.UBAfiuba
FACULTAD DE INGENIERÍA

La batalla contra el reloj

Algoritmos y programación II - 1C2022

Trabajo práctico Nº3

grupal

Introducción

Luego de unos meses de estar en el club de lectura Sid comenzó a aficionarse a llevar un registro de cuánto tiempo tardaría en leer todos los libros de los que se hablaba en la reunión. Pero como buen perezoso comenzó a aburrirse de llevar la cuenta y comenzó a buscar ayuda…

Enunciado

Para este trabajo práctico ayudaremos a Sid en su última aventura de esta mitad del año, como ya lo hemos ayudado en anteriores ocasiones nos basaremos en esos proyectos para hacerlo nuevamente.

En esta ocasión, Sid tendrá los mismos archivos que se presentaron para el proyecto anterior (escritores.txt y lecturas.txt) con unas pequeñas modificaciones:

Modificaciones de escritores:

Ahora el archivo de escritores.txt contara con el ISNI del autor (un numero de 4 dígitos que identifica al mismo) y sus datos deberán ser guardados en una **más adelante.**

Nuevo formato:

Archivo *escritores.txt*

```
(ISNI) → número al que se hará referencia en el archivo de lecturas
Nombres y apellidos
Nacionalidad → puede ser "¿?"
Año de nacimiento → puede ser -1 o no aparecer si se desconoce[1].
Año de fallecimiento → puede no aparecer
// línea en blanco
```

Ejemplo:

```
(1568)
Julio Cortázar
argentino
1914
1984

(1544)
Mary Shelley
británica
-1
1851

(1757)
Stephen Edwin King
estadounidense
1947

(4567)
Alfonsina Storni
¿?
1892
1938

(5474)
Ken Follet
```

[1] Si no aparece es porque tampoco se conoce el año de fallecimiento, si se conoce el año de fallecimiento se mostrará con un -1 para no confundir los datos.

galés

Modificaciones de lecturas:

En este archivo la referencia a los autores será el ISNI de los mismos, el resto se mantendrá.

Sid nos informó que según la lectura que acaba de terminar y la próxima que comenzará es el tiempo que requiere dormir:

- Pasar de **cuento** a **novela** y viceversa requiere **10** minutos de siesta.
- Pasar de **cuento** a **novela histórica** y viceversa requiere **15** minutos de siesta.
- Pasar de **cuento** a **poema** y viceversa no requiere siesta.
- Pasar de **poema** a **novela** y viceversa requiere **5** minutos de siesta.
- Pasar de **poema** a **novela histórica** y viceversa requiere **20** minutos de siesta.
- Pasar de **novela** a **novela histórica** y viceversa requiere **60** minutos de siesta.
- Pasar de **poema** a **poema** requiere **1** minuto de siesta
- Pasar de **cuento** a **cuento** requiere **8** minutos de siesta
- Pasar de **novela** a **novela** requiere **30** minutos de siesta
- Pasar de **novela histórica** a **novela histórica** requiere **80** minutos de siesta.

Se deberá armar un grafo utilizando como peso los datos recién mencionados y armar el **Árbol de expansión mínima** del mismo para que cuando Sid pida conocer en qué orden debería leer las lecturas para tardar lo menos posible le indique el orden y el tiempo total [2]que tardaría.

Hashing

Para la tabla de hashing deberán tener en cuenta:

- A lo sumo habrá 20 escritores, por lo que se debe definir una tabla de tamaño adecuado según lo visto en clases. Este tamaño lo llamaremos n.
- La función de hashing será h(k) = k % n, donde k es la clave del objeto a guardar.
- La solución a las colisiones será con direccionamiento cerrado, es decir, en cada posición de la tabla habrá una lista de objetos.
- En la documentación deben indicar, de forma teórica, cuál es el orden del alta, consulta y baja en la tabla, explicando el mejor caso, el peor caso y el caso promedio.

Árbol de expansión mínima

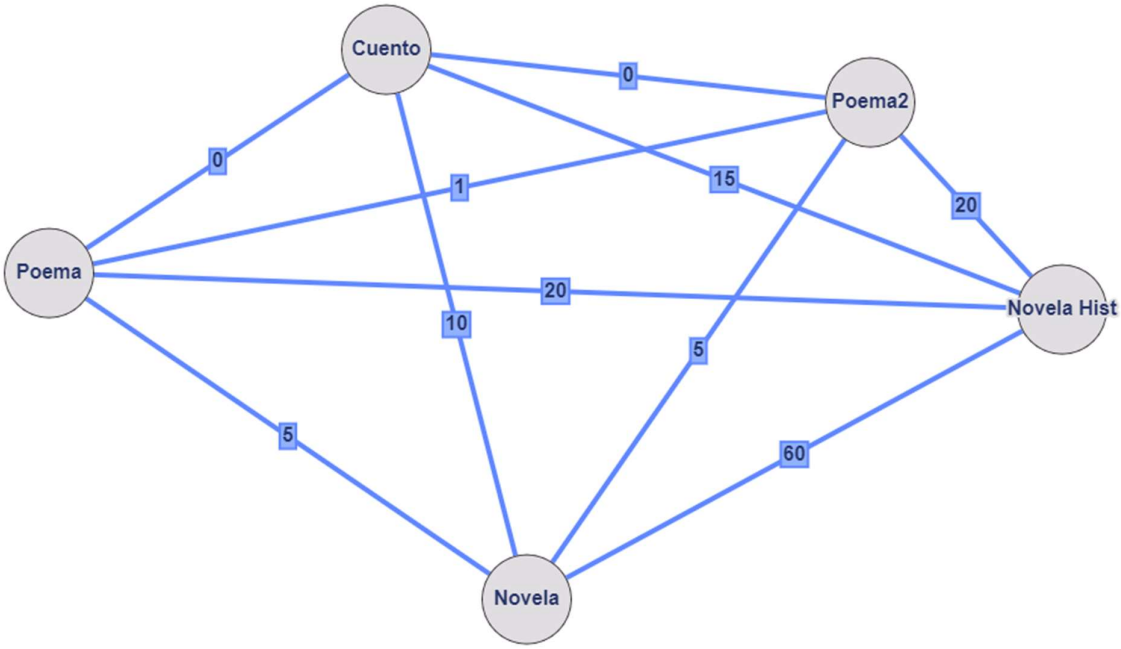
En este caso se tendrá un grafo no dirigido cuyos nodos serán las lecturas y sus aristas tendrán como valor el tiempo de siesta que requiere Sid. **Pueden elegir entre implementar el algoritmo de Kruskal o el de Prim.**

Se agregará en el menú una opción para poder solicitar el orden y tiempo mínimo que nos llevaría leer todas las lecturas.

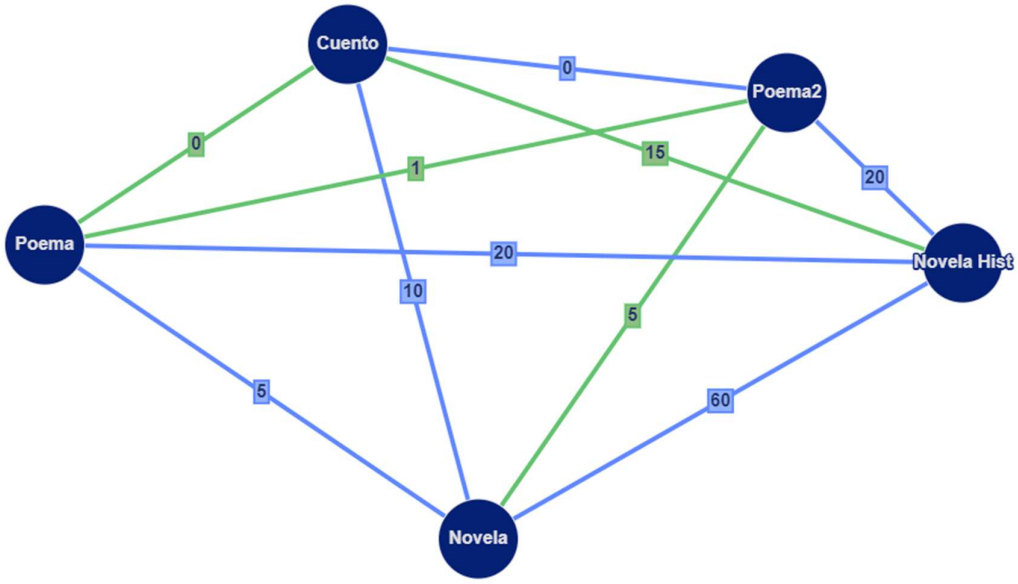
Si el libro ya fue leído no se puede volver a leer.

[2] Tiempo de cada una de las lecturas más el tiempo total de siesta que requiere.

Ejemplo:



En este caso el árbol de expansión mínima sería:



Dando como resultado:

- 21 minutos de siesta para leer todas las lecturas

Aclaraciones importantes

- Los vectores deben crearse utilizando memoria dinámica. La misma se debe liberar al final del programa.
- El trabajo es de carácter **grupal.** Los mismos serán de 3 integrantes.
- Se valorará, entre otras cosas, la eficiencia del algoritmo.
- No se deben subir archivos de configuración de los IDEs (solo subir .cpp y .h).
- El trabajo debe compilar con los flags **-Wall -Werror -Wconversion**.

¿Qué se evaluará?

- Compilación: sin warnings ni errores.
- Funcionalidad.
- Eficiencia espacial.
- Eficiencia temporal.
- Buenas prácticas de programación (nombres descriptivos, indentación, etc.).
- Separación de operadores.
- Modularización.
- Participación y conocimiento de hash, árboles y grafos.
- Precondiciones y postcondiciones.
- POO
- Memoria dinámica
- Trabajo en equipo

Normas de entrega

Se deberá subir al campus un único archivo comprimido (.zip) en la sección TPs.

Este archivo deberá tener un nombre formado de la siguiente manera:

Nombre_grupo_TP3

Por ejemplo:

grupo_TP3.zip

El archivo deberá contener solo los archivos fuente. Es decir, solo .cpp y .h. **NO** subir los archivos de configuración de sus IDEs. (por ejemplo: CMakeList y cmake-build para Clion, .vscode para VisualStudioCode).

La fecha de **entrega** vence el **viernes 1 de julio** a las 23.55hs.

Puntaje: **70** Puntos.