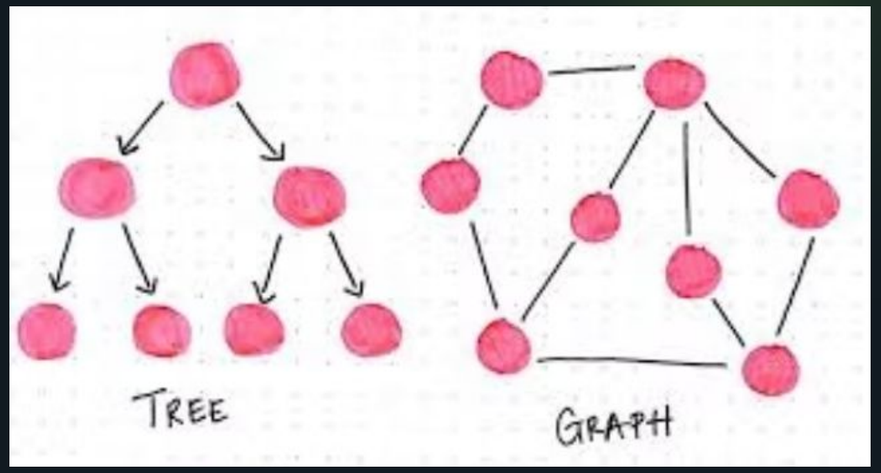
Implementacja grafów i algorytmu Dijkstry w C++

Czym jest graf?

1. Graf to sposób przedstawienia połączeń
2. Składa się z punktów (wierzchołków) i linii między nimi (krawędzi)
3. Może mieć kierunek albo nie
4. Czasem krawędzie mają "koszt" - np. odległość, czas
5. Przykład: miasta jako punkty, drogi jako połączenia



Reprezentacja grafu w C++

Jak komputer przechowuje graf?

Graf jest reprezentowany za pomocą listy sąsiedztwa. Używamy **unordered_map**, gdzie miasto wskazuje na listę sąsiadów.

```
class Graph{
    private:
        unordered_map<string, vector<pair<string, int>>> graphData;
```

```
"Warszawa" → [("Kraków", 300), ("Gdańsk", 350)]
```

Algorytm Dijkstry – zasada

1. Start z węzła początkowego
2. Eksploruj najbliższych sąsiadów
3. Aktualizuj odległości

```cpp
unordered_map<string, int> Graph::Dijkstra(const string& start) {
    unordered_map<string, int> distances;
    unordered_map<string, bool> visited;
    priority_queue<pair<int, string>, vector<pair<int, string>>, greater<pair<int, string>>> pq;

    for (auto& [city, _] : graphData) {
        distances[city] = numeric_limits<int>::max();
        visited[city] = false;
    }
    distances[start] = 0;
    pq.push({0, start});

    while (!pq.empty()) {
        auto [current_dist, current_city] = pq.top();
        pq.pop();

        if (visited[current_city]) continue;
        visited[current_city] = true;

        for (auto& [neighbour, distance] : graphData[current_city]) {
            if (!visited[neighbour] && current_dist + distance < distances[neighbour]) {
                distances[neighbour] = current_dist + distance;
                pq.push({distances[neighbour], neighbour});
            }
        }
    }

    return distances;
}
```

Wczytywanie danych z pliku

Program pozwala wczytywać dane listy sąsiedztwa z pliku.

Muszą być one w formacie:

Miasto1;Miasto2;Koszt

Na przykład:

Warszawa;Kraków;500

Szczecin;Poznań;250

lub - dla lokalnych połączeń:

Szczecin, Wyszyńskiego;Szczecin, Centrum;2

```
void Graph::loadFromFile(const string& filename) {
    ifstream file(filename);
    if (!file.is_open()) {
        cerr << "Nie można otworzyć pliku: " << filename << endl;
        return;
    }

    string line;
    while (getline(file, line)) {
        stringstream ss(line);
        string city1, city2;
        int distance;

        if (getline(ss, city1, ';') && getline(ss, city2, ';') && (ss >> distance)) {
            addGraphEdge(city1, city2, distance);
        }
    }
    file.close();
}
```

Demo programu

Program wczytuje dane grafu, wyświetla je, pozwala wybrać miasto startowe, a następnie oblicza najkrótsze ścieżki do pozostałych miast.

Dodatkowo, za pomocą biblioteki SFML wyświetla graficzną reprezentację grafu.

```
kajetanstepien@MacBookAir src % ./program
1. Wczytaj dane z pliku
2. Wprowadź dane ręcznie
Wybierz opcję: 1
Podaj nazwę pliku: miasta.txt
Miasto Białystok jest połączone z: Warszawa (200km) Lublin (250km) Gdańsk (350km) Olsztyn (150km)
Miasto Rzeszów jest połączone z: Lublin (170km) Kraków (160km) Katowice (200km)
Miasto Lublin jest połączone z: Warszawa (170km) Łódź (210km) Kraków (270km) Rzeszów (170km) Białystok (250km)
Miasto Warszawa jest połączone z: Kraków (300km) Gdańsk (350km) Poznań (310km) Wrocław (350km) Łódź (130km) Toruń (200km) Częstochowa (220km) Lublin (1
70km) Białystok (200km) Olsztyn (180km)
Miasto Bydgoszcz jest połączone z: Łódź (200km) Szczecin (300km) Gdańsk (150km) Toruń (50km)
Miasto Katowice jest połączone z: Kraków (80km) Wrocław (200km) Łódź (180km) Częstochowa (80km) Rzeszów (200km)
Miasto Szczecin jest połączone z: Gdańsk (340km) Poznań (260km) Łódź (400km) Bydgoszcz (300km)
Miasto Łódź jest połączone z: Warszawa (130km) Kraków (210km) Poznań (200km) Wrocław (220km) Szczecin (400km) Bydgoszcz (200km) Toruń (150km) Katowice
(180km) Częstochowa (120km) Lublin (210km)
Miasto Toruń jest połączone z: Bydgoszcz (50km) Warszawa (200km) Łódź (150km)
Miasto Wrocław jest połączone z: Warszawa (350km) Kraków (270km) Gdańsk (480km) Poznań (180km) Łódź (220km) Katowice (200km)
Miasto Gdańsk jest połączone z: Warszawa (350km) Kraków (550km) Poznań (300km) Wrocław (480km) Szczecin (340km) Bydgoszcz (150km) Białystok (350km) Ols
ztyn (150km)
Miasto Poznań jest połączone z: Warszawa (310km) Kraków (470km) Gdańsk (300km) Wrocław (180km) Łódź (200km) Szczecin (260km)
Miasto Olsztyn jest połączone z: Gdańsk (150km) Warszawa (180km) Białystok (150km)
Miasto Częstochowa jest połączone z: Katowice (80km) Łódź (120km) Warszawa (220km)
Miasto Kraków jest połączone z: Warszawa (300km) Gdańsk (550km) Poznań (470km) Wrocław (270km) Łódź (210km) Katowice (80km) Lublin (270km) Rzeszów (160
km)

Podaj miasto startowe: Szczecin
Najkrótsze odległości od Szczecin:
Do Kraków: 610 km
Do Częstochowa: 520 km
Do Olsztyn: 490 km
Do Poznań: 260 km
Do Szczecin: 0 km
Do Gdańsk: 340 km
Do Katowice: 580 km
Do Bydgoszcz: 300 km
Do Warszawa: 530 km
Do Toruń: 350 km
Do Wrocław: 440 km
Do Łódź: 400 km
Do Białystok: 640 km
Do Lublin: 610 km
Do Rzeszów: 770 km
```

Zastosowania

Nawigacja

Sieci komputerowe

Logistyka

Podsumowanie

1. Grafy = uniwersalny model zależności
2. Dijkstra = efektywny algorytm
3. C++ STL ułatwia implementację

Grafy są uniwersalnym modelem zależności, a algorytm Dijkstry to efektywny sposób na znajdowanie najkrótszych ścieżek. C++ STL ułatwia implementację.

https://github.com/KajetanStepienPM/31287_cpp

Dziękuję za uwagę.