Ģirts Rudzišs

gr16014

Ātru algoritmu konstruēšana un analīze

Programmēšanas darba apraksts

Vinnija pūka grafu uzdevumam risinājums izstrādāts programmēšanas valodā C++.

Izstrādātā algoritma darbība

1. No ieejas datiem tiek ielasīts virsotņu skaits n un izveidota n*n saistību matrica *graph*, tā tiek aizpildīta ar neeksistējošas šķautnes vērtību NO_VERTEX (101);
2. No ieejas datiem tiek ielasīta katra šķautne (a_i, b_i, w_i), svaru w_i ierakstot matricas a_i - 1 rindas b_i - 1 kolonnā un b_i - 1 rindas a_i - 1 kolonnā (C++ masīvu notācija sākas no 0, tāpēc -1);
3. Tiek izveidots masīvs *result_edges*, kurā glabāju Vinnijam optimālās šķautnes gala rezultātam
4. Tiek apstaigāta matrica no i = [0; n) un j = [i; n), apskatot *graph[i][j]* svaru:
    a. Ja *graph[i][j] < 0:* šķautnes svars ir negatīvs. Šķautne tiek saglabāta *result_edges* masīvā un tā tiek dzēsta no grafa (gan [i][j], gan[j][i] matricas elementi tiek pārveidoti uz NO_VERTEX;
    b. Citādi, ja *graph[i][j] != NO_VERTEX*: sķautnes svars ir nenegatīvs (bet tā eksistē). Šķautnes svaram pārveidojam uz negatīvu, to pareizinot ar -1 (gan [i][j], gan[j][i] matricas elementi tiek pareizināti ar -1);
5. Tiek palaists Prima algoritms, lai atrastu minimālo savienoto koku (*minimum spanning tree*) šajā grafā ar negatīviem svariem:
    a. Izveidoju n elementu masīvus *min_neighbor* (apzīmē virsotnes kaimiņu, uz kuru pievienotās šķautnes svars ir vismazākais), *min_edge* (apzīmē mazāko šķautnes svaru ar virsotnes kaimiņiem) un *visited* (vai virsotne jau ir iekļauta minimum spanning tree)
    b. Inicializē *min_edge[0]* = 0 un *min_neighbor[0] = -1*, lai kā pirmo apskatītu 1. virsotni
    c. No count = [1; n]:

i. Inicializē min uz integer datu tipa maksimālo vērtību INT_MAX, inicializē u;
ii. No v = [0; n):
    1. Ja v nav apskatīta un *min_edge[v] < min: min = min_edge[v] un u = v;*
iii. Atzīmē u virsotni kā apskatītu
iv. No v = [0; n):
    1. Ja eksistē šķautne (u, v), virsotne v vēl nav apskatīta un šķautnes (u,v) svars < *min_edge[v]*: min_neighbor[v] = u un min_edge[v] = šķautnes (u,v) svars;

d. No i = [1; n):
    i. Ja min_edge[i] != INT_MAX, tad šķautne (*min_neighbor[i] + 1, i +1*) ar svaru *graph*[*min_neighbor[i]][i]* tiek pievienota minimum spanning tree šķautņu masīvam;

6. No grafa tiek dzēstas visas šķautnes, ko atradis Prima algoritms, secīgi dzēšot katru Prima algoritma atrasto šķautni;
7. Tiek apstaigāta matrica no i = [0; n) un j = [i; n):
    a. Ja *graph != NO_VERTEX*: šķautne eksistē un tā tiek saglabāta *result_edges* masīvā
8. Inicializē *sum = 0;*
9. Apstaigā *result_edges* masīvu, katrā solī pieskaitot pie *sum* šķautnes svaru *w;*
10. Izvaddatu failā *output.txt* tiek ierakstīta summa *sum,* kā arī vēlreiz tiek apstaigāts *result_edges* masīvs un failā tiek ierakstītas šo šķautņu a un b virsotnes.

Algoritma korektuma skaidrojums

Negatīvo šķautņu iekļaušana rezultātā ir acīmredzama, jo šīs šķautnes “palīdz” Vinnijam izvietot medus krājumus, samazinot kopējo grūtības pakāpes summu. Negatīvās šķautnes arī tiek atmestas, jo, pat ja tās ir kāda cikla sastāvdaļa, to izņemšana un pievienošana rezultātam apmierina uzdevuma 1. nosacījumu.

Lai pilnībā izpildītu 1. nosacījuma prasību, jāatrod šķautņu kopa F, ko atmetot no sākotnējā grafa G, iegūst grafu, kas ir aciklisks (proti, H = (V, E-F) būs aciklisks). Tātad, tas ir koks. Turklāt, F šķautnēm jābūt ar minimālo iespējamo svaru (lai izpildītos 2. nosacījums), kas nozīmē, ka H ir jābūt ar maksimālo iespējamo svaru. Uzdevums reducējas uz maximum

spanning tree atrašanu, ko arī veicu ar Prima algoritmu, visas nenegatīvās šķautnes pārveidojot par negatīvām un atrodot minimum spanning tree uz negatīvā grafa. Iegūtās šķautnes ir grafa H = (V, E-F) šķautnes, kuras atmetot no G iegūstam vajadzīgo F šķautņu kopu ar minimālo svaru.

Algoritma sarežģītības novērtējums

n - virsotņu skaits, m - sķautņu skaits

1. solis = O(1)
2. solis = O(m)
3. solis = O(1)
4. solis = $O(n^2)$ (tehniski apstaigājam tikai pusi matricas, bet tik un tā tas < $O(n^2)$)
5. solis (Prima algoritms) = $O(n^2)$ (tā kā izmantoju vienkāršus nesakārtotus masīvus virsotņu un mazāko šķautņu glabāšanai, tad šī Prima algoritma implementācija novērtējama ar $O(n^2)$)
6. solis = O(m) (sliktākajā gadījumā minimum spanning tree var saturēt visas šķautnes)
7. solis = $O(n^2)$
8. solis = O(1)
9. solis = O(m) (sliktākajā gadījumā visas šķautnes ir *result_edges* masīvā)
10. solis = O(m)

Kopējā risinājuma sarežģītība: $O(n^2 + m)$