Universidad Andrés Bello

Algoritmos y Estructuras de Datos
Laboratorio 1 – Programación en C: Aplicación de Listas Dinámicas.

Profesores: Elizabeth Montero e Irene Zuccar

Ayudantes: Matías Vargas, Kevin Muena y Matías Poblete

Universidad Andrés Bello, Facultad de Ingeniería
Santiago-Viña del Mar, Chile.

FECHA DE ENTREGA: Domingo 14 de abril, 23:55 hrs

1. Objetivo

El objetivo de este Laboratorio es:

- Diseñar e implementar un TDA lineal que permita representar los principales componentes del problema enunciado a resolver.
- Diseñar e implementar las operaciones necesarias para la implementacón del algoritmo que resuelve el problema.
- Comprender y solucionar un problema clásico de las Ciencias de la Computación.

2. Enunciado

Encontrar el camino más corto desde un punto de inicio hasta un punto de destino dado un conjunto de puntos y las longitudes de ruta que los conectan es un problema ya conocido, e incluso es parte de nuestra vida cotidiana, ya que los programas de ruta más corta están ampliamente disponibles en la actualidad.

El problema de encontrar el camino más corto que visita exactamente una vez cada conjunto de nodos y vuelve al nodo inicial al final del recorrido se conoce como el problema del vendedor viajero y se ha estudiado ampliamente en Ciencias de la Computación.

Problema del Vendedor Viajero (Traveling Salesman Problem)

En el TSP se busca encontrar el ciclo hamiltoniano de costo mínimo. En el TSP simétrico la distacia de ir desde la ciudad i a la ciudad j es la misma que ir desde la ciudad j a la ciudad i. En este Laboratorio se resuelven instancias del TSP simétrico. La figura 1 muestra el ciclo hamiltoniano de costo mínimo para una versión modificada de la instancia `burma14.tsp`.

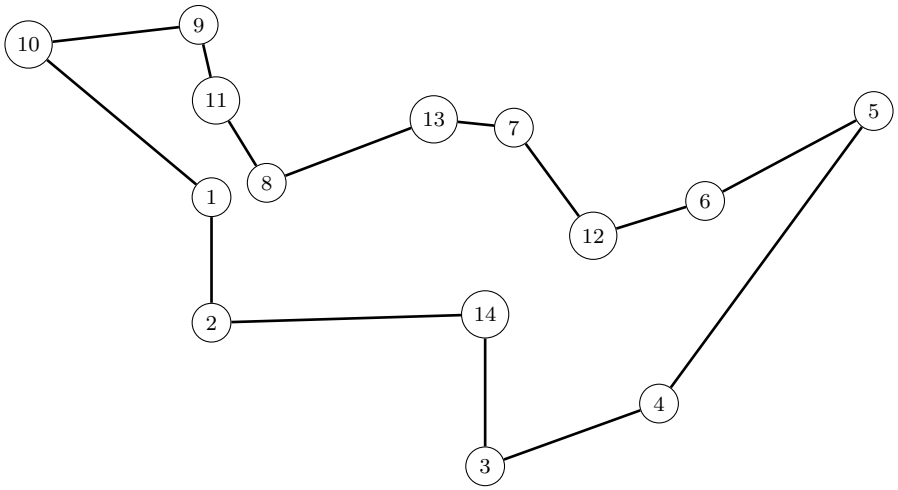


***Figura* 1:** Ejemplo instancia burma14

Heurística inserción más barata

Un algoritmo greedy consiste en un procedimiento iterativo que, comenzando con una solución inicialmente vacía, agrega componentes que optimizan una heurística particular.

El criterio heurístico *inserción más barata*, para el vendedor viajero, consiste en insertar en la mejor posición posible la ciudad que genera el menor costo. Así, comenzando con tres ciudades cualquiera, el procedimiento *insercion-mas-barata* se encarga de agregar en la solución actual la ciudad cuya inserción genera menor costo. El procedimiento finaliza cuando se han agregado todas las ciudades.

Para este Laboratorio se le pide encontrar el ciclo que se obtiene usando una simplificación de la heurística *inserción más barata* que, partiendo de tres ciudades aleatorias, inserta cada ciudad, en el orden en que aparecen en el archivo de entrada, en la mejor posición posible del ciclo actual.

2.0.1. Ejemplo

La figura 2 muestra un ejemplo para la instancia burma14 considerando como punto de partida los nodos 3, 8 y 14. Existe sólo una forma de conectar tres ciudades. Entre paréntesis se indica la distancia euclidiana total del ciclo que visita 3, 8 y 14.

A partir de estos tres nodos se debe elegir donde insertar la ciudad 1 que genera menor costo. Los esquemas de la figura 3 muestran las tres posibles opciones que se deben analizar para la inserción de la ciudad 1. En cada caso se indica la distancia total del nuevo ciclo.

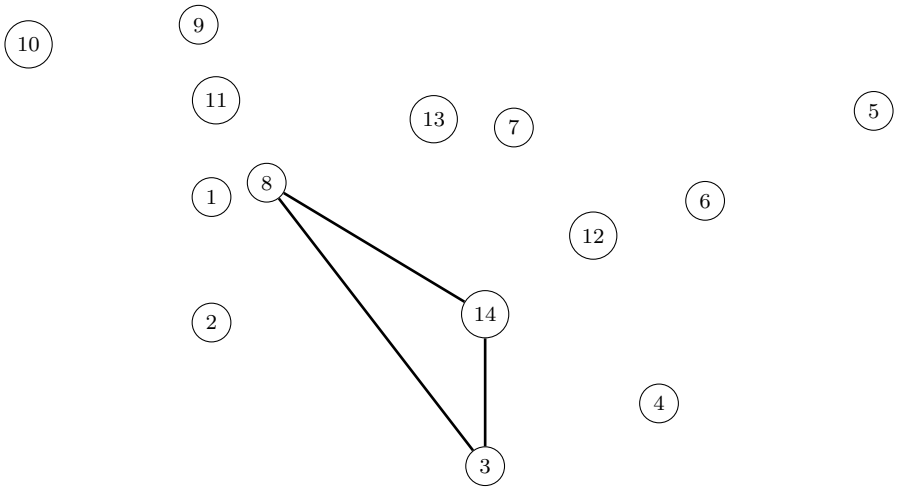
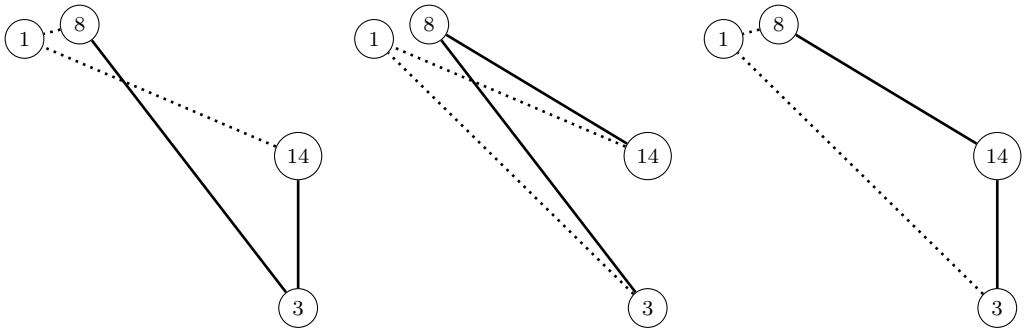


***Figura* 2:** Ejemplo – configuración inicial: 8-14-3 (10.11)



(a) 8-**1**-14-3 (11.44) (b) 8-14-**1**-3 (17.12) (c) 8-14-3-**1** (11.21)

***Figura* 3:** Ejemplo – posibles inserciones de la ciudad 1

Luego de decidir la inserción menos costosa (que en este caso corresponde a la de la figura 3, caso (c)) se repite el procedimiento.

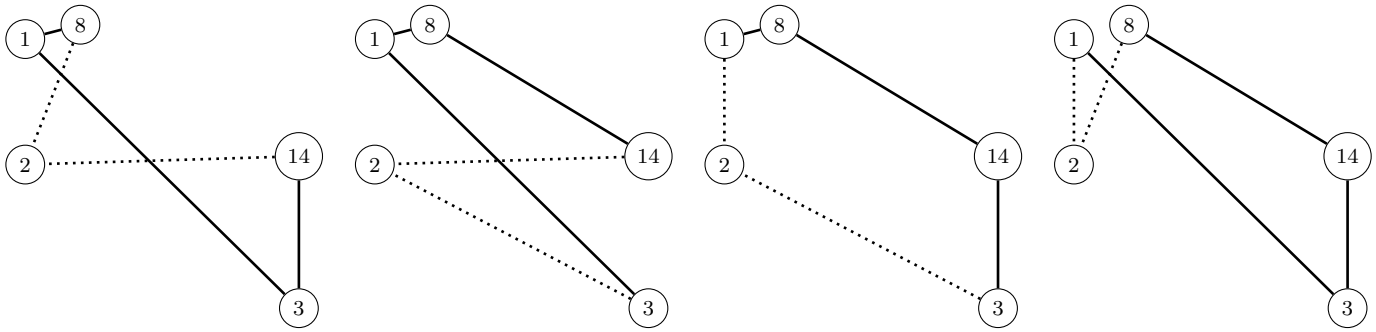
Esta vez se debe elegir la posición en la cual insertar la ciudad 2. Los esquemas de la figura 4 muestran las cuatro posibles opciones que se deben analizar para la inserción de la ciudad 2. En cada caso se indica la distancia total del nuevo ciclo.

Así, la mejor opción es la de la figura 4 (c). El proceso continúa hasta insertar todas las ciudades en el ciclo.

2.1. Entrada

La primera línea de un caso de prueba contiene un entero N($2 \leq N \leq 500$) indicando la cantidad de ciudades del problema. Cada una de las líneas siguientes indica el identificador de la ciudad y las coordenadas de la ciudad en un mapa cartesiano. La figura 5 muestra un ejemplo de la entrada especificada para el problema que se muestra en la figura 1.

En la segunda línea del archivo, vendrán los identificadores de las 3 ciudades iniciales, que debe considerar como punto de partida para construir la solución.



(a) 8-**2**-14-3-1 (13.45) (b) 8-14-**2**-3-1 (16.91) (c) 8-14-3-**2**-1 (11.87) (d) 8-14-3-1-**2** (14.11)

***Figura* 4:** Ejemplo – posibles inserciones de la ciudad 2

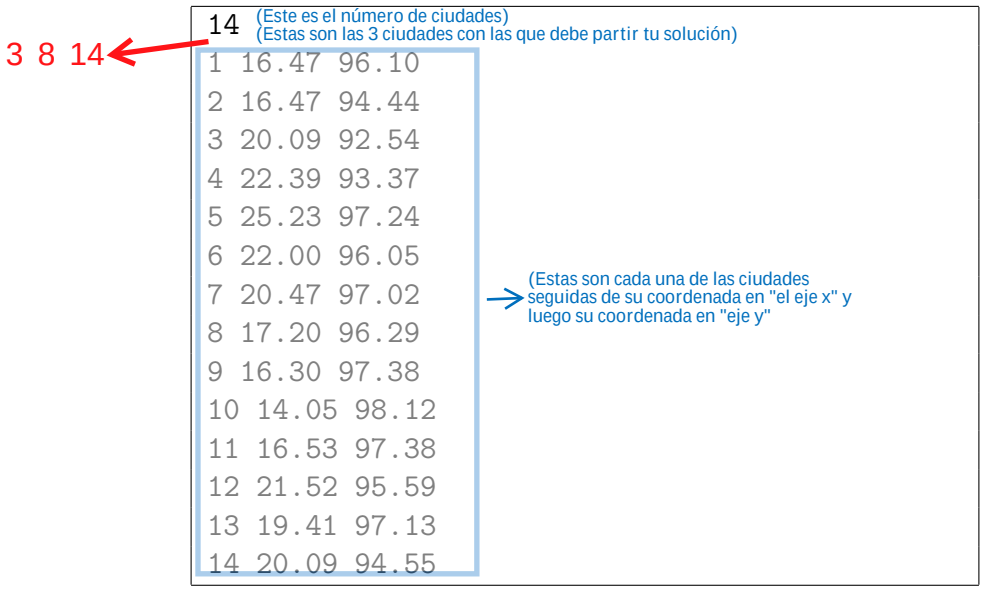


***Figura* 5:** Ejemplo entrada

2.2. Salida

Para cada caso de prueba, su programa debe imprimir una línea, indicando la distancia del ciclio hamiltoniano más corto encontrado seguido del camino correspondiente. La salida debe mostrarse por salida estándar. La figura 6 muestra un ejemplo de la salida especificada.

```
elizabeth@cancun: $ ./a.out
31.54
1 2 14 3 4 5 6 12 7 13 8 11 9 10
```

***Figura* 6:** Ejemplo salida

3. Lo que se pide en este laboratorio

Se espera que usted construya un programa escrito en C, que resuelva el problema del vendedor viajero usando un algoritmo *greedy*. Para esto, su programa debe leer un archivo de texto con los datos de la red y determinar el camino más corto usando la heurística especificada.

Dentro de su programa usted debe diseñar e implementar un TDA para el manejo de las listas usadas por el algoritmo. Este TDA se debe basar en una implementación de listas dinámicas, que corresponde a listas dinámicas simplemente enlazadas.

4. Acerca de la Entrega

4.1. Reglas

- Se realiza en equipos de dos personas.
- Se entrega a través de moodle a más tardar a las 23:59 h. del día especificado.
- Se debe entregar como un archivo comprimido que contenga el manual de usuario en pdf y el código fuente de la implementación. Los entregables se detallan en la sección 4.2.
- El nombre del archivo debe ser "L1-{`Apellido1`}-{`Apellido2`}.{`extension`}". Donde `Apellido1` y `Apellido2` corresponden a los apellidos de cada uno de los integrantes del equipo y `extension` $\in$ {~~`zip`~~, `rar`, `tar.gz`}.
- No se revisarán tareas atrasadas.

Además considere las siguientes reglas de evaluación:

- Ante la ausencia de alguno de los archivos requeridos, la nota del Laboratorio será un 1.0.
- Si el programa no se puede ejecutar, se evaluará el Laboratorio con nota máxima 4.0. Si el programa funciona, se evaluará su ejecución, y también el código fuente.
- Si obtiene una nota menor a 4.0 en el programa, no se evaluará el manual de usuario. Si obtiene una nota igual o superior a 4.0, la nota de este laboratorio será 80 % de la nota en el programa y un 20 % de la nota obtenida en el manual de usuario.
- Si existe sospecha de copia (con otros compañeros, o desde internet) se evaluará el Laboratorio con nota mínima: 1.0.
- Las consultas las debe realizar directamente al ayudante, de lunes a viernes a los correos especificados o directamente en horario de laboratorio.

4.2. Entregables

La entrega considera el código fuente ~~un manual de usuario.~~ y un archivo makefile para compilar (en linux)

4.3. Código fuente

1. Debe implementar este trabajo en lenguaje C (no C++).
2. Debe incluir un *makefile* para compilar.
3. Debe construir una función para cada tarea que realice su código, cuidando los tipos de datos de las entradas y de las salidas.
4. Debe usar identificadores representativos para sus variables, parámetros de entrada y para sus funciones.
5. Debe comentar cada una de sus funciones, estructuras y tipos de datos que defina, indicando una descripción de la labor que lleva a cabo, sus entradas, y sus salidas.
6. Su código debe estar correctamente indentado (uso de sangrías para cada subbloque de instrucciones).
7. Puede trabajar con el IDE, entorno y compilador de C, que más le acomode. No obstante lo anterior, su código debe poder ser compilado y ejecutado sin problemas en Linux.
8. El sistema debe ser robusto, se penalizarán los errores no manejados, de cualquier tipo.

4.4. Manual de usuario

1. El manual de usuario debe incluir una detallada explicación de la forma en la que se debe ejecutar su programa.
2. Debe mostrar, al menos, un ejemplo de lo que se debe esperar como salida para una entrada determinada.
3. Debe indicar el comportamiento de su programa, ante los posibles escenarios que puedan ocurrir cada vez que el usuario digite algo, ya sea válido o inválido, con al menos un ejemplo.
4. Se considera dentro de la evaluación del manual de usuario, ortografía y redacción.

No se pedirá manual de usuario. Sin embargo, tu programa debe tener exactamente la siguiente interfaz, al momento de ser ejecutado:

Ingrese nombre de archivo: Aquí el usuario digitará el nombre del archivo de texto (con punto y extensión)
El ciclo hamiltoniano es: Aquí tu programa debe mostrar la secuencia de ciudades que construyó como solución.
El costo del ciclo es: Aquí tu programa debe imprimir con 2 decimales de precisión el costo total de la solución.