Universidad Simón Bolívar
Departamento de Computación
y Tecnología de la Información
CI-2693: Laboratorio de
Algoritmos III

Proyecto 2 Septiembre – Diciembre 2018

Situación del agua

1 Introducción

La maravillosa y confiable Compañía Hidrológica de La Región Capital (Hidrocapital) ha tenido algunas fallas menores ultimamente. Para sus clientes organizacionales distribuidos entre varios edificios (llamados "campus"), esto implica la necesidad de mover su personal de un edificio a otro para poder hacer uso de los baños. Debido a las lluvias, las conexiones entre los edificios pueden verse inhabilitadas, por lo que los planes de movimiento deben cambiarse con frecuencia (estas condiciones cambiantes se denominan los "casos"). Esto hace necesario el desarrollo de un programa que indique hacia dónde moverse para poder hacer uso de los baños y mantener la organización funcionando. Sin embargo, debe mantenerse la imagen pública de las organizaciones, por lo que hay un número limitado de personas totales que pueden enviarse entre cada par de edificios.

2 Requerimientos del programa

Usted debe desarrollar un programa que, dado una lista de casos de un campus, un edificio de partida, y un número de personas en ese edificio, le asigne el baño a todas el baño más cercano disponible en todos los casos.

El programa se debe poder ejecutar desde la cónsola con el siguiente comando:

```
java EncontrarAgua <grafo> <casos> <edif> <personas>
```

donde

- <grafo> es la ruta del archivo con el grafo del campus
- <casos> es la ruta del archivo con la lista de casos
- <edif> es el código del edificio interesado en encontrar baños para su personal
- <personas> es el número de personas en el edificio

Para cada caso, el programa debe realizar una copia del grafo, realizando los cambios pertinentes al caso (así, si una conexión entre dos edificios se indica como afectada por las lluvias, esa arista debe ser eliminada). Luego, debe efectuar el algoritmo Bellman Ford desde el `edif` indicado para evaluar la distancia de recorrido hasta los edificios que tienen agua, y escoger el más cercano.

Suponga que todos los baños se encuentran disponibles para uso de las `personas` que realizan la consulta a menos que el caso indique lo contrario, incluyendo los baños del edificio de partida. Si a un edificio se envían tantas personas como tiene capacidad, el edificio ya no podrá ser usado por ninguna otra persona. En consecuencia, si se deben asignar, por ejemplo, 600 personas, y el edificio más cercano solo tiene capacidad para 60, debe volverse a evaluar para las 540 personas restantes. Similarmente,

debe evitar exceder la capacidad total de las conexiones entre edificios: si una arista tiene capacidad para 150 personas, y se envían 60 personas a lo largo de ella al primer edificio, al buscar un destino para las restantes, debe considerarse que esta arista sólo le queda capacidad para 90 personas.

Si no es posible asignarle baño a todas, indique cuántas personas se quedan sin baño, y las rutas encontradas para los que sí.

2.1 Formato de Archivos

Se proporcionan siempre dos archivos: uno conteniendo el campus, representado como un grafo, y uno conteniendo el cronograma, presentado como una serie de casos.

2.1.1 Archivo de Grafo

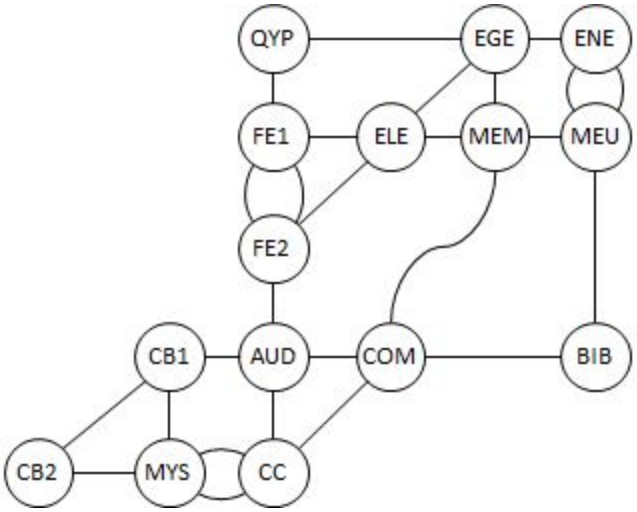
Los grafos son no-dirigidos y tienen el siguiente formato:

```
n
m
v_1 dv_1 pv_1
⋮
v_n dv_n pv_n
vi_1 vf_2 dl_1 pl_1
⋮
vi_1 vf_2 dl_m pl_m
```

donde

- `n` es el número de vértices
- `m` es el número de lados
- v_i es el identificador del vértice, dado por el código del edificio
- dv_i es la capacidad del edificio i, indicando el número de personas totales que pueden ir al baño en él
- pv_i es el peso del vértice vi, dado por el número de pisos que se deben subir para encontrar un baño en condiciones normales (si el edificio solo ofrece baños en el sótano, pvi puede ser negativo)
- vi_i es el vértice inicial del lado i
- vf_i es el vértice final del lado i
- dl_1 es la capacidad del lado, indicando cuántas personas totales pueden enviarse por ese camino antes de que se vuelva notable que están buscando un baño
- pl_i es el peso del lado i, dado por la distancia en metros de la ruta

Considere, por ejemplo, el siguiente grafo



Representado por

```
15
23
CB2 36 0
MYS 84 1
CC 16 0
CB1 72 1
AUD 16 0
COM 24 0
BIB 72 0
FE2 54 2
FE1 54 2
ELE 72 1
MEU 96 0
MEM 96 3
QYP 72 3
EGE 48 1
ENE 128 1
CB2 MYS 150 150
CB2 CB1 150 87.5
MYS CC 150 156
MYS CC 150 92
MYS CB1 300 48.5
CC AUD 50 88
CC COM 100 171
CB1 AUD 300 200
AUD COM 300 144.5
AUD FE2 125 167
COM MEM 75 157
COM BIB 150 260
BIB MEU 150 230
FE2 FE1 75 119
FE1 FE2 90 137
FE1 QYP 150 90
FE1 ELE 150 75
ELE EGE 100 100
ELE MEM 200 57
MEM EGE 200 57
MEM MEU 200 80
MEU ENE 200 157
ENE MEU 75 123
QYP EGE 12 163.5
EGE ENE 75 169.5
```

Debe sumar 25 metros por cada piso que se debe subir en el edificio destino. las aristas múltiples entre algunos pares de edificios indican accesos por medio de puertas traseras o laterales que pueden ser afectadas independientemente.

2.1.2 Archivo de casos

Los archivos de casos tienen el siguiente formato

id_1
e_1
a_1
$v_{1,1}$ $[pv_{1,1}]$
⋮
$v_{1,e}$ $[pv_{1,e}]$
$l_{1,1}$
⋮
$l_{1,a}$

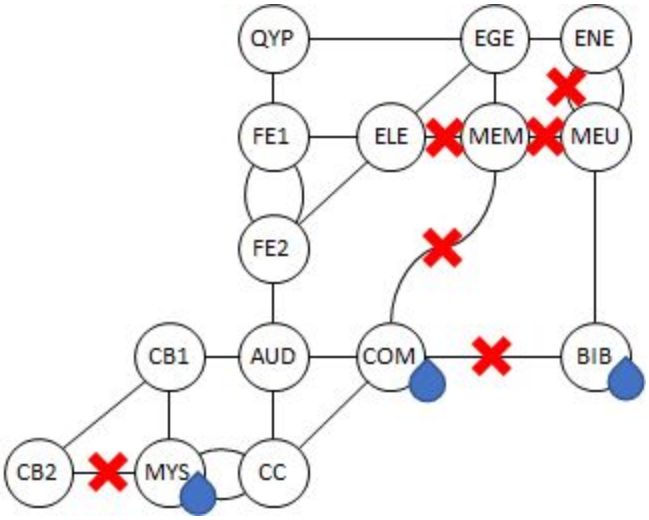
id_2
e_2
a_2
$v_{2,1}$ $[pv_{2,1}]$
⋮
$v_{2,e}$ $[pv_{2,e}]$
$l_{2,1}$
⋮
$l_{2,a}$

id_3
⋮

donde

- id_i es el identificador del i-ésimo caso
- e_i es el número de edificios que tienen agua en el i-ésimo caso
- a_i es el número de arcos afectados (pudiendo ser 0) en el i-ésimo caso
- $v_{i,j}$ es el identificador del j-ésimo edificio con agua en el i-ésimo caso
- $pv_{i,j}$ es la modificación al peso (por ejemplo, si un edificio normalmente tiene peso 0 y pv_i es +1, eso indica que ese día los baños de planta baja estarán cerrados.) pv1 podría no estar presente, en cuyo caso debe suponer que el peso no se modifica
- l_i es el número de línea del arco afectado en el archivo original, entre 0 (inclusive) y m (exclusive)

así, por ejemplo, una modificación al grafo anterior del siguiente tipo:



Podría ser indicado por un caso así

```
lunes                                   0
3                                       9
6                                       11
BIB +1                                  17
MYS -1                                  19
COM                                     20
```

Puede haber cualquier número de casos en un archivo. Los casos se separan con una línea en blanco.

2.2 Formato de salida

Debe indicar, para cada caso, cuántas personas son enviadas a cada edificio, a través de qué ruta, la longitud de la ruta (incluyendo las escaleras al final) y –si hay personas que no pueden ser asignadas a un edificio– cuántas son.

Para una llamada como

```
java EncontrarAgua USB.txt semana.txt ENE 200
```

Se puede dar una respuesta como

```
lunes
      72 personas a BIB
            Ruta: ENE - MEU - BIB (437 m)
      24 personas a COM
            Ruta: ENE - EGE - ELE - FE2 - AUD - COM (731 m)
      84 personas a MYS
            Ruta: ENE - EGE - ELE - FE2 - AUD - CC - MYS (766 m)

      20 personas sin asignar
```

3 Requerimientos de la entrega

Debe usar las clases desarrolladas en el proyecto 1. Debe encontrar los caminos usando el algoritmo de Bellman-Ford. Todos sus códigos deben estar debidamente documentado.

El proyecto debe ser subido al Moodle de la materia en la sección marcada como "Proyecto 2" hasta el jueves 15 de Noviembre. Deberá entregar la Declaración de autenticidad de entregas. Sólo deberá efectuar una entrega por grupo.

4 Evaluación

El proyecto tiene una ponderación de 20 puntos. El programa debe correr sin errores. Se asignarán:

- 6 puntos por código
  - 1 punto por crear copias adecuadas a cada caso
  - 1 puntos por la implementación de Bellman-Ford
  - 1 punto por modificar Bellman-Ford para buscar solo rutas hasta los edificios con agua
  - 1 punto por modificar Bellman-Ford para acatar las capacidades de aristas
  - 1 punto por ajustar las capacidades de los edificios de forma acorde luego de asignar personas a cada edificio
  - 1 punto por ajustar las capacidades de las aristas de forma acorde luego de asignar las rutas
- 11 puntos por ejecución
  - 1 punto por escoger correctamente los edificios con agua más cercanos
  - 1 punto por encontrar correctamente el camino más corto hasta ellos
  - 1 punto por calcular correctamente la distancia de cada camino, incluyendo la distancia que debe considerarse por subir o bajar escaleras
  - 1 punto por no utilizar las aristas afectadas aunque estas darían un camino más corto
  - 1 punto por indicar correctamente que no se puede asignar un edificio si todas las aristas entre el punto de origen y los edificios con agua restantes se encuentran afectadas
  - 1 punto por indicar correctamente que no se puede asignar un edificio si ningún edificio tiene agua en el caso
  - 1 punto por indicar correctamente cuál es el baño más cercano en situaciones donde el desempate venga dado por el costo de subir o bajar las escaleras
  - 1 punto por asignar las personas al edificio inicial cuando éste tenga agua
  - 1 punto por no descartar ir entre dos edificios cuando existan n aristas entre ellos cuando $m<n$ aristas se encuentren afectadas
  - 1 punto por poder manejar campus compuestos de varias componentes conexas
  - 1 punto por manejar correctamente los casos borde (por ejemplo, solicitar asignar baños a cero personas)
- 3 puntos por documentación
  - 1 punto por documentar apropiadamente su modificación del algoritmo Bellman-Ford
  - 1 punto por documentar apropiadamente los métodos que utilizan las clases de grafos
  - 1 punto por documentar apropiadamente su programa principal