EDA-2019-FEBEjercicios.pdf

Anónimo

Estructura de Datos y Algoritmos

2º Grado en Desarrollo de Videojuegos

Facultad de Informática
Universidad Complutense de Madrid

Estructuras de Datos y Algoritmos — Examen final ordinario 2018/19
Grado en Desarrollo de Videojuegos. 2º V
Facultad de Informática, UCM

Instrucciones:

- Esta segunda parte del examen dura **2 horas y 10 minutos**.
- En el enlace «**Material para descargar**» dentro del juez tenéis un ZIP con las transparencias de la asignatura, el código de los TADs, las plantillas de solución del juez y otros documentos.
- En el enlace «**C++ reference**» dentro del juez (botón en la parte superior) tenéis disponible la documentación completa de la STL.
- Se valorará la calidad del código, la eficiencia, la inclusión del coste de las funciones/métodos involucrados y la corrección con respecto a los casos de prueba.
- Si no os da tiempo a terminar el ejercicio incluid un comentario lo más completo posible describiendo la idea de la solución y su coste. Aunque no haya nada de código, este comentario puede ser evaluable.
- El fichero enviado debe contener una cabecera indicando vuestro nombre completo.

1. (4 pt) Un camión tiene capacidad para C m^3 de artículos. Existen N artículos que se pueden cargar, cada una con un beneficio de b_i € y un volumen de v_i m^3. Además, una vez cargado el camión se debe llenar todo el espacio vacío con espuma de embalar para evitar que la mercancía se mueva durante el transporte, pero cada m^3 de espuma cuesta E €. Si el camión viaja vacío no hace falta llenar el espacio con espuma, ya que no hay nada que se pueda mover.

Aplica la técnica algorítmica que mejor se adapte al problema («*divide y vencerás*» o «*vuelta atrás*») para desarrollar un programa que calcule el máximo beneficio que se puede obtener al cargar el camión. Tened en cuenta que dependiendo del precio de la espuma el beneficio de cargar artículos puede ser negativo, y en esas ocasiones es preferible hacer el trayecto con el camión vacío y tener un beneficio de 0 €.

Entrada
La entrada consta de varios casos de prueba. Cada caso comienza con una línea «`C E N`» de valores separados por espacios indicando la capacidad del camión ($C \geq 1$), el precio de un m^3 de espuma ($E \geq 0$) y el número de artículos disponibles para cargar ($1 \leq N \leq 70$). A continuación aparecen N líneas «b_i v_i» indicando el beneficio ($b_i \geq 1$) y el volumen ($v_i \geq 1$) de cada artículo. La entrada termina con un caso especial con los valores $C = E = N = 0$ que no debe procesarse.

Salida
Para cada caso de prueba la salida será una línea indicando el máximo beneficio que se puede obtener al cargar el camión con los artículos disponibles, sin poder sobrepasar en ningún caso la capacidad máxima.

Ejemplo de entrada

```
10 1 3
```

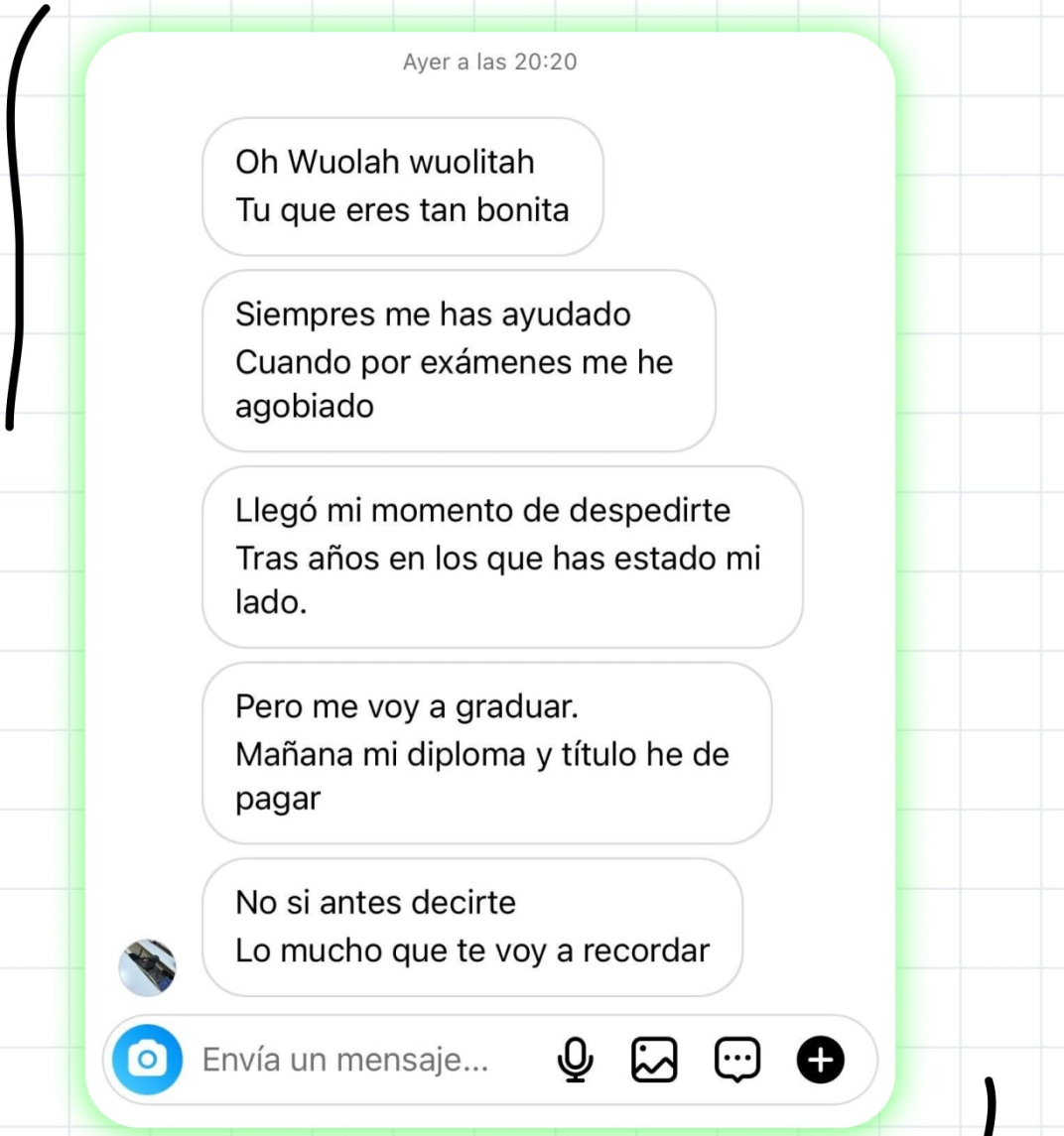
```
14 7
7 4
7 4
10 5 2
20 6
10 7
10 1 3
20 10
1 8
2 9
10 0 3
10 6
5 2
1 2
0 0 0
```

Ejemplo de salida

```
12
0
20
16
```

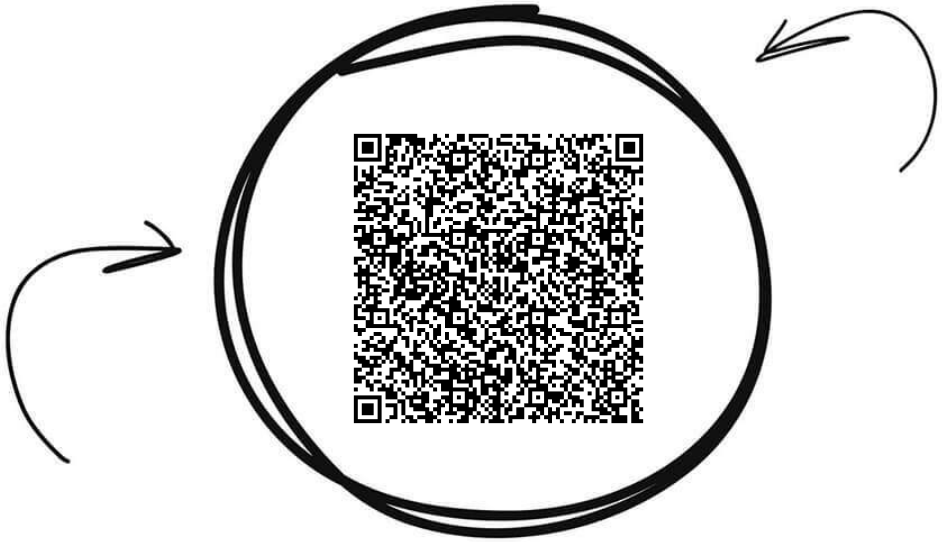
Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

Estructura de Datos y Algori...



Banco de apuntes de la UCM

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Estructuras de Datos y Algoritmos — Examen final ordinario 2018/19
Grado en Desarrollo de Videojuegos. 2º V
Facultad de Informática, UCM

Instrucciones:

- Esta segunda parte del examen dura **2 horas y 10 minutos**.
- En el enlace «**Material para descargar**» dentro del juez tenéis un ZIP con las transparencias de la asignatura, el código de los TADs, las plantillas de solución del juez y otros documentos.
- En el enlace «**C++ reference**» dentro del juez (botón en la parte superior) tenéis disponible la documentación completa de la STL.
- Se valorará la calidad del código, la eficiencia, la inclusión del coste de las funciones/métodos involucrados y la corrección con respecto a los casos de prueba.
- Si no os da tiempo a terminar el ejercicio incluid un comentario lo más completo posible describiendo la idea de la solución y su coste. Aunque no haya nada de código, este comentario puede ser evaluable.
- El fichero enviado debe contener una cabecera indicando vuestro nombre completo.

1. (3 pt) El lanzamiento de la nueva consola retro «PAYStation 2 mini» puede ser un caos, así que la compañía Pony ha decidido gestionar de manera centralizada todas las reservas y las unidades que envía a cada tienda. Para ello necesita un sistema que reciba estas acciones y las registre convenientemente. Concretamente hay dos tipos de acciones:

- Un **cliente** realiza una **reserva** en una **tienda**.
- Se **envía** una consola al cliente con la reserva más antigua en una **tienda**. Una vez se envía una consola, la reserva afectada desaparece.

Pony necesita un programa que registre estas acciones y al final muestre un resumen de tiendas junto con los clientes que tienen una reserva pendiente, ordenados por antigüedad. Para resolver este problema es necesario utilizar **únicamente estructuras de la STL de C++**.

Entrada
La entrada consta de varios casos de prueba. Cada caso comienza con una línea conteniendo el número $1 \leq A \leq 1000$ de acciones a procesar. A continuación aparecen A líneas indicando una acción cada una. Las acciones se representan como:

- «`RESERVA cliente tienda`»: registra la reserva de `cliente` para la `tienda` indicada.
- «`ENVIA tienda`»: envía una consola al siguiente cliente con reserva en `tienda`. Si dicha `tienda` no tiene ninguna reserva en ese momento, esta acción se ignora.

Tanto los nombres de tiendas como de clientes son cadenas de letras en minúsculas con una longitud entre 1 y 100 letras. La entrada termina con un caso especial con valor $A = 0$ que no debe procesarse.

Salida

Para cada caso de prueba la salida será la descripción de las reservas que quedan. Se mostrarán las tiendas en orden alfabético, y para cada una de ellas se indicará el nombre de los clientes que están a la espera en el orden en que recibirían la consola. El formato de la línea asociada a cada tienda es «`tienda -> LISTA_CLIENTES`», donde `LISTA_CLIENTES` es una secuencia de nombres de clientes separados por un espacio. Todas las tiendas que han recibido alguna reserva deben aparecer en el listado, aunque en el momento final ya no tengan ninguna reserva pendiente. **Después de cada caso de prueba se debe dejar una línea en blanco**.

Ejemplo de entrada

```
5
RESERVA juanra game
RESERVA loli fnac
RESERVA ana fnac
ENVIA game
RESERVA eva eci
2
RESERVA monchi fnac
ENVIA eci
3
RESERVA monchi fnac
ENVIA fnac
ENVIA fnac
0
```

Ejemplo de salida

```
eci -> eva
fnac -> loli ana
game ->

fnac -> monchi

fnac ->
```