**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

Algoritmų sudarymas ir analizė (P170B400)

2 laboratorinio darbo ataskaita

(11.15 užduotis)

Atliko:

IF-4/12 gr. studentas

Mangirdas Kazlauskas

2016 m. gegužės 09 d.

Priėmė:

Doc. Vytautas Pilkauskas

KAUNAS

2016

1. Laboratorinio darbo užduotis:

Naudojant programavimo algoritmų sudarymo metodą, atlikti laboratorinio darbo užduotį.

Laboratorinio darbo užduotis susideda iš dviejų dalių:

1. Naudojant dinaminio programavimo rekursiją (skaidant uždavinį į mažesnius uždavinius), rasti trumpiausius kelius nuo visų grafo viršūnių iki nurodytos grafo viršūnės k. Grafe negali būti neigiamų ciklų.
2. Pritaikyti a) varianto sprendimą 11.14 užduotyje nurodytam grafui, kuriame reikia rasti trumpiausius kelius nuo visų grafo viršūnių iki 6-tosios grafo viršūnės.
3. Laboratorinio darbo realizacija:

Laboratoriniam darbui atlikti naudojama programavimo kalba – JAVA. Programavimo aplinka – IntelliJ IDEA 15.0.3.

1. Užduoties analizė:

Kaip nurodyta užduotyje, užduoties sprendimui reikia naudoti dinaminio programavimo principą – uždavinį skaidyti į mažesnius uždavinius. Kadangi šios užduoties esmė – trumpiausiojo kelio paieška grafe, todėl visa užduotis, taikant dinaminio programavimo principą, sprendžiama palaipsniui: pirmiausia skaičiuojami trumpiausių kelių trumpiausi atstumai, susidedadantys daugiausiai iš vienos briaunos. Toliau ieškomi visi trumpiausi atstumai, susidedantys bent iš dviejų briaunų ir t.t.

Pažymėkime viršūnių skaičių V. Taip pat darome prielaidą, kad grafe nėra neigiamo ilgio ciklų. Viename kelyje nuo konkrečios viršūnės iki kitos grafos viršūnės daugiausiai gali būti V-1 briaunų (kitu atveju jau eitume perteklinėmis briaunomis, kas reikštų, jog toks kelias tikrai nebus trumpiausias). Todėl per visas briaunas perėjus V-1 kartą būsime užtikrinti, kad bus surasti visi trumpiausi atstumai nuo visų grafo viršūnių iki duotosios. Užduoties sprendimui naudosime kiek pakeistą Bellman-Ford algoritmą (originalus algoritmas ieško trumpiausių atstumų NUO duotosios viršūnės iki visų kitų likusių grafo viršūnių, tačiau mums reikia ieškoti atvirkščiai).

Pažymėkime Pi,v – optimalaus kelio ilgis iki nurodytos viršūnės v, susidedantis ne daugiau nei iš i briaunų. V – grafo viršūnių aibė, w – bet kuri grafo viršūnė, esanti priešpaskutinė kelyje nuo bet kurio grafo viršūnės iki nurodytosios viršūnės v, cij – grafo briaunos, esančios tarp viršūnių i ir j, svoris. Kiekvienam , dinaminio programavimo rekursinė užduoties sprendimo formulė:

1. Užduoties sprendimo algoritmo realizavimas:
2. Grafų generavimas:

Kadangi, remiantis a) užduoties dalies aprašymu, grafe negali būti neigiamų ciklų, todėl, kad generacija būtų paprastesnė, generuojamo grafo briaunų svoriai yra teigiami. Grafo generavimo funkcijai reikia nurodyti viršūnių skaičių n. Funkcija sugeneruoja n viršūnių grafą su 2n-1 briauna. Generavimo funkcijos realizacija:

**public static** Grafas generuotiGrafa(**int** n){  
 **int** b = 2\*n-1;  
 Grafas grafas = **new** Grafas(n, b);  
 **for** (**int** i = 1; i <= n-1; i++){  
 **int** pr, pb, svoris;  
 Random rnd = **new** Random();  
 pr = i+1;  
 pb = i;  
 svoris = rnd.nextInt(n+1);  
 grafas.**briaunos**[i-1].**pr** = pr;  
 grafas.**briaunos**[i-1].**pb** = pb;  
 grafas.**briaunos**[i-1].**svoris** = svoris;  
 }  
 **for** (**int** i = 1; i <= n; i++){  
 **int** pr, pb, svoris;  
 Random rnd = **new** Random();  
 pr = rnd.nextInt(n+1);  
 pb = i;  
 **while** (pr == i){  
 pr = rnd.nextInt(n+1);  
 }  
 svoris = rnd.nextInt(n+1);  
 **if** (pr == 0)  
 pr += 1;  
 grafas.**briaunos**[n-2+i].**pr** = pr;  
 grafas.**briaunos**[n-2+i].**pb** = pb;  
 grafas.**briaunos**[n-2+i].**svoris** = svoris;  
 }  
 **return** grafas;  
}

1. Trumpiausių kelių paieškos grafe funkcijos realizacija bei jos sudėtingumo įvertinimas:

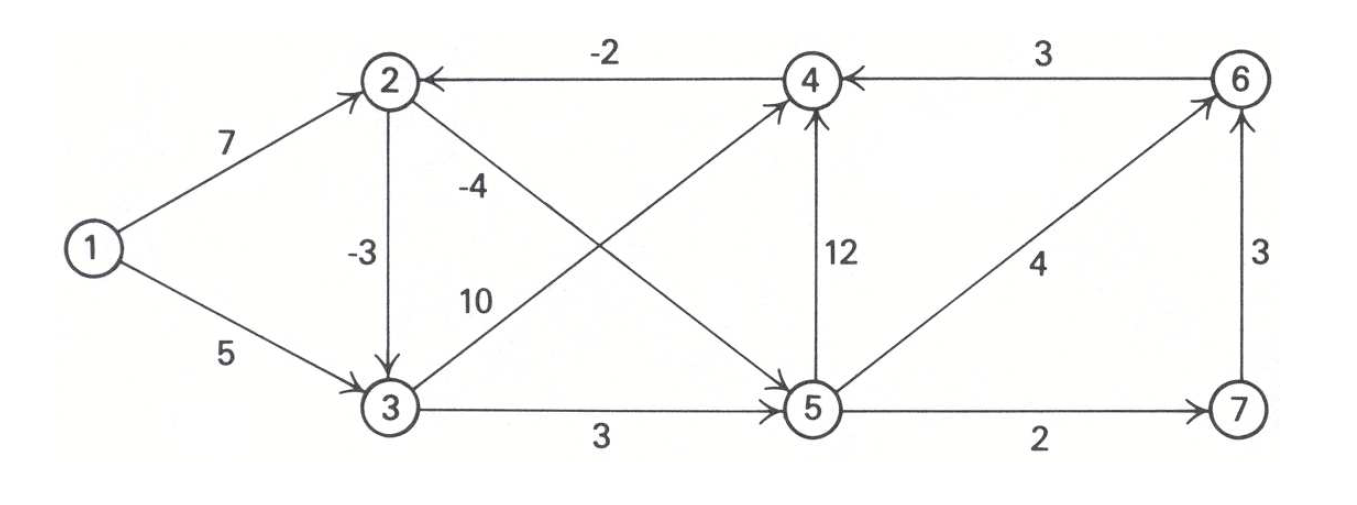
|  |  |  |  |
| --- | --- | --- | --- |
| Eil. nr. | Programos kodas | Laikas | Kartai |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | **public static int**[][] trumpiausiKeliai(Grafas grafas, **int** k){  **int** v = grafas.**v**;  **int** b = grafas.**b**;  **int** atstumai[] = **new int**[v];  **int** prec[] = **new int**[v];  **int** ats[][] = **new int**[2][v];   **boolean** pakeitimas;  **boolean** turiNeigiamaCikla = **false**;   **for** (**int** i = 0; i < v; i++){  atstumai[i] = Integer.***MAX\_VALUE***;  prec[i] = 0;  }  atstumai[k-1] = 0;  prec[k-1] = k-1;  *// Briaunu relaksacija* **for** (**int** i = 1; i < v; i++) {  pakeitimas = **false**;  **for** (**int** j = 0; j < b; j++) {  **int** pr = grafas.**briaunos**[j].**pr**-1;  **int** pb = grafas.**briaunos**[j].**pb**-1;  **int** svoris = grafas.**briaunos**[j].**svoris**;  **if** (atstumai[pb] != Integer.***MAX\_VALUE*** &&  atstumai[pb] + svoris < atstumai[pr]) {  atstumai[pr] = atstumai[pb] + svoris;  prec[pr] = pb+1;  pakeitimas = **true**;  }  }  **if** (!pakeitimas)  **break**;  }   *// Patikrinimas del neigiamu ciklu* **for** (**int** i = 0; i < b; i++){  **int** pr = grafas.**briaunos**[i].**pr**-1;  **int** pb = grafas.**briaunos**[i].**pb**-1;  **int** svoris = grafas.**briaunos**[i].**svoris**;  **if** (atstumai[pb] != Integer.***MAX\_VALUE*** && atstumai[pb] + svoris < atstumai[pr]) {  turiNeigiamaCikla = **true**;  **break**;  }  }  **if** (turiNeigiamaCikla) {  ats = **null**;  **return** ats;  }  **else** {  ats[0] = atstumai;  ats[1] = prec;  **return** ats;  } } | C1  C1  C1  C1  C1  C1  C1  C2  C1  C1  C1  C1  C2  C1  C2  C1  C1  C1  C3  C1  C1  C1  C3  C4  C2  C1  C1  C1  C3  C1  C4  C3  C1  C4  C1  C1  C4 | 1  1  1  1  1  1  1  v+1  v  v  1  1  v  v-1  (v-1)(b+1)  (v-1)b  (v-1)b  (v-1)b  (v-1)b  (v-1)b  (v-1)b  (v-1)b  v-1  1  b+1  b  b  b  b  1  1  1  1  1  1  1  1 |

Algoritmo funkcijos sudėtingumas:

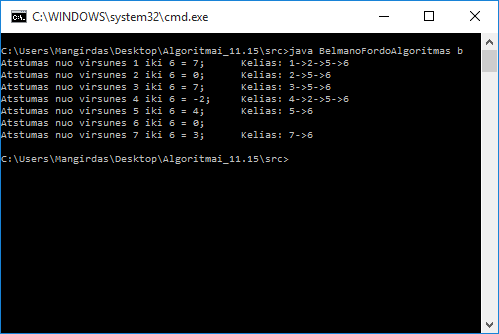
Funkciją *T(n)* galime išreikšti pavidalu , todėl šio trumpiausių kelių paieškos algoritmo sudėtingumas – *T(n)=O(vb)*, kur v – grafo viršūnių skaičius, b – grafo briaunų skaičius.

1. Rezultatai:

b) užduoties dalyje buvo duotas grafas:



Programa, pagal užduoties sąlygą, ieškojo trumpiausių atstumų nuo visų duotojo grafo viršūnių iki viršūnės nr. 6. Programos vykdymo rezultatai:



Palyginus duotojo grafo duomenis su gautais rezultatais, matome, kad rasti teisingi trumpiausi keliai nuo visų duotojo grafo viršūnių iki 6-tosios viršūnės.

Taip pat buvo tiriama ir algoritmo greitaveika. Testas buvo atliekamas su atsitiktinai sugeneruotais grafais, turinčiais 100000 viršūnių (199999 briaunos), 200000 viršūnių (399999 briaunos), 300000 viršūnių (599999 briaunos), 400000 viršūnių (799999 briaunos), 600000 viršūnių (1199999 briaunos), 800000 viršūnių (1599999 briaunos), 1200000 viršūnių (2399999 briaunos), 1600000 viršūnių (3199999 briaunos), 2400000 viršūnių (4799999 briaunos) bei 3200000 viršūnių (6399999 briaunos). Testo rezultatai pavaizduoti grafike:

1. Išvados:

Iš grafiko matome, kad teorinis algoritmo sudėtingumo įvertinimas pasitvirtino – apskaičiuotas teorinis algoritmo sudėtingumas atitinka realaus testavimo rezultatų grafiko kreivę. Todėl šio paieškos algoritmo sudėtingumas – *O(vb)*, kur v – grafo viršūnių skaičius, o b – grafo briaunų skaičius.