Documentatie implementare secventiala Algoritm Dijkstra

1. Detalii despre IDE-ul si sistemul pe care a fost rulat algoritmul

Algoritmul a fost implementat si testat in C++ folosind IDE-ul Visual Studio 2019 pe laptopul personal. Laptopul are urmatoarea configuratie:

```
Operating System: Windows 10 Pro 64-bit (10.0, Build 18363)
Language: English (Regional Setting: English)
System Manufacturer: Dell Inc.
System Model: Latitude 5500
BIOS: 1.6.5
Processor: Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz (8 CPUs), ~2.1GHz
Memory: 16384MB RAM
Page file: 9841MB used, 8785MB available
DirectX Version: DirectX 12
```

2. Reprezentarea datelor:

A) de intrare:

- Se citesc din fisiere .txt
- Structura fisierelor .txt este urmatoarea:
  - Pe prima line se afla numarul de noduri si numarul de arce (n si m), 0<n<=297, 0<m<=2148
  - Pe a doua linie se afla nodul sursa si destinatie (source si destination), 0<destination, source<=n
  - Pe urmatoarele m linii se afla muchia si costul acesteia. Muchia este reprezentata prin perechea de noduri (x,y), iar costul prin cost (0<x,y<=n si cost <= 9999)
  - Datele de pe fiecare linie sunt separate prin spatiu

B) de iesire:

- Se afiseaza atat in fisiere .txt cat si in fisier .csv fiecare continand outputul corespunzator
- Structura fisierelor .txt este urmatoarea:
  - Daca exista drum intre nodul sursa si destinatie atunci:
    - Prima linie contine costul total al drumului. Prin cost total se intelege suma tuturor costurilor de pe fiecare muchie de pe cel mai scurt drum identificat.

        - A doua linie, separat prin spatiu, contine drumul de la nodul sursa la destinatie. Prin drum se intelege multimea nodurilor care il reprezinta.
    - Daca nu exista drum intre nodul sursa si destinatie atunci fisierul .txt va contine o singura linie cu un mesaj de atentionare.
- Rolul fisierului .csv este de a salva timpii de executie precum si dimensiunea grafului. Structura fisierului este urmatoarea:
    - Prima coloana reprezinta numarul de noduri n
    - A doua coloana, numarul de arce
    - A treia, nodul sursa
    - A patra, nodul destinatie
    - A cincea, timpul de executie reprezentat prin microsecunde

C) structuri de date folosite:

- Vectori normali din C/C++ (p[MAX], dist[MAX] unde MAX este o constanta definita de catre mine)
- Un dictionar care are ca si cheie primara un nod x, iar ca si valoare un vector de structuri de noduri, unde prin structuri de noduri se intelege nodul adiacent nodului reprezentat de cheie si costul aferent acestei muchii. Acest dictionar este simulat printr-un vector din stl astfel: vector<Node>G[MAX], unde Node este o structura care are urmatoarea reprezentare:

```
/*the struct of the node*/
struct Node {
    int y;
    int cost;
    Node(const int& y, const int& cost) {
        this->y = y;
        this->cost = cost;
    }
    /*overriding the < operator*/
    bool operator < (const Node& other) const {
        return this->cost > other.cost;
    }
};
```

- Coada cu prioritati, unde prioritatea este definita in felul urmator: are prioritate muchia cu costul cel mai mic. Aceasta este definita: priority_queue<Node>Q
- Un iterator peste dictionar definit vector<Node>::iterator it
- Stiva din stl pentru a afisa drumul de la sursa la destinatie si nu invers. Stiva este definita astfel: stack<int>s

3. Proiectarea implementarii