1 Problema

En esta tarea diseñarás algoritmos para resolver el problema del *knapsack*. Tendrás un *knapsack* con espacio limitado y una colección de cosas con diferentes valores y pesos. Tu tarea es maximizar el valor de las cosas seleccionadas en tu *knapsack* sin exceder la capacidad del mismo.

2 Formalización

El problema se puede formalizar como sigue:

$$\begin{aligned} \textbf{maximize} \quad & \textstyle\sum_{i \in I} v_i \cdot x_i \\ \textbf{s.t.} \quad & \sum_{i \in I} w_i \cdot x_i \quad \leq \quad Q \\ & x_i = \{0, 1\} \quad i \in I \end{aligned} \tag{1}$$

3 Formatos de datos

3.1 Datos de entrada

Cada problema está especificado en un archivo. Los archivos contienen $n + 1$ líneas. La primera línea contiene dos números: n, el número de objetos, y Q la capacidad del *knapsack*.
El resto de las líneas representan los datos para cada uno de los objetos. Cada línea tiene dos enteros, su valor v_i y su peso w_i.

3.2 Datos de salida

Tu programa deberá de generar un archivo de salida con la solución. El archivo tendrá dos líneas. La primera línea tendrá el valor solución del problema (i.e. el valor de todas las cosas seleccionadas en el *knapsack*, i.e. el valor de la función objetivo). La segunda línea tendrá una lista de n números con valor $0/1$ para cada una de las variables.

3.3 Ejemplo

Datos de entrada (archivo `data/ks_4_0`)

```
4 11
8 4
10 5
15 8
4 3
```

Datos de salida

```
19
0 0 1 1
```

4 Instrucciones

Acepta la invitación de **Github Classroom**, el repositorio tiene un nombre con el prefijo: `or-2020-knapsack-problems`.
Copia los archivos dentro de la carpeta `tareas/Tarea_1` a tu repositorio de la tarea.
Utiliza `solver.py` y modifícalo agregando tus algoritmos de optimización.
Debes de poder ejecutarlo con

```
python ./solver.py ./data/<archivo>  <algorithm>
```

donde `<algorithm>` es el nombre de la función que implementa tu algoritmo.
Crea un archivo de texto llamado `soluciones.txt` que contenga por cada archivo de datos una línea como la mostrada:

```
./data/ks_4_0  <algorithm-1>
./data/ks_40_0  <algorithm-2>
... # etc
```

Si sólo tienes un algoritmo para solucionar, todas las líneas tendrán el mismo último argumento.
NOTA: Este archivo será el utilizado para evaluar tu tarea
NOTA: Tu algoritmo debe de terminar en menos de 5 horas.
NOTA: Si decides usar otro lenguaje de programación agrega un archivo `bash` con la línea de ejecución, deberás de proveer instrucciones sobre el ambiente necesario para ejecutar tu solución y deberás respetar el formato de entrada y salida.

4.1 Recursos

El archivo `solver.py` está en la carpeta `Tarea_1`, junto con el directorio `data` y este archivo.
La especificación de los problemas del *knapsack* se encuentran en el directorio `data`.

5 Calificación

- Soluciones *infeasible* recibirán 0 puntos.
- Soluciones que no concluyan en el tiempo límite, recibirán 0 puntos
- Soluciones *feasible* recibirán por lo menos 3 puntos.
- Soluciones *feasible* que alcancen un mínimo de calidad recibirán 7 puntos
- Soluciones *feasible* que superen la barrera de alta calidad recibirán 10 puntos

6 Políticas de colaboración

Creo que los problemas de la vida real se resuelven colaborando, intercambiando ideas y cotejando soluciones, por lo tanto, los incito a realizar estas actividades. Por otro lado, la tarea es *individual*, entonces, **NO** hagan:

1. Usar código que no es suyo y que no entienden
2. Pasar el código a otros compañeros o *postearlo* en algún foro.
3. Pasar *pseudocódigo* (es lo mismo que lo anterior)

7 Advertencias

1. No modifiquen los archivos de la carpeta `data`.
2. No cambien el nombre del archivo `solver.py`
3. No uses variables globales
4. No cambies los archivos de lugar, en tu repositorio `solver.py` y la carpeta `data` deben de estar en la raíz del repositorio.

8 Créditos

Los problemas de esta tarea fueron tomados de la clase *Discrete Optimization* de los profesores **Pascal Van Hentenryck** y **Carleton Coffrin**.