UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

EVALUACIÓN	Obligatorio	**GRUPO**	Todos	**FECHA**	2/9
MATERIA	Estructura de Datos y Algoritmos 2				
CARRERA	Ingeniería en Sistemas				
CONDICIONES	- Puntos: Máximo: 30 Mínimo: 1 - Fecha máxima de entrega: 25/11 LA ENTREGA SE REALIZA EN FORMA ONLINE EN ARCHIVO NO MAYOR A 40MB EN FORMATO ZIP, RAR O PDF. **IMPORTANTE:** - Inscribirse - Formar grupos de hasta dos personas. - Subir el trabajo a Gestión antes de la hora indicada, ver hoja al final del documento: “RECORDATORIO”				

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Obligatorio

El obligatorio de Estructuras de Datos y Algoritmos 2, para el semestre 2 de 2021, está compuesto por un conjunto de **11** problemas a resolver.

El desarrollo del obligatorio:

- Se deberá realizar en grupos de **2 estudiantes**.
- Se podrá utilizar tanto C++ (C++11) como Java (jdk8).
  Para asegurar que utilizan C++11 deberán tener instalado el compilador C++ de GNU e invocar la siguiente orden: **`g++ -std=c++11 ...`**
- No se puede usar ninguna librería, clase, estructura, función o variable no definida por el estudiante, a excepción de <iostream>, <string> y <cstring> para C++. y System.in, System.out y String para Java.
- Si se quiere usar algo que no sea lo listado anteriormente, consultar previamente al profesor.
- Se deberá utilizar el formato de ejecución trabajado durante el curso:
  `miSolucion < input > output`
- La cátedra proveerá un conjunto de casos de prueba, pares de entrada y salida esperada. Se deberá comparar la salida de la solución contra una salida esperada.

La entrega:

- El formato de entrega se encuentra definido en este enlace.

La corrección:

- La corrección implica la verificación contra la salida esperada, así también como la corrección del código fuente para verificar que se cumplan los requerimientos solicitados (órdenes de tiempo de ejecución, usos de estructuras o algoritmos en particular, etc.).
- Se verificarán TODOS los casos de prueba. Si el programa no termina para un caso de prueba, puede implicar la pérdida de puntos.
- Si el código fuente se encuentra en un estado que dificulta la comprensión, podrá perder puntos.
- Se utilizará MOSS para detectar copias.

La defensa:

- Luego de la entrega, se realizará una defensa de autoría a cada estudiante de manera individual.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

1. SORTING **Nombre de archivo:** ejercicio1.cpp/Ejercicio1.java

Se desea ordenar un conjunto de números enteros, no acotados. Se solicita que implemente un algoritmo basado en la estrategia de *ordenación por inserción en árboles* en $O(N \log N)$ en el *peor caso*. Para lograr el objetivo, se deberá utilizar un AVL.

Formato de entrada

N n_1 n_2 … n_N

La primera línea indica la cantidad de elementos a ordenar. Las siguientes N líneas son los elementos a ordenar.

Formato de salida

La salida contendrá N líneas, siendo estos los elementos ordenados de menor a mayor.

UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

2. CONTADOR DOBLE

Nombre de archivo: ejercicio2.cpp/Ejercicio2.java

Dada una secuencia de palabras se desea saber aquellas que aparecieron exactamente dos veces.
Restricciones: $O(N)$ caso promedio en tiempo de ejecución.

Formato de entrada

```
N
palabra_1
palabra_2
...
palabra_N
```

La primera línea indica la cantidad de palabras a leer. Las siguientes N líneas son el conjunto de palabras a trabajar.

Formato de salida

La salida contendrá 1 línea, que es la cantidad de palabras que ocurrieron exactamente dos veces en el formato de entrada.

Ejemplo de entrada/salida

7 AAA BAB ABC AAA ABC BCA BBB	2

4 AAA AAA AAA BAC	0

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

3 AAA BBB CCC	0

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

3. INTERCALAR **Nom. de arch.**: ejercicio3.cpp/Ejercicio3.java

Se desea construir un programa que se encargue de intercalar un conjunto de K listas ordenadas, de diferente tamaño. Se debe resolver en $O(P \log_2 K)$, siendo P la cantidad total de elementos.

El algoritmo esperado se describe en:
https://en.wikipedia.org/wiki/K-way_merge_algorithm

Formato de entrada

```
K
N1
Elemento 1 de L1
Elemento 2 de L1
...
Elemento N1 de L1
N2
Elemento 1 de L2
Elemento 2 de L2
...
Elemento N2 de L2
...
```

La primera línea K indica cuántas listas contendrá el archivo de entrada.

La siguiente línea, N_1 nos indica el tamaño de la lista L_1. Luego, las siguientes N_i líneas, contienen los elementos de dicha lista. Esto se repite K veces, una vez por cada lista.

Formato de salida

La salida contendrá P líneas, cada una con los elementos de la lista resultante de intercalar las K listas ordenadamente.

UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Formato de entrada de ejercicios de grafos

Todos los ejercicios de grafos tendrán la misma codificación para los grafos. Es decir, una parte del formato de entrada, la que corresponde a la información del grafo será siempre igual. A continuación se describe:

```
V
E
v1 w1 [c1]
v2 w2 [c2]
...
vi wi [ci]
...
vE wE [cE]
```

Cada grafo comienza con la cantidad de vértices, V. Los vértices siempre serán números, a menos que se especifique lo contrario. Por ejemplo, si $V=3$, entonces los vértices serán: $\{1, 2, 3\}$ (siempre serán numerados a partir de 1).

La siguiente línea corresponde a la cantidad de aristas, `E`. Las siguientes `E` líneas en el formato `v w c` corresponden a las aristas (v,w) con costo c si el grafo *es ponderado*, o en el formato `v w`, correspondiente a la arista (v,w) si *no es ponderado*. Es decir, c es opcional (`[c]`).

El grafo será dirigido o no, dependiendo el problema en particular. En caso de ser *no dirigido* solo solo se pasará un sentido de la arista, es decir, (v,w) pero no (w,v) (queda implícito). Por ejemplo:

```
2
1
1 2
```

Representa al grafo completo de dos vértices y aristas: $\{(1,2), (2,1)\}$.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

4. GRAFO CÍCLICO

Nom. de arch.: ejercicio4.cpp/Ejercicio4.java

Dado un grafo disperso, dirigido y no ponderado, saber si dicho grafo contiene al menos un ciclo.

Restricciones: $O(V+E)$

Formato de entrada

(Ver Formato de entrada)

Formato de salida

Contendrá una única línea:
“1” en el caso de que el grafo sea cíclico.
“0” en caso contrario.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

5. CAMINO + CORTO **Nom. de arch.**: ejercicio5.cpp/Ejercicio5.java

Dado un grafo *dirigido, ponderado (sin costos negativos) y disperso* implementar un algoritmo que devuelve el *costo total* de los caminos más cortos de un vértice dado a todos los demás.

El algoritmo debe tener un tiempo de ejecución de $O(E \log V)$ por cada vértice a calcular.

Formato de entrada

(Ver Formato de entrada)

```
<formato de entrada de un grafo>
N
v1
v2
…
vN
```

La siguiente línea contiene un número N, indicando la cantidad de vértices desde la cual se desea conocer la menor distancia hacia los demás. Seguido a este número continúan N líneas, cada una representando un vértice a partir del cual se quiere conocer la menor distancia hacia los demás.

Formato de salida

La salida contendrá NV líneas, donde cada línea contendrá el costo total $C_{i,j}$ de ir desde v_i hasta j, o -1 si no existe tal camino o $i=j$.
Los costos $C_{i,j}$ deben aparecer ordenados en el archivo de salida según el orden lexicográfico del par (i,j), con $1 \leq i \leq N$ y $1 \leq j \leq V$.

```
C1,1 si hay un camino de v1 a 1; -1 si no o v1=1
C1,2 si hay un camino de v1 a 2; -1 si no o v1=2
…
C1,i si hay un camino de v1 a i; -1 si no o v1=i
…
C1,V si hay un camino de v1 a V; -1 si no o v1=V
…
C2,1 si hay un camino de v2 a 1; -1 si no o v2=1
…
Ck,V si hay un camino de vk a V; -1 si no o vk=V
```

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

…
$C_{N,V}$ si hay un camino de v_N a V; -1 si no o v_N=V

Claramente, se puede observar que la línea m corresponde al costo del camino más corto de $v_{m/N}$ a w=m%N.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

6. ÁRBOL CUBR. MÍN. **Nom. de arch.**: ejercicio6.cpp/Ejercicio6.java

Implementar un algoritmo que dado un grafo *no dirigido y ponderado* devuelva el *costo total* del árbol de cubrimiento mínimo.

No se exige un algoritmo en particular.

Formato de entrada

(Ver Formato de entrada)

Formato de salida

El archivo de salida debe contener un solo número *sin EOL (fin de línea)* indicando el costo total (la suma del costo de cada arista) del árbol en caso de ser un grafo conexo, -1 en caso contrario (grafo no conexo).

UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

7. MIN PASES **Nom. de archivo**: ejercicio7.cpp/Ejercicio7.java

Dada una matriz de enteros de tamaño MxN, en la cual cada celda puede contener un valor positivo, negativo o cero, determinar la mínima cantidad de pases necesarios para convertir todo valor negativo a positivo.

En cada pase, las celdas con valores positivos podrán convertir los valores de sus celdas adyacentes, de ser necesario, de valores negativos a positivos.
Es decir, para una celda con posición (i, j), si la misma tiene valor positivo, podrá convertir los valores de las celdas (i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1).

Las celdas que contienen 0 se considerarán vacías y deberán ser ignoradas.
En la matriz recibida siempre se podrá llegar de cualquier celda con número distinto a 0 a cualquier otra con número distinto a 0; el camino no podrá estar completamente bloqueado. A su vez, siempre existirá al menos una celda con número positivo.

Ejemplo:
Matriz parámetro:

-1	-9	0	-17	0
-5	-3	-2	9	-7
2	0	0	-6	0
0	-4	-3	5	4

Ignorando las celdas con valor 0, tenemos la siguiente matriz:

-1	-9		-17	
-5	-3	-2	9	-7
2			-6	
	-4	-3	5	4

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

En el primer pase, solo los números 2, 9 y 5 son positivos; por lo tanto solo podremos cambiar a positivo las celdas adyacentes a estos

-1	-9		17	
5	-3	2	**9**	7
2			6	
	-4	3	**5**	4

En el segundo pase, el 5, 2 y el 3 tienen celdas adyacentes con valores negativos para cambiar:

1	-9		17	
5	3	**2**	9	7
2			6	
	4	**3**	5	4

Finalmente, en el tercer pase, 1 y 3 pueden convertir el valor de -9 a positivo por ser adyacentes al mismo:

1	9		17	
5	**3**	2	9	7
2			6	
	4	3	5	4

Por tanto, para la matriz recibida, el mínimo número de pases necesarios para convertir todo valor negativo a positivo es 3.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Formato de entrada

```
M
N
K11 K12 ... K1N
K21 K22 ... K2N
...
KM1 KM2 ... KMN
```

La primera línea M indica la cantidad de filas de la matriz. La segunda línea N indica la cantidad de columnas. Las siguientes M líneas corresponden a las filas de la matriz. Por cada fila, habrá N números que corresponden a los elementos pertenecientes a cada columna de dicha fila, separados por un espacio.

Formato de salida

La salida contendrá un solo número sin EOL que corresponderá a la mínima cantidad de pases necesarios para convertir todos los valores negativos a positivos.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

8. COMP. CONEXAS

Nom. de arch.: ejercicio8.cpp/Ejercicio8.java

Dado un grafo *no dirigido y no ponderado* implementar un algoritmo que devuelve la *cantidad* de componentes conexas.

Formato de entrada

(Ver Formato de entrada)

Formato de salida

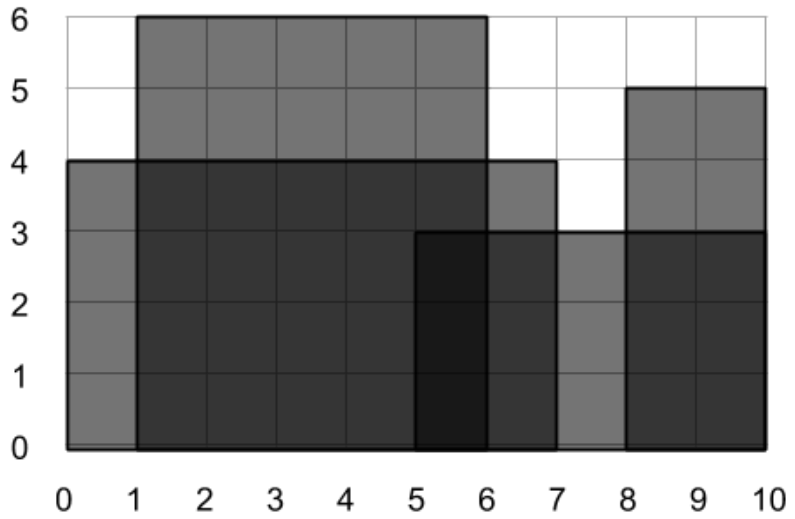
El archivo de salida debe contener un solo número sin EOL indicando la cantidad de componentes conexas.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

9. SILUETAS

Nom. de archivo: ejercicio9.cpp/Ejercicio9.java

Dada una vista horizontal de una ciudad que consta únicamente de edificios rectangulares, le piden diseñar un algoritmo que dada la ubicación y altura exacta de los edificios de la ciudad, halle la "silueta" o línea de corte de todos los edificios con el cielo, es decir, la silueta de todos los edificios juntos, pero eliminando las intersecciones ocultas entre los mismos. Por ejemplo, sean los siguientes cuatro edificios:

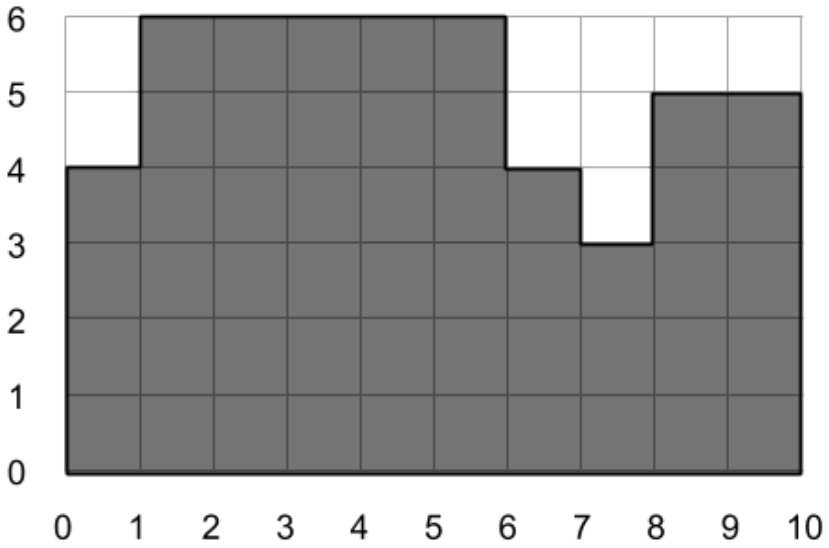


Los mismos se representarán mediante la secuencia de tripletas $[(0,7,4), (5,10,3), (1,6,6), (8,10,5)]$, donde cada componente de la tripleta (I,F,H) representa: I la coordenada en el eje horizontal donde comienza el edificio, F la coordenada donde termina y H la altura del mismo.

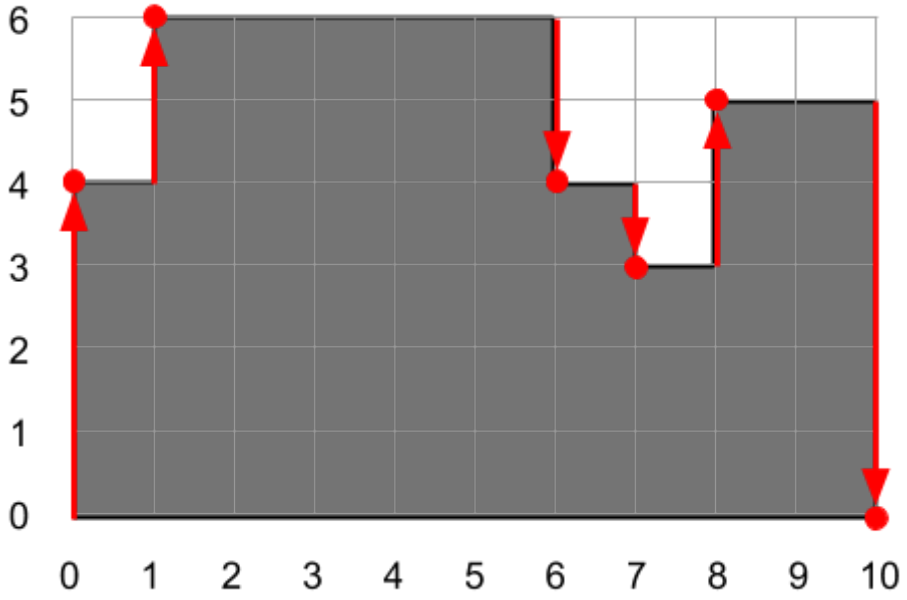
Se debe resolver en tiempo $O(N \log N)$ *usando la técnica "Dividir para conquistar" o "Divide and conquer"*, siendo N la cantidad de edificios.

La solución al problema anterior deberá ser la siguiente:

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy



La solución retornada debe ser una secuencia de *tuplas*. Cada tupla representa una "tira" o vector en el plano dada por el punto (x,h) de destino. El punto de origen quedará determinado por la altura de la tira anterior. Para entenderlo mejor, veamos qué debería retornar el ejemplo anterior. La solución es la siguiente: $[(0,4), (1,6), (6,4), (7,3), (8,5),(10,0)]$.



Inicialmente, convendremos que la silueta comienza con $h_0=0$. La primera tira de la secuencia es $(0,4)$. Esto quiere decir que el vector comienza desde $(0,0)$ y va hasta $(0,4)$. En la segunda tira, tenemos $(1,6)$. Como la altura actual de la silueta $h_1=4$ (punto anterior), ahora la nueva tira va desde $(1,4)$ hasta $(1,6)$. Y así, sucesivamente.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Formato de entrada

```
N
I1 F1 H1
I2 F2 H2
...
IN FN HN
```

La primera línea, N, indica la cantidad de edificios que se recibirán. Las siguientes N líneas contiene tres números I_i F_i H_i representando a la tripleta (I_i,F_i,H_i) que describe al edificio i, es decir, siendo I_i su coordenada inicial, F_i la final y H_i su altura.

Formato de salida

La salida contendrá una secuencia de tuplas describiendo la silueta final, cada una separada por EOL.
La secuencia devuelta no puede contener tiras consecutivas con la misma altura, es decir, no puede existir ninguna ocurrencia $[..., (x_i,h),(x_{i+1},h), ...]$.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

10. A ÚLTIMO MOMENTO

Nom. de archivo: ejercicio10.cpp/Ejercicio10.java

El obligatorio de Estructuras de datos y algoritmos 3 es muy similar al de su materia antecesora, con la pequeña diferencia de que no todos los ejercicios tienen el mismo puntaje.

Un día antes de la entrega, el profesor le recuerda que la entrega de todos los archivos no debe superar los S MB (megabytes) ni las L líneas de código entre todos ellos.

Como usted olvidó este detalle y no tiene tiempo a refactorizar su código para que cumpla con las restricciones, decide elegir aquellos ejercicios/archivos que le garanticen un mejor puntaje.

De cada ejercicio/archivo, usted sabe su tamaño, cantidad de líneas de código y el puntaje.

Restricciones:

- $O(N*S*L)$ temporal y espacial
- Debe realizarse con la táctica de **programación dinámica**.

Formato de entrada

```
N
S
L
t1 l1 p1
t2 l2 p2
...
tN lN pN
```

La primera línea (N) es la cantidad de archivos, la segunda (S) y tercera (L) es la cantidad máxima de tamaño y líneas de código respectivamente para la entrega. Por último siguen N líneas con la información de tamaño, líneas de código y puntaje de cada archivo/entrega.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Formato de salida

Contendrá 1 sola línea con el puntaje máximo que se podría obtener dada la entrega realizada.

Nota: Cada ejercicio es un archivo para la entrega.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

11. A ÚLTIMO MOMENTO (BACKTRACKING)

Nom. de archivo: ejercicio11.cpp/Ejercicio11.java

Usted se da cuenta que se perdió la clase de Programación Dinámica del curso por asistir a un recital por Zoom de Nestor en Bloque. Negoció con el profesor y éste lo autorizó a realizar el ejercicio 10 con BackTracking.

Restricciones:

- Debe realizarse con la táctica de **backtracking.**

Formato de entrada

```
N
S
L
t1 l1 p1
t2 l2 p2
…
tN lN pN
```

La primera línea (N) es la cantidad de archivos, la segunda (S) y tercera (L) es la cantidad máxima de tamaño y líneas de código respectivamente para la entrega. Por último siguen N líneas con la información de tamaño, líneas de código y puntaje de cada archivo/entrega.

Formato de salida

Contendrá 1 sola línea con el puntaje máximo que se podría obtener dada la entrega realizada.

Nota: Cada ejercicio es un archivo para la entrega.

UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

RECORDATORIO: IMPORTANTE PARA LA ENTREGA

Obligatorios (Cap.IV.1, Doc. 220)

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. La entrega se realizará desde **gestion.ort.edu.uy**
2. Previo a la conformación de grupos cada estudiante deberá estar inscripto a la evaluación. S**ugerimos realizarlo con anticipación.**
3. **Uno de los integrantes del grupo de obligatorio será el administrador del mismo** y es quien formará el equipo y subirá la entrega
4. Cada equipo debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar)
5. El archivo a subir debe tener **un tamaño máximo de 40mb**
6. Les sugerimos **realicen una 'prueba de subida' al menos un día antes**, donde conformarán el **'grupo de obligatorio'.**
7. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
8. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc)
9. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor pasar por la oficina del Coordinador o por Coordinación adjunta **antes de las 20:00hs.** del día de la entrega

Si tuvieras una situación particular de fuerza mayor, debes dirigirte con suficiente antelación al plazo de entrega, al Coordinador de Cursos o Secretario Docente.