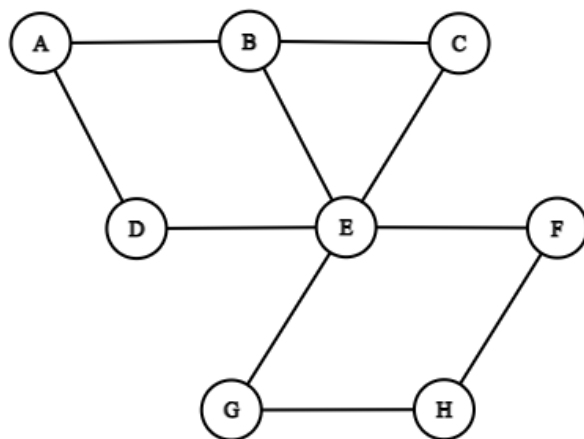


Užduotis: Rasti maksimalų grafo G viršūnių poravimą, t.y. maksimalų briaunų aibės E poaibį E' , kurio visos briaunos nėra incidentinės

Briaunų E' poaibis, kurio visos briaunos nėra incidentinės reiškia tai, kad jokios 2 briaunos negali turėti bendros viršūnės.

Pavyzdys: Tarkime turime tokį grafą



Šis grafas turi tokias briaunas: AB, BC, AD, BE, CE, DE, EF, FH, HG, GE
ir tokias viršūnes: A, B, C, D, E, F, G, H

Kaip vyksta poravimas: Imame briauną AB ir pridedame jos viršūnes į *panaudoti* sąrašą, tada panaudotų viršūnių sąrašą bus 2 viršūnės: ["A", "B"], toliau imame briauną iš sąrašo, kuriame nebėra briaunų, kurios turi viršūnių iš panaudotų viršūnių sąrašo.

Taigi po pirmo briaunos paėmimo turime tokią situaciją:

Panaudotos viršūnės = ["A", "B"]

Galimos briaunos = ["CE", "DE", "EF", "FH", "HG", "GE"]

Imame CE briauną :

Panaudotos viršūnės = ["A", "B", "C", "E"]

Galimos briaunos = ["FH", "HG"]

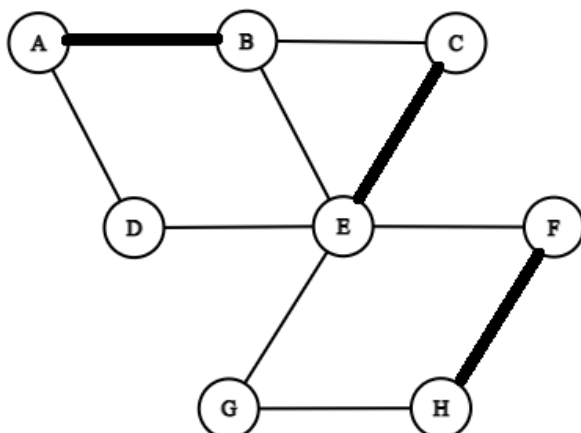
Imame FH briauną:

Panaudotos viršūnės = ["A", "B", "C", "E", "F", "H"]

Galimos briaunos = []

Kadangi į panaudotų viršūnių sąrašą dėjome viršūnės eilės tvarka, dabar poruodami po 2 elementus galime gauti E' poaibį: AB, CE, FH

Taip atrodo grafa, su paryškintu poravimu:

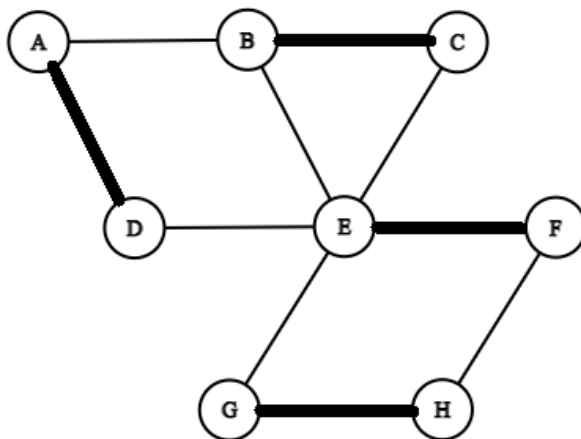


Tai nėra pabaiga, nes šis poravimas nėra maksimalus, tai reiškia, kad įmanoma taip suporuoti pradinę grafą, kad atsakyme būtų daugiau nei 3 briaunos.

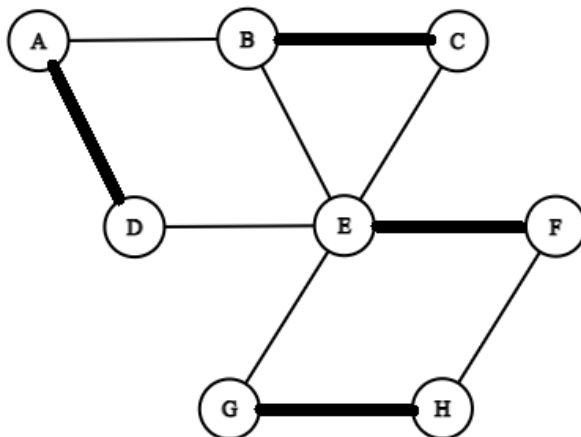
Kad rastume maksimalų poravimą, aš suskaičiuoju tiek poravimų, kiek yra briaunų, tad šiame pavyzdyje reikia apskaičiuoti poravimą 10 kartų, kiekvieną kartą startuojant nuo vis kitos briaunos.

Jau apskaičiavau kai pradėdame nuo AB, dabar liko dar 9 kartai:

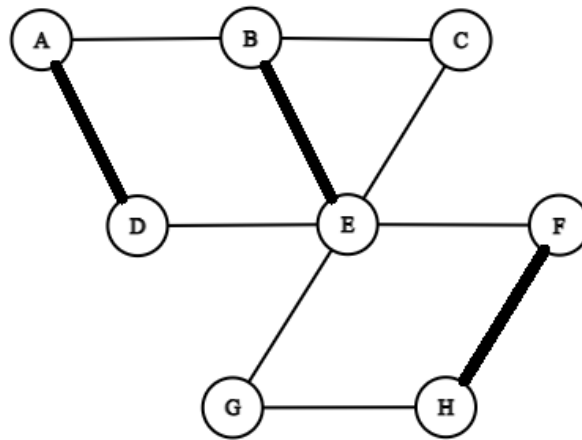
1) Pradedu nuo BC



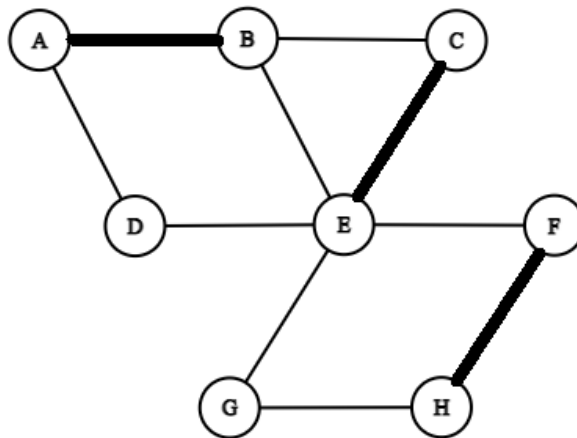
2) Pradedu nuo AD



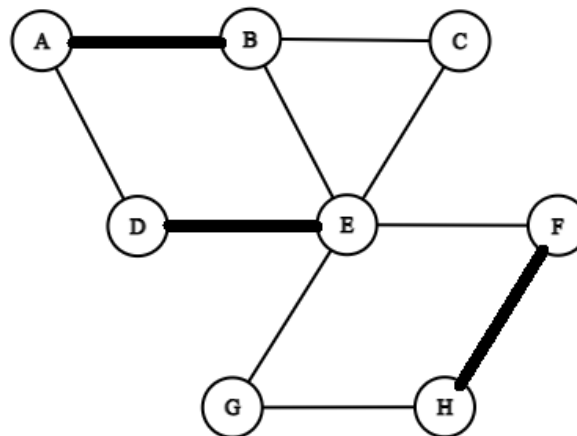
3) Pradedu nuo BE



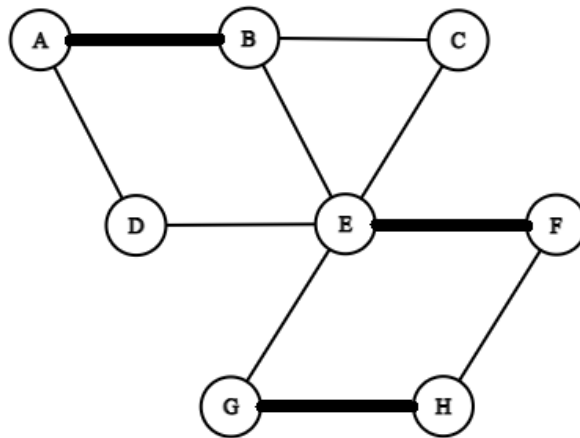
4) Pradedu nuo CE



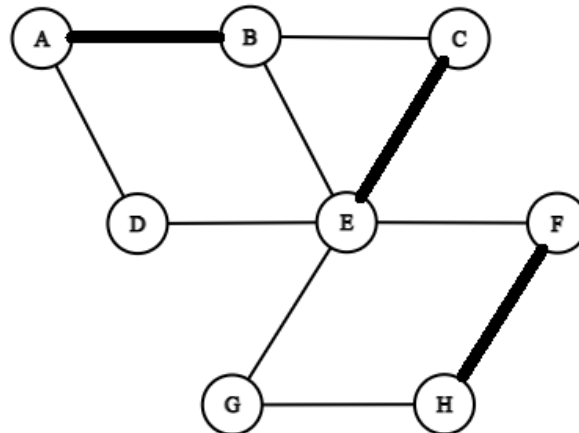
5) Pradedu nuo DE



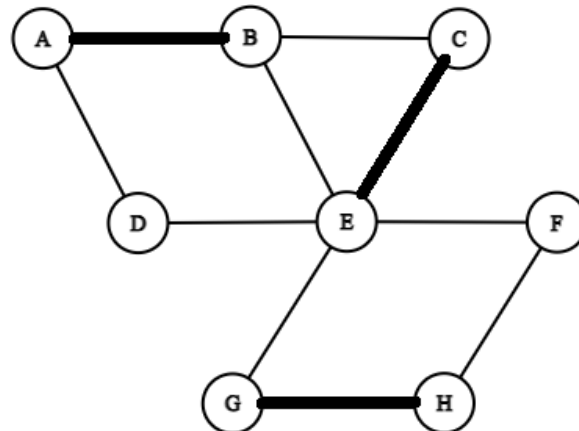
6) Pradedu nuo EF



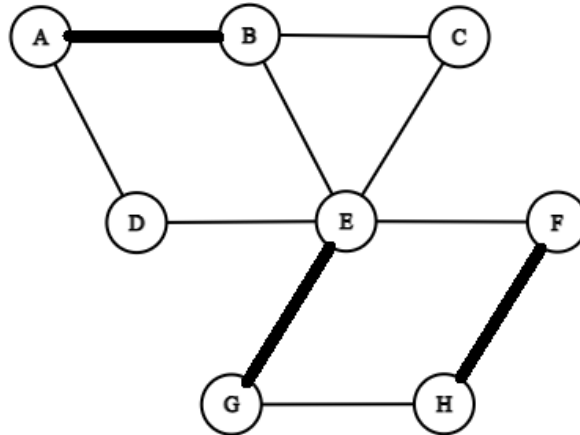
7) Pradedu nuo FH



8) Pradedu nuo HG



9) Pradedu nuo GE



Matome kad didžiausios E' aibės susidaro pradedant poravimą nuo BC arba AD.
Nėra skirtumo kurį rezultatą rašysime, tad rezultatas šio grafo: ["BC", "AD", "EF", "HG"]

Algoritmo kintamieji:

Algoritmą sprendamas naudoju tokius sąrašus kaip:

1. MM - kiekvienai sąlygai čia bus m poravimų, kiekvienas turės po E' briaunų sąrašą ir E' poaibio dydį, kur m - briaunų skaičius
2. panaudoti - čia skaičiuojant kiekvieną poravimą, bus kaupiamos panaudotos viršūnės.

Grafą įvedimo faile vaizduosiu briaunų masyvo pavidalu.

Masyvas *in_list* saugos dabar skaičiuojamos grafos įvedimo briaunų masyvą.

Algoritmo tikslas, rasti grafo maksimalų briaunų aibės E poaibį E' , kurio visos briaunos nėra incidentinės.

Maksimalaus poravimo algoritmas:

Gavę įvestą briaunų masyvą skaičiuosime poravimą tiek kartų, kiek bus briaunų, pradėdami nuo vis kitos briaunos, tai reiškias turėsime patys pasigaminti sąrašą, kuris atitiktų skaičiavimo pradėjimą nuo vis kitos briaunos. Tai pasiekiu m kartų iškėlęs vis kitą briauną į sąrašo pradžią.

Toliau einu per briaunas, ir paimdamas briauną, jos viršūnes pridedu į sąrašą *panaudoti*, taip vėliau žinau, kokios galimos briaunos liks kitai iteracijai, ir kaupiu rezultato sąrašą. Radęs skaičiuojama poravimą, į rezultatų sąrašą pridedu poravimo briaunų sąrašą ir poravimo sąrašo dydį kaip vieną objektą. Gale algoritmo randu didžiausią poravimo sąrašo dydį ir išvedu į ekraną atitinkantį poravimą.

Briaunos iškėlimo į priekį algoritmas (python 3.6 kodas):

```
def apkeisti(E, i):
    if i == 0:
        return E
    Rez = [E[i]] + E[:i] + E[i + 1:]
    return Rez
```

Maksimalaus viršūnių poravimo skaičiavimo algoritmas (python 3.6 kodas):

```
def skaiciuoti(B):
    MM = []
    listas = B
    for i in range(0, len(listas)):
        C = apkeisti(listas, i)
        panaudoti = []
        for b in C:
            if b[0] not in panaudoti and b[1] not in panaudoti:
                panaudoti.append(b[0])
                panaudoti.append(b[1])
            result = []
            for i in range(0, len(panaudoti), 2):
                result.append(panaudoti[i] + panaudoti[i + 1])
            MM.append([result, len(result)])
    print(max(MM, key=itemgetter(1))[0])
```

Duomenų nuskaitymo ir siuntimo skaičiavimui algoritmas (python 3.6 kodas):


```
with open('duomenys.txt') as f:
    for line in f.read().splitlines():
        in_list = str(line.splitlines())
        in_list = in_list[2:len(str(line.splitlines()))-2]
        in_list = in_list.split(',')
        skaiciuoti(in_list)
```

Laiko tyrimo kodas (python 3.6 kodas):

```
def generuoti_detailed():
    for kartai in range(10):
        V = ['AB', 'CD', 'EF', 'GH', 'IJ', 'KL', 'MN', 'OP', 'RS', 'TU', 'VZ', '12', '34', '56', '67', '89',
            'ab', 'cd', 'ef', 'gh', 'ij', 'kl', 'mn', 'op', 'rs', 'tu', 'vz']
        for l in range(len(V)):
            for i in range(5, 100, 5):
                cut = V[0:l]
                final = []
                for v in cut:
                    if random.randint(0, 100) > i:
                        final.append(v)
                start = datetime.datetime.now()
                for j in range(1000):
                    skaiciuoti(final)
                end = datetime.datetime.now()
                elapsed = end - start
                print(l, elapsed.total_seconds())
```

Programos naudojimas, parametrai:

Surašyti duomenis reikia į duomenys.txt failą. Grafo briaunas atskirti kableliu, o skirtingas grafas nauja linija. 'duomenys.txt' pavyzdys:

 duomenys.txt - Notepad

File Edit Format View Help

```
AB,BC,AD,BE,CE,DE,EF,FH,HG,EG
DE,EA,EJ,JK,EK,AG,AB,BG,KG,BC,GH,CH,KM,CF,HI,IF,IM
AC,CD,DB,BA,CB
AC,CB,AB,BD,BE,BF
BA,BC,AC,AD,CE,DE,DF
```

Nuėjus į programos failo vietą programos paleidimo komanda:

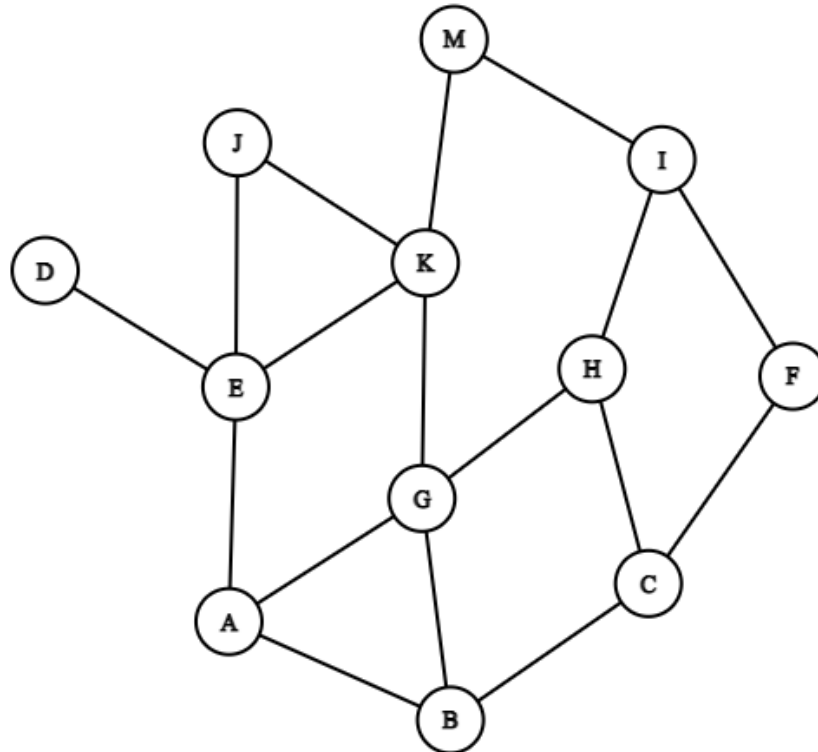
python MM.py

Eksperimentai:

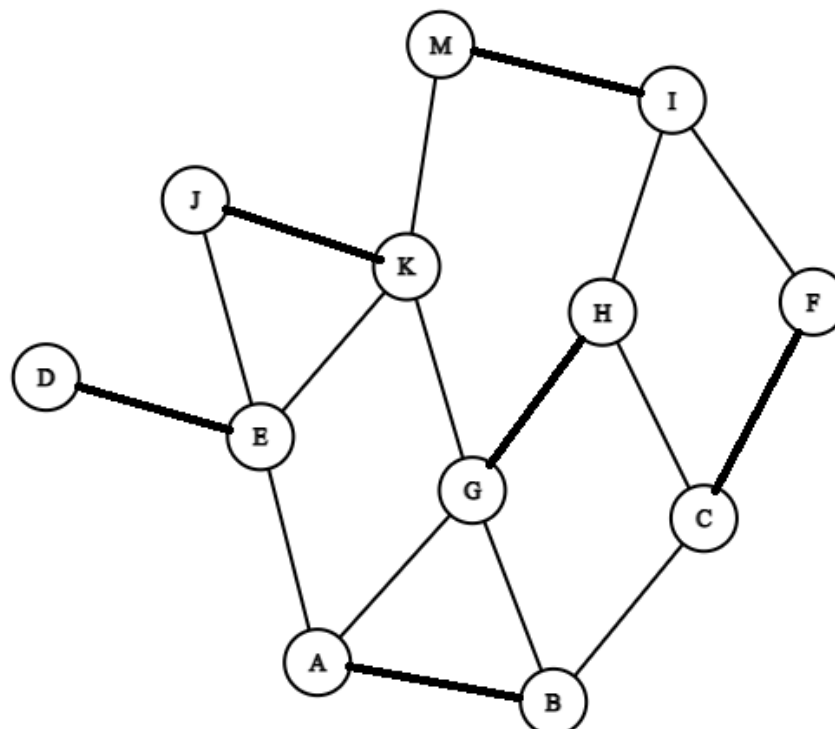
Eksperimentai daryti pagal viršų pateiktą duomenys.txt failo pavyzdį pradedant nuo antros eilutės, nes pirmoji išnagrinėta skiltyje „Pavyzdys“.

Grafas 1:

Duota:

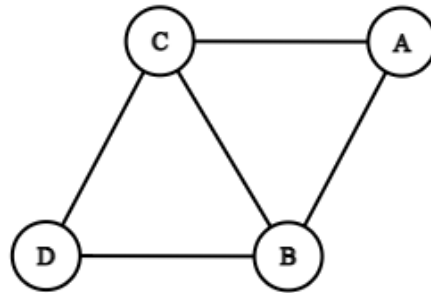


Rezultatas:

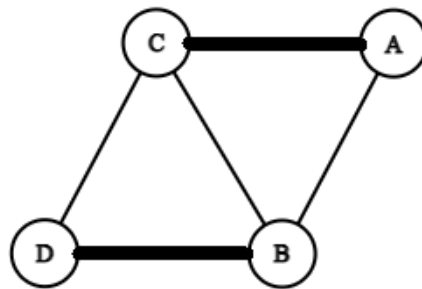


Grafas 2:

Duota: .

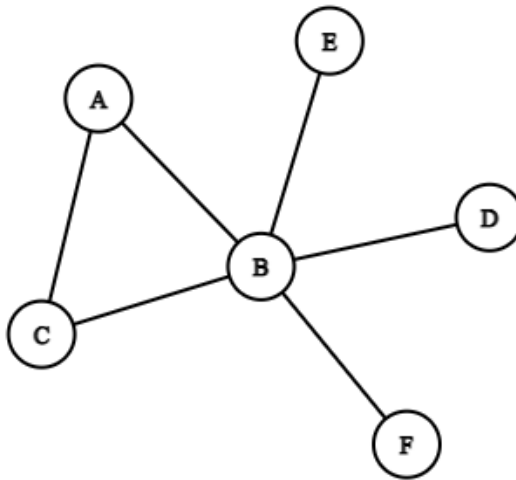


Rezultatas:

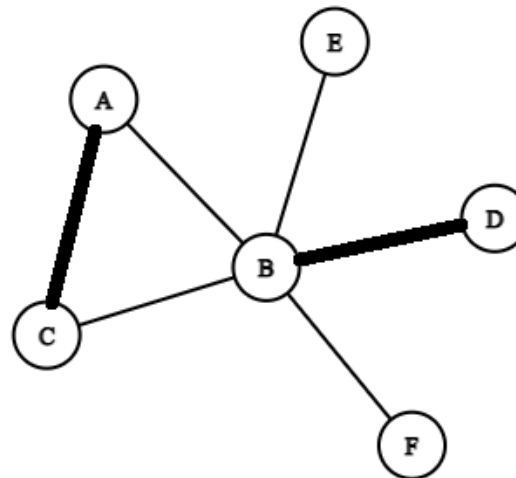


Grafas 3:

Duota:

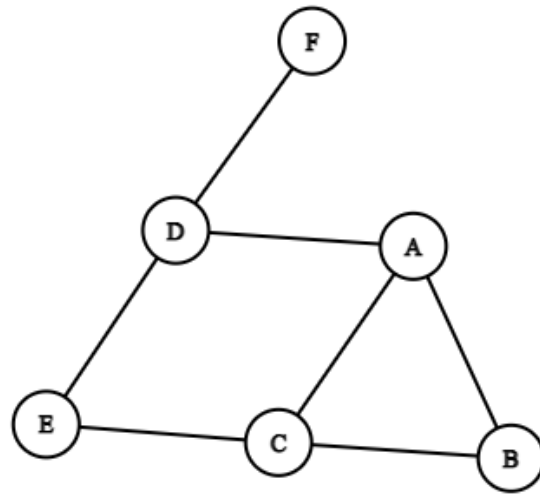


Rezultatas:

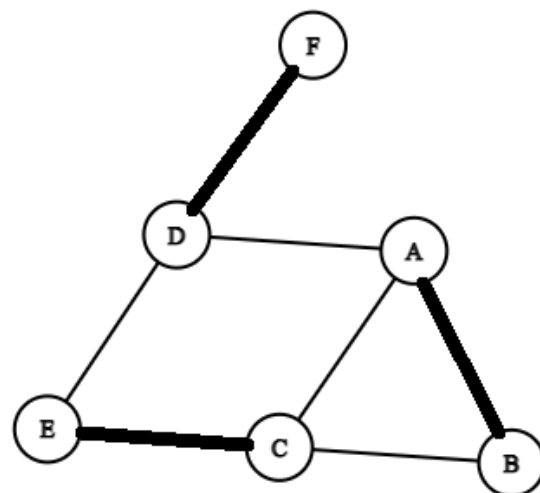


Grafas 4:

Duota:



Rezultatas:



Uždavinio sudėtingumas:

Nagrinėjame 3 FOR ciklus kuriuos turiu algoritme:

Pirmasis for ciklas priklauso nuo briaunų sąrašo dydžio, tad priklauso nuo m .

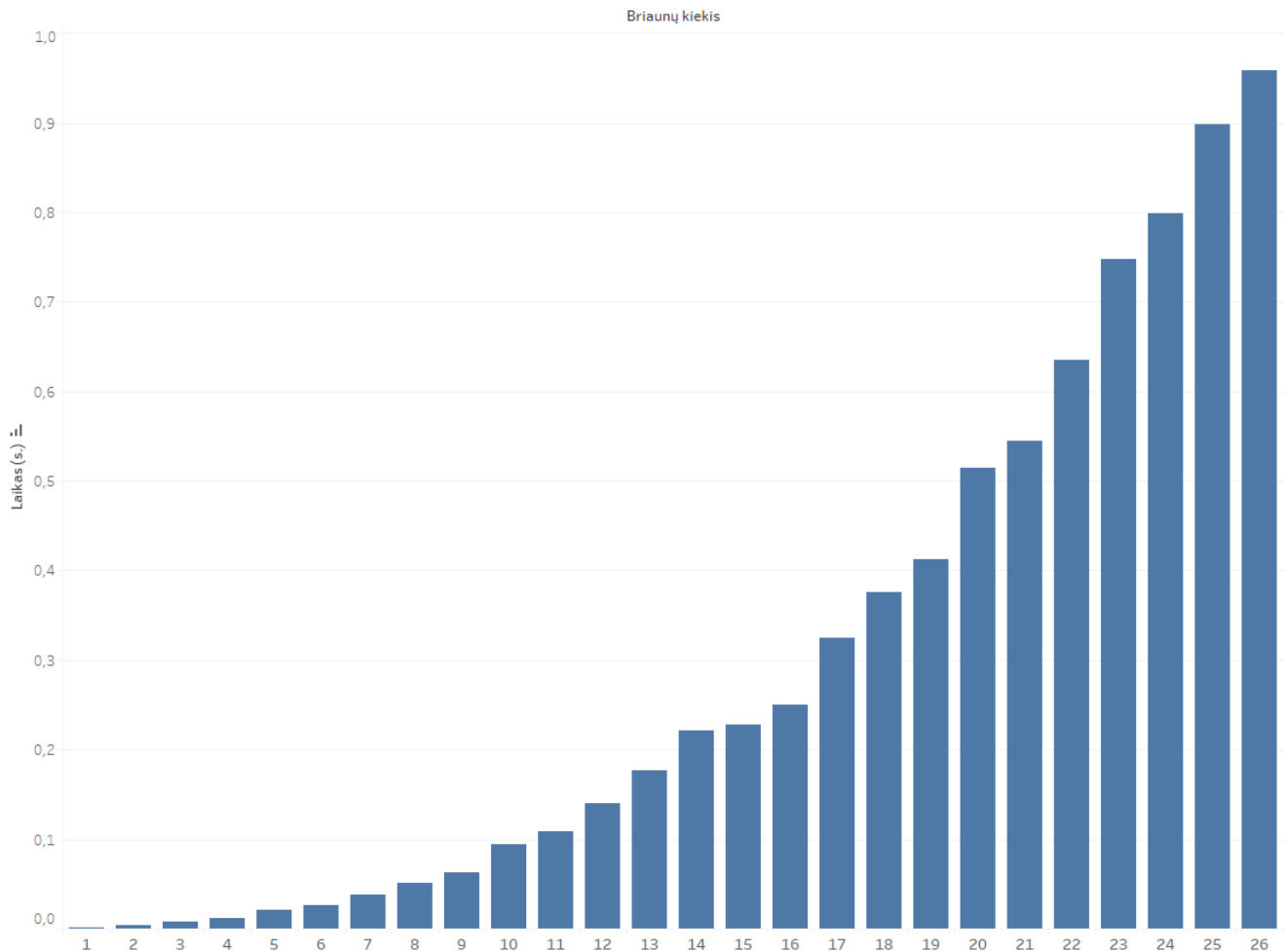
Antrasis for ciklas taip pat priklauso nuo briaunų sąrašo dydžio, tad priklauso nuo m .

Trečiasis for ciklas parodo kad

Šie trys for ciklai yra vienas kitame, tad gauname laiką $O(m*m*n) = O(m^2*n)$

Toliau reikia atsižvelgti kad m kinta nuo 0 iki $(n^2)/2$, tad galutinį laiką gauname $O((n^5)/4) = O(n^5)$

Taigi laikas *polinominis*



Vidurkių išvedimai ir vizualizacija padaryta su Tableau.

Literatūra:

- 1) https://www.youtube.com/watch?v=jtgBCGVux-8&ab_channel=Udacity
- 2) https://www.youtube.com/watch?v=bOJC93XxoFc&ab_channel=Udacity
- 3) https://www.youtube.com/watch?v=chdr2aj4FUc&t=710s&ab_channel=WraithofMath