Tarea 5
Algoritmos y Complejidad

Dilema

Sassy Complexes

1. Introducción

Mas de un cuarto de los problemas que aparecen en competencias y en paginas de jueces online[1] están etiquetados con la categoría **Dynamic Programing**. Los problemas de DP son populares entre los creadores de problemas por que cada problema de DP es único en cierta manera y hay que pensar mucho para inventar una solución. Por eso, sus aproblemados ayudantes les obligan a resolver uno de estos problemas por que es una habilidad útil para dominar (y por que quieren sentirse bien consigo mismos y creer que están haciendo bien su trabajo).

2. Problema

Carina, John y Rodrigo tienen n kilogramos de dulces ácidos que les obsequió el malvado doctor Von Brand. Ellos quieren repartirse el botín, pero como son personajes muy especiales quieren repartírselo en porciones de cierto peso que cumplan con ciertas condiciones:

- Carina solo quiere porciones que pesen a kilos, John quiere varias porciones de b kilos y Rodrigo solo aceptará porciones que tengan c kilos.
- Por supuesto, siendo una tarea de algoritmos, no quieren desperdiciar dulce alguno (El doctor siempre les dará una cantidad valida que puedan repartirse según sus condiciones) y maximizar la cantidad de porciones que cumplan con los pesos indicados.

2.1. Preguntas

1. Proponga y programe un algoritmo de **fuerza bruta** para resolver este problema. Utilice el lenguaje de programación que estime conveniente pero atengase a las condiciones de entrega y los formatos de entrada y salida explicados mas adelante. Guíese por los ejemplos.
2. Proponga y programe un algoritmo de **programación dinámica** para resolver este problema. Utilice el lenguaje de programación que estime conveniente pero atengase a las condiciones de entrega y los formatos de entrada y salida explicados mas adelante. Guíese por los ejemplos.
3. Haga una comparación superficial del rendimiento empírico de sus programas, muestre sus resultados y explique brevemente por qué cree que está obteniendo mejor rendimiento en uno u otro. Se evaluará lo creíble y honesta de su explicación.

2.2. Formato

Recibirá por **entrada estándar** una linea que contiene **cuatro** enteros separados n,a,b y c ($1 \leq n,a,b,c \leq 4000$), que son: la cantidad total de kilogramos a repartir, y los pesos de las respectivas porciones aceptadas por los personajes (podría suceder que todos acepten porciones iguales).
Se pide que imprima por **salida estándar** el máximo numero de porciones repartidas.

[1] Como codeforces.com/, codechef.com/ y topcoder.com/

INCUMPLIR CON EL FORMATO SE CONSIDERA UN ERROR DE PROGRAMACIÓN GRAVE Y SE EVALUARÁ CON NOTA CERO

2.3. Ejemplos

2.3.1. Ejemplo 1

```
5 5 3 2
```

Listing 1: STDIN

```
2
```

Listing 2: STDOUT

En este primer ejemplo habían 5 kilos de dulces, Rodrigo recibió una porción de 2 kilos y John recibió una de 3 kilos.

2.3.2. Ejemplo 2

```
8 2 3 5
```

Listing 3: STDIN

```
4
```

Listing 4: STDOUT

Aquí hay 8 kilos, por lo que Carina recibirá 4 porciones de 2 kilos.

2.3.3. Ejemplo 3

```
7 5 5 2
```

Listing 5: STDIN

```
2
```

Listing 6: STDOUT

7 kilos. Rodrigo recibirá 1 porción de 2 kilos y Carina **o** John recibirá una porción de 5 kilos, no importa quien.

3. Condiciones de Evaluación de código

Se evaluará:

- **Ejecución correcta**: que funcionen los casos de prueba y **NO** sea posible encontrar casos en que el programa entregue una respuesta equivocada.
- **Complejidad computacional adecuada**: Que el algoritmo implementado tenga una complejidad igual o mejor que la esperada, y que sea ad-hoc a la materia que se está evaluando (e.g. no utilizar programación dinámica si se pide programar un algoritmo voraz).
- **Calidad del programa**: uso adecuado de funciones, uso de estructuras de control, uso de estructuras de datos, código claro y simple.
- **Código ordenado**: nombres adecuados, indentación correcta, comentarios suficientes, ausencia de código comentado.

Para más información refiérase a las condiciones de evaluación del código publicadas en moodle

4. Condiciones de entrega

- La tarea se realizará *individualmente* (esto es grupos de una persona), sin excepciones.
- La entrega debe realizarse vía Moodle en un *tarball* en el área designada al efecto, bajo el formato `tarea-5-`*`rol`*`.tar.gz` (`rol` con dígito verificador y sin guión).

  Dicho *tarball* debe contener las fuentes en LaTeX (al menos `tarea.tex`) de la parte escrita de su entrega, además de un archivo `tarea-5.pdf`, correspondiente a la compilación de esas fuentes.
- En caso de haber programas, su ejecutable *debe* llamarse `tarea-5`, de haber varias preguntas solicitando programas, estos deben llamarse `tarea-5-1`, `tarea-5-2`, etc. Si hay programas compilados, incluya una `Makefile` que efectúe las compilaciones correspondientes.

  Los programas se evalúan según que tan claros (bien escritos) son, si se compilan y ejecutan sin errores o advertencias según corresponda. Parte del puntaje es por ejecución correcta con casos de prueba. Si el programa no se ciñe a los requerimientos de entrada y salida, la nota respectiva es cero.
- Además de esto, la parte escrita de la tarea debe en hojas de tamaño carta en Secretaría Docente de Informática (Piso 1, edificio F3).
- Tanto el *tarball* como la entrega física deben realizarse el día indicado en Moodle. No entregar la parte escrita en papel o no entregar en formato electrónico tiene un descuento de 50 puntos.

  Por cada día de atraso se descontarán 20 puntos. A partir del tercer día de atraso no se reciben más tareas, y la nota de la tarea es cero.
- Nos reservamos el derecho de llamar a interrogación sobre algunas de las tareas entregadas. En tal caso, la nota base (antes de descuentos por atraso y otros) es la de la interrogación. No presentarse a la interrogación sin justificación previa significa automáticamente nota cero.