DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

Algoritmos y Estructuras de Datos III
Segundo Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Bouzón, María Belén	128/13	belenbouzon@hotmail.com
Jiménez, Paula	655/10	puly05@gmail.com
Montepagano, Pablo	205/12	pablo@montepagano.com.ar
Rey, Maximiliano	037/13	rey.maximiliano@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Problema 1: Saliendo del Freezer

1.1. Descripción de la problemática

En este ejercicio se nos presenta una edificación de N niveles cuyo flujo de transporte entre distintos pisos - no necesariamente consecutivos - se encuentra dado por teletransportación a través de portales unidireccionales (es decir, que sólo permiten el ascenso).
A partir de este contexto y asegurando que existe al menos una solución para cada instancia, se nos pide diseñar un algoritmo de complejidad O(n^2) que calcule la mayor cantidad de portales que pueden ser usados para subir desde planta baja al piso N, sin descender en ningún momento.

1.2. Resolución propuesta y justificación

Para garantizar la cota superior de complejidad solicitada propoponemos resolver el problema utilizando un algoritmo de programación dinámica cuyo comportamiento esté regido por las siguientes instrucciones:

- tomar un piso (partiendo por el superior)
- para cada uno, fijarse a qué otros niveles superiores comunica.
- para cada uno de ellos comparar la máxima cantidad de portales que los separa del último nivel
- tomar el máximo de ellos y almacenarlo
- al llegar al primer piso, devolver el máximo calculado.

este comportamiento puede organizarse y entenderse a través del siguiente pseudocódigo:
completar una vez qe tenga el algoritmo
bla

1.3. Análisis de la complejidad

1.4. Código fuente

1.5. Experimentación

1.5.1. Constrastación Empírica de la complejidad

2. Problema 2: Algo Rush

2.1. Descripción de la problemática

2.2. Resolución propuesta y justificación

2.3. Análisis de la complejidad

2.4. Código fuente

2.5. Experimentación

2.5.1. Constrastación Empírica de la complejidad

3. Problema 3: Perdidos en los Pasillos

3.1. Descripción de la problemática

En este ejercicio se nos presenta un pabellón con M pasillos de distintas longitudes (potencialmente) y un conjunto vértices que pueden ser tanto intersecciones en las cuales dos o más de ellos convergen, o extremos incididos por un sólo corredor. A partir de este contexto se nos pide desarrollar un algoritmo que elimine cualquier ciclo posible del grafo dado, logrando crear partiendo del mismo un árbol generador cuyo peso (dado por la sumatoria de los pesos de cada arista) sea mayor o igual al de cualquier otro árbol generador posible (conocido como árbol generador máximo).

3.2. Resolución propuesta y justificación

Para resolver el problema, desarrollamos un método que hace uso de una adaptación del algoritmo de Kruskal, mediante el cuál es posible encontrar un árbol generador mínimo con una complejidad de O(m log m) (siendo m la cantidad de aristas).
El algoritmo propuesto ordena inicialmente cada pasillo de acuerdo a sus longitudes y los toma uno a uno de mayor a menor verificando si conectan vértices entre los cuales ya existe un camino o no. De ser cierto, la arista en cuestión es descartada. De ser falso, la misma pasa a ser parte del conjunto solución y se repite el procedimiento con la siguiente hasta que se hayan analizado todas.
Para que las complejidades respetaran los requerimientos, fue necesario desarrollar... (find & union) COMPLETAR
El pseudocódigo de nuestro algoritmo es el siguiente:

```
Ordenar pasillos de acuerdo a su longitud.
for pasillo p in pasillos do
  if El pasillo conecta dos intersecciones que no tenían demarcado un camino previamente then
    Agregar el pasillo al conjunto solución.
    Sumar la longitud del pasillo a la solución.
  end if
end for
```

3.3. Análisis de la complejidad

3.4. Código fuente

3.5. Experimentación

Pensar casos borde

3.5.1. Constrastación Empírica de la complejidad