EXPLICACION DE CASOS DE EJECUCION

La eficiencia del algoritmo obliga a que el número de ciudades con el que se realizan los casos de ejecución sea muy limitado, pero a continuación podemos ver los resultados y procedimientos que se obtienen de nuestro algoritmo con diferentes cantidades de ciudades.

1.- Nº CIUDADES = 4 (MOSTRANDO CAMINOS POSIBLES)

En este caso se ha realizado un programa que realiza el mismo procedimiento para obtener las distancias que en el resto de casos, pero la salida por pantalla está adaptada a poder ver los caminos posibles y cuál de ellos está comprobando en cada momento.

En la primera imagen vemos, con 4 ciudades, cuál es el árbol que contiene todos los caminos posibles que parten de la ciudad 1 y, sin repetir ciudad, vuelven a la ciudad 1 pasando por todas las restantes.

En las siguientes capturas se ve qué distancia supondría coger cada uno de los caminos. En eso consiste el algoritmo, la comprobación de combinar las ciudades de todas las formas posibles sin repeticiones y quedarse con el camino más corto. Por ello, dibujar el árbol con todos los caminos de 7 ciudades, por ejemplo, contendría 720 caminos, lo cual es imposible mostrar por pantalla. Esta es la razón por la que observamos las posibilidades con 4 ciudades únicamente para hacerlo más gráfico, pero a partir de este ejemplo las ejecuciones se limitan a mostrar distancias y recorridos.

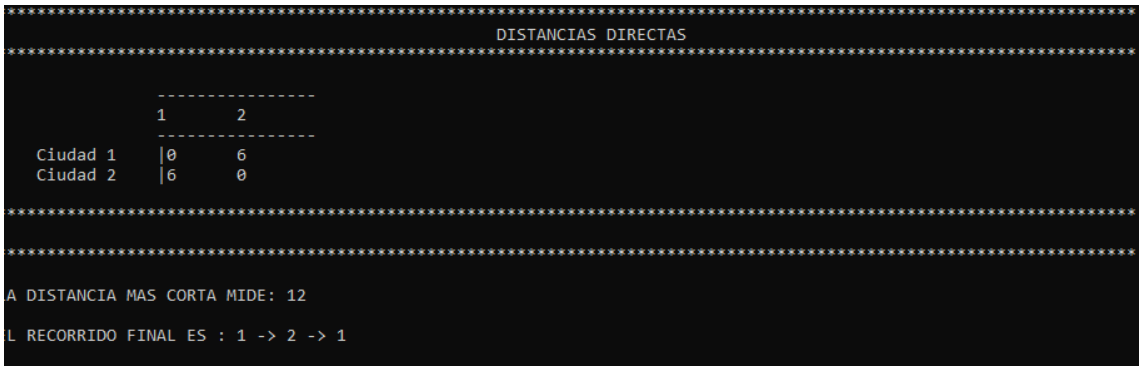
```
******************************************************************************************************************
        *1*
       / | \
      /  |  \
     /   |   \
    /    |    \
  *2*    3     4
  / \   / \   / \
*3*  4 2   4 2   3
 |   | |   | |   |
*4*  3 4   2 3   2
 |   \ \   / /   |
 \___ \ \ / / ___/
      \\_|_//
         |
        *1*
Cogiendo el camino marcado con '*' la longitud es: 15
******************************************************************************************************************
        *1*
       / | \
      /  |  \
     /   |   \
    /    |    \
  *2*    3     4
  / \   / \   / \
 3  *4*2   4 2   3
 |   | |   | |   |
 4  *3*4   2 3   2
 |   \ \   / /   |
 \___ \ \ / / ___/
      \\_|_//
         |
        *1*
Cogiendo el camino marcado con '*' la longitud es: 19
******************************************************************************************************************
        *1*
       / | \
      /  |  \
     /   |   \
    /    |    \
   2    *3*    4
  / \   / \   / \
 3   4*2*  4 2   3
 |   | |   | |   |
 4   3*4*  2 3   2
 |   \ \   / /   |
 \___ \ \ / / ___/
      \\_|_//
         |
        *1*
Cogiendo el camino marcado con '*' la longitud es: 14
******************************************************************************************************************
        *1*
       / | \
      /  |  \
     /   |   \
    /    |    \
   2    *3*    4
  / \   / \   / \
 3   4 2  *4*2   3
 |   | |   | |   |
 4   3 4  *2*3   2
 |   \ \   / /   |
 \___ \ \ / / ___/
      \\_|_//
         |
        *1*
Cogiendo el camino marcado con '*' la longitud es: 19
```

Es fácil observar a raíz de las imágenes anteriores que con pocas ciudades es altamente probable que diferentes caminos resulten en las mismas distancias. Ante estas situaciones de conflicto, el programa se queda con el primer camino obtenido y evita rehacer la secuencia final para obtener mismas distancias.

De entre todas las opciones, la más corta es aquella que partiendo de una pasa por las ciudades 3, 2, 4 y regresa a 1, con una distancia de 14.

2.- Nº CIUDADES = 2

Ahora, comprobamos la ejecución del caso mínimo, un recorrido que solamente pase por una ciudad diferente a esta de la que se parte.



Como era de esperar, el camino directamente va de la ciudad origen a la restante y de vuelta, lo cual tiene una longitud del doble de la distancia de una ciudad a la otra (ida y vuelta).

3.- Nº CIUDADES = 4

Otra ejecución con un fichero que contiene 4 ciudades.

```
**************************************************************************************************************************
                                                DISTANCIAS DIRECTAS
**************************************************************************************************************************

                ---------------------------------
                1       2       3       4
                ---------------------------------
|   Ciudad 1    |0      6       5       4
|   Ciudad 2    |6      0       1       4
|   Ciudad 3    |5      1       0       4
|   Ciudad 4    |4      4       4       0

**************************************************************************************************************************

**************************************************************************************************************************

LA DISTANCIA MAS CORTA MIDE: 14

EL RECORRIDO FINAL ES : 1 -> 3 -> 2 -> 4 -> 1
```

Nº CIUDADES = 4 PERO SALIENDO DE LA CIUDAD 3

Una ventaja del código (aunque poco necesaria) es la flexibilidad que ofrece a la hora de seleccionar la ciudad de partida. Como el camino que busca es mínimo, lo será saliendo de cualquiera de las ciudades, como comprobaremos más tarde, pero puedes probarlo teniendo en cuenta la parte del código mostrada a continuación. El conjunto de funciones involucradas en el cálculo del camino está adaptado a construirse alrededor de la ciudad de origen y se puede modificar esta cambiando el valor en la variable marcada a continuación. Esta implementación hace fácil que el usuario pueda indicar como parámetro la ciudad que quiere como primera, pero no tiene por qué conocer cuantas ciudades hay en el fichero seleccionado, lo cual podría provocar problemas. (Al modificarlo hay que tener en cuenta que ciudadInicio = índice de la ciudad – 1)

```
/**
* @brief Funcion para mostrar la informacion final por pantalla
* @param solucion pair que contiene tanto la distancia minima como la secuencia de ciudades a seguir
* @param ciudadInicio variable que contiene la ciudad de la que se inicia el recorrido
*/
void ResultadoFinal(pair<int,vector<int> > solucion, int ciudadInicio){

    cout << "\nLA DISTANCIA MAS CORTA MIDE: " << solucion.first << endl ;
    cout << "\nEL RECORRIDO FINAL ES : " ;
    for(int i=solucion.second.size()-1; i>=0; i--)         //Muestra el recorrido al reves porque la recursividad hace que los nodos se almacenen de destino a origen, al reves
        cout << solucion.second[i]+1 << " -> " ;

    cout<< ciudadInicio+1 << "\n";     //Aniade al recorrido final que se muestra por pantalla la vuelta al nodo de inicio

}


int main(int argc, char * argv[]){

    vector<Ciudad> ciudadesRecogidas;                       //vector de ciudades que recogeremos del fichero
    int dimension = obtenerInfoFichero(ciudadesRecogidas, argv[1]); //rellenamos el vector ciudadesRecogidas con las ciudades del fichero y devuelve la cantidad de ciudades que hay
    int ciudadInicio = 2;                                   //Variable que indica la ciudad de la que parte el camino (en nuestro caso el camino empieza y termina en la primera)
    vector<int> vectorNodosPaso;                            //Vector que contiene el resto de ciudades por las que pasar
    pair<int,vector<int> > solucion;                        //Pair que contendra la solucion final al problema. En el primer elemento la distancia mas corta y en el segundo el recorrido final
    int **distancias_directas;                              //Creo la matriz de distancias entre ciudades

    //Reserva para la matriz de distancias
    distancias_directas = new int *[dimension];

    for (int i = 0; i < dimension; i++)
```

Con esto simplemente se quiere demostrar que saliendo de cualquiera de las ciudades se obtiene el camino más corto, pero con un recorrido adaptado a las necesidades de salir de otra que no sea la 1.

```
****************************************************************************************************
                                            DISTANCIAS DIRECTAS
****************************************************************************************************


              -------------------------------
              1       2       3       4
              -------------------------------
|   Ciudad 1    |0      6       5       4
|   Ciudad 2    |6      0       1       4
|   Ciudad 3    |5      1       0       4
|   Ciudad 4    |4      4       4       0

****************************************************************************************************

****************************************************************************************************

LA DISTANCIA MAS CORTA MIDE: 14

EL RECORRIDO FINAL ES : 3 -> 1 -> 4 -> 2 -> 3
```

__4.- Nº CIUDADES = 7__

Ejemplo como los anteriores pero con 7 ciudades.

```
****************************************************************************************************
                                            DISTANCIAS DIRECTAS
****************************************************************************************************


              -------------------------------------------------------
              1       2       3       4       5       6       7
              -------------------------------------------------------
|   Ciudad 1    |0      6       5       4       11      8       7
|   Ciudad 2    |6      0       1       4       17      14      13
|   Ciudad 3    |5      1       0       4       17      13      12
|   Ciudad 4    |4      4       4       0       13      11      10
|   Ciudad 5    |11     17      17      13      0       4       6
|   Ciudad 6    |8      14      13      11      4       0       1
|   Ciudad 7    |7      13      12      10      6       1       0

****************************************************************************************************

****************************************************************************************************

LA DISTANCIA MAS CORTA MIDE: 35

EL RECORRIDO FINAL ES : 1 -> 3 -> 2 -> 4 -> 5 -> 6 -> 7 -> 1
```

__Nº CIUDADES = 7 PERO SALIENDO DE LA CIUDAD 3__

Y de nuevo salimos de otra ciudad y comprobamos que da la misma distancia.

```
****************************************************************************************************
                                            DISTANCIAS DIRECTAS
****************************************************************************************************


              -------------------------------------------------------
              1       2       3       4       5       6       7
              -------------------------------------------------------
|   Ciudad 1    |0      6       5       4       11      8       7
|   Ciudad 2    |6      0       1       4       17      14      13
|   Ciudad 3    |5      1       0       4       17      13      12
|   Ciudad 4    |4      4       4       0       13      11      10
|   Ciudad 5    |11     17      17      13      0       4       6
|   Ciudad 6    |8      14      13      11      4       0       1
|   Ciudad 7    |7      13      12      10      6       1       0

****************************************************************************************************

****************************************************************************************************

LA DISTANCIA MAS CORTA MIDE: 35

EL RECORRIDO FINAL ES : 3 -> 1 -> 7 -> 6 -> 5 -> 4 -> 2 -> 3
```

5.- Nº CIUDADES = 9

Una vez comprobado que la distancia es la misma desde cualquier ciudad, volvemos al caso de partir de la 1 y probamos con 9 ciudades.

```
DISTANCIAS DIRECTAS

             1     2     3     4     5     6     7     8     9
| Ciudad 1  |0     6     5     4     11    8     7     1     11
| Ciudad 2  |6     0     1     4     17    14    13    6     17
| Ciudad 3  |5     1     0     4     17    13    12    6     16
| Ciudad 4  |4     4     4     0     13    11    10    3     15
| Ciudad 5  |11    17    17    13    0     4     6     11    8
| Ciudad 6  |8     14    13    11    4     0     1     8     5
| Ciudad 7  |7     13    12    10    6     1     0     7     5
| Ciudad 8  |1     6     6     3     11    8     7     0     12
| Ciudad 9  |11    17    16    15    8     5     5     12    0

LA DISTANCIA MAS CORTA MIDE: 44

EL RECORRIDO FINAL ES : 1 -> 3 -> 2 -> 4 -> 8 -> 7 -> 6 -> 5 -> 9 -> 1
```

6.- Nº CIUDADES = 13

En esta última ejecución se probó un fichero de 13 ciudades, pero deja de ser eficiente con cantidades de ciudades grandes. El tiempo obtenido hace incoherente esta opción de cálculo para 13 ciudades, ya que la espera obtenida es superior a los 30 minutos.

```
C:\Users\manue\OneDrive\Desktop\P4\TSP_PD_TIEMPOS.exe
13  1913.73

--------------------------------
Process exited after 1914 seconds with return value 0
Presione una tecla para continuar . . .
```