U.B.A. FACULTAD DE INGENIERÍA

Algoritmos y Programación II (75.41)

Curso 4 - Wachenchauzer

Trabajo Práctico Nº 3

Curso 2013 - 2do Cuatrimestre

Integrantes:

- Amura Federico, 95202
- Rupcic Florencia, 94525

Fecha de entrega: 29 de noviembre de 2013

Corrector:
Luciano

Análisis y diseño.

Para este trabajo práctico se decidió utilizar Python, ya que cuenta con las colecciones de datos necesarias (listas, tuplas, diccionarios) predefinidas.

Creamos un grafo que será un conjunto de vértices y aristas. Los vértices serán los encargados de representar todas las esquinas que se carguen a través del archivo mapa.csv, y las calles serán las aristas que conecten las diversas esquinas. Las esquinas, además, contienen datos de longitud y latitud, y coordenadas x e y.

Las fábricas contienen un diccionario de juguetes, un id, y horarios de entrada y de salida. Al cargar el archivo fabricas.csv, se van añadiendo las diversas fábricas con sus datos correspondientes a las esquinas especificadas.

En cuanto a los juguetes, éstos simplemente tienen un id, un valor y un peso. Al cargar el archivo juguetes.csv, se crea el juguete y se almacena en la fábrica que corresponda con el id de fábrica.

En la oficina de Papa Noel cargamos los archivos y armamos el grafo explicado anteriormente. Además, proveemos a Papa Noel con una agenda: un diccionario cuyas claves son los id de las fábricas, donde se guardan las esquinas en donde están posicionadas las fábricas.

Para la carga de los archivos de las fábricas y juguetes, se leyó por línea para almacenar los datos necesarios en cada caso y poder crear las fábricas y juguetes correspondientes.
Para la carga de archivo mapa, en cambio, hubo que crear una variable index que fuera contando la cantidad de líneas que se fueran leyendo, pues el formato del archivo era distinto. En este caso, la primera línea correspondía a la cantidad N de esquinas, seguida por los datos de las N esquinas. Una vez que se terminaban de leer dichas líneas, se procedía a leer la cantidad de calles y la lista de calles con sus datos correspondientes.

Comandos de administración.

• listar_fabricas
Muestra un listado ideal de las fabricas a visitar, considerando que se busca el mayor número de fabricas
Mediante una lista ordenada, por horario de cerrado, luego por horario de apertura y finalmente por idfabrica, se van quitando las que cierran primero para luego agregarlas, si es que puede ser visitada ya que puede abrir antes de que haya cerrado la última fabrica que visitamos, a una lista, esta vez por orden de inserción, que termina siendo la lista óptima de fábricas a visitar con las consideraciones mencionadas. Al mismo tiempo se lleva un contador para saber la cantidad de fábricas que se visitaran.

• valuar_juguetes
Indica el valor total máximo de los juguetes a retirar de la fábrica.
Se deberá pasar por parámetro el id de la fábrica para la cual se desea obtener el valor. En el caso de que la agenda contenga la clave de la fábrica, se utiliza el “problema de la mochila” con programación dinámica para devolver la cantidad de sonrisas de la fábrica y una lista con todos los juguetes que se necesitan cargar para ello. El problema consiste en llenar una mochila que puede cargar únicamente con un determinado peso P con diversos objetos. Estos objetos tienen dos características importantes: peso y valor. Se busca de esta forma maximizar el valor de la

mochila evitando que la suma de los pesos exceda el peso P determinado previamente. En nuestro caso sucede algo muy similar. Se busca que los juguetes que se necesitan cargar en el trineo no excedan cierto peso, y que el valor de sonrisas que se carga sea el mayor posible.

• valuar_juguetes_total
Indica el valor total de los juguetes a retirar de todas las fábricas. Resulta de sumar los resultados óptimos de cada fábrica.
Se seleccionarán las fábricas óptimas a visitar con la misma función que se utiliza para listar las fábricas en el primer comando descripto. Para cada fábrica contenida en el camino conseguido, se utilizará la misma función que se realizó para valuar los juguetes y se irán sumando los resultados de los valores máximos.

• camino_optimo
Indica el camino óptimo (el más corto) desde el Polo Norte hasta la fábrica especificada.
Es básicamente un algoritmo de dijkstra en el que se toma cada vértice con distancia infinita desde ningún padre y a partir del vértice origen se van armando los caminos más cortos que se pueden hacia los vecinos y luego con estos sucesivamente, procesando siempre el que a menor distancia se encuentra (y no fue visitado antes). Esto se hace hasta saber las distancias mínimas a cada vértice y viniendo desde que otro vértice, el padre, para luego simplemente empezar a insertar en una lista el destino final, luego su padre, y así repetimos hasta llegar al vértice origen. Esto nos da el camino optimo pues el peso de las aristas va a ser el menor, y en este caso, el peso de las aristas es la distancia, por lo que el camino será el más corto.

Opcional:

• listar_juguetes
Lista los juguetes a retirar de una fábrica, que dan valor de felicidad máximo.
Para conseguir las sonrisas máximas, anteriormente se describió que se devolvería una lista con los juguetes además de la cantidad de sonrisas de la fábrica. Eso formaría una tupla. La función para valuar juguetes devolvería una segunda tupla, la cual contendría el resultado del comando (0 o -1 dependiendo del caso) y la primer tupla. Es por eso que, haciendo uso de la función de valuar juguetes, se podrá listar en el main los diversos juguetes que será necesario retirar de la fábrica con el id pasado por parámetro. Recorriendo la lista de juguetes, se imprime por pantalla el id de los mismos.

Implementación.

Ver anexo.