PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2133 - ESTRUCTURAS DE DATOS Y ALGORITMOS

Tarea 4

1^{er} **semestre 2018**

18 de junio de 2018

Entrega código: Domingo 8 de julio a las 23:59

Introducción

Luego de triunfar en su enfrentamiento con los Asteroidius, la flota de naves del planeta Espacial vuelve a casa. Pero al llegar, se encuentran con un panorama desolador: en su ausencia, se perdió el control de las cañerías que alimentan de agua a las ciudades del planeta y los habitantes ya casi han agotado sus reservas. En una nueva misión, tendrás que ayudar a los pobres Espaciales a reestablecer su sistema de agua para que sobrevivan.

Problema

En el planeta existe una red de nodos que corresponden a dos tipos: ciudades y fuentes de agua; y tuberías que conectan estos nodos, las cuales se encuentran severamente dañadas. Debido a que los recursos luego del combate son escasos, tu deber es decidir qué tuberías reparar para que el agua viaje desde las fuentes hasta todas las ciudades considerando que el costo de reparación es conocido para cada tubería.
Al seleccionar las tuberías a reparar, se deben tener en cuenta dos puntos:

1. Todas las ciudades deben recibir agua directamente (al reparar una tubería que conecta a la ciudad con una fuente) o indirectamente (al reparar una tubería que conecta la ciudad con una red ya alimentada).

2. La red obtenida al reparar las tuberías puede ser conexa (en cuyo caso solo se utiliza una fuente para alimentar la red) o no conexa (si se usa más de una fuente).

En la figura 1 se muestra un ejemplo de una red de tuberías dañadas con sus costos, mientras que en la figura 2 se muestra una configuración de tuberías reparadas que minimiza el costo total de la operación de reparación. En este ejemplo se verifica que la red obtenida no es conexa.

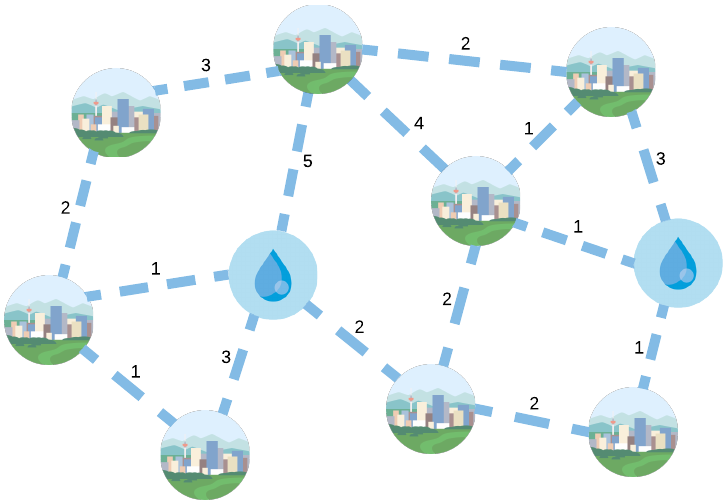

Figura 1: Situación inicial

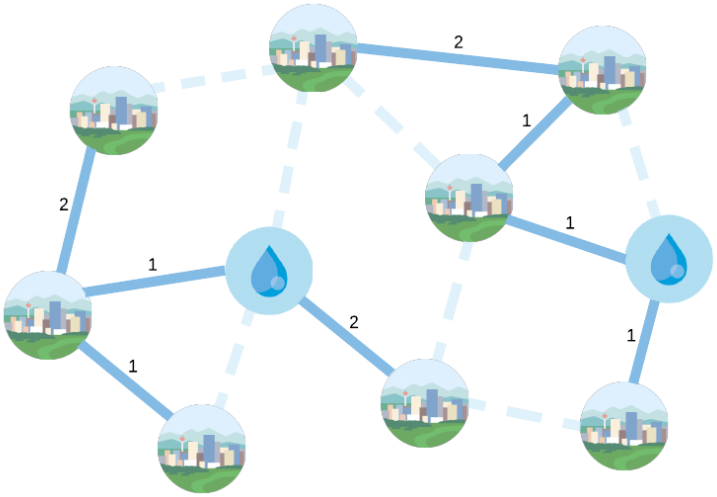

Figura 2: Red solución con costo mínimo

Deberás escribir un programa en C que dada una configuración de ciudades, fuentes y tuberías dañadas, entregue la red de costo mínimo que cumple las restricciones mencionadas.
Tip: Este problema se puede resolver haciendo una modificación pequeña a los algoritmos vistos de árboles de cobertura mínima para que creen bosques de cobertura mínima.

Input

Tu programa debe recibir como único parámetro la ruta del problema a resolver, en formato `.txt`.
La primera línea de los archivos de input tiene tres naturales c f e que indican la cantidad de ciudades, de fuentes, y de tuberías respectivamente. A continuación le siguen e líneas que describen las tuberías. Cada

una estas líneas tiene tres naturales, donde los dos primeros son los nodos de la red unidos por la tubería y el tercero es su costo de reparación. La numeración de los nodos es tal que

- los números $i \in \{0, \ldots, c-1\}$ corresponden a las ciudades y
- los números $j \in \{c, \ldots, c+f-1\}$ corresponden a las fuentes.

El archivo `.txt` del ejemplo mostrado en la figura 1 contendría:

```
8 2 15
0 2 1
0 8 1
0 1 2
1 3 3
3 8 5
2 8 3
8 6 2
2 5 4
3 4 2
5 4 1
5 6 2
6 7 2
5 9 1
4 9 3
9 7 1
```

donde los nodos 0 a 7 son ciudades y los nodos 8 y 9 son las dos fuentes.

Output

Se espera como output un archivo `result.txt` que contenga k líneas con el número de las tuberías que se repararon. La numeración de las tuberías se hace respecto a su aparición en el archivo de input (la tubería 0 es la descrita por la línea `0 2 1` en el ejemplo mostrado). Para el ejemplo, el output esperado es

```
0
1
2
6
8
9
12
14
```

El output puede tener las tuberías en cualquier orden.

Evaluación

Se probará tu programa con distintos tests con un tiempo límite de 10 segundos por test. Para cada uno se revisará que el archivo de output cumpla (1) que cada ciudad reciba esté conectada a alguna red alimentada por una fuente y (2) que el costo total de las tuberías sea óptimo. Esto último se verifica pues para una misma instancia del problema, podría haber más de una configuración óptima.

Entrega

Deberás entregar tu tarea en el repositorio que se te será asignado; asegúrate de seguir la estructura inicial de éste.

Se espera que tu código compile con `make` dentro de la carpeta **Programa** y genere un ejecutable de nombre `solver` en esa misma carpeta. **No modifiques código fuera de la carpeta *src/solver*.**

Estar reglas ayudan a recolectar y corregir las tareas de forma automática, por lo que su incumplimiento implica un descuento en tu nota.

Se recogerá el estado de la rama `master` de tu repositorio, 1 minuto pasadas las 23:59 horas del día de entrega. Recuerda dejar ahí la versión final de tu tarea. No se permitirá entregar tareas atrasadas.

Bonus

Manejo de memoria perfecto (+5 % a la nota de *Código*)

Se aplicará este bonus si **valgrind** reporta en tu código 0 leaks y 0 errores de memoria, considerando que tu programa haga lo que tiene que hacer.