTF

Dada la descripción de los directorios de un ordenador, ¿puedes calcular cuantos pasos hay que dar para llegar al fichero flagCTF?

B. CTF

Si vemos los elementos del ordenador como nodos tenemos una estructura de grafo!!



B. CTF

- En el ordenador hay directorios y **ficheros**, pero solo se describen los directorios!!!!! O reservamos suficiente memoria, o utilizamos una estructura dinámica.
- Utilizar Strings como nodos es infernal!!!! Mapealos a enteros.
- El grafo puede tener ciclos!!!! El enunciado nos avisa en: "Cabe destacar que gracias a los accesos directos, un directorio puede accederse desde más de un origen"

B. CTF

Idea básica: Lanzamos un BFS desde cada padre, identificando por cuantos nodos pasamos en cada camino y nos quedamos el mínimo:

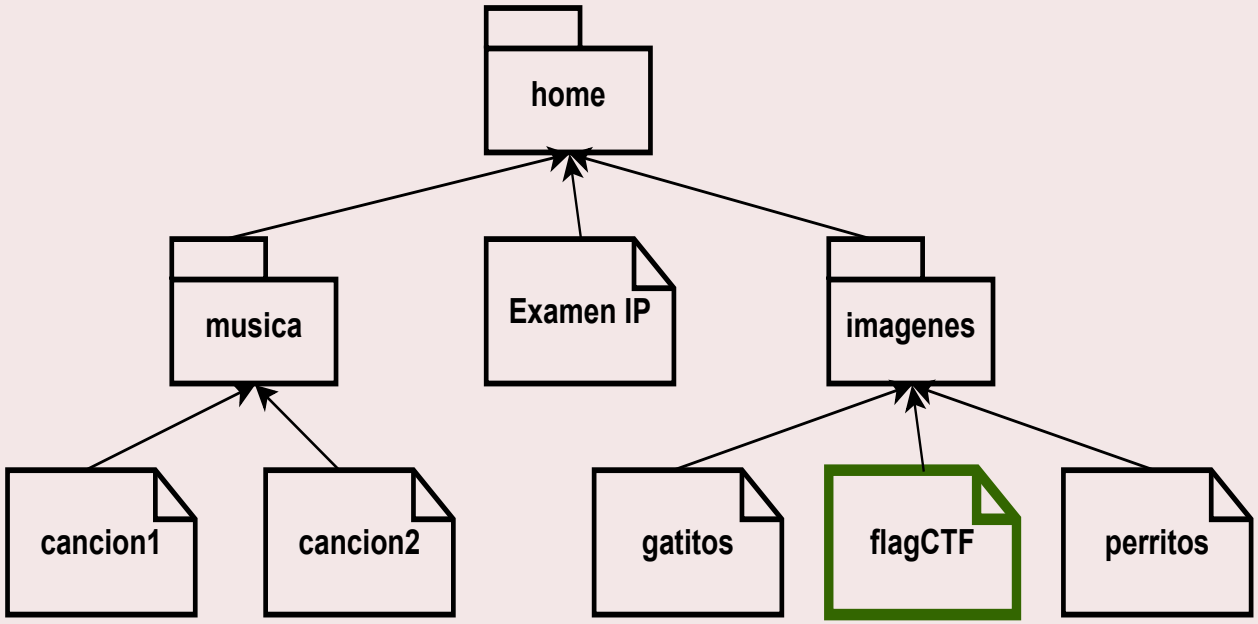
$$\textit{Complejidad}: \quad \mathcal{O}(P \cdot |V|)$$

B. CTF

TLE

B. CTF

Podemos hacerlo mejor: En el grafo tenemos varias entradas pero una única salida!!!
Idea: Podemos lanzar el BFS desde el fichero "flagCTF", pero debemos construir el grafo al reves!!!! Si llegamos a un nodo padre este será el más cercano.



B. CTF

Podemos hacerlo mejor: En el grafo tenemos varias entradas pero una única salida!!!
Idea: Podemos lanzar el BFS desde el fichero "flagCTF", pero debemos construir el grafo al reves!!!! Si llegamos a un nodo padre este será el más cercano.

$$\textit{Complejidad}: \quad \mathcal{O}(|V|)$$

B. CTF

```
grafo={}
mapa={}
padres={}

for(directorio : d)
    mapa.map(d,index)
    for(hijo : h)
        mapa.map(h,index)
        grafo[map(h)].add(map(d))

for(padre p)
    padres.add(map(p))


ans = 0;
flag=false;
visitado = [];
cola = {map('flagCTF'),-1};
```

B. CTF

```
while(|cola| > 1)
    top = cola.remove()
    if(top in padre)
        flag=true;
        break;
    if(top==-1)
        ans++
        cola.add(-1)
    else
        for(adyacentes(top) : a)
            if(!visitado[a]) cola.add(a)


if(flag) imprimir(ans)
else imprimir(-1)
```

B. CTF

¿Y SI NO SE ME OCURRE?

B. CTF

Este problema se puede solucionar mediante el algoritmo de Dijkstra con un poco de cuidado y sin necesitar modelar el algoritmo al revés!

$$Complejidad: \quad \mathcal{O}(|V| \cdot log(|V|))$$

Este algoritmo será parte de la teoría del próximo día :)
Es peor complejidad, pero en este problema el veredicto habría sido:

AC

C. BENJAMINO

Envíos	Válidos	% éxito
19	3	16 %

C. BENJAMINO

Dada la edad inicial de Benjamín y sabiendo que en cada cumpleaños puede sumar un año más, restar cinco o multiplicar su edad por dos, ¿Puede llegar a la edad que quiere? ¿En cuántos cumpleaños?

C. BENJAMINO

Idea:
- BFS por niveles
- No hay que construir el grafo, añadir los nodos a la cola sobre la marcha

Observaciones:
- Siempre hay que mantenerse en los límites del problema ($1 \leq$ edad ≤ 1000)
- Nunca se va a imprimir -1, siempre es posible llegar

C. BENJAMINO

```
para cada caso de prueba (N):
    edadInicial = A, edadObjetivo = B
    visitado[edadInicial] = true, cola.add(edadInicial), cola.add(-1)
    mientras(cola.size() > 1 Y no visitado[edadObjetivo]):
        edad = cola.extraer()
        si(edad==-1):
            cumples++
            cola.add(-1)
        si no:
            si(edad+1 <= 1000 Y no visitado[edad+1])
                visitado[edad+1] = true
                cola.add(edad+1)
            si(edad*2 <= 1000 Y no visitado[edad*2])
                visitado[edad*2] = true
                cola.add(edad*2)
            si(edad-5 >= 1 Y no visitado[edad-5])
                visitado[edad-5] = true
                cola.add(edad-5)
            si(visitado[edadObjetivo]) cumples++
    imprimir(cumples)
```

D. Fiesta

Envíos	Válidos	% éxito
26	8	31 %

D. Fiesta

Dadas las relaciones entre las personas de la fiesta, ¿podrías contar el número de grupos de amigos?

D. Fiesta

Ideas?

D. Fiesta

Ideas?

- Grafo **no dirigido**

D. Fiesta

Ideas?

- Grafo **no dirigido**
- Contar **componentes conexas**

D. Fiesta

```
cont=0;

for(int i=0; i<n; i++){
    if(!visitado[i]){
        //contamos la componente
        cont++;
        //lanzamos un recorrido desde i
        BFS(i) / DFS(i)
    }
}
```

E. Tareas

Envíos	Válidos	% éxito
8	0	0 %

E. Tareas

Dada una serie de tareas, donde se nos indica qué tarea se debe realizar antes que otra, imprimir la secuencia de las tareas en el orden a realizar. ¿Solución?

E. Tareas

TOPOLOGICAL SORT

E. Tareas

¿Problema? Que dado dos posibles soluciones debemos imprimir la menor lexicográficamente. **¿Solución?**

E. Tareas

Basta con tener una cola de prioridad para ir metiendo los nodos sobre los que no incide ninguna arista, asi cuando saquemos de cola de prioridad, siempre sacaremos el menor lexicograficamente, en vez de una pila normal del TopoSort habitual.

E. Tareas

```
grafo={}
visit={}
inc={}
ts={}
indegree={}
pq={}

for(int i=0; i<N; i++)
    if(indegree[i]==0)
        pq.add(i)
```

E. Tareas

```
String sol=""

while(!pq.isEmpty()) {

   int n = pq.poll()
   sol.append(n+" ")

   for(int hijo:grafo[n]) {
      indegree[hijo]--
      if(indegree[hijo]==0 && !visitados[hijo])
          pq.add(hijo)
          visitados[hijo] = true
   }

}
imprimir(sol)
```