Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Mircoelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

despre lucrarea de laborator nr.4

**Disciplina:** Analiza și Proiectarea Algoritmilor

**Tema: „Metoda programării dinamice”**

A îndeplinit st. Gr. TI-206, Rutcovschi Samuil-Pavel

A verificat asis. univ., Bîtca Ernest

Chișinău – 2021

**Scopul lucrării:**

1. Studierea metodei programării dinamice.
2. Analiza şi implementarea algoritmilor de programare dinamică.
3. Compararea tehnicii greedy cu metoda de programare dinamică.

**Sarcina de bază:**

1. De studiat metoda programării dinamice de proiectare a algoritmilor.
2. De implementat într-un limbaj de programare algoritmii dijkstra și floyd.
3. De făcut analiza empirică a acestor algoritmi pentru un graf rar şi pentru un graf dens.
4. De alcătuit un raport.



**Ce este programarea dinamică?**

* Programarea dinamică a fost dezvoltată de către **Richard Bellman** în 1950.
* În **programarea dinamică** cuvântul programare se referă la planificare și nu la porgramare în sens informatic
* Cuvântul **dinamic** se referă la maniera în care sunt contruite tabelele în care se rețin informațiile referitoare la soluțiile parțiale.

**Richard E. Bellman**1920-1984

**Rezumatul succint la temă:**

Pentru studierea programării dinamice au fost propuși 2 algoritmi: Dijkstra și Floy.

Algoritmul Dijkstra calculează drumul minim într-un graf de la un anumit vârf spre toate celelalte vârfuri și are complexitatea O(V^2). Dar algoritmul Floy face tot același lucru doar că calculează drumurile din toate vârfurile spre toate celelalte varfuri, acest algoritm are complexitatea O(V^3).

Dar pentru a putea compara acești algoritmi a fost porpusă ca sarcină de a modifica algoritmul Dijkstra ca să afle și el costul minim din toate vârfurile spre toate celelalte. În așa mod s-a majorat complexitatea algoritmului Dijkstra la O(V^3).

În final am primit doi algoritmi cu aceeși complexitatea. A rămas doar de aflat care din ei este mai eficient în diferite situații create.

**Întrebări de control:**

1. ***Descrieţi metoda programare dinamică.***

Programarea dinamică rezolvă problemele prin descompunerea lor în subprobleme și rezolvarea acestora. Se aplică în general problemelor de optimizare, atunci când dorim să determinăm rapid soluția optimă pentru o problemă, dar putem afla în același timp și mai multe soluții optime dacă acesta le conține.

Pentru aplicarea acestei metode este necesar să ținem cont de următorii pași:  
 1) Analiza structurii soluției

2) Indentificarea relației de recurență

3) Dezvoltarea relației de recurență

4) Construirea propriu-zisă a soluției

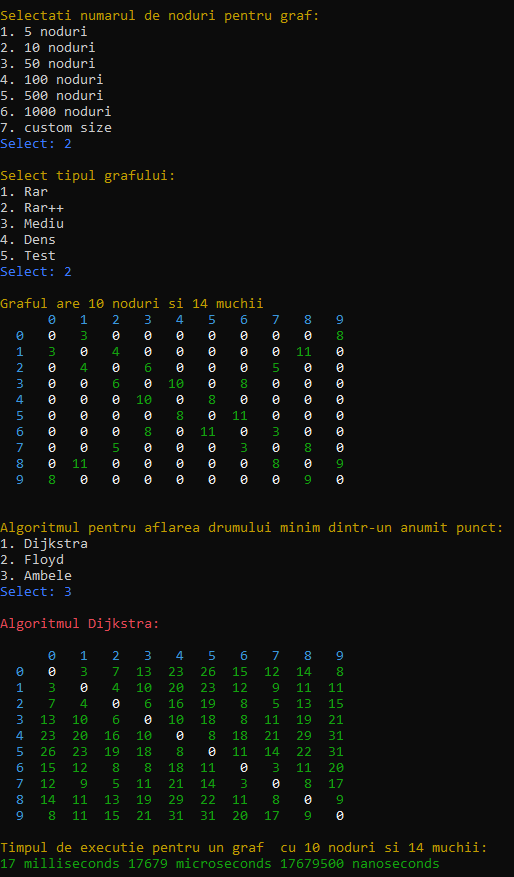
1. ***De ce această metodă se numeşte programare dinamică?***

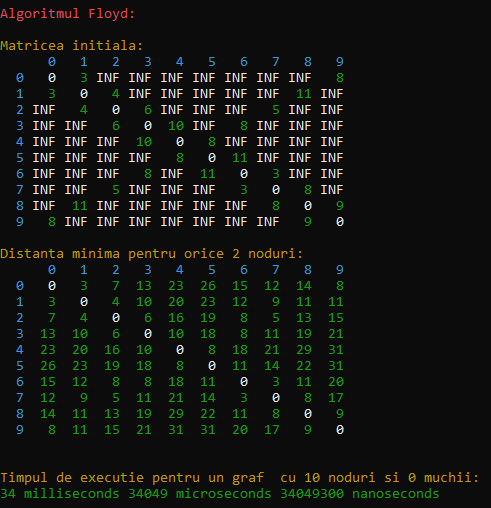
Programarea dinamincă se numeste așa din cauza la maniera în care sunt contruite tabelele în care se rețin datele referitoare la soluțiile parțiale.

1. ***Care este diferenţa între*** ***metoda divide et impera şi metoda programării dinamice?***În cazul programării dinamice problemele nu mai sunt independente, ele se suprapun.
2. ***Care este clasificarea problemelor de programare dinamică?***Găsirea unei valori maxime sau minime, sau numărarea elementelor unei mulțimi.
3. ***Ce se întâmplă în cazul când costurile arcelor grafurilor prelucrate cu algoritmii Dijkstra şi Floyd sunt negative?***

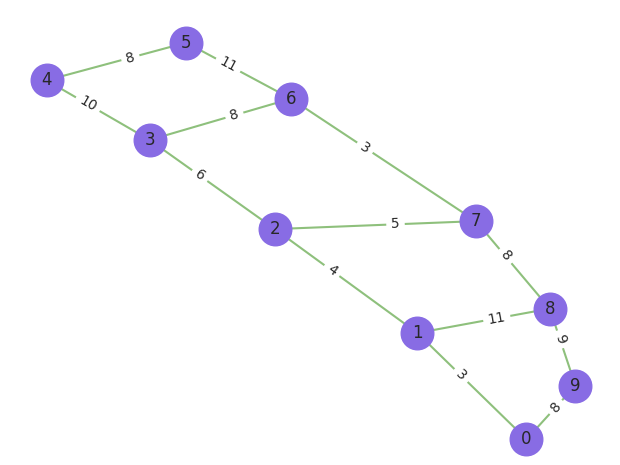
Nu vom primit rezultatul corect. În cazul meu am folosit graf neorientat din acestă cauză se poate trece prin nodul cu cost negativ până la inifinit iar drumul minim nu va fi determinat.

**Rezultatul execuției programului:**





*Fig1. Exemplu de rularea programului cu afișarea activată*



*Fig2. Graful construit automat în urma execuției programului*

***Compararea algorimilor:***

Pentru a putea compara acești algoritmi am ales să calculez timpul de execuție pentru un graf creat în mai multe moduri:

1. Număr diferit de noduri (50,100,250,500,1000).
2. Seturi diferite de muchii:
   1. graf rar - numărul de muchii este minim posibil.
   2. graf mediu – un număr mediu de muchii.
   3. graf dens - numărul de muchii este maxim posibil.

Iar pentru a obține un rezultat cât mai exact am hotâr să execut algoritmii pentru toate aceste moduri de 1000 de ori, iar rezultatele le-am exportat într-un file excel unde am calculat timpul mediu de execuție. Iar în final mai jos am construit tabelele și graficele cu datele primite.

În imaginea de mai jos am arătat cum arată file-ul exel în care am salvat timpul de execuție în microsecunde a algoritmilor de o mie de ori, iar în linia 3 se calculează timpul mediu de execuție pentru fiecare algoritm.

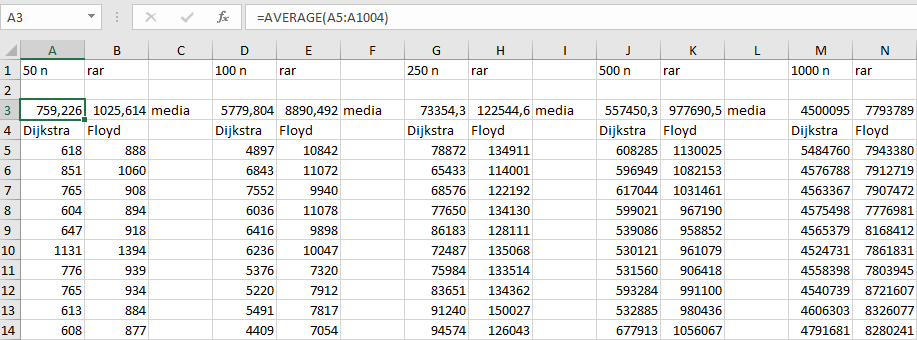


Fig3. File-ul excel creat în urma execuției programului

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tabel 1. Timpul mediu de execuție în microsecunde a alg. pentru un graf rar | | | | | |
|  | Timpul de execuție în milisecunde | | | | |
| Noduri  Algoritmul | 50 | 100 | 250 | 500 | 1000 |
| Dijkstra | 0.6 | 4.5 | 60 | 460 | 3689 |
| Floyd | 0.7 | 5.8 | 81 | 648 | 5238 |

În tabelul de mai sus am introdus timpul de execuție pentru un graf rar. Deci într-un graf rar pentru orice număr de noduri am primit că nu foarte semnificativ dar algoritmul Dijkstra are un timp de execuție mai rapid de cât Floyd.

*Graficul 1.* Timpul de execuție a algoritmelor pentru un graf rar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tabel 2. Timpul de execuție în microsecunde a alg. pentru un graf mediu | | | | | |
|  | Timpul de execuție în milisecunde | | | | |
| Noduri  Algoritmul | 50 | 100 | 250 | 500 | 1000 |
| Dijkstra | 1.4 | 10.1 | 142 | 1075 | 8697 |
| Floyd | 0.9 | 6.5 | 92 | 700 | 5652 |

Pentru un graf mediu pentru orice număr de noduri am primit că algorimtul Dijkstra are un timp de execuție mai lent de cât Floyd. Acest rezultat este așa mai mult din cauza proiectării mele a grafului.

*Graficul 2.* Timpul de execuție a algoritmelor pentru un graf mediu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tabel 3. Timpul de execuție în microsecunde a alg. pentru un graf dens | | | | | |
|  | Timpul de execuție în milisecunde | | | | |
| Noduri  Algoritmul | 50 | 100 | 250 | 500 | 1000 |
| Dijkstra | 1.1 | 7.0 | 103 | 751 | 6041 |
| Floyd | 0.8 | 5.5 | 84 | 653 | 5821 |

Iar pentru un graf dens algoritmii au timpul de execuție aproape la fel, cu cat mai multe noduri sunt în graf cu atât acestă diferență de timp este mai mică intre algoritmi.

*Graficul 3.* Timpul de execuție a algoritmelor pentru un graf mediu

**Concluzie:**

În cadrul acestui laborator am studiat tehnica de programare dinamică dezvoltată de către Richard Bellman. Această tehnică de programare rezolvă problemele prin descompunerea lor în subprobleme și rezolvarea acestora.

Am observat că programarea dinamică se aseamănă cu tehnica divide et impera deoarece la fel se bazează pe divizarea problemei inițiale în subprobleme. Dar în cazul porgramării dinamice subprobelemele se suprapun, în sens ca soluția unei subprobleme se utilizează la aflarea soluțiilor altor subprobleme. În comparație cu tehnica greedy principala diferență este că programarea dinamică reexaminează soluțiile ce asigură o soluție optimă la problemă.

Pentru studierea acestei tehnici au fost propuși doi algortimi: dijkstra și floyd. Ambii au comlexitatea O(V^3). Deci teoretic mă așteptam la un timp de execuție asemănător. În practică am obținut următoarele concluzii:

* Când avem un graf rar cu un număr minim de muchii putem folosi orice algoritm deoarece au aproape același timp de execuție.
* Când avem un graf mediu cu un număr mediu de muchii eset mai relevant de utilicat algoritmul Floyd deoarece are un timp de execuție mai rapid.
* Dar când avem un graf dens cu un numar maxim de muchii ca și în cazul unui graf rar nu are importanță ce algoritm vom folosi deoarece ambii au același timp de execuție.

**Bibliografie:**

Floyd Warshall Algorithm | DP-16:

<https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/>

Dijkstra’s shortest path algorithm | Greedy Algo-7:

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>