Universidad de Buenos Aires
Facultad de
Ciencias Exactas y Naturales
Departamento de Computación

Algoritmos y Estucturas de datos III

Segundo Cuatrimestre de 2011

Trabajo Práctico 3

Cartero Chino en grafo mixto.

Grupo: '1'

Integrante	LU	Correo electrónico
Cammi, Martín	676/02	martincammi@gmail.com
Garbi, Sebastián	179/05	garbyseba@gmail.com
Kretschmayer, Daniel	310/99	daniak@gmail.com

Índice

Ejecución del TP

0.1. Lenguaje utilizado

El lenguaje utilizado para el trabajo práctico ha sido *Java*, compilando con la versión 1.5 de la Virtual Machine.

El trabajo se acompaña con los fuentes de la solución que puede importarse en IDE de Eclipse o ejecutarse desde línea de comandos.

0.2. Como ejecutar el TP

Desde línea de comandos

- Posicionarse en el directorio Algo3Tp3
- Copiar allí el archivo de entrada para el problema i, por ejemplo Ej1.in
- Ejecutar el comando: *java -cp ./bin problema1.Ej1*

Esto generará el archivo Ej1.out con la solución en el mismo directorio Algo3Tp3.

Desde el Eclipse

Primero importaremos el proyecto:

- Seleccionar File ⇒ Import.
- Seleccionar General ⇒ Existing Projects into Workspace ⇒ Next.
- Seleccionar el directorio llamado Algo3Tp3.
- Finish.

Desde la vista de **Package Explorer** bajo el paquete **src** aparecerán tres paquetes más y dentro de cada uno de ellos los siguientes archivos de java:

COLOCAR IMAGEN CORRECTA

Para ejecutar un problema:

- Posicionarse en el directorio Algo3Tp3
- Copiar en el directorio Algo3Tp3 el archivo de entrada para el problema i, por ejemplo Ej1.in
- Con botón derecho Run As ⇒ Java Application. Se ejecutará el problema seleccionado.

Esto generará el archivo Ej1.out con la solución en el mismo directorio Algo3Tp3.

1. Introducción

En 1736 Leonhard Euler publicó un trabajo en el cual resolvía el denominado *problema de los puentes de Königsberg*. La ciudad de Köningsberg, actual Kaliningrado, Rusia, está dividida por el río Pregel en 4 zonas, dos orillas (A y B), la isla de Kneiphof (C) y otras dos partes divididas por el río. (D y F). Las orillas estaban conectadas mediante 7 puentes. El problema consistía partir de una orilla y recorrer todos los puentes, recorriéndo una sola vez cada uno y volviendo al punto de partida.

Este fué el puntapié inicial para un tipo de problema que más adelante de modelaría usando la teoría de grafos. Este problema fue caracterizado por Euler y consistía en encontrar un circuito que pasara por todas las aristas de un grafo una sóla vez volviendo al punto de partida.

Más de doscientos años más tarde en 1962 el matemático chino Kwan Mei-Koo publicó un paper en el que abordaba un problema ligeramente similar, proponía que si un grafo no poseía un circuito Euleriano, podria quizás encontrar el circuito más corto que pasara por todas las aristas aunque repitiera algunas de ellas. Este problema fue luego nombrado como "Problema del cartero chino".

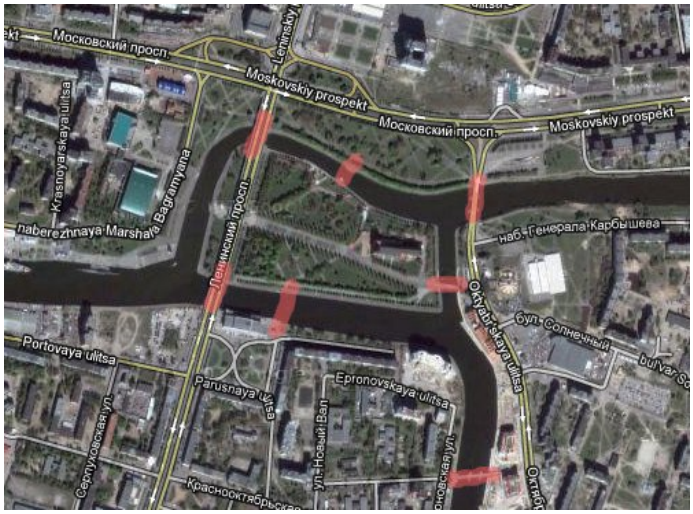

Imagen actual de la ciudad de Kaliningrado, en rojo figuran los puentes del problema original de Königsberg algunos cuales ya no existen hoy en día.

El problema tiene varias variantes, en este trabajo abordaremos la variante *mixta* que consiste en intentar encontrar el mínimo circuito que pase por todas las aristas (no orientadas) y arcos (orientadas) de un grafo donde cada una tiene peso asignado.

2. Situaciones de la vida real

El algoritmo del cartero chino tiene múltiples aplicaciones en la vida cotidiana por ser

- Cálculo de rutas óptimas para recorridos de camiones de basura: en donde los camiones deban recorrer las calles de una ciudad y volver al centro de tratamiento de basura. Las calles

podrían bien tener diferentes sentidos o sentidos únicos, y los pesos podrán corresponder a la cantidad de basura que deban recolectar en ciertas zonas.

3. Heurística Constructiva

Para la etapa constructiva, hemos realizado en primer lugar, una orientación de todas las aristas utilizando 5 maneras distintas, todas ellas con respecto a los grados de los nodos. Entonces, según un parámetro de entrada que indica cual de estas opciones usará, encontrará los nodos del grafo que cumplan lo siguiente, pasándole otro parámetro y buscará:

- Los nodos que tienen igual cantidad de aristas sin orientar que orientadas, y en caso de haber muchos, solo tomará los primeros k nodos de ellos, donde k es el parámetro de entrada de la función.
- Los nodos a partir de un cierto número, que es definido por el parámetro de entrada de la función.
- Los nodos que tengan grado de nodo (sin orientar) mayor o igual al parámetro de entrada de la función.
- Los nodos que tengan grado de entrada (orientado) mayor o igual al parámetro de entrada de la función.
- Los nodos que tengan grado de salida (orientado) mayor o igual al parámetro de entrada de la función.

En caso de no encontrar ningún nodo a orientar con estas opciones, encontrá los k primeros nodos (empezando desde 0), sea cual sea el grado de sus nodos. Con cualquiera de estas posibilidades, siempre se intenta hacer un balance entre los grados que quedarán (solo quedarán grados de entrada y salida, ya que será completamente orientado). Esta decisión se basa, que luego le calcularemos el circuito Euleriano al grafo resultante, para poder encontrar un circuito que pase por todas las aristas. Una vez que tenemos todo el grafo orientado, realizamos luego una etapa de matching entre los nodos que quedan con diferentes grado de entrada y de salida, y balanceamos a todos los nodos, para que queden con iguales grados de entrada como de salida, y el grafo queda Euleriano. Este matching lo calculamos con la distancia mínima de entre cada uno de los nodos, pero en el grafo original, por lo que todas las aristas agregadas, tienen distancia mínima entre ellas.

4. Heurística de busqueda Local

En nuestra búsqueda local, lo que realizaremos es cambiar el matching que hace entre los nodos, para obtener otro donde sea mínimo y así poder acercarnos mejor a la solución requerida. Notemos que, como los matching entre los nodos donde se tenía que los grados de entrada y de

salida no coincidían se efectúa sobre el grafo original (y no con el orientado), esta es una solución óptima para esa orientación. Luego, con el grafo totalmente orientado y Euleriano, simplemente se calcula el peso de todas las aristas, y llegamos al peso del camino requerido.

//DESDE ACA ES LO QUE HABIA Nuestro algoritmo de busqueda local se basa en eliminar aristas y arcos agregados para igualar los grados de salida a los de entrada de algunos nodos y cambiarlos por un conjunto de aristas que cumplen la misma propiedad pero cuya suma es menor. Para ello el algoritmo constructivo realiza los siguientes pasos dado un grafo mixto:

1) Orientar todas las aristas no orientadas del grafo. 2) Calcular un matching entre los nodos que tengan din != dout. 3) Calcular el peso del camino mínimo entre los nodos del matching.

4) Calcular los caminos mínimos entre cada par del matching y con ellos y armar el Circuito euleriano. //DESDE ACA ES LO QUE HABIA

5. Metaheuristica Grasp