Tema de casă

Vasilescu Teodor Bogdan, CR2.3

1 Enunțul problemei

Presupunem că 2 prieteni locuiesc în orașe diferite pe o hartă, de exemplu harta României din figura 2. La fiecare tură, putem muta în același timp fiecare prieten într-un oraș vecin de pe hartă. Timpul necesar mutării dintr-un oraș i într-un oraș vecin j este egal cu distanța drumului d(i, j) dintre cele 2 orașe, dar la fiecare tură prietenul care ajunge primul trebuie să aștepte până când celălalt ajunge (și îl sună pe telefonul lui/ei) înainte ca următoarea tură să înceapă. Dorim ca cei doi prieteni să se întâlnească cât mai repede posibil.

Deoarece relația dintre distanța parcursă și duata călătoriei este direct proporțională (chiar egală), pentru a ne asigura că cei 2 prieteni se vor întâlni cât mai repede, este de ajuns să determinăm traseul pe care vor merge cei doi prieteni, acesta fiind drumul cel mai scurt dintre orașele în care se află cei doi. În acest caz vom folosi algoritmul lui Dijkstra deoarece este un algoritm ușor de înțeles și ușor de implementat. Prin urmare, deși acest algoritm este făcut ca să determine distanța minimă în toate nodurile din graf de la un nod sursă, acesta poate fi modificat în așa fel încât să putem determina drumul minim dintre 2 noduri. Un mare dezavantaj al algoritmului lui Dijkstra este faptul că pierde timp calculând calea minimă la toate nodurile, însă are o complexitate de timp relativ mică (O($ElogV$) dacă graful este reprezentat prin lista de adiacență).

Având în vedere faptul că fiecare prieten trebuie să se mute cel mult un oraș pe tură, cei doi prieteni se vor întâlni la mijlocul traseului stabilit anterior (mijlocul traseului înseamnă orașul cu numărul $v/2 + 1$, dacă v este număr impar sau orașul cu numărul $v/2$ sau $v/2 + 1$ dacă v este număr par, unde v este numărul de orașe pe care le are traseul). Pentru a simplifica codul, vom considera că mijlocul traseului este întotdeauna orașul cu numărul $v/2 + 1$.

2 Pseudocodul algoritmilor

Pe lângă algoritmul ce se ocupă de calculul efectiv al drumului minim mai avem algoritmul constructorului grafului, algoritmul funcției de adăugare de muchii/drumuri directe, algoritmul de alcătuire al drumului minim(definit separat de algoritmul Dijkstra deoarece trebuie apelat recursiv) și algoritmul generatorului de date de intrare.

```
ConstructorGraf(int nrVarfuri)
1  Graf.nrVarfuri < −nrVarfuri;
2  Alocare memorie pentru lista de adiacență
```

```
AdaugareMuchie(int varf1, int varf2, int lungime) returns void
1  Adaugare in lista de adiacenta la varf1 perechea (varf2, lungime)
2  Adaugare in lista de adiacenta la varf2 perechea (varf1, lungime)
```

```
CreareTraseu(int parent[], int path[], int v)
1  if parent[v] = NULL
2        Adaugare in path a lui parent[v]
3        CreareTraseu(parent, path, parent[v])
```

GeneratorDateDeIntrare returns void

1 **for** $i = 1$ **to** 23
2 Write the value of random generated number from $[1, 200]$ interval in DateX.in
3 **for** $i = 1$ **to** 2
4 Write the value of random generated number from $[0, 19]$ interval in DateX.in

3 Structura aplicației

Ca limbaj de programare am ales să folosesc C++, deoarece consider că este un limbaj de programare făcut special pentru a rezolva aceste tipuri de probleme, mai ales dacă se folosește STL.

Din punct de vedere structural, programul nostru este alcătuit din 2 fișiere sursă (Source.cpp, Graph.cpp) și 2 fișiere header (Graph.h, problemconfiguration.h).

Pentru testarea programului implementat, se va folosi structura grafului din Figure 2: Romania Map, însă mi-am luat libertatea de a modifica valoarea lungimilor muchiilor. Astfel, s-au creat 10 fisiere cu formatul DateX.in, fiecare fisier conținând 23 de numere in intervalul $[1, 200]$ reprezentând lungimile muchiilor grafului nostru în ordinea în care au fost definite în programul implementat, și alte 2 numere ce corespund orașelor în care locuiesc cei doi prieteni.

Pentru fișierul Date1.in s-au folosit valori ale lungimilor muchiilor oferite de Figure 2: Romania map, iar ca orașe de plecare pentru cei 2 prieteni s-au ales Craiova și Arad. Pentru celelalte fișiere se va folosi un mic program de generare a numerelor aleatoare, a cărui structură este definită în secțiunea anterioară.

Datele de ieșire sunt stocate în fișiere cu numele DateX.out, fiecare fișier conținând calea cea mai scurtă calculată prin dijkstra (pe prima linie) și orașul în care se vor întâlni cei doi prieteni (pe a doua linie), prezentate într-un format custom. Fișierul DateX.out reprezintă rezultatele programului nostru cu datele de intrare extrase din fișierul DateX.in.

Modulul problemconfiguration.h se folosește pentru configurarea programului la problema noastră. Acest modul conține:

- Includerea bibliotecilor C++ necesare programului nostru.
- Definirea de valorilor întregi din intervalul [0, 19] ca nume de orașe din Figure 2: Romania Map, conform numerotării de mai jos:
  - $Arad -> 0$
  - $Zerind -> 1$
  - $Sibiu -> 2$
  - $Timisoara -> 3$
  - $Oradea -> 4$
  - $Fagaras -> 5$
  - $RamnicuValcea -> 6$
  - $Lugoj -> 7$
  - $Bucharest -> 8$
  - $Pitesti -> 9$

- $Craiova -> 10$
- $Mehadia -> 11$
- $Drobeta -> 12$
- $Giurgiu -> 13$
- $Urziceni -> 14$
- $Vaslui -> 15$
- $Hirsova -> 16$
- $Iasi -> 17$
- $Neamt -> 18$
- $Eforie -> 19$

- Funcția PrintCity este o funcție utilă ce se ocupă cu afișarea orașului conform cu numărul atribuit în numerotarea de mai sus. Această funcție primește numărul specific orașului X și returnează ca string nume acestuia.

Modulul Graph.h are rolul de a defini harta noastră și funcțiile necesare, inclusiv funcția de calculare al drumului minim. Acest modul conține definirea clasei graph specifică hărții noastre. Această clasă are ca atribute numărul de vârfruri ale grafului (numărul de orașe ale hărții) și lista de adiacență a grafului. De asemenea, clasa graph conține funcția de adăugare de muchii, funcția de calculare a drumului minim și funcția de creare a drumului minim.

Modulul Graph.cpp conține implementarea funcțiilor definite în graph.h .

- Funcția addEdge implementează algoritmul AdaugareMuchie de la secțiunea Pseudocodul algoritmilor. Variabilele întregi v și u reprezintă nodurile componente ale muchiei, iar variabila întreagă w conține lungime drumului.

- Funcția shortestPath este o funcție ce implementează algoritmul lui Dijkstra și returnează calea minimă căutată, având ca parametrii întregi nodul sursă (s) și nodul destinație (d).

- Funcția createPath este o funcție utilă ce se ocupă de construirea căii minime folosind vectorul parent, definit în shortestPath. Pe lângă calea pe care o modifică (path) și vectorul parent, acesta mai are ca parametru și nodul de destinație al drumului minim, nod din care se începe construcția. Deși această funcție returnează void, parametrul path este transmis prin referință, ceea ce înseamnă că la apelarea acestei funcții, parametrul path se va modifica atât în createPath cât și în funcția ce îl apelează.

Modulul Source.cpp conține locul în care se citesc datele din fișier, se calculează drumul minim, se calculează orașul de întâlnire și se scrie în fișier rezultatele obținute.

4 Concluzii

În urma acestei lucrări, am reușit să implementez Dijkstra pe un graf reprezentat prin listă de adiacență și să o modific astfel încât să rezolve cerințele problemei mele (determinarea drumului minim între 2 orașe).

În timpul efectuării temei de casă, pot să spun că am avut câteva probleme în ceea ce privește folosirea noțiunilor de STL în rezolvarea problemei, lovindu-mă de foarte multe ori de erori de tipul. De asemenea, după cum se poate observa din datele de ieșire, există posibilitatea ca una din orașele extremitate ale căii (practic una din cele 2 orașe din care calculăm calea) să nu apară în componența căii.

Pe viitor, o să încerc să extind acest program să calculeze timpul necesar efectuării călătoriei.

5 Referinte bibliografice

References

[1] *TOTUL DESPRE C și C++ Manualul fundamental de programare în C și C++* Dr. Kris Jamsa Lars Klander

[2] `www.overleaf.com`, accesat în Aprilie 2020

[3] `https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/`, accesat in Mai 2021

[4] `https://www.youtube.com/watch?v=pVfj6mxhdMw`, accesat in Iunie 2021

[5] `https://www.geeksforgeeks.org/the-c-standard-template-library-stl/`, accesat in Iunie 2021