9

Problema 53

SOTILLO SOTILLO, CLARA (TAIS96)

ID envio	Usuario/a	Hora envío	Veredicto
64239	TAIS96	2022-11-30 10:38	AC
64231	TAIS96	2022-11-30 10:30	AC
64211	TAIS96	2022-11-30 10:08	AC

Fichero Source.cpp

```
*
* Indicad el nombre completo y usuario del juez de quienes habéis hecho esta solución:
* Estudiante 1:Carlos Gomez TAIS45
* Estudiante 2:Clara Sotillo TAIS96
*
  El coste de esta funcion en tiempo en el peor de los casos es O(n^3) siendo n el numero
  de poblados que hay
  El coste en espacio adicional de la funcion es O(1)

  Recurrencia:

      aventura(i,j): el coste minimo de cada posible viaje entre los poblados i y j.

      aventura(i,j)       {  aventura(i,i) = 0
                          {  min( aventura(i,k) + aventura(k,j) + ci)
                             i<k<j

      Tenemos un caso base , en el cual se quiere viajar al mismo poblado en el que se esta
  . En este caso
      no necesitamos el coste de ese viaje.

              aventura(i,i) = 0

      Como caso recursivo tenemos que comprobar el minimo coste posible entre viajar
  directamente de i a j o
      ir viajando desde i por k pueblos hasta llegar a j.

           aventura(i,j) = min( aventura(i,k) + aventura(k,j) + ci)
                           i<k<j

      Implementamos la funcion de manera ascendente rellenando la tabla de tamaño (n+1)(n
  +1) ya que vamos a
      recorrer del poblado mas cercano al mas lejano.

      Primero creamos una matriz en la que guardamos el coste de los alquileres en la
  diagonal superior de esta.
      Una vez rellenada, implementamos la funcion aventura descrita anteriormente.
      De esta funcion obtenemos una matriz, con los resultados requeridos.
```

```
void aventura(vector<vector<int>> const& alquiler, Matriz<int>& m) {

    int n = alquiler.size();
    m = Matriz<int>(n + 1, n + 1, 0);
    for (int d = 1; d <= n - 1; ++d){ // recorre diagonales
        for (int i = 1; i <= n - d; ++i) { // recorre elementos de diagonal
            int j = i + d;
            m[i][j] = alquiler[i - 1][j - 1];
            for (int k = i; k <= j - 1; ++k) {
                int temp = m[i][k] + m[k][j];
                if (temp < m[i][j]) { // es mejor partir por k
                    m[i][j] = temp;
                }
            }
        }
    }
}

bool resuelveCaso() {

    // leemos la entrada
    int N;
    cin >> N;

    if (!cin)
        return false;

    // leemos los alquileres
    vector<vector<int>> alquiler(N, vector<int>(N, 0));
    for (int i = 0; i < N - 1; ++i) {
        for (int j = i + 1; j < N; ++j) {
            cin >> alquiler[i][j];
        }
    }

    Matriz<int> m;
    aventura(alquiler, m);

    for (int i = 1; i <= N ; ++i) {
        for (int j = i + 1; j <= N; ++j) {
            cout << m[i][j]<<"␣";
        }
        cout << "\n";
    }

    // resolver el caso
```

```
    return true;
}
```