INF-221 Algoritmos y Complejidad, 2022-1

Tarea 1

Profesores: Diego Arroyuelo, Juan Pablo Castillo
Ayudantes: Javier Pérez, Bayron Valenzuela
javier.perezp@usm.cl
bayron.valenzuela@sansano.usm.cl

Fecha de Inicio: 30 de septiembre, 2022
Fecha de Entrega: 19 de octubre, 2022
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.

1. Problema 1: El Problema de dar Cambio

Suponga que necesita dar cambio (o vuelto) exacto para una cantidad de dinero n, y para eso dispone de monedas con denominaciones $d_1 < d_2 < \cdots < d_m$, asumiendo que siempre se va a cumplir $d_1 = 1$. Asumiendo que cuenta con una cantidad infinita de monedas de cada denominación, el problema consiste en dar cambio exacto usando la mínima cantidad de monedas posibles. Defina e implemente un algoritmo de **fuerza bruta recursiva** que reciba un arreglo $D[1..m] = [d_1, d_2, \ldots, d_m]$ con las posibles denominaciones y el valor entero $n \geq 0$, y retorne la cantidad mínima de monedas de esas denominaciones que deben usarse.

Formato de Entrada

Los datos serán leídos desde la entrada standard, en donde cada línea tendrá el siguiente formato. La primera línea contiene un único entero positivo n, indicando la cantidad de dinero que debemos dar como cambio. La segunda línea contiene un único entero m, indicando la cantidad de denominaciones distintas de las que se dispone. Luego, le siguen m líneas, cada una conteniendo un número entero correspondientes a las denominaciones $d_1, d_2, \ldots, d_m$. Un ejemplo es el siguiente:

```
30
3
1
10
25
```

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 se hará a través de la salida estándar (`stdout`). Ésta debe mostrar una única línea que contiene un número entero correspondiente la cantidad mínima de monedas a dar para la cantidad de dinero indicada. Un ejemplo de salida para el ejemplo mostrado anteriormente es:

```
3
```

2. Problema 2: Las Formas de Sumar un Número

Dada un número entero t y un arreglo L de n enteros, queremos encontrar las distintas formas de sumar t usando valores de L. Por ejemplo, si $L = \langle 6, 4, 3, 3, 3, 2, 2, 2 \rangle$ y $t = 6$, hay 4 formas distintas de obtener 6 sumando elementos de L:

- 6,
- 4+2,
- 3+3,
- 2+2+2.

L puede contener valores repetidos, por lo que un mismo valor puede ser usado en una suma tantas veces como sea necesario, siempre respetando la cantidad de veces que éste aparece en L (en el ejemplo anterior, vea los casos 2+2+2 y 3+3). Use **backtracking recursivo** para resolver el problema.

2.1. Formato de Entrada

La entrada de datos será a través de la entrada estándar (`stdin`), y contendrá varios casos de prueba. Cada caso de prueba ocupará una línea separada de la entrada, y tendrá el siguiente formato. Comienza con el valor entero t ($1 \leq t \leq 1000$), seguido por el entero n ($0 \leq n \leq 20$), indicando el tamaño de la lista L. A continuación le siguen los n valores enteros $x_1, \ldots, x_n$ que conforman la lista. Se cumple que $1 \leq x_i \leq 100$. Los valores dentro de una línea de la entrada están separados por un único espacio. Un valor $n = 0$ indica el fin de la entrada. Los valores $x_1, \ldots, x_n$ están ordenados de forma no creciente, y pueden haber valores repetidos.

Un ejemplo de entrada es el siguiente:

```
6 8 6 4 3 3 3 2 2 2
8 6 8 6 4 4 2 2
10 3 4 2 2
800 12 100 100 100 100 100 100 50 50 50 50 50 50
0 0
```

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-2.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:

```
./problema2 < input-2.dat
```

2.2. Formato de Salida

La salida se hará a través de la salida estándar (`stdout`). Debe mostrar el resultado para cada uno de los casos de prueba dados en la entrada, con el siguiente formato. Por cada caso, debe imprimir una línea que contenga el texto "`Suma de`", seguido por el valor de t correspondiente, seguido por el caracter '`:`'. Luego, se imprimen todas las maneras de sumar t encontradas por el algoritmo. Las sumas deben estar ordenadas de forma no creciente de acuerdo a los números que la conforman. Aún cuando la lista de entrada puede contener elementos repetidos, todas las sumas mostradas deber ser distintas: no se admiten sumas repetidas. Si para algún caso de prueba no hay ninguna manera de sumar elementos de la lista que permita obtener t, se debe imprimir "`NADA`".

La salida correspondiente al ejemplo mostrado anteriormente es:

```
Suma de 6:
6
4+2
3+3
2+2+2
Suma de 8:
8
6+2
4+4
4+2+2
Suma de 10:
NADA
Suma de 800:
100+100+100+100+100+100+50+50+50+50
100+100+100+100+100+50+50+50+50+50+50
```

3. 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 19 de octubre, 2022, a las 23:59:00 hrs (Chile Continental), el cual contenga:

- 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.