Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 4

IIC2133 - Estructuras de datos y algoritmos

Primer semestre, 2016

Entrega: Jueves 30 de Junio

Objetivos

- Resolver un problema de optimización
- Encontrar una aproximación a la solución de un problema

Problema

Se quiere trazar una red entre diversos puntos de interés para poder abastecerlos de **Slurm**, el agua del futuro. Uno de estos puntos es la fábrica: el origen de la red, mientras que los otros puntos actúan como redistribuidores, conocidos como *estaciones*.

Para esto se ha decidido arrendar las tuberías del antiguo sistema de alcantarillado de la ciudad, las cuales varían en capacidad y recorrido.

Tu objetivo es escoger que tuberías han de usarse en la red de manera de maximizar el volumen de *Slurm* disponible en cada estación, y de manera secundaria minimizar el costo de mantención de la red. Para cada tubería, el costo es directamente proporcional a su capacidad, por lo que se considerarán equivalentes.

El volumen disponible en una estación está sujeto a la capacidad de la tubería más pequeña que la conecta con el origen. Claramente este volumen no puede sobrepasar a la capacidad original de la fábrica.

This assignment has been brought to you by *Slurm*
It's highly addictive!

Ahora en serio

Sea $S = \{s_1, s_2, ..., s_{n-1}\}$ las estaciones redistribuidoras de la red.

Sea f la fábrica de *Slurm*, la estación origen de la red.

Sea entonces $G(V, E)$ el grafo no direccional que representa el problema, altamente redundante.

$V = S \cup \{f\}$

Cada arista $(u, v) \in E$ es una serie de tuberías que conectan los nodos en u, v, donde cada tubería pertenece a solo una arista de E.

Se define $w(e),\ e \in E$ como la capacidad máxima de flujo de la arista e.

Se define $h(v),\ v \in V$ como el volumen disponible en la estación v.

$h(f) = k$ por definición, k es la capacidad de output de la fábrica.

Sea $\{e_1, e_2, ..., e_m\}$ la serie de tuberías que conecta la estación s_i con f.

$h(s_i) = min(k, w(e_1), w(e_2), ..., w(e_m))$

Se busca encontrar $G'(V, E'), E' \in E$ un grafo acíclico tal que

$\sum_{v \in V} h(v)$ sea máximo, y sujeto a ello, que
$\sum_{e \in E'} w(e)$ sea mínimo.

Solución

Debes inventar un algoritmo, o adaptar un algoritmo conocido para solucionar este problema e implementarlo en C. No se espera que encuentres el óptimo, pero debes tratar de acercarte a él en lo posible.

Análisis

Deberás entregar un informe donde analices tu algoritmo. Se espera que respondas al menos lo siguiente.

1. ¿Qué método usaste para representar el grafo? ¿Por qué?
2. ¿Tu algoritmo es codicioso? Si no, ¿Pertenece a alguna familia de los algoritmos que hemos estudiado? Justifica.
3. ¿Cuál es la complejidad de tu algoritmo? Justifica.

Evaluación

El 60 % de tu nota proviene de los resultados de tu programa, mientras que el otro 40 % viene del informe.

La nota de código será relativa al resto del curso. Existirán dos programas de prueba, uno para el 4 y otro para el 7. Según como quede tu código posicionado entre ambos será el puntaje obtenido.

Tu código será evaluado con 4 tests de dificultad creciente.

Input

Tu programa deberá recibir como primer parámetro la ruta a un archivo de prueba. Este archivo sigue la siguiente estructura:

La primera línea indica la cantidad de nodos en el grafo.
`V 4`
La segunda línea indica cual es el nodo origen de la red.
`F 0`
La siguiente línea indica la capacidad de output de f
`K 50`
Luego viene la cantidad de aristas de la red
`E 9`
Las siguientes `E` líneas contienen la información de las aristas de la siguiente forma: `índice nodou nodov capacidad`.

```
0 0 1 70
1 0 1 80
2 0 1 40
3 1 2 90
4 1 2 130
5 1 2 70
6 2 3 100
7 2 3 50
8 2 3 60
```

Output

Tu programa deberá imprimir los índices de las aristas que se deben conservar del grafo original.
Con el ejemplo anterior, un posible output para tu programa sería

```
0
5
7
```

Entrega

Deberás entregar en tu repositorio de Github una carpeta de nombre **Tarea04**. Y dentro de esta carpeta, una carpeta **Programa** y una carpeta **Informe**. Dentro de *Programa* deberás tener un *Makefile* y una carpeta **src** con todo el código de tu tarea. Ésta se compilará con el comando `make` dentro del mismo directorio, y tu programa deberá generarse en un archivo de nombre `slurmer`.

En la carpeta *Informe* deberás tener un archivo de nombre *Informe.pdf*.

Se recogerá el estado de la rama `master` de tu repositorio, 1 minuto pasadas las 24 horas del día de entrega. Recuerda dejar ahí la versión final de tu tarea.

Bonus

A continuación, formas de aumentar la nota obtenida en tu tarea

Powerhouse (+20 % a la *Nota Final*)

Tu algoritmo debe ser óptimo, polinomial, y deberás demostrar que lo es. Si crees que no es posible hacer uno, justificarlo.

Crystal Clear (+15 % a la nota de *Código*)

Tu código deberá ser de buena calidad, es decir, no deberá tener

- Memory leaks
  - `valgrind` deberá decir `All heap blocks were freed -- no leaks are possible`
- Errores de memoria
  - `valgrind` deberá decir `ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)`
- Advertencias de compilador
  - Esto compilando con las *flags* `-Wall` y `-Wextra` (Ponlas en `CFLAGS` en tu *Makefile*)

Ortografía perfecta (+5 % a la nota de *Informe*)

La nota de tu informe aumentará en un 5 % si no tienes faltas de ortografía. Faltas gramaticales te harán perder este bonus.

Buen uso del espacio y del formato (+5 % a la nota de *Informe*)

La nota de tu informe aumentará en un 5 % si tu informe está bien presentado y usa el espacio y formato a favor de entregar la información. A juicio del corrector.