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**	27/4
MATERIA	Estructura de Datos y Algoritmos 2				
CARRERA	Ingeniería en Sistemas				
CONDICIONES	- Puntos: Máximo: 30 Mínimo: 1 - Fecha máxima de entrega: 01/7 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 1 de 2020, está compuesto por un conjunto de **13** 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 “miSolucion < input > output” trabajado durante el curso.
- 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 solución de cada ejercicio deberá estar contenida en un único archivo.

La entrega:

- Se deberá entregar un zip, únicamente con el código fuente de cada ejercicio, en una estructura de archivos como la siguiente:

```
├──── ejercicio1.cpp
├──── ejercicio2.cpp
...
└──── ejercicioN.cpp
```

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.

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

- Se utilizará MOSS para detectar copias.

La defensa:

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

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

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
n1
n2
...
nN
```

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.

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. DICCIONARIO **Nom. de arch.**: ejercicio2.cpp/Ejercicio2.java

Se desea construir un programa que opere eficientemente buscando palabras de un diccionario utilizando la técnica de hashing.

Formato de entrada

```
N
palabra_1
palabra_2
...
palabra_N
M
palabra_test_1
palabra_test_2
...
palabra_test_M
```

La primera línea, indica cuántas palabras contendrá el diccionario: $N \leq 10^6$. Cada una de las siguientes N líneas son las palabras del diccionario. Las palabras tendrán no más de 20 caracteres (recordar el carácter de finalización)

Luego viene otro número, $M \leq 10^6$, que indica cuántas palabras se desean buscar o determinar si existen en el diccionario. Es decir, primero se cargan N palabras y luego se consultan M palabras. Claramente, para que el ejercicio sea producente, algunas de las M palabras a buscar estarán en el diccionario y otras no.

Formato de salida

La salida contendrá M líneas indicando en cada línea `i` si $\texttt{palabra\_test}_i$ pertenece al diccionario (`1`) o no (`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 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: $\{(1,2), (2,1)\}$.

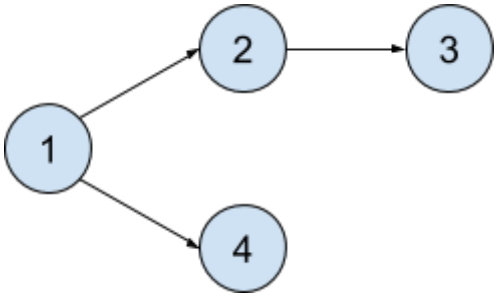
Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

4. ORDEN TOP.

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

Implementar un algoritmo que dado un grafo *dirigido, no ponderado, débilmente conexo, disperso y acíclico* devuelva el orden topológico.

Ante la posibilidad de varios órdenes topológicos, desarrollar la estrategia que se describe a continuación. Supongamos el siguiente grafo:



Existen tres órdenes, a saber: <1,2,3,4>, <1,4,2,3> y <1,2,4,3>. Nuestro algoritmo debe retornar el último. La estrategia consiste en las siguientes dos condiciones:

1. Ante la posibilidad de dos o más órdenes, siempre retornar el que contenga más cantidad de nodos "consecutivos en el mismo nivel". Con esto último queremos decir que tanto <1,2,4,3> como <1,4,2,3> mantienen a 2 y 4 consecutivos, mientras que <1,2,3,4> no. Decimos que 2 y 4 pertenecen la mismo nivel porque se encuentra a la misma cantidad de saltos desde el primer vértice en el orden[1].
2. En caso de que existan varios órdenes *luego de aplicar la condición anterior*, elegir el que sitúe a los nodos consecutivos en orden ascendente. En el ejemplo, el orden final será <1,2,4,3>.

El algoritmo debe ejecutarse en $O(E + V \log V)$ tiempo de ejecución.

Formado de entrada

(Ver Formato de entrada)

[1] Notar que si el grafo tuviera varios vértices iniciales, i.e, de grado de incidencia cero, esto no afectará la determinación de la condición o restricción.

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

La salida contendrá V líneas, donde cada línea en la posición i representa al vértice v_i que se encuentra en la posición i del orden topológico.

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 $O(E \log V)$.

Formato de entrada

(Ver Formato de entrada)

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

La siguiente línea contiene un número $N \ll V$ (mucho menor a V), 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 vi=i
...
C1,V si hay un camino de v1 a V; -1 si no o vi=V
...
C2,1 si hay un camino de v2 a 1; -1 si no o vi=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

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

Claramente, se puede observer 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, ponderado y conexo* devuelva el *costo total* del árbol de cubrimiento mínimo.

El algoritmo debe ejecutarse en $O(E \log V)$ tiempo de ejecución. *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.

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. COMP. CONEXAS **Nom. de arch.**: ejercicio7.cpp/Ejercicio7.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

8. CICLICIDAD

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

Dado un grafo *disperso, sin ponderar y dirigido*, determinar en tiempo $O(E+V)$ si tiene uno o más ciclos.

Formato de entrada

(Ver Formato de entrada)

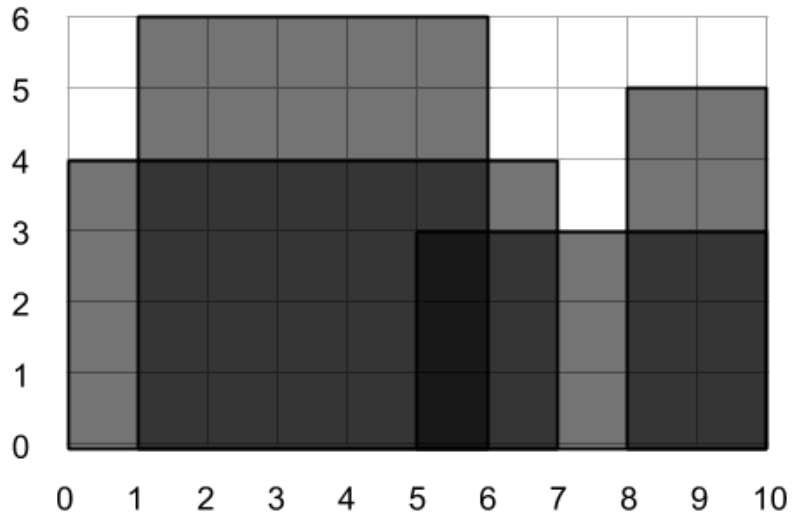
Formato de salida

La salida contendrá un solo número sin EOL: `1` si tiene ciclos o `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

9. SOMBRAS **Nom. de archivo**: ejercicio9.cpp/Ejercicio9.java

Dado un conjunto de sombras de bloques de distintos tamaño proyectados sobre un pared, se desea obtener el contorno total de las mismas. Por ejemplo, sean las siguientes cuatro sombras:



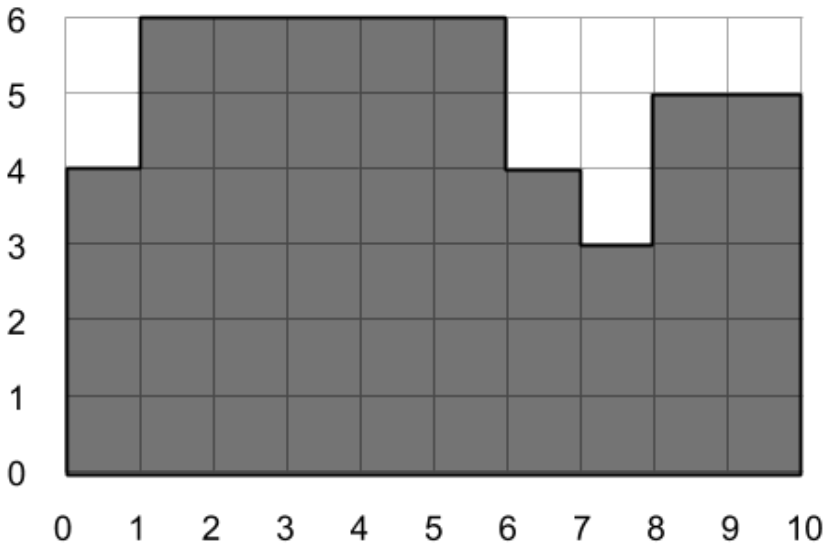
Las mismas se representarán mediante la secuencia de tripletas $[(0,7,4), (5,10,3), (1,6,6), (8,10,5)]$, donde cada componente de la tripleta (I,F,H) representa: I la coordenada en el eje horizontal donde comienza la sombra, F la coordenada donde termina y H la altura de la misma.

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

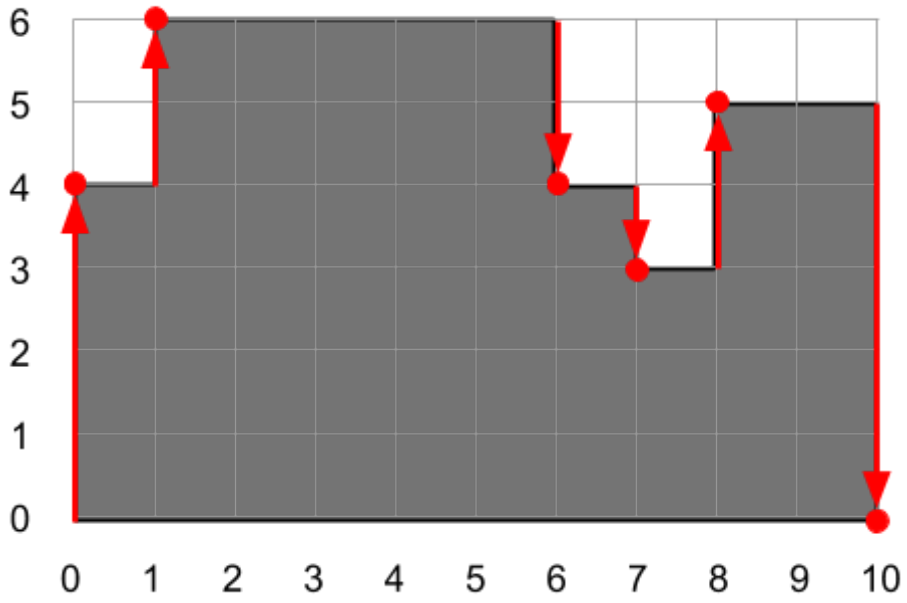
Puede asumir que *N siempre será potencia de 2*, es decir, $\exists k.2^k=N$. Por ejemplo, $N=1,2,4,8,\ldots,2^k$.

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 NH1
I2 F2 H2
...
IN FN HN
```

La primera línea, N, indica la cantidad sombras 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 a la sombra 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 sombra final, cada una separada por EOL.
La secuencia devuelta no puede contener tiras consecutivas con 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. LA CONSTR.

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

En un edificio en construcción se desea subir y bajar materiales de un piso a otro. Los materiales son situados en un sistema electrónico de poleas *instalado en un piso*, el cual es controlado por un *operador*, situado en el mismo. En un piso dado puede haber o bien un operador, o bien una *cuadrilla de obreros* que necesitan los materiales, pero *no ambos*. Debido a la alta demanda de materiales por parte de los obreros, cada uno de estos sistemas *solo puede atender a una cuadrilla*.
Dado un edificio de N pisos, con cierta instalación de poleas y organización de obreros, y sabiendo que cada sistema solo puede trasladar materiales K pisos arriba o abajo, se desea saber la máxima cantidad de pisos con cuadrillas que se pueden atender.
Los edificios serán representados con una secuencia de letras O y C, las cuales representan a los operadores y cuadrillas, respectivamente.

Nota: Se sugiere que su solución sea greedy.

Formato de entrada

Se tiene un archivo el cual contendrá todos los casos de prueba. La primera línea tendrá un número representando la cantidad de casos de prueba, llamémosle P. Las siguientes 3P líneas contendrán a los casos de prueba. Cada caso de prueba consta de tres líneas, en donde: la primera representa el K mencionado previamente en la letra, la segunda N, y finalmente la tercera representa al edificio.

```
P
K1
N1
X1 X2 X3 X4 ... XK1
K2
N2
X1 X2 X3 X4 ... XK2
...
KP
NP
X1 X2 X3 X4 ... XNP
```

Con X_i=O ó X_i=C.

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

Por ejemplo, el siguiente archivo representa un solo caso, el cual es un edificio de cinco pisos con cierta organización y con máquinas que pueden transportar materiales solo un piso (hacia arriba o abajo).

```
1
1
5
O C C O C
```

Formato de salida

La salida tendrá P líneas, cada una correspondiendo a cada caso de prueba.

Siguiendo el ejemplo anterior, la misma deberá ser:

```
2
```

Es decir, para el primer y único caso corresponde la primera y también única línea con “2”, indicando que en el edificio anterior la máxima cantidad de piso atendidos es 2.

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

11. COMBINATORIA **Nom. de arch.**: ejercicio11.cpp/Ejercicio11.java

Dados dos números, N y K, se pide retornar el resultado de calcular $C(N,K)$ o $\binom{N}{K}$, es decir, la cantidad de maneras de tomar N elementos en grupos de K sin repetir, con $0 \leq K \leq N$.

Se pide resolver en tiempo $O(NK)$ y espacio $O(N)$.

Por ejemplo, $C(5,2)=10$. Corresponde a los siguientes conjuntos: $|\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,5\}|=10$.

Formato de entrada

```
N
P
N1 K1
N2 K2
...
NP KP
```

La primera línea indica las cota N para los siguientes P casos, es decir, $N_i \leq N$ y $K_i \leq N$ para todo i con $1 \leq i \leq P$. La segunda indica la cantidad de casos de prueba, P. Las siguientes P líneas contienen cada caso dado por dos números N_i y K_i con $1 \leq i \leq P$ y $0 \leq K_i \leq N_i \leq N$ *ordenados* según el par (N_i,K_i) para los cuales se desea calcular la cantidad de combinaciones $C(N_i,K_i)$.

Formato de salida

```
C(N1,K1)
C(N2,K2)
...
C(NP,KP)
```

La salida contendrá P líneas, cada una indicando la cantidad de combinaciones $C(N_i,K_i)$ con $1 \leq i \leq P$.

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

12. SUBCONJ. DE SUMA M **Nom.**: ejercicio12.cpp/Ejercicio12.java

Dados 1) un conjunto C de $N>0$ enteros positivos estrictos K_i, es decir, $K_i>0$ para todo i con $1 \leq i \leq N$, y 2) un entero positivo M, se desea saber si existe un subconjunto S tal que $\sum s_j = M$, con $s_k \in S$.

Se debe resolver en tiempo y espacio $O(NM)$.

Por ejemplo, Si $C=\{3,34,4,12,5,2\}$ y $M=9$ entonces sí existe $S=\{4,5\}$.

Formato de entrada

```
N
K1
K2
...
KN
M
P
M1
M2
...
MP
```

La primera línea N indica cuántos elementos tiene C. Las siguientes N líneas son los elementos. Luego, sigue un número M que es una cota de todos los casos M_i que finalizan a la entrada. La siguiente línea indica la cantidad P de casos de prueba. Los siguientes P números indican los casos M_i con $1 \leq i \leq P$ para los cuales se desea conocer si existe S.

Formato de salida

La salida contendrá P líneas indicando si existe S (`1`) o no (`0`).

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

13. LABERINTO

Nom. de archivo: ejercicio13.cpp/Ejercicio13.java

Dada una matriz de MxN casilleros, cada uno con un número $C_{x,y}>0$ con $1 \le x \le M$ y $1 \le y \le N$ o un espacio en blanco (representado por 0), se desea obtener el camino desde un casillero (x_i,y_i) hacia uno (x_f,y_f) de menor suma cuyo valor no supere un umbral K>0 y que no pase por casilleros en blanco. Dado un casillero, solamente se puede mover hacia arriba, abajo, izquierda o derecha (no se permiten movimientos diagonales).

Por ejemplo, dado el siguiente tablero de 4x5 con (1,3) y (4,1) posiciones inicial y final respectivamente, y un umbral K=10, la solución se encuentra coloreada en :

0	1	2	1
0	2	1	0
1	1	3	1
0	1	0	0
0	1	0	0

Formato de entrada

```
M N
K
C1,1 C1,1 C2,1 … CM,1
C1,2 C1,2 C2,2 … CM,2
C1,3 C1,3 C2,3 … CM,3
…
C1,N C1,N C2,N … CM,N
P
xi,1 yi,1 xf,1 yf,1
xi,2 yi,2 xf,2 yf,2
…
xi,P yi,P xf,P yf,P
```

Los primeros dos números indican las dimensiones de la matriz. Le sigue el umbral K. Los siguientes MxN números representan a la matriz como se indica en la tabla. Luego sigue P indicando la cantidad de casos de prueba. Por último, siguen 4P

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

números representando las ordenadas y coordenadas de los puntos iniciales y finales, respectivamente, en el orden indicado.

Formato de salida

La salida contendrá un solo número sin EOL con la *suma total* del camino o 0 si no existe.

Claramente si la suma del camino mínimo es mayor a K entonces se debe retornar 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

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.