**Vilniaus Universitetas**



# Matematikos ir Informatikos Fakultetas Informatikos Katedra

## III kursas 1 grupė

### Rytis Petrauskas

Užduotis 1-15:

VISŲ MAKSIMALIŲ GRAFO KLIKŲ GENERAVIMAS

2019

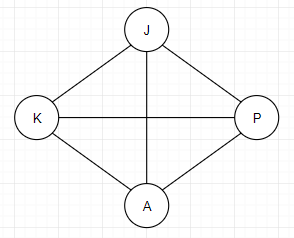
# Užduoties aprašymas

**Duota:** Neorientuotas grafas G = (V,E), kur |V| = n, |E|=m, turintis n viršūnių ir m briaunų.

**Rasti:** Visas maksimalias grafo G klikas, t.y., tokius viršūnių aibės poaibius, kurie grafe G sudaro pilną pografį ir kurių nebegalima praplėsti taip, kad vėl gautųsi pilnas pografis.

Ši problema dažnai naudojama rasti dideles žmonių grupes, kurie yra pažįstami socialiniuose tinkluose, pavyzdžiui facebook‘e, chemijoje šį problema yra ekvivalenti panašių molekulių struktūrų paieškai, bioinformatikoje rasti panašius evoliucijos medžius, baltymų struktūras.

**Pavyzdys:** Tarkime turime socialinį tinklalapį, kuriame žmones gali tapti vienas kito draugais. Vienas iš algoritmų, kuris padetu surasti tarpusavyje pažįstamu žmonių grupes socialiniame tinkle, yra ekvivalentus maksimalių klikų radimui. Šiame grafe matome, jog Jonas, Petras, Kęstas ir Algis visi yra sujungti briauna tarpusavyje. Tai reiškia, jog šiame grafe yra viena maksimali klika. Tada socialinis tinklas, radęs tokias grupes gali atlikti jiems naudingas funckcijas kaip reklamų taikymas, rekomendacijos. Praktikoje naudojama šiek tiek modifikuoti algoritmai, kurie ieško maksimalių klikų, kuriose maksimalus atstumas tarp draugų gali būti tam tikras skaičius s. Vis gi šiame darbe kalbėsime apie klasikinį Bron-Kerbosch algoritmą, kuris randa visas maksimalias klikas.



# Algoritmo Aprašymas

g

### Algoritmo kintamieji

Šiam uždaviniui spręsti naudosime duomenų struktūra ***Vertex***, kuri apibrėžia grafo viršūnę. Struktūra turi tris parametrus: savo pavadinimą ***x*** (numeruojama nuo 0), visų savo kaimynių sąrašą ***nbrs***(ArrayList) bei laipsnį ***degree.*** Norint inicijuoti grafą, turime pateikti failą, kuriame turi būti pateikta tokia informacija: viršūnių skaičius, kitoje eilutėje briaunų skaičius ir po jos išvardintos visos briaunos. Nuskaičius failą, grafui atvaizduoti turėsime šių viršūnių sąrašą ***graph***(ArrayList). Uždavinio tikslas, naudoti rekursija ir radus maksimalią kliką, ja išspausdinti ekrane.

class Vertex implements Comparable<Vertex> {

private int x;

private int degree;

ArrayList<Vertex> nbrs = new ArrayList<>();

(...)

}

### Klikų radimo algoritmas

Algoritmo funkcijai yra duodami 3 sąrašai: R, kuris potencialiai bus maksimali klika, P, kuriame laikome visas grafo viršūnes, ir X. R ir X yra tušti, o P turi visas viršūnes. Taigi pradėdami algoritmą patikrininame ar P ir X yra tušti, tokiu atvėju R yra maksimali klika, nes tada nebera elementų, kuriais galėtume papildyti R. Jei tai nėra tiesa, išrenkame viršūnę su didžiausiu laipsniu, tai bus mūsų atraminė viršūnė. Prieš tai sukuriame kopija P sąrašo, jį pavadiname P1. Tada iš P pašaliname visas atraminės viršūnės kaimynes, kitu atvėju algoritmas rekursyviai kreiptųsi į visas klikas, nors jos nebūti maksimalios, taip sumažindami kreipinių skaičių. Tada einam per visus P elementus ir cikle atliekame tokius veiksmus: įdedame į R einamąja viršūnę ir rekursyviai kreipiames į funkcija su jau papilditu R, P tampa P1 ir atraminės viršūnės kaimynių sankirta, o X tampa X ir atraminės viršūnės kaimynių sankirta. Poto ji panaikina viršūnę iš R ir P1, ja ideda į X, tam, kad tolimesniuose kreipiniuose ji nebebūtų naudojama. Algoritmas rašytas java kalba atrodo taip:

**Pseudokodas:**

BronKerbosch(R,P,X):

if P and X are both empty:

report R as a maximal clique

choose a pivot vertex u in P ⋃ X

for each vertex v in P \ N(u):

BronKerbosch2(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))

P := P \ {v}

X := X ⋃ {v}

private void Bron\_KerboschWithPivot(ArrayList<Vertex> R, ArrayList<Vertex> P,

ArrayList<Vertex> X, String pre) {

System.out.print(pre + " " + printSet(R) + ", " + printSet(P) + ", "

+ printSet(X));

if ((P.size() == 0) && (X.size() == 0)) {

printClique(R);

return;

}

System.out.println();

ArrayList<Vertex> P1 = new ArrayList<>(P);

Vertex u = getMaxDegreeVertex(union(P, X));

P = removeNbrs(P, u);

for (Vertex v : P) {

R.add(v);

Bron\_KerboschWithPivot(R, intersect(P1, getNbrs(v)),

intersect(X, getNbrs(v)), pre + "\t");

R.remove(v);

P1.remove(v);

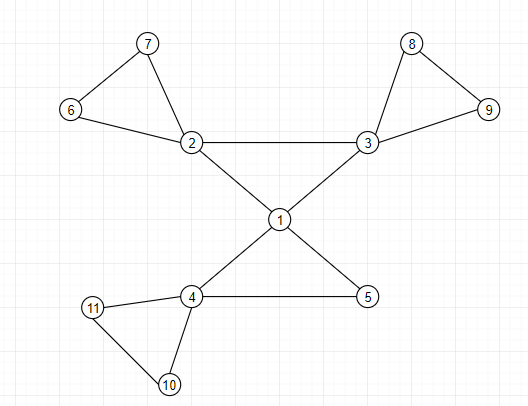
X.add(v);

}

}

# Ekspermentas

Buvo paimtas grafas, kuris turi 11 viršūnių ir 15 briaunų. Algoritmas buvo leistas 100000 kartų.



{}, {1,2,3,4,5,6,7,8,9,10,11}, {}

{2}, {1,3,6,7}, {}

{2,3}, {1}, {}

{2,3,1}, {}, {} -------------- Maximal Clique : 2 3 1

{2,6}, {7}, {}

{2,6,7}, {}, {} -------------- Maximal Clique : 2 6 7

{2,7}, {}, {6}

{3}, {1,8,9}, {2}

{3,8}, {9}, {}

{3,8,9}, {}, {} -------------- Maximal Clique : 3 8 9

{3,9}, {}, {8}

{4}, {1,5,10,11}, {}

{4,1}, {5}, {}

{4,1,5}, {}, {} -------------- Maximal Clique : 4 1 5

{4,10}, {11}, {}

{4,10,11}, {}, {} -------------- Maximal Clique : 4 10 11

{4,11}, {}, {10}

{6}, {7}, {2}

{7}, {}, {2,6}

{8}, {9}, {3}

{9}, {}, {3,8}

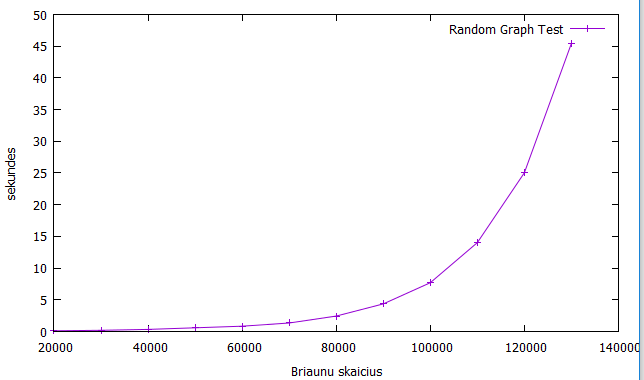
---------------------------------------------------

Briaunų: 15

Maksimalių klikų: 5

paieškos trukme : 1.353sek.

# Uždavinio sudėtingumas



Pagal J.W. Moon ir L. Moser tyrimą, atliktą 1965, kiekvienas grafas, su n viršūnių maksimaliai gali turėti 3^(n/3) maksimalių klikų. Kadangi šis algoritmas, kaip medyje, iteruoja iki kiekvienos maksimalios klikos, blogiausiu atvėju jam reikės pereiti per jas visas

Taigi šio algoritmo sudėtingumas blogiausiu atvėju yra O(3^(n/3)). Visgi praktikoje šis algoritmas yra pakankamai greitas.

### Programos naudojimas, parametrai

$ Bronn.exe failas [[-s] [-f]] [n]

Programą paleisti galime dviem būdais:

* Neįvede jokių parametrų, ekspermentas bus vykdomas su atsitiktiniais grafais, kurie turi 1000 viršūnių, kuris bus vykdomas 12 kartų kaskart didinant briaunų kiekį 10000, pradedant nuo 20000.
* Kitu atvėju testavimui turime įvesti failo vardą, pasirinkti kur spausdinti rezutatus, -s į ekraną, -f į failą bei kiek kartų vykdyti testavimą. Tokiu atvėju turime būtinai nurodyti reikšmes. Pavyzdžiui: (Bronn.exe grafas1.txt -f 1000).