Vilniaus universitetas

Algoritmų teorija
Duomenų mokslas 1 grupė

Užduotis 6
Maksimalaus poravimo grafe uždavinys

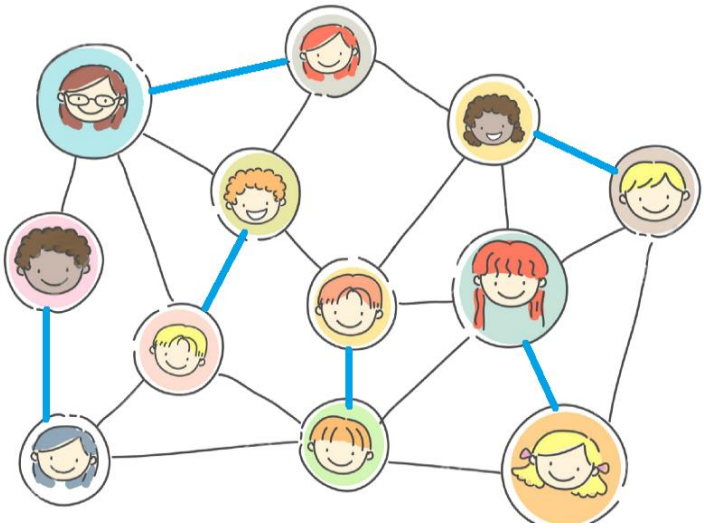
Darbą atliko:
Jurgis Adomaitis

Vilnius, 2023

Užduoties formuluotė:

Duotame neorentuotame grafe G, kuris turi n viršūnių ir m briaunų rasti maksimalų grafo poravimą – maksimalų briaunų E poaibį E`, tokį, kad visos briaunos neincidentinės.

Maksimalų poravimą galima pritaikyti daugelyje sričių, nes toks poravimas neatsiejamas nuo resursų paskirstymo – kiekvienam kažkokiam objektui priskirti tam tikrą kiekį resursų. Toks poravimas taip pat matomas būtent pačio žodžio prasmėje – žmonių poravime. Tarkime vaikų stovykloje norime sudaryti poras užsiemimui ir norime, kad kuo daugiau vaikų būtų poroje su žmogum, kuriuo jis pageidautų būti kartu. Vaikus traktuokime kaip viršūnes, o jų sudarytus pageidaujamų draugų sąrašus kaip viršūnes (žr. 1 pav.). Taip, galime atrinkti visus variantus, bet tai gali ilgai užtrukti, tad pasitelkus toliau pateikiamą Edmonds “Blossom” algoritmą, mes šią užduoti atliksime greičiau.

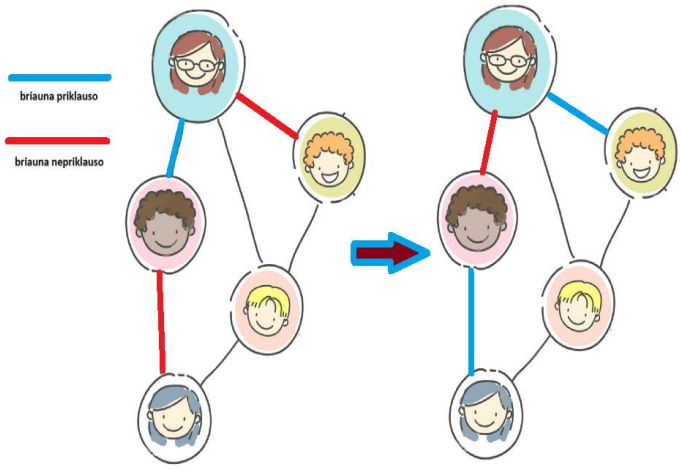


1 pav.Maksimalus poravimas grafe

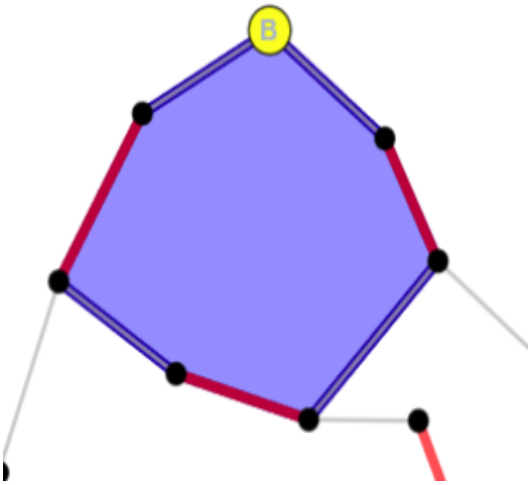
Algoritmo aprašymas

Labai plačia prasme, algoritmas kiekvieną kartą ieško suporavimo, kuris būtų viena pora didesnis nei turimas suporavimas, to neradus, paieška stabdoma ir teigiama, kad poravimas yra maksimalus. Ieškant maksimalaus poravimo negalime surasti dviejų besijungiančių dar nesuporuotų viršūnių ir pridėti jas į mūsų porų aibę, nes aklai besirenkant briaunas mes galime gauti ne optimalų sprendinį. Šis algoritmas remiasi M-augmenting kelių ieškojimu, kur kelią vadiname M-augmenting, jei jo briaunos kaitaliojasi tarp nesamų/esamų turime poravime ir

prasideda bei baigiasi su nesuporuotomis viršūnėmis (iš to išplaukia, jog briaunų skaičius yra nelyginis), taip pat šis kelias turi būti paprastas – kiekviena viršūnė aplankoma tik 1 kartą. Tokį kelią radus, mes galime porų skaičiu padidinti vienetu, nes apkeičiame poravime esamas briaunas su nesamomis (žr. 2 pav.).



2 pav. M-augmenting kelio pavyzdys



3 pav. Pradedama paieška iš nesuporuotos B ir grįžatama į ją

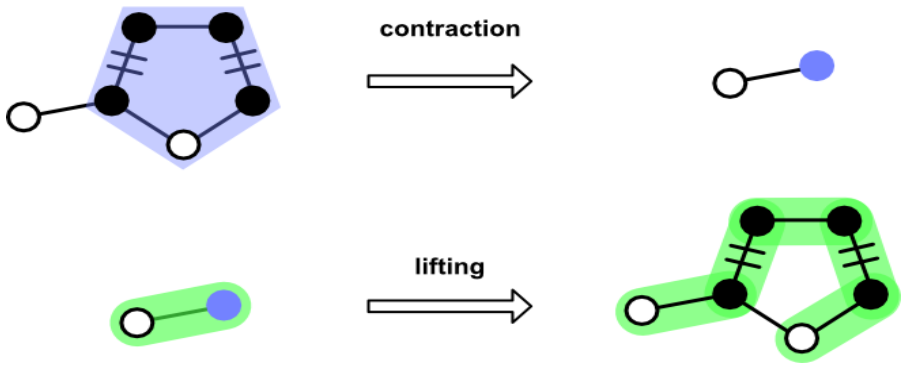
Jei suradę nesuporuotą viršūnę atliktume paieška platyn, užtikrindami, kad iš poravime nesančioje briaunos eitume į esančią, bet neužtikrintume, kad kelias bus paprastas, tai viršūnę galime aplankyti kelis kartus – pvz. joje būnant, kai einame iš suporuotos briaunos į nesuporuotą ir po to iš suporuotos į nesuporuotą arba atvirkščiai (žr 3 pav.). Tokiu atveju apkeičiant briaunas mes gautume dvi šalia esančias suporuotas briaunas. Edmonds šią problemą išsprendė pristatydamas „Žiedus“ (angl. „Blossoms“).

Žiedu mes vadinsime tokį kelią , jei jo briaunos kaitaliojasi tarp nesamų/esamų turime poravime, v0, . . . , vt−1 visos skirtingos, v0 nesuporuota su cikle esančiomis viršūnėmis, ir vt = v0. Iš to išplaukia, kad tai yra ciklas, kuris turi 2k+1 briauną, kur k - poravime esančių briaunų skaičius. Radus tokį ciklą mes jį sutraukiame (žr. 4 pav.) ir tada rekursyviai ieškome M-augmenting kelių [2].

Taigi „Blossom“ algoritmas yra toks:

Pradedame paieška platyn iš nesuporuotss viršūnės, jei randame, kad jo briaunas kaitaliojasi tarp nesamų/esamų turime poravime, žiūrime, ar šis kelias yra M-augmenting. Jei ne, radome žiedą ir jį sutraukiame bei tęsiame paiešką. Jei kelias yra M-augmenting, mes keičiame jo

suporuotas briaunas nesuporuotomis atkeliant (žr 4 pav.) reikiamus žiedus. Neradus kelio, kuris briaunas kaitaliojasi tarp nesamų/esamų turime poravime, mes teigiame, kad poravimas yra maksimalus.



4 pav. žiedo sutraukimas ir atkelimas

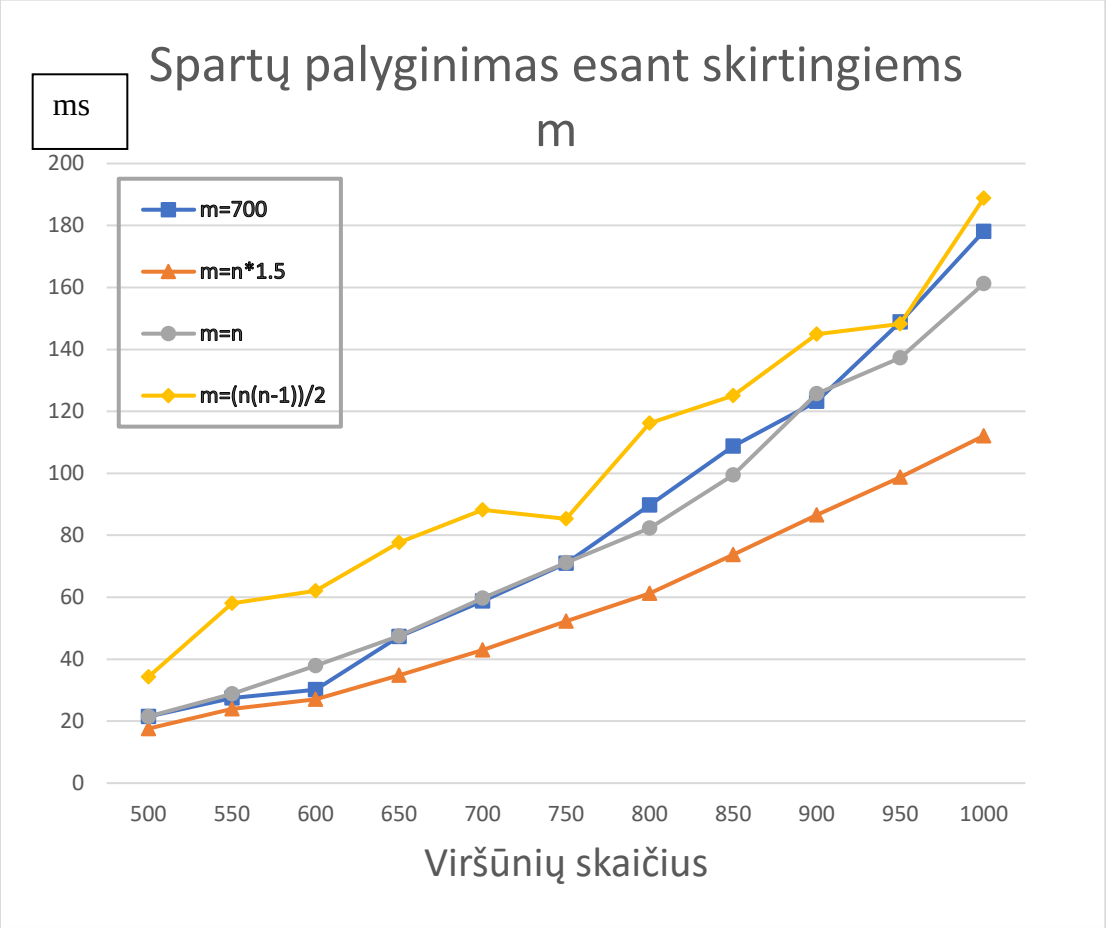
Algoritmo sudėtingumo tyrimas.

Teoriškai iš viso galimų iteracijų yra n/2, nes kiekvienojke iteracijoje galima pridėti vieną porą, o maksimalus porų skaičius yra n/2, taigi sudėtingumas O(n). Paprasta paieška platyn yra O(n^2), nes kiekviena viršūnė gali pasirodyti paieškoje tik po kartą, o iš jos atliekama paieška į visas likusias viršūnes. Augemnting kelią galime rasti tik kartą kiekvienoje iteracijoje, radimo laikas užtrunka O(n). Žiedų sutraukimas užtranka $O(n|b_c|)$, kur $|b_c|$ yra žiedo c ilgis, kadangi kiekvienai viršūnei/žiedui yra priskiriama maksimaliai vienas žiedas, tai b_c maksimali suma = n, reiškia žiedų sutraukimas yra O(n^2). Kadangi visumoje yra O(n) iteracijų, tai viso algoritmo sudėtingumas yra O(n^3).

Praiktinis tyrimas

Toliau tyrimas su skirtingais m ir n parametrais, kur n – viršūnių, o m – briaunų skaičius.

Testai atlikti kievieno dydžio failui sugeneravus 10 skirtingų grafų, atliekant testą su visais jais po 10 kartų ir paimant laiką kaip vidurkį.



Pastebėta, kad laikas kinta, keičiant m skaičių. Didinant n, bet paliekant m tokį patį veikimo greitis auga greičiau, nei kai m = n, ar m=n*1.5. Kai grafas yra pilnas, veikimo laikas su šiuo algoritmu yra ilgiausias. Iš keturių paimtų m variantų, greičiausiai algoritmas veikė su m=n*1.5.

Kodo paleidimas

Paleidimo Instrukcija: Atsidarę terminalą ar komandinę eilutę, patekite į programos aplankalą. Paleiskite programą naudodami šią eilutę:

./blossoms.exe.

Pasirinkimo Galimybės: Programa išves meniu su trimis galimybėmis:

Choose an option: 1. Use a txt file

2. Manually input n, m, and edges

3. Generate n m graph Enter the option number (1, 2, or 3:

Pasirinkite Galimybę: Įveskite norimą galimybės numerį: 1, 2 arba 3.

Atitinkamas Kodo Vykdymas Pagal Pasirinkimą:

Pasirinkimas 1: Jei pasirinkote skaityti iš tekstinio failo: Įveskite failo pavadinimą, kuriame saugomi grafo duomenys. Programa nuskaitys n, m ir visas briaunas iš failo ir tęs vykdymą pagal šiuos duomenis.

Pasirinkimas 2: Jei pasirinkote rankinį duomenų įvedimą: Programa naudos "manual_input" funkciją, leidžiančią jums įvesti grafo duomenis rankiniu būdu. Grafo duomenys bus išsaugoti failo pavadinimu "manual_input.txt".

Pasirinkimas 3: Jei pasirinkote sugeneruoti grafą: Įveskite norimų viršūnių skaičių n ir briaunų skaičių m. Sugeneruotas grafas bus išsaugotas failo pavadinimu "generated_graph.txt".

Sprendimas bus išsaugotas "solution_" + filename faile, o pačios poros graph_solved.txt faile.

Literatūros sąrašas:

1. A. Schrijver, A Course in Combinatorial Optimization, 2004, pp. 79—85.
2. https://brilliant.org/wiki/blossom-algorithm/ (žiūrėta 18-12-2023)