INF-221 Algoritmos y Complejidad, 2023-2

Tarea 1

Profesor/es: Juan P. Castillo, Miguel Truffa
Ayudantes: Carlos Lagos, Vicente Lineros, Bayron Valenzuela
carlos.lagosc@usm.cl
vicente.lineros@usm.cl
bayron.valenzuela@sansano.usm.cl

Fecha de Inicio: 2 de octubre, 2023
Fecha de Entrega: 20 de octubre, 2023
Plazo máximo de entrega: 5 días.

Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con el ayudante **ANTES** de comenzar la tarea. No se permiten de ninguna manera grupos de más de 3 personas. Pueden usarse los lenguajes de programación C, C++, Python, y Java.

Problema 1: Fuerza Bruta Recursiva

Suponga que tiene un conjunto $S[1..n]$ de n cajas 3D rectangulares, donde la i-ésima caja tiene: altura h_i, ancho w_i y profundidad d_i, tal que $h_i, w_i, d_i \in \mathbb{N}$. Dado el conjunto $S[1..n]$, se necesita encontrar cual es la pila de cajas más alta que se puede hacer utilizando las cajas de S (pueden quedar cajas sin utilizar), siguiendo las siguientes restricciones:

I) Solo se puede apilar una caja arriba de otra si:

- Las dimensiones de la base de la caja de abajo es estrictamente más grande que la caja que se colocará encima, es decir, $w_{j-1} > w_j$ y $d_{j-1} > d_j$.

II) Se pueden rotar las cajas para que cualquiera de sus caras tenga la función de base.

- Estamos interesados específicamente en las 3 rotaciones donde el valor de h_i cambia, es decir, el orden en que w_i y d_i se intercambien no nos interesa.

III) Puede utilizar más de una instancia de la misma caja, es decir, puede utilizar las rotaciones de una misma caja como si fueran distintas. Sin embargo, no puede utilizar la misma rotación múltiples veces.

Implemente un algoritmo de **fuerza bruta recursiva** que encuentre la altura máxima de la pila de cajas que se puede crear, siguiendo las restricciones antes mencionadas.

Hint: podría ser de utilidad estudiar (por su cuenta) el problema de la *subsecuencia creciente más larga*, o *longest increasing subsequence* (LIS) en inglés. Hay mucho material disponible al respecto en la web.

Formato de Entrada

La entrada es leída desde la entrada standard, y contiene varios casos de prueba. Por cada caso hay un entero n que indica la cantidad de cajas, luego le siguen n cajas (separadas por un salto de línea). Cada caja esta dada por 3 enteros que indican el alto (h_i), ancho (w_i) y profundidad (d_i) respectivamente (separados por un único espacio). En este caso $1 \leq n \leq 10000$. La entrada es terminada por `EOF`.

Un ejemplo de entrada es el siguiente:

```
3
1 2 3
4 5 6
1 5 4
4
3 4 5
4 4 4
1 3 4
1 1 1
```

Hint: para probar su programa de una mejor manera, ingrese los datos de entrada con el formato indicado en un archivo de texto (por ejemplo, el archivo `input-1.dat`). Luego, ejecute su programa desde la terminal, redirigiendo la entrada standard como a continuación, evitando tener que entrar los datos manualmente cada vez que prueba su programa:

```
./problema1 < input-1.dat
```

Formato de Salida

La salida debe mostrarse a través de la salida standard. Para cada caso de prueba, se debe mostrar una línea que contiene la altura de la pila más grande que se puede crear con las cajas disponibles (siguiendo las restricciones).

La salida correspondiente a la entrada mostrada anteriormente es:

```
15
12
```

Problema 2: Programación Dinámica

Resuelva el Problema 1 usando la técnica de programación dinámica. Los formatos de entrada y salida son los mismos que en el Problema 1.

Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea1-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda), o alternativamente usando formato zip, en el sitio Aula USM del curso, a más tardar el día 20 de octubre, 2023, a las 23:59:00 hrs (Chile Continental), el cual contenga:

- El plazo máximo de entrega es de a lo más 5 días desde la fecha original de entrega (20 de octubre, 2023). Por cada día (o fracción) de atraso se descontarán 20 puntos de la nota de la tarea.
- Los archivos con el código fuente necesarios para el funcionamiento de la tarea.

- `NOMBRES.txt`, Nombre y ROL de cada integrante del grupo. También se debe indicar qué hizo cada integrante del grupo.
- `README.txt`, Instrucciones de compilación en caso de ser necesarias.
- `Makefile`, Instrucciones para compilación automática, en caso de ser necesarias.