**TSP**

**Manual de Usuario**

1. El siguiente manual explica cómo utilizar dos funciones que nos permiten recorrido más corto por una serie de nodos de modo que el nodo de inicio sea igual al nodo de fin y que no se pase dos veces por el mismo nodo. Este es conocido como el Problema del Agente Viajero y se trata de un problema NP Duro, por lo que hasta ahora no se conocen soluciones en tiempo polinomial. Una de las funciones encuentra este recorrido y costo mínimo usando fuerza bruta al probar con todas las permutaciones posibles. Estas funciones se realizaron a través de una clase en el lenguaje de programación Java. Para utilizar esta clase es necesario tener Java instalado en el equipo y colocar el archivo de texto con los datos de entrada dentro de la misma carpeta en donde se encuentre el archivo Java que acompaña a este manual. Esta API fue construida y probada utilizando el IDE NetBeans 8.0.2.

**Importante: Para hacer funcionar este programa fue necesario utilizar un equipo HP con 16 GB de memoria RAM y dentro de la configuración del IDE se necesitó configurar el tamaño máximo del heap a 14 GB.**

Se mostrarán las funciones de las clases y algunos de sus ejemplos de uso.

1. **La clase TSP**

Esta clase es la básica primaria que guardará todos las funciones de nuestra implementación. Cada función lee el grafo del archivo que recibe como parámetro, trata de obtener el recorrido más corto y muestra su costo así como el orden de los nodos en que se recorrieron. Además, imprime el tiempo de ejecución del programa.

1. **Formato del archivo de texto**

Para implementar las funciones que se mencionaran es necesario que el archivo de texto que se pasa como parámetro cumpla con las siguientes especificaciones que sirven para describir un grafo.

1. La primera línea contiene un entero positivo **N**, que corresponde a la cantidad de vértices en el grafo.
2. Las siguientes **N** líneas consisten de dos valores enteros o flotantes: **xi** y **yi**, mismos que están separados por un espacio simple. Estos tres valores constituyen un punto del grafo y de esta forma la distancia entre dos nodos se calcula como la distancia euclideana.
3. Importante: En Netbeans es necesario escribir los números flotantes usando una coma para separar la parte entera de la parte decimal del número. Este problema no ocurre en Eclipse pero es necesario aclararlo.
4. **Función void fuerzaBruta(String archivo)**

Recibe el archivo y por medio de búsqueda exhaustiva encuentra el recorrido más corto. La función imprime los valores requeridos: costo mínimo, orden en que se deben visitar los nodos así como el tiempo de ejecución. El orden en que se visitan los nodos es siempre iniciando en el vértice 0 y terminando en ese mismo vértice. El orden asintótico de este algoritmo es de **O(N!)**

1. **Función void DP(String archivo)**

Recibe el archivo y por medio de programación dinámica encuentra el recorrido con menor costo y regresa el orden en que se visitaron los nodos, comenzando siempre con el nodo número 0. El orden asintótico de este algoritmo es de **O(N22N)** en tiempo y **O(N2N)** en memoria.

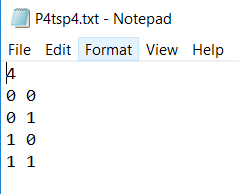
1. **Análisis:**

Para el algoritmo con fuerza bruta,El orden asintótico es de **O(N!)** tanto en tiempo como en memoria pues la recuersión requiere de una pila que se llena con cada permutación.

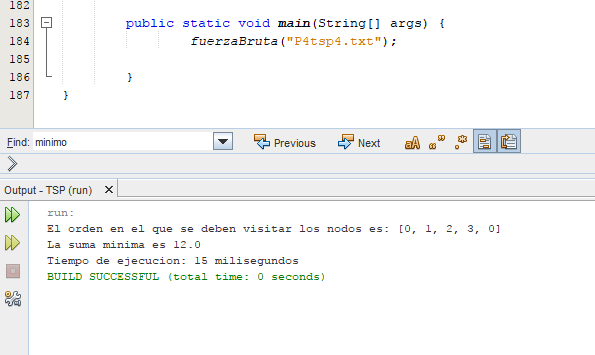
Para el algoritmo con programación dinámica, el orden asintótico es de **O(N22N)** en tiempo y **O(N2N)** en memoria.

1. **Ejemplo de implementación N=4**

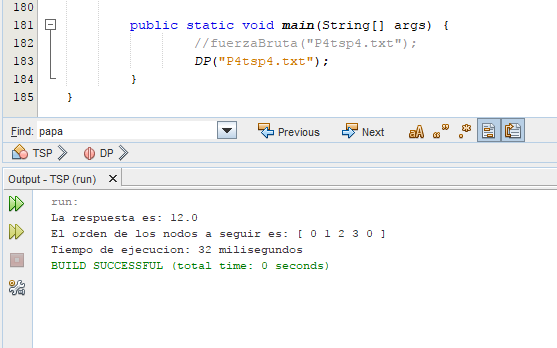
Archivo de texto:



Usando Fuerza Bruta:

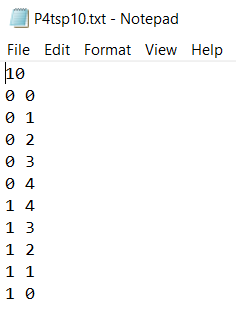


Usando Fuerza Programación Dinámica:

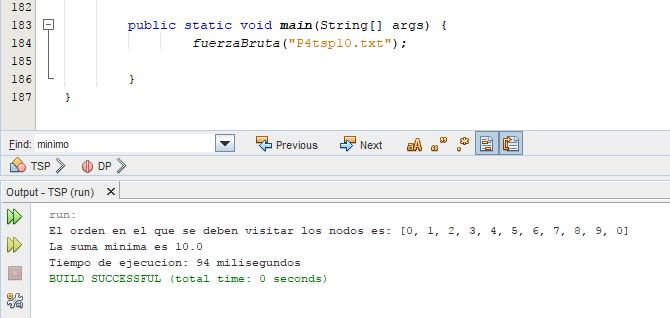


1. **Ejemplo de implementación N=10**

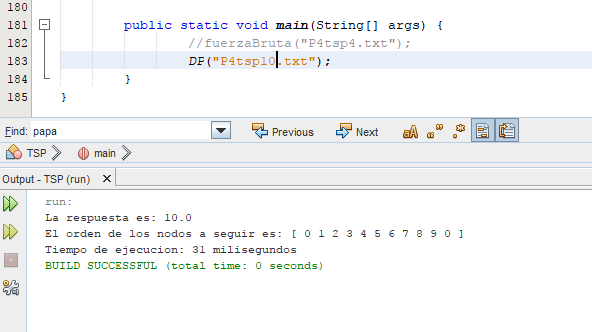
Archivo de texto:



Usando Fuerza Bruta:

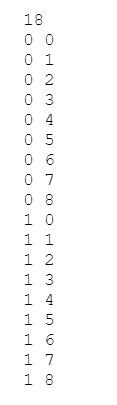


Usando Fuerza Programación Dinámica:



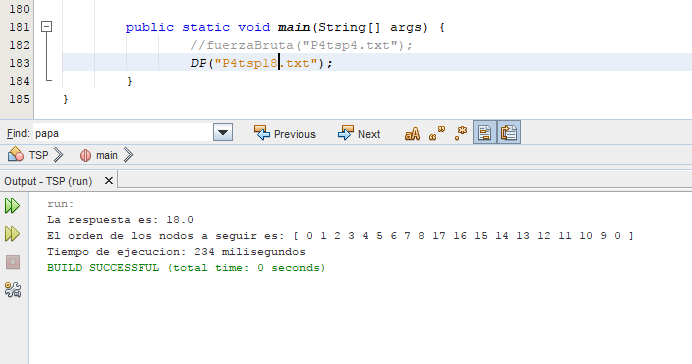
1. **Ejemplo de implementación N=18**

Archivo de texto:



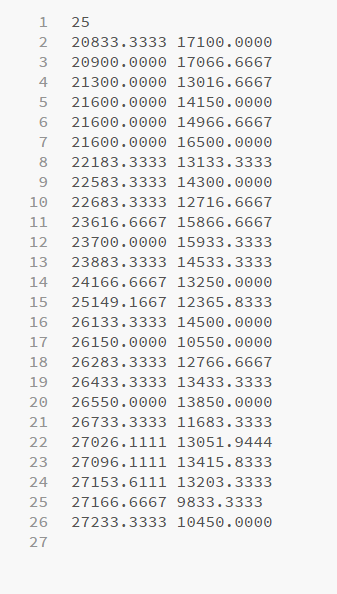
Usando Fuerza Bruta: Imposible por limitantes de tiempo.

Usando Fuerza Programación Dinámica:



1. **Ejemplo de implementación N=25**

Archivo de texto:



Usando Fuerza Bruta: Imposible por limitantes de tiempo.

Usando Fuerza Programación Dinámica:

