Ministerul Educației și Cercetării al Republicii Moldova
CENTRUL DE EXCELENȚĂ ÎN INFORMATICĂ ȘI TEHNOLOGII INFORMAȚIONALE

Lucrare individuală N-3

Tema: Tehnici de Programare

Disciplina: Programarea calculatoarelor
Varianta: 26

Grupa: P-2423
Elev: Pricop Maxim

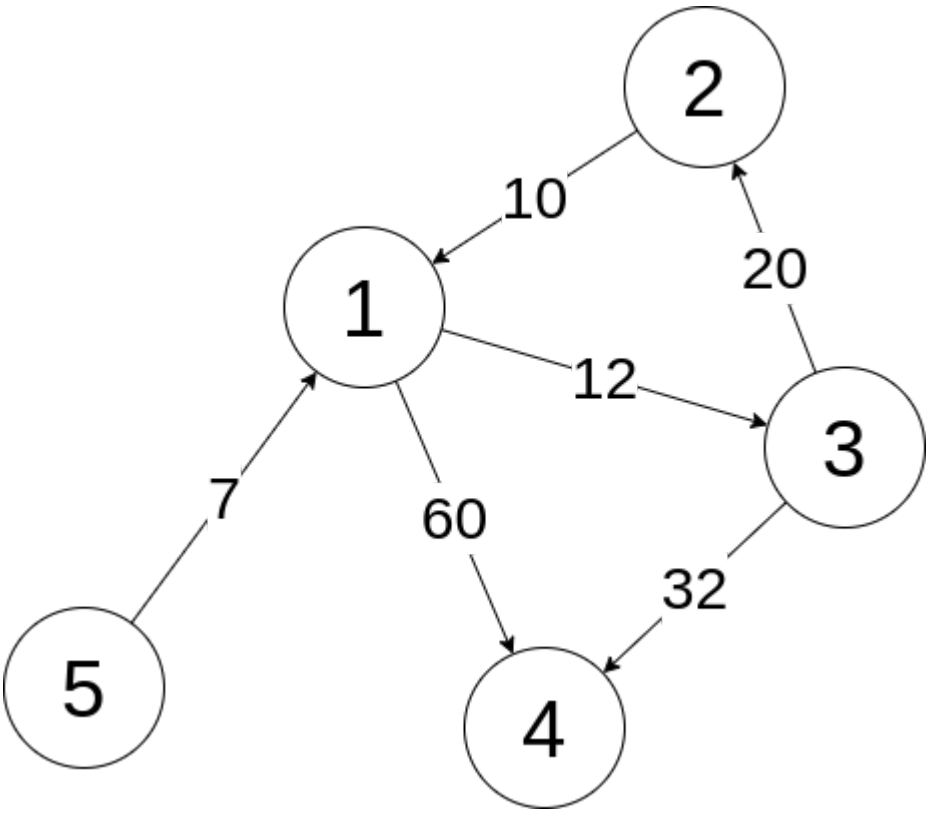
Profesor: Cojocaru Liuba

Chișinău, 2025

Sarcina

Se dă un graf orientat ponderat – în care fiecare arc are asociat un cost, număr natural strict pozitiv, și un nod p. Să se determine, folosind algoritmul lui Dijkstra, costul minim al drumului de la p la fiecare nod al grafului.

Rezultat




```
┌──(max㉿nurofen)-[~/…/max_programare_ceiti/Anul 2/programareaCalculatoarelor/Studiu Individual 3]
└─$ cmake --build out/build/clang-linux-debug --target run
[100%] Built target StudiuIndividual
Introdu numarul de noduri: 5
Introdu numarul de muchii: 6
Introdu nodul de start: 1
Introdu muchia 1 (format: nodulOrigine, nodulFinal, greutatea/costul ): 5 1 7
Introdu muchia 2 (format: nodulOrigine, nodulFinal, greutatea/costul ): 2 1 10
Introdu muchia 3 (format: nodulOrigine, nodulFinal, greutatea/costul ): 3 2 20
Introdu muchia 4 (format: nodulOrigine, nodulFinal, greutatea/costul ): 1 3 12
Introdu muchia 5 (format: nodulOrigine, nodulFinal, greutatea/costul ): 1 4 60
Introdu muchia 6 (format: nodulOrigine, nodulFinal, greutatea/costul ): 3 4 32
Distanta de la nodul 1 la nodul 1 este: 0
Distanta de la nodul 1 la nodul 2 este: 32
Distanta de la nodul 1 la nodul 3 este: 12
Distanta de la nodul 1 la nodul 4 este: 44
Distanta de la nodul 1 la nodul 5 este: -1

[100%] Built target run
```

Rezolvare

Programul este mic, fiind alcătuit doar dintr-un fișier (main.cpp):

```
#include <iostream>
#include <limits>
#include <queue>
#include <vector>

const long long INF = std::numeric_limits<long long>::max() / 4;

int main() {
   int numNodes, numEdges, startNode;

   std::cout << "Introdu numarul de noduri: ";
   std::cin >> numNodes;

   std::cout << "Introdu numarul de muchii: ";
   std::cin >> numEdges;

   std::cout << "Introdu nodul de start: ";
   std::cin >> startNode;

   std::vector<std::vector<std::pair<int, long long>>> adjacencyList(numNodes + 1);
   for (int edgeIndex = 0; edgeIndex < numEdges; ++edgeIndex) {
       int from, to;
       long long weight;

       std::cout << "Introdu muchia " << edgeIndex + 1 << " (format: nodulOrigine,
nodulFinal, greutatea/costul ): ";
       std::cin >> from >> to >> weight;

       if (from >= 1 && from <= numNodes) adjacencyList[from].emplace_back(to, weight);
   }

   std::vector<long long> distance(numNodes + 1, INF);
   std::priority_queue<std::pair<long long, int>, std::vector<std::pair<long long, int>>,
std::greater<std::pair<long long, int>>>
       priorityQueue;
   distance[startNode] = 0;
   priorityQueue.emplace(0, startNode);

   while (!priorityQueue.empty()) {
       std::pair<long long, int> topPair = priorityQueue.top();
       long long currentDistance = topPair.first;
       int currentNode = topPair.second;

       priorityQueue.pop();
```

```
        if (currentDistance != distance[currentNode]) continue;

        for (auto &edge : adjacencyList[currentNode]) {
            int to = edge.first;
            long long weight = edge.second;

            if (to < 1 || to > numNodes) continue;

            if (distance[to] > currentDistance + weight) {
                distance[to] = currentDistance + weight;
                priorityQueue.emplace(distance[to], to);
            }
        }
    }

    for (int nodeIndex = 1; nodeIndex <= numNodes; nodeIndex++) {
        std::cout << "Distanta de la nodul " << startNode << " la nodul " << nodeIndex <<
" este: ";
        if (distance[nodeIndex] == INF) std::cout << -1;
        else std::cout << distance[nodeIndex];

        std::cout << '\n';
    }

    std::cout << '\n';

    return 0;
}
```

Concluzie

Studiul Individual 3 prezintă o implementare clară și funcțională a algoritmului lui Dijkstra pentru determinarea costurilor minime într-un graf orientat ponderat.

Programul realizează o citire robustă a datelor de intrare (număr de noduri, muchii, nod sursă și muchii cu greutăți), construiește o listă de adiacență și calculează distanțele minime folosind o coadă de priorități eficientă. Au fost incluse verificări pentru a preveni accesările invalide ale listei și reprezentarea valorilor infinite, iar nodurile inaccesibile sunt semnalate clar prin afișarea valorii -1.

Testarea manuală pe scenarii tipice și de margine a confirmat corectitudinea propagării costurilor și stabilitatea programului la intrări invalide.

În ansamblu, lucrarea atinge obiectivele Studiului Individual: demonstrează înțelegerea algoritmului și a structurii de date asociate, produce rezultate predictibile și constituie o bază solidă pentru extensii viitoare.