Vilniaus Universitetas

Matematikos ir informatikos fakultetas

Ilgiausias kelias

Laimonas Beniušis, Kompiuterių Mokslas

Užduoties aprašymas

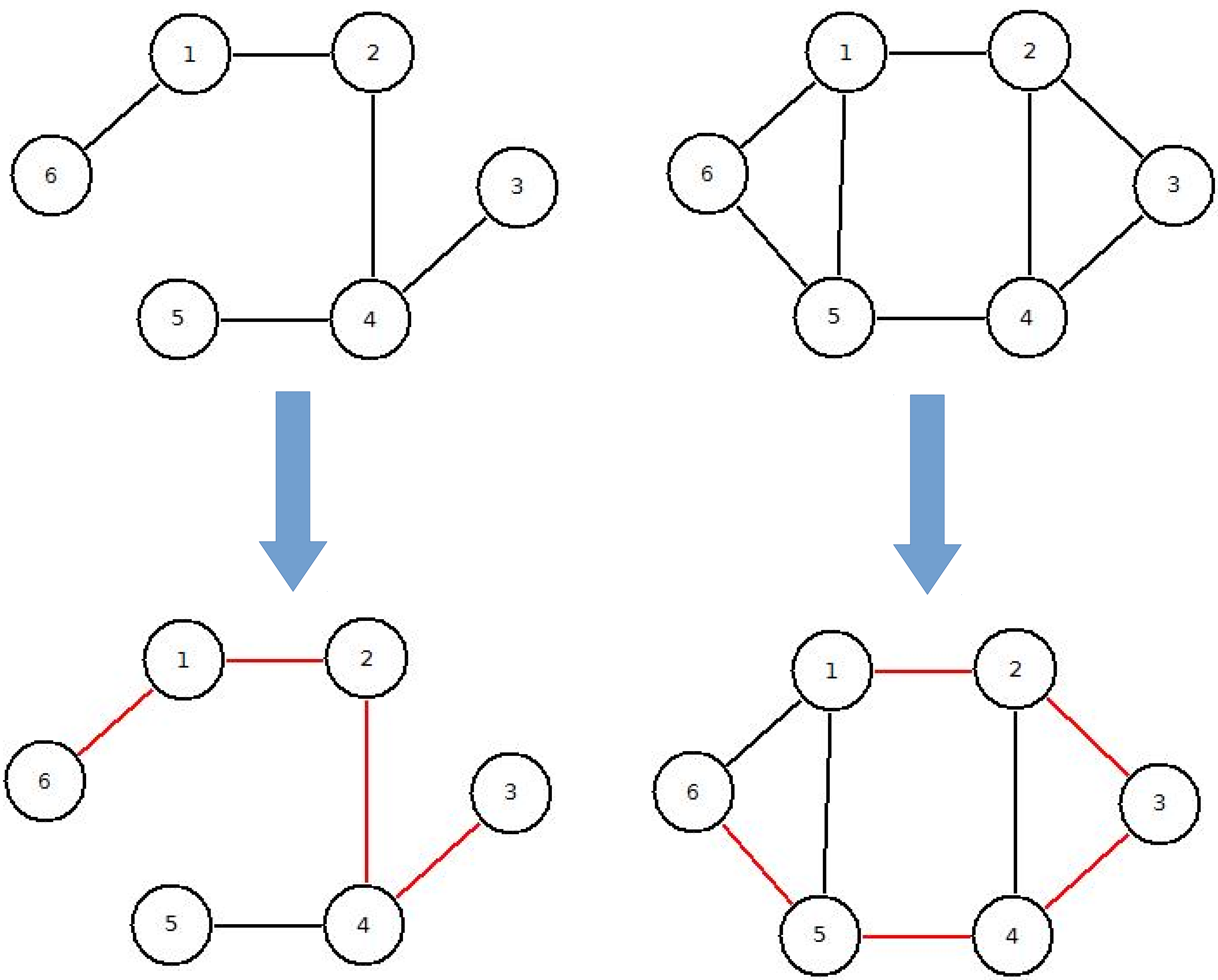
Duota: Neorientuotas grafas G = (V, E)
Rasti: Ilgiausią paprastą kelią

http://www.csc.kth.se/~viggo/wwwcompendium/node114.html

Paprastas kelias – gretimų viršūnių seka, kurioje viršūnė nesikartoja

Ilgiausio kelio problema yra ne tik sunkiai išsprendžiama, bet ir sunkiai aproksimuojama. Yra tam tikri grafo atvejai, kada yra optimali struktūra šiam uždaviniui spręsti. Vienas iš jų – beciklis orientuotas grafas. Šiuo atveju tereikia perrinkti viršūnes topologine tvarka.
Bendru atveju, polinominio greičio sprendimo nėra, o visų kelių perrinkimas yra O(V!) sudėtingumo.

Pavyzdžiai



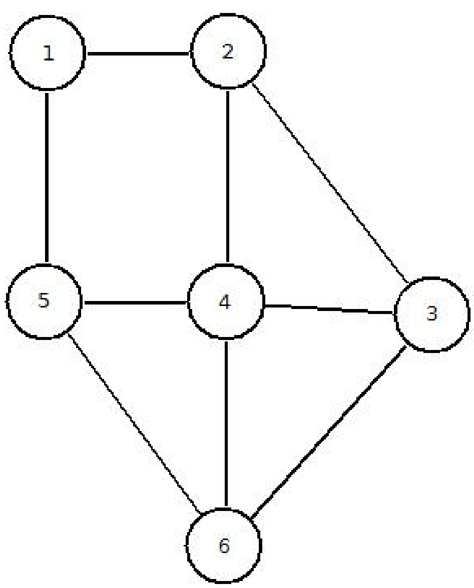
Euristins algoritmas

Veikimo principas: pirmiausiai sujungiam mažiausiai jungias viršūnes tikėdamiesi, kad labiau jungios viršūnės galiausiai apjungs tokiu būdu sudarytus pografius.

1. Etapas:
   1. Susikuriame V dydžio masyvą *P*
   2. Apskaičiuojami visų grafo viršūnių laipsniai ir išsaugojam juos masyve *P*
   3. Sudaromi dinamiški (lengvai plėčiami ir sutraukiami) sąrašai kiekvienai viršūnei
   4. Baigiame 1 etapą

2. Etapas:
   1. Iteracijos pradžia
   2. Visus sąrašus pažymim, kaip nedalyvavusius iteracijoje
   3. Vykdome kol yra iteracijoje nedalyvavusių sąrašų
   4. Pasirenkamas sąrašas S1 (kuris dar nebuvo panaudotas šioje iteracijoje), kurį yra bandoma sujungti:
      1. S1 pažymimas kaip panaudotas iteracijoje. Randami visi kiti iteracijoje nepanaudoti sąrašai, tiriamos S1 **kraštinės** viršūnės v1,v2:
         1. Surandamos visos **kraštinės** viršūnės, kurios yra gretimos v1 arba v2
         2. Iš visų šitų porų, pasirenkama turinti mažiausią viršūnių laipsnių sumą
         3. Jeigu tokia viršūnių pora iš sąrašų S1 ir S2 yra, tai ji tampa jų jungtimi
         4. S1 ir S2 yra sujungiami į vieną
         5. Prieš jungimą, jeigu yra jungiamas to paties tipo kraštas (pradža su pradžia arba pabaiga su pabaiga), tada S2 sąrašo tvarka yra apverčiama, kad išlaikytų sąrašo eiliškumą pagal jungtis
   5. Iteracijos pabaiga
   6. Jeigu sąrašų kiekis itercijos metu nesumažėjo, darbą baigiame ir grąžiname ilgiausią sąrašą
   7. Priešingu atveju pereiname į iteracijos pradžią

Pavyzdys:

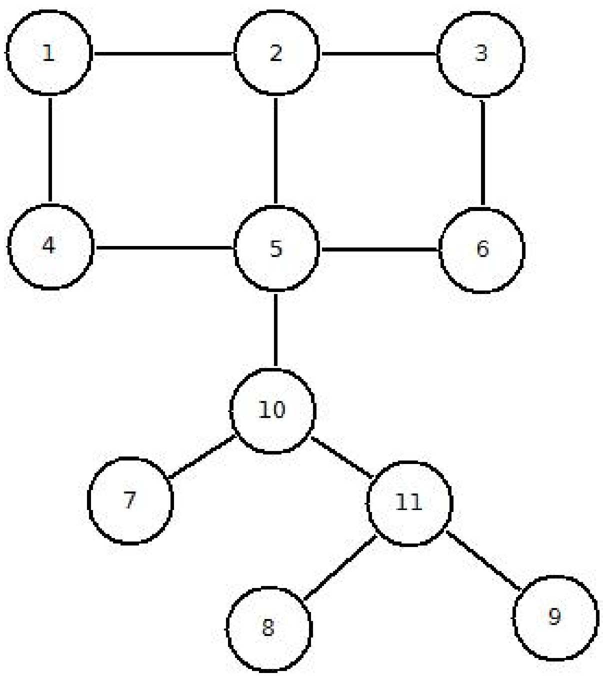


Inicializacija:	{1}; {2}; {3}; {4}; {5}; {6};
Iteracija 1:	{1} + {2}; {3} + {6}; {4} + {5};
Po jos:	{1,2}; {3,6}; {4,5};
Iteracija 2:	{1,2} + {4,5};
Po jos:	{4,5,1,2}; {3,6};
Iteracija 3:	{4,5,1,2} + {3,6};
Po jos:	{4,5,1,2,3,6}
Iteracija 4:	jokių pokyčių

Baigiam darbą
Rastas ilgiausias kelias: {4,5,1,2,3,6}

Šis algoritmas blogai randa ilgiausius kelius, jeigu grafas yra medžio formos. Tokiu atveju, galima bandyti vykdyti algoritmą dar kartą, tačiau su pakitusia sąlygą: sąrašų jungimo metu randa **didžiausią** viršūnių laipsnių sumą.

Iš tokio pastebėjimo, grafas, kurio optimalaus rezultato šis algoritmas neaptiks, gali būti toks:



R – reverse (sąrašas jungiamas atvirkščia tvarka)
Inicializacija: {1}; {2}; {3}; {4}; {5}; {6}; {7]; {8}; {9}; {10}; {11};
Iteracija 1: {1} + {4}; {2} + {3}; {5} + {6}; {7} + {10}; {8} + {11};
Po jos: {1,4}; {2,3}; {5,6}; {7,10}; {8,11}; {9};
Iteracija 2: R{2,3} + {1,4}; {7,10} + {5,6}; {8,11} + {9};
Po jos: {3,2,1,4}; {7,10,5,6}; {8,11,9};
Iteracija 3: {7,10,5,6} + {3,2,1,4};
Po jos: {7,10,5,6,3,2,1,4}; {8,11,9};
Iteracija 4: jokių pokyčių

Baigiam darbą
Rastas ilgiausias kelias: {7,10,5,6,3,2,1,4};
Ilgiausio kelio variantas: {8,11,10,5,6,3,2,1,4};

Euristinio algoritmo sudėtingumo analizė

Sąrašų sujungimo paieška:
Kiekvienas sąrašas blogiausiu atveju turi turi 2 višūnes, kurių incidentines viršūnes reikia tikrinti.
Pereinama per visus nenaudotus iteracijoje sąrašus, kurių gali būti S. S – likusių sarašų kiekis tos iteracijos metu.
Tarkime turime pasirinkto sąrašo kraštines viršūnes v1 ir v2. Iš likusių nenaudotų sąrašų pasirenkame visas kitas kraštines viršūnes, ir gauname potencialių poros viršūnių sąrašą {p1, p2, … , pN} kurio ilgis yra 2S.
Patikriname, ar v1 ir p1..pN yra incidentiškos (egzistuoja tokia briauna), jeigu yra, fiksuojame viršūnių poros laipsnių sumą. Analogiškai su v2. Iš viso gauname $2 \cdot 2S$ patikrinimų.
Iš to išplaukia, kad sąrašui jungimo poros radimas užtrunka 4S kartų. Blogiausiu atveju, iteracijos metu sąrašų kiekis sumažės tik 1 ir tai nutiks per paskutinį patikrinimą, todėl maksimalus sąrašų kiekio kitimas iteracijos metu yra aritmetinės progresijos V-j, j=1..V-1 suma T.y. (V-1 + V-2 + … + 1)

Kadangi sąrašų kiekis mažėja po 1, tai patikrinimų kiekis irgi mažėja kas kiekviena iteracija.
Pvz. Jeigu turime 6 viršūnes, taip kinta maksimalus palyginmų skačius:
4(5 + 4 + 3 + 2 + 1)
4(4 + 3 + 2 + 1)
4(3 + 2 + 1)
4(2 + 1)
4(1)

Galime suvesti tokį dėsningumą į formulę:

$$4\sum_{i=1}^{V-1} i(V-i)=\frac{4}{6}V(V-1)(V+1)=\frac{2}{3}(V^3-V)$$

Taigi, euristinio algoritmo sudėtingumas yra $O(V^3)$.