UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructuras de Datos I

Primer cuatrimestre de 2019

24 de Junio de 2019

Competencia de Programación

1. Entrega

La solución debe realizarse en los archivos `risas.cpp, mayas.cpp, robotitos.cpp, libros.cpp` siguiendo el encabezado provisto por la cátedra. También se debe implementar en algunos casos las dos funciones de lectura de los datos y de lectura de la respuesta correcta, siguiendo el formato de los encabezados.

Para la evaluación de la solución propuesta, los archivos deben ser enviados en formato zip a la siguiente dirección de correo electrónico: `algo1challengetm@gmail.com` hasta las 12 hs.

2. Análisis de la risa

Científicos rusos están investigando propiedades de la risa. Para ello, analizan el habla humana e identifican qué sonidos pertenecen a una risa.

Estos científicos ya hicieron un software que traduce el habla humana en texto. Consideran que la risa es una secuencia alternada de las letras "h" y "a". Por ejemplo, "ahahaha", "hah" y "a" son risas, pero "abacaba" y "hh" no.

Dado un string s encontrar la longitud de la risa más larga, es decir, un substring de s que sea risa y que sea más larga que el resto de las risas.

Deben implementar la función:

```
int risaMasLarga(string s)
```

Ejemplos

1. `risaMasLarga("ahaha")` debe ser igual a 5.
2. `risaMasLarga("ahahrunawayahahsofasthah")` debe ser igual a 4.
3. `risaMasLarga("ahahaahaha")` debe ser igual a 5.

Datos y Tests

En la carpeta `datos` se encuentran dos archivos de prueba como ejemplo del formato de entrada que sus funciones de E/S deben saber levantar. Este formato consiste en una sola línea donde cada caracter esta separado por un espacio. Por ejemplo, el contenido del archivo `01.in` es:

```
a h a h a
```

La respuesta esperada para esa entrada se encuentra en el archivo `01.a` cuyo contenido es un entero indicando la risa más larga.

Puntaje

Se va a dar un puntaje diferente de acuerdo a la cantidad de caracteres que tengan los archivos de test que pasen por el sistema de verificación de la cátedra:

Archivos	Puntaje individual por test
De "01" a "09"	0.1
De "10" a "19"	0.2
De "20" a "29"	0.3
De "30" a "39"	0.4
De "40" a "49"	0.5
De "50" a "53"	0.6
"54"	0.7
TOTAL	18

3. Descifrando la escritura Maya

Descifrar la escritura Maya siempre ha resultado ser una tarea desafiante. Durante mucho tiempo muy poco se ha sabido del tema y los avances más importantes se han dado en las últimas tres décadas.

La escritura Maya está basada en pequeños dibujos conocidos como jeroglíficos. Cada uno de ellos representa un sonido. Uno de los tantos problemas que existe al intentar descifrarla tiene que ver con el orden de lectura. Al ubicar los distintos jeroglíficos para formar una palabra, los Mayas decidían el orden principalmente por cuestiones estéticas y no por alguna regla gramatical. De esta manera, aunque los sonidos para cada jeroglífico son conocidos, los arqueólogos a veces no están seguros de cómo se pronuncia determinada palabra.

En la práctica, cuando los arqueólogos están buscando cierta palabra W, conocen cuáles son los jeroglíficos que la conforman pero no saben todas las posibles combinaciones en las que se puede encontrar la palabra. Sabiendo que se avecinaba el fin del cuatrimestre, los arqueólogos pidieron por favor que los ayudemos con este problema, pero dejaron bien en claro que solo aceptarían ayuda de los alumnos de Algo 1, en quienes confían ciegamente. Ellos les proveerán de los g jeroglíficos que conforman la palabra W y una secuencia S de todos los jeroglíficos que están estudiando. Ustedes tienen que contar la cantidad de posibles apariciones de la palabra W.

Escribir un programa que dado los jeroglíficos que conforman W y una secuencia S de jeroglíficos escritos en la pared de un templo, cuente la cantidad de posibles apariciones de W en S, es decir, cada secuencia de g jeroglíficos consecutivos que sea una permutación de los jeroglíficos de W.

Deben implementar la función:

```
int posiblesApariciones(string W, string S)
```

Ejemplo

1. `posiblesApariciones("cAda", "AbrAcadAbRa")` debe ser igual 2.

Datos y Tests

Los datos de cada transcripción están volcados en un archivo `writingX.in` que posee la siguiente estructura:

```
4 11
cAda
AbraAcadAbRa
```

La primera línea indica $|W|$ $|S|$ respectivamente (esto es a título informativo, puede no ser necesario). Luego viene W y S. En el archivo `writingX.res` se encuentra la solución, como un dato entero. Se proveen 2 archivos de test para los alumnos.

Puntaje

Se va a dar un puntaje diferente de acuerdo a la cantidad de caracteres que tengan los archivos de test que pasen por el sistema de verificación de la cátedra:

Archivos	Puntaje individual por test
De “1” a “4”	1
De “5” a “9”	2
De “10” a “12”	3
TOTAL	23

4. Robots

¡El hermanito de Marita ha dejado juguetes tirados en todo el piso del living de su casa! Afortunadamente, Marita construyó robots especiales para recogerlos. Ella necesita determinar cuál robot debe recoger cada juguete.

Hay T juguetes, cada uno con un peso $W[i]$ y un volumen $S[i]$ (ambos son números enteros). Hay robots de dos tipos: débiles y pequeños.

- Hay A robots débiles. Cada robot débil tiene un límite de carga $X[i]$, y puede cargar cualquier juguete que pese estrictamente menos que $X[i]$. El tamaño del juguete no importa.
- Hay B robots pequeños. Cada robot pequeño tiene un límite de volumen $Y[i]$, y puede cargar cualquier juguete de volumen estrictamente menor que $Y[i]$. El peso del juguete no importa.

Cada uno de los robots de Marita se demora un minuto para guardar cada juguete. Robots diferentes pueden guardar al mismo tiempo diferentes juguetes.

La tarea consiste en determinar si los robots de Marita pueden guardar todos los juguetes, y en caso de ser posible, determinar el tiempo más corto en que pueden hacerlo.

Deben implementar la función:

```
int calcularTiempoRecoleccion(int A, int B, int T,
    vector<int> X, vector<int> Y, vector<int> W, vector<int> S)
```

donde X es un vector con los límites de carga de cada robot débil, Y el límite de volumen de cada robot pequeño, W el peso de cada juguete y S su volumen. La función debe devolver el tiempo más corto en que los robots pueden juntar los juguetes y, en caso de no ser posible, debe devolver -1.

Ejemplos.

1.

Supongamos que hay $A = 3$ robots débiles con límite de carga $X = [6, 2, 9]$ y $B = 2$ robots pequeños con límite de volumen $Y = [4, 7]$ y $T = 10$ juguetes:

Número de Juguetes	Peso	Volumen
0	4	6
1	8	5
2	2	3
3	7	9
4	1	8
5	5	1
6	3	3
7	8	7
8	7	6
9	10	5

El menor tiempo para guardar todos los juguetes es 3 minutos.

	Robot débil 0	**Robot débil 1**	**Robot débil 2**	**Robot pequeño 0**	**Robot pequeño 1**
Minuto 1	Juguete 0	Juguete 4	Juguete 1	Juguete 6	Juguete 2
Minuto 2	Juguete 5		Juguete 3		Juguete 8
Minuto 3			Juguete 7		Juguete 9

2.

Ahora supongamos que hay $A = 2$ robots débiles con los límites de carga $X = [2, 5]$ y $B = 1$ robot pequeño con límite de volumen $Y = [2]$ y $T = 3$ juguetes:

Número de Juguetes	Peso	Volumen
0	3	1
1	5	3
2	2	2

Ningún robot puede recoger el juguete de peso 5 y volumen 3, por lo que es imposible para los robots poder recoger todos los juguetes.

Datos y Tests

Los archivos de entrada de datos denominados `sampleX.in` tienen la siguiente estructura:

```
2 1 3
2 5
2
3 1
5 3
2 2
```

Donde en la primera línea se dan los valores de A, B y T respectivamente. La línea siguiente presenta los A valores de $X[i]$. La tercera línea son los B valores de $Y[i]$. Las T líneas siguientes muestran las propiedades de los T juguetes presentes como un par de enteros, cada uno con su $W[j]$ y su $S[j]$. La respuesta de cada test está en el archivo `sampleX.out`.

Atención!! En este ejemplo, no hay robots pequeños:

```
2 0 2
10 20

10 7
10 13
```

Los tests se dividieron en dos subtareas, `st1` y `st3`. Hay 7 tests `st1` y 6 tests `st3`.

Puntaje

Subtarea	Puntaje individual por test
st1	2
st3	3
TOTAL	32

5. Ordenando Libros Antiguos

La ciudad de Alejandría en Egipto, es el hogar de la Biblioteca Real Nacional que hace un homenaje a la legendaria Biblioteca que intentó concentrar todo el saber escrito en la antiguedad. El tesoro principal de esta biblioteca se encuentra en un salón largo con una fila de n mesas, enumeradas de 0 a $n-1$ de izquierda a derecha. En cada mesa se expone un antiguo libro manuscrito. Los lectores pueden consultar libremente los libros, pero luego deben devolverlos al salón. Estos libros se ordenan en función del orden alfabético de sus títulos, pero, al final del día siempre terminan desordenados.

Mohamed Salah, uno de los bibliotecarios, debe reordenar los libros en las mesas para el otro día. Para ello, ha creado una lista p de longitud n, que contiene enteros diferentes de 0 a $n-1$. Esta lista describe los cambios necesarios para reorganizar los libros en orden alfabético: para cada $0 \leq i < n$, el libro que se encuentra actualmente en la mesa i debe ser movido a la mesa $p[1]$. Mohamed comienza a ordenar los libros en la mesa s. Él quiere volver a la misma mesa después de terminar el trabajo. Dado que los libros son muy valiosos, no puede llevar más de un libro a la vez. Mientras ordena los libros, Mohamed realizará una secuencia de acciones. Cada una de esas acciones debe ser una de las siguientes:

- Si no lleva un libro y hay un libro sobre la mesa en la que él está, puede recoger este libro.
- Si está llevando un libro y hay otro libro sobre la mesa en la que él está, puede cambiar el libro que lleva con el que está sobre la mesa.
- Si está llevando un libro y él está en una mesa vacía, puede poner el libro que lleva sobre la mesa.
- Puede caminar a cualquier mesa. Puede llevar un solo libro mientras camina.

Para cada $0 \leq i, j < n$, la distancia entre las mesas i y j es precisamente metros $|i-j|$. La tarea a resolver es ayudar a Mohamed a generar la lista p de movimientos que genere un recorrido en distancia mínimo para ordenar todos los libros.

Deben implementar la función:

```
long caminoMasCorto(vector<int> p, int s)
```

donde *long* es un tipo de entero de 64 bits, p es un vector entero de longitud n. El libro que se encuentra inicialmente en la mesa i debe ser llevado por Mohamed a la mesa $p[i]$ (para todo $0 \leq i < n$). s es el numero de la mesa donde Mohamed está al inicio, y donde debe estar después de ordenar los libros. Esta función debe devolver la mínima distancia total (en metros) que Mohamed tiene que caminar.

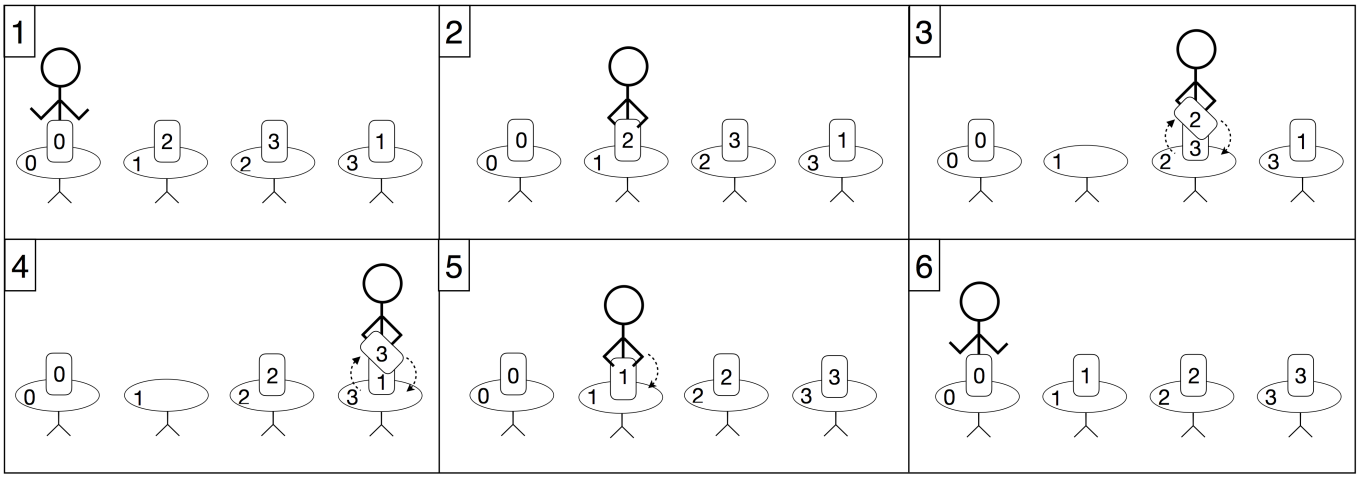
Ejemplo

```
caminoMasCorto([0, 2, 3, 1], 0)
```

En este ejemplo, y Mohamed empieza en la mesa 0. Él ordena los libros de la siguiente manera:

- Camina a la mesa 1 y recoge el libro que hay sobre ella. Este libro debe ponerse en la mesa 2.
- Luego, camina a la mesa 2 y cambia el libro que lleva con el libro sobre la mesa. El nuevo libro que lleva debe ser puesto en la mesa 3.
- Luego, camina a la mesa 3 y cambia el libro que lleva con el libro sobre la mesa. El nuevo libro que está llevando debe ponerse en la mesa 1.
- Luego, camina a la mesa 1 y pone el libro que está llevando sobre la mesa.
- Finalmente, regresa a la mesa 0.

Tener en cuenta que el libro en la mesa 0 ya está en el lugar correcto, la mesa 0, por lo que Aryan no tiene que recogerlo. La distancia total que camina en esta solución es 6 metros. Esta es la solución óptima; por lo tanto, la función debe devolver 6.



Datos de entrada

El evaluador de ejemplo lee la entrada a partir de un archivo texto en el siguiente formato:

```
n s
p[0] p[1] ... p[n-1]
```

Del ejemplo anterior, el contenido del archivo **1-01.in** es:

```
4 0
0 2 3 1
```

El largo del recorrido correcto es 6 y se encuentra como único dato del archivo **1-01.out**.

Tareas y puntajes

Los tests que debe cumplir la función desarrollada poseen las siguientes características y aportan los respectivos puntajes:

1. (0.75 puntos) $n \leq 4$, y $s = 0$
2. (0.5 puntos) $n \leq 1000$, y $s = 0$
3. (1 puntos) $n \leq 1000$
4. (1.5 puntos) Sin restricciones

Total: **54** puntos.