

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

## Algoritmos y Estructuras 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	<a href="mailto:martincammi@gmail.com">martincammi@gmail.com</a>
Garbi, Sebastián	179/05	<a href="mailto:garbyseba@gmail.com">garbyseba@gmail.com</a>
Kretschmayer, Daniel	310/99	<a href="mailto:daniak@gmail.com">daniak@gmail.com</a>

# Índice

0.1. Lenguaje utilizado . . . . .	3
0.2. Como ejecutar el TP . . . . .	3
<b>1. Introducción</b>	<b>5</b>
<b>2. Situaciones de la vida real</b>	<b>5</b>
<b>3. Heurística Constructiva</b>	<b>6</b>
<b>4. Heurística de búsqueda Local</b>	<b>6</b>
<b>5. Metaheurística Grasp</b>	<b>6</b>

## 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 Algo3Tp2
- 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 Algo3Tp2.

#### Desde el Eclipse

Primero importaremos el proyecto:

- Seleccionar File  $\Rightarrow$  Import.
- Seleccionar General  $\Rightarrow$  Existing Projects into Workspace  $\Rightarrow$  Next.
- Seleccionar el directorio llamado Algo3Tp2.
- 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 Algo3Tp2
- Copiar en el directorio Algo3Tp2 el archivo de entrada para el problema i, por ejemplo Ej1.in
- Con botón derecho Run As  $\Rightarrow$  Java Application. Se ejecutará el problema seleccionado.

Esto generará el archivo Ej1.out con la solución en el mismo directorio Algo3Tp2.

## 1. Introducción

En 1736 Leonhard Euler publicó un trabajo en el cual resolvía el denominado *problema de los puentes de Königsberg*. La ciudad de Königsberg, 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, recorriéndolo una sola vez cada uno y volviendo al punto de partida.

Este fué el puntapié inicial para un tipo de problema que más adelante se 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 sola 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, podría 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".

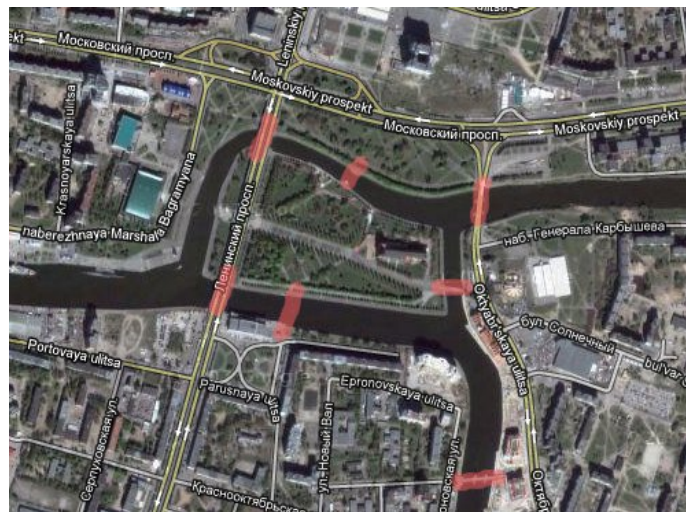


Imagen actual de la ciudad de Kaliningrado, en rojo figuran los puentes del problema original de Königsberg algunos de los cuales ya no existen hoy en día.

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**

### **4. Heurística de búsqueda Local**

Nuestro algoritmo de búsqueda 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  $d_{in} \neq d_{out}$ . 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.

### **5. Metaheurística Grasp**