KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Algoritmų sudarymas ir analizė (P170B400)

Individualaus projekto ataskaita

Atliko:
IFF-1/1 gr. Studentas
Vytenis Kriščiūnas
2023 m. gegužės 12 d.

KAUNAS 2023

TURINYS

1. Individualus projektas

1.1. Darbo užduotis

Faile places_data.xlsx pateikta informacija apie lankytinas vietas (1 lentelė). Tikslas: kaip galima pigesnio maršruto sudarymas kai:

- priimama, kad kelionės tarp vietų kaina lygi kvadratinei šakniai iš kelionės atstumo;
- reikia aplankyti visas vietas;
- ta pati vieta negali būti aplankyta daugiau nei vieną kartą (tariama, kad vieta aplankyta, jei ją aplankė bet kuris iš keliautojų);
- kelionės pradžios ir pabaigos vieta sutampa (su grįžimu atgal);
- maršrutas planuojamas 2 keliautojam.

1.2. Apskaičiuotas asimptotinis programos vykdymo laiko sudėtingumas

Asimptotinis sudėtingumas: T(n) = O(n^3)

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IP
{
    internal class Location
    {
        public string Name { get; set; }

        public long Id { get; set; }

        public double X { get; set; }

        public double Y { get; set; }

    }

    internal class Program
    {
        const string InputFile = "IP_duomenys.txt";
// C | 1
        public static void Main(string[] args)
        {
            Stopwatch sw = Stopwatch.StartNew();
// C | 1
            sw.Start();
// C | 1
            List<Location> places = InOutUtils.ReadPlacesFromTextFile(InputFile);
// f(n) = n + C | 1

            List<long> visitedIds1 = new List<long>();
// C | 1
            List<long> visitedIds2 = new List<long>();
// C | 1
            long firstName = 5844429476;
// C | 1
            long secondName = 1013895687;
// C | 1

            TaskUtils.Solve(places, firstName, secondName, visitedIds1, visitedIds2);
// f(n) = n^3 + n^2 + n + C | 1

            InOutUtils.PrintResults(places, visitedIds1, TaskUtils.CalculateCost(places,
visitedIds1));     // f(n) = n^2 + n + C && T(n) = n^2 + n + C | 1
            InOutUtils.PrintResults(places, visitedIds2, TaskUtils.CalculateCost(places,
visitedIds2));     // f(n) = n^2 + n + C && T(n) = n^2 + n + C | 1


            sw.Stop(); Console.WriteLine($"Trukmė: {sw.Elapsed.TotalSeconds}");
// C | 1
```

```
        }
        // T(n) = n^3 + n^2 + n + C = O(n^3)
    }

    internal class TaskUtils
    {
        public static double CalculateCost(List<Location> places, List<long> visitedIds)
        {
            double cost = 0.0;                                                  // C | 1
            for (int i = 0; i < visitedIds.Count - 1; i++)                      // C | n
+ 1
            {
                long id1 = visitedIds[i];                                       // C | n
                long id2 = visitedIds[i + 1];                                   // C | n
                Location place1 = places.First(p => p.Id == id1);               // C |
n*n
                Location place2 = places.First(p => p.Id == id2);               // C |
n*n
                if (place1 != null && place2 != null)                           // C | n
                {
                    double dx = place2.X - place1.X;                            // C | n
                    double dy = place2.Y - place1.Y;                            // C | n
                    cost += Math.Sqrt(dx * dx + dy * dy);                       // C | n
                }
            }
            return cost;                                                        // C | 1
        }
        // T(n) = n^2 + n + C

        public static void Solve(List<Location> places, long nameOfLocation1, long
nameOfLocation2, List<long> visitedIds1, List<long> visitedIds2)
        {
            double[,] distances = new double[places.Count, places.Count];
// C | 1
            for (int i = 0; i < places.Count; i++)
// C | n+1
            {
                for (int j = 0; j < places.Count; j++)
// C | (n+1)*n
                {
                    double dx = places[j].X - places[i].X;
// C | n*n
                    double dy = places[j].Y - places[i].Y;
// C | n*n
                    distances[i, j] = Math.Sqrt(dx * dx + dy * dy);
// C | n*n
                }
            }

            int placeId1 = places.FindIndex(p => p.Id == nameOfLocation1);
// C | n
            visitedIds1.Add(places[placeId1].Id);
// C | 1

            int placeId2 = places.FindIndex(p => p.Id == nameOfLocation2);
// C | n
            visitedIds2.Add(places[placeId2].Id);
// C | 1

            while (visitedIds1.Count + visitedIds2.Count < places.Count)
// C | n + 1
```

```
            {

                long nextId1 = NextId(visitedIds1, visitedIds2, places, distances);
// f(n) = n^2 + n + C | n
                if (nextId1 != -1)
// C | n
                {
                    visitedIds1.Add(nextId1);
// C | n
                }

                long nextId2 = NextId(visitedIds2, visitedIds1, places, distances);
// f(n) = n^2 + n + C | n
                if (nextId2 != -1)
// C | n
                {
                    visitedIds2.Add(nextId2);
// C | n
                }

            }
            visitedIds1.Add(places[placeId1].Id);
// C | 1
            visitedIds2.Add(places[placeId2].Id);
// C | 1
        }
        // T(n) = n^3 + n^2 + n + C


        private static long NextId(List<long> visitedIds, List<long> otherVisitedIds,
List<Location> places, double[,] distances)
        {
            long nextId = -1;
// C | 1
            double minCost = double.MaxValue;
// C | 1
            long lastId = visitedIds.Last();
// C | 1
            int lastIdx = places.FindIndex(p => p.Id == lastId);
// C | n

            for (int i = 0; i < places.Count; i++)
// C | n + 1
            {
                if (visitedIds.Contains(places[i].Id) ||
otherVisitedIds.Contains(places[i].Id))    // C | n * n
                {
                    continue;
// C | n
                }
                double cost = distances[lastIdx, i];
// C | n
                if (cost < minCost)
// C | n
                {
                    nextId = places[i].Id;
// C | n
                    minCost = cost;
// C | n
                }
```

```
            }

            return nextId;
// C | 1
        }
    }
    // T(n) = n^2 + n + C

    internal class InOutUtils
    {
        public static List<Location> ReadPlacesFromTextFile(string fileName)
        {
            List<Location> places = new List<Location>();                    // C | 1
            using (StreamReader reader = new StreamReader(fileName))         // C | 1
            {
                reader.ReadLine();                                           // C | 1
                string line;                                                 // C | 1
                while ((line = reader.ReadLine()) != null)                   // C | n
                {
                    string[] values = line.Split('\t');                      // C | n
                    string name = values[0];                                 // C | n
                    long id = long.Parse(values[1]);                         // C | n
                    double x = double.Parse(values[2]);                      // C | n
                    double y = double.Parse(values[3]);                      // C | n
                    places.Add(new Location { Name = name, Id = id, X = x, Y = y });
// C | n
                }
            }
            return places;                                                   // C | 1
        }
        // T(n) = n + C


        public static void PrintResults(List<Location> places, List<long> visitedIds,
double totalCost)
        {
            Console.WriteLine($"Aplankytu vietu skaicius: {visitedIds.Count}, Pilna
kaina: {totalCost}");   // C | 1
            foreach (long id in visitedIds)
// C | n
            {
                Location place = places.First(p => p.Id == id);
// C | n*n
                if (place != null)
// C | n
                {
                    Console.WriteLine($"{id}: {place.Name}");
// C | n
                }
                else
// C | n
                {
                    Console.WriteLine($"Negalima surasti vietos pagal identifikatoriu
{id}");                // C | n
                }
            }
            Console.WriteLine();
// C | 1
        }
        // T(n) = n^2 + n + C
```

```
    }
}
```

1.3. Realizuotų programų abstraktus aprašas („pseudo“ kodas ar „workflow“ diagrama)

Apibrėžkite „Location“ klasę su šiomis savybėmis:

Vietos vardas : Name

Vietos identifikavimo numeris: Id

Kelionės koordinatės : X ir Y

1. „InOutUtils“ klasėje, „ReadPlacesFromTextFile“ metode nuskaityk vietoves iš tekstinio failo.
2. „Main“ klasėje išsirink dvi vietoves ir nustatyk jas kaip kelionės pradžios ir pabaigos taškus.
3. Sukurk du sąrašus aplankytų vietovių
4. „TaskUtils“ klasėje, „Solve“ metode apskaičiuok visus galimus atstumus tarp vietovių
5. Kol bendras dviejų keliautojų aplankytų vietovių skaičius yra mažesnis už bendrą visų vietovių skaičių, tol leisk „While“ ciklą.
    a. „While“ ciklo viduje pirmam keliautojui ieškok trumpiausių atstumų nuo esamų vietovių iki tolimesnių vietovių, taip kad šios vietovės dar nebūtų aplankytos nei vieno iš keliautojų
    b. „While“ ciklo viduje antram keliautojui ieškok trumpiausių atstumų nuo esamų vietovių iki tolimesnių vietovių, taip kad šios vietovės dar nebūtų aplankytos nei vieno iš keliautojų.
    c. Kai reikiamos vietovės yra randamos, jos yra atitinkamai talpinamos į pirmojo ar antrojo keliautojo sąrašus.
6. Gražink abiejų keliautojų aplankytų vietovių sąrašus.
7. „CalculateCost“ metode apskaičiuok atskirai pirmo ir antro keliautojų aplankytų vietovių kainas susumuojant atstumus tarp vietovių.
8. „InOutUtils“ klasėje, „PrintResults” metode išspausdink abejų keliautojų aplankytų vietovių sąrašų rezulatus.

1.4. Skirtingų metodų rezultatų analizė uždavinio gerumo ir vykdymo laiko prasmė esant kelioms skirtingoms pradinėms sąlygoms.

Kadangi tik „Solve“ metodas turi didžiausią reikšmę programos asimptotiniam sudėtingumui, todėl būtent jis yra testuojamas.

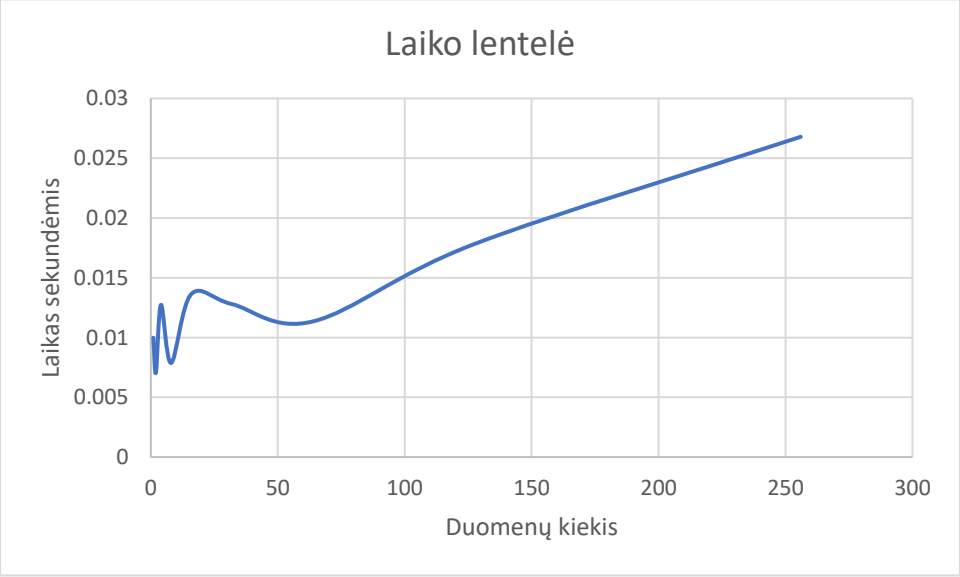
1. Keičiamas elementų kiekis:

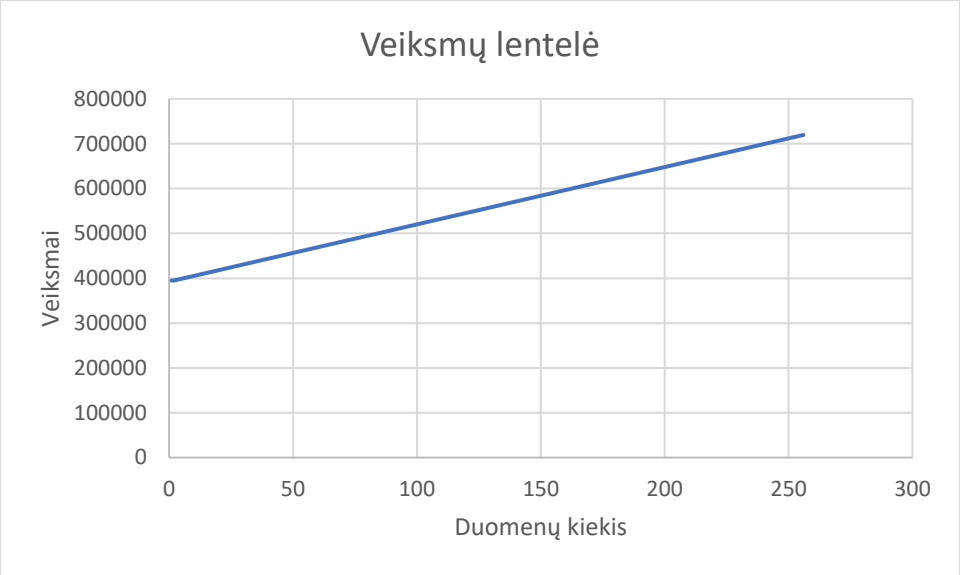
```
for (int i = 1; i < places.Count; i*=2)
{
    int count = i;
    List<long> visitedIds1 = new List<long>();
    List<long> visitedIds2 = new List<long>();
    long firstName = 5844429476;
    long secondName = 1013895687;
    long sk = 0;
    Stopwatch watch = Stopwatch.StartNew();
    watch.Start();
    TaskUtils.Solve(places, firstName, secondName, visitedIds1, visitedIds2, count, ref sk);
    watch.Stop();
    Console.WriteLine($"Trukmė: {watch.Elapsed.TotalSeconds}");
    Console.WriteLine($"Veiksmu skaičius: {sk}");

}
```

Metodui „Solve“ yra duodamas vis kitoks elementų skaičius, norint gauti tam tikras laiko ir veiksmų skaičių tendencijas.

Gautos lentelės pagal duomenis:





Pagal gautus rezultatus galima teigti, kad veiksmų skaičius ir laikas priklauso nuo O(n) asimptotinio sudėtingumo.

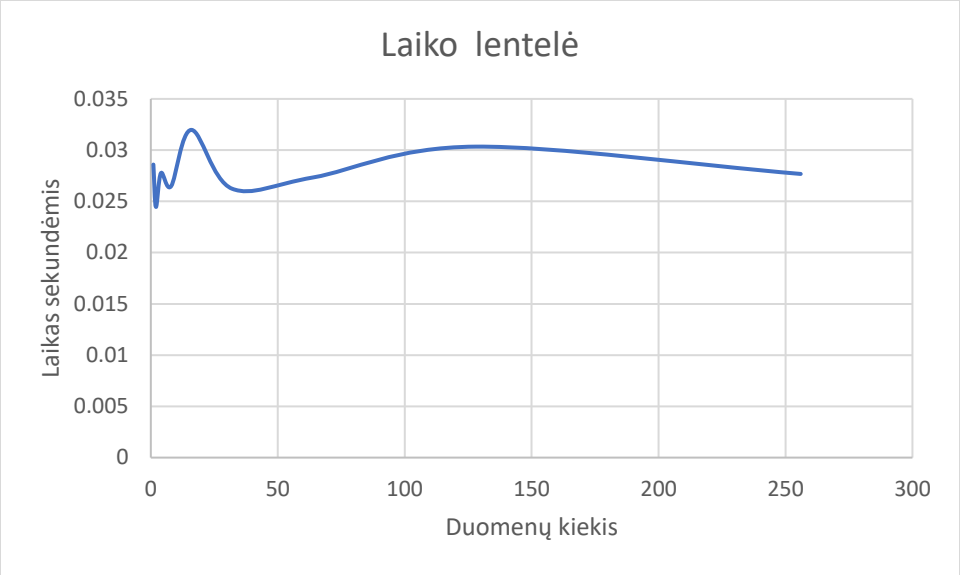
2. Keičiamos pradinės keliautojų vietovių reikšmės:

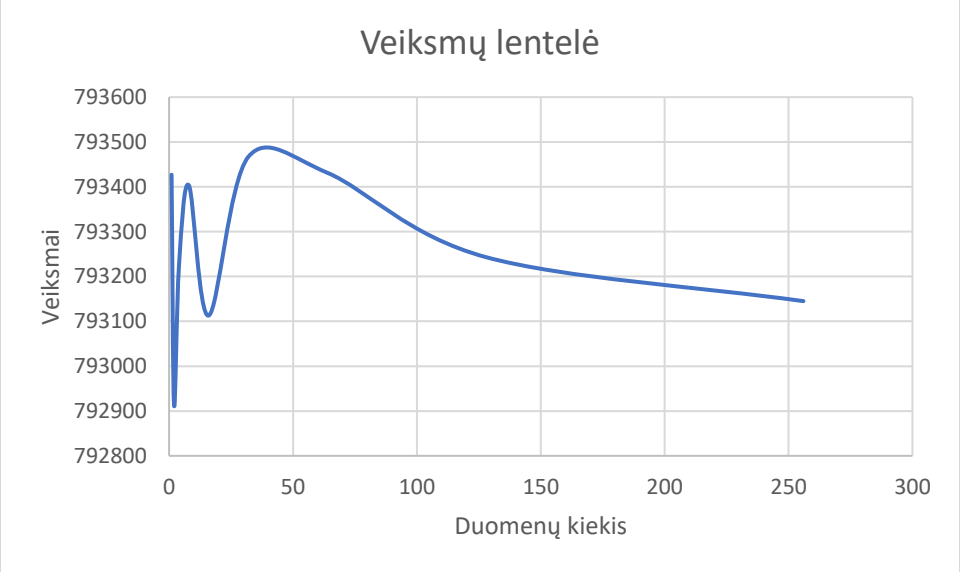
```
for (int i = 1; i < places.Count - 1; i*=2)
{
    List<long> visitedIds1 = new List<long>();
    List<long> visitedIds2 = new List<long>();
    long firstName = 5844429476;
    long secondName = 1013895687;
    long sk = 0;
    Stopwatch watch = Stopwatch.StartNew();
    watch.Start();
    TaskUtils.Solve(places, firstName, secondName, visitedIds1, visitedIds2, i, i + 1, ref sk);
    watch.Stop();
    //Console.WriteLine(watch.Elapsed.TotalSeconds);
    Console.WriteLine(sk);

}
```

Metodui „Solve“ yra duodami vis kitokie pradžios indeksai, kad būtų galima ištestuoti, kaip pasikeis veiksmų skaičius ir laikas priklausomai nuo pradinių keliautojų vietovių.

Gautos lentelės pagal duomenis:





Apibendrinus galima teigti, kad atsižvelgus į pradines vietoves pastebėti laiko ir veiksmų tendencijas yra sudėtinga, nes rezultatai labai skiriasi.

1.5. Testuojami rezultatai

Pagal IP_duomenys.txt testuojama programa.

Pirmas keliautojas pradės kelionę nuo Abromiškio atodanga (id: 4314994364) ir antras pradės kelionę nuo Akmeninė užtvanka (id: 3713676194)

Gauti rezultatai;

Pradzios vietove: Abromiskio atodanga, Aplankytu vietu skaicius: 158, Pilna kaina: 27782025211.30829
4314994364: Abromiskio atodanga
4314994341: Daumantu atodanga
4811244924: Pagojes tvenkinio uztvanka
2972634548: Vaizdas i Anykscius
4315141797: Gyliu atodanga
1796291688: Variaus atodanga
1355474516: Kupos upes slenio parkas
3832353383: Aukstupenu piliakalnis
3561937361: Girininko bokstelis
4558151069: Velniakampio atodanga
2059656225: Regykla ant ispudingo aukscio slaito
2665518099: Vaizdas i Galves ezera is Uzutrakio dvaro
1354078668: Ausiutiskiu regykla
2109098819: Karmazinu atodanga
792100839: Azuolu kalnas
4964629666: Verksioniu regykla
5172670279: Dukstu atodanga
1468775760: Kernaves apzvalgos aikstele
1468796966: Baltas kalnas
1659291458: Vaizdas i Merki
4549105391: Kartuviu kalnas
5028700985: Gelioniu kalva
2710734036: Ulos skardziu regykla
3984811657: Manciagires skardziai
2645268792: Vaizdas i Cepkeliu raista
5684711972: Ilgio ezero apzvalgos aikstele
1309813996: Puvociu (Omnitel) apzvalgos bokstas
2473433289: Vaizdas i pelke
2119876187: Bakanausko ezero apzvalgos aikstele
1687532290: Merkines piliakalnis
3178402508: Merkines apzvalgos bokstas
1688321196: Poteroniu piliakalnis
6075811579: Punios piliakalnio virsutine aikstele
1134203199: Alytaus piliakalnio apzvalgos aikstele
5178044397: Panemuninku skardis
1690106146: Birstono piliakalnis
5717687729: Kernuviu atodanga
4964629667: Skevoniu atodanga
2496012802: Skevoniu atodanga
5179618538: Pauksciu stebejimo vieta
312531257: Panorama
1688104974: Aukstuju Sanciu piliakalnis
1690291739: Lentainiu piliakalnis
2471915996: Jiesios atodanga

4571659367: Prisikelimo baznycios apzvalgos aikstele
1688078085: Eiguliu piliakalnis
2713017369: Kalnelis
1260064245: Vaizdas i Kauna is Milikoniu
5171030470: Jezuitu vienuolyno apzvalgos aikstele
1013895687: Aleksoto apzvalgos aikstele
2000463622: Nemuno ir Neries santaka
374121724: Balbieriskio atodanga
2579910797: Vaizdas aplink Papeciu piliakalni
2659338565: Meteliu apzvalgos bokstas
4348203490: Vaizdas i Ancios ezera
1746798593: Prelomciskes piliakalnis
2595529674: Zuvinto rezervato apzvalgos bokstas
6792766086: Sv. Jurgio baznycios apzvalgos aikstele
1733573451: Nemuno apzvalgos aikstele
3486143720: Apzvalgos aikstele ‚Gairiakalnis"
3662663128: Apzvalgos bokstelis Novaraiscio ornitologiniame draustinyje
3089770393: Seredziaus apzvalgos bokstelis
1688075575: Seredziaus piliakalnis I
4055210014: Siaules apylinkiu apzvalgos bokstas
1993447244: Saules musio vieta
3990113032: Pilsupiu atodanga
1962237461: Netoniu slaitas
1907459503: Vaizdas i Nemuna
4353564489: Snow Arena
5028701032: Miezoniu regykla
1686128731: Liskiavos piliakalnis
3027164894: Vaizdas i Pazaislio vienuolyna
718512601: Vaizdas i Kauno marias
1074248500: Lakstingalu slenis
833247708: Ziegzdriu atodanga
2803225316: Kalniskiu konglomeratu atodanga
3192087181: Draseikiu atodanga
1323938974: Gastiloniu atodanga
1314231552: Rumsiskiu apzvalgos bokstas
5740561866: Vaizdas i Kauno marias
382399179: Mergakalnio regykla
5179618536: Lasiniu konglomeratu atodanga
6403605800: Vaizdas i Neri
2375321445: Kopa
2375321446: Matelio reva
2375321447: Pageleziu miskas
3205944559: Vaizdas i Senvages ezera
2283843877: Levens apzvalgos aikstele
4312281699: Kalnelio misko regykla
5162560420: Murmuliu atodanga

2659304435: Krekenavos apzvalgos bokstas
4996806565: Nevezio upes slenio apzvalgos bokstelis
2712793129: Vaizdas i Kauno marias
2712793134: Vaizdas i Kauno marias nuo Pazaislio
5844429476: Apascios ir Nemunelio santaka
4947550831: Muoriskiu atodanga
4953135081: Tabokines atodanga
2108908928: Sviliskiu krastovaizdzio draustinio regykla
4515277514: TV bokstas
4166852260: Pramonine panorama
3421869471: Neries apzvalgos aikstele
2303146285: Vilniaus panorama
2470948927: Miesto panorama
4340144872: Tauro kalno panorama
1277407583: Vaizdas i Prezidentura
3651935479: Spaustuves kiemas
3651935481: M.K. Sarbievijaus kiemas
2471812239: Sv. Jonu baznycios varpine
674111736: Barbakano apzvalgos aikstele
5028214986: Verkiu dvaro parko apzvalgos aikstele
4448856990: Obelu laukyme
738082549: Puckoriu atodanga
2475356686: Tuputiskiu serpantinas
3949668360: Vaizdas nuo pilkapyno
3774775516: Rokantiskiu piliavietes regykla
5216159177: Liudgardo slaitas
3700633124: Mindunu apzvalgos bokstas
5838804285: Vyzuonos parko apzvalgos aikstele
3238695453: Atodanga
5695821857: Juozapines kalnas
2788401067: Juozapines kalno III regykla
963718797: Atodanga
866097561: Vaikeso ez. salos piliakalnis
437564602: Siliniskiu (Bites) apzvalgos bokstas
5003241197: Kaukiskes apzvalgos aikstele
1314771408: Vilkakalnio apzvalgos bokstas
4258220410: Apzvalgos vieta
5028701031: Staskines ezero regykla
4361505839: Saules puke
2788401061: Juozapines kalno II regykla
1416904471: Gedimino kapo kalno regykla
4402505089: Auksciausia Anyksciu apzvalgos aikstele
429331341: Vetygalos atodanga
726034642: Traku m. apzvalgos aikstele
2061136835: Paneriu silo regykla
2807219131: Kadagiu slenio panorama

4966875773: Nemunaicio atodanga
2659362312: Veisieju apzvalgos bokstas
4829246478: Dirvonu klonis
5735780281: Vaizdas i Aukstumalos pelke
976164591: Bokstelis su vaizdu i Kursiu marias
2837567137: Italiskas vaizdelis
3680502700: Pustlaukio duobes regykla
5178044407: Sveikatos tako apzvalgos aikstele
6371353705: Birstono apzvalgos bokstas
1690102715: Lepelioniu piliakalnis
1238977771: Vaizdas i Galves ezera
1690271333: Buivydu piliakalnis
686546498: Mikieriu atodanga
2928063927: Stirniskiu atodanga
866133511: Zaibiskiu II piliakalnis
5179618537: Raudonpamusio atodanga
3517028307: Apzvalgos boksteliai
2659282431: Kamanu apzvalgos bokstas
1917291702: Dyburiu atodanga
2756825217: Hageno kalnas
5028701034: Regykla ,Trys ezerai"
4314994364: Abromiskio atodanga

Pradzios vietove: Akmenine uztvanka, Aplankytu vietu skaicius: 158, Pilna kaina: 30869189250.541218
3713676194: Akmenine uztvanka
5028701023: Zasugalos kalnas
3631511904: Medvegalio piliakalnis
2219448197: Debesnu botaninio tako apzvalgos bokstelis
5551814772: Vaizdas i Juros upes sleni
5028700202: Akmenos apzvalgos aikstele
2370216014: Pagramancio atodanga
3436031886: Geniu Juros Atodanga
3449141499: Apzvalgos bokstas
3448572815: Vilkyskiu apzvalgos bokstas
4968605721: Sereitlaukio apzvalgos bokstas
5028700215: Merguvos ezero apzvalgos bokstas
5028214987: Bitenu apzvalgos aikstele
3013040133: Nemuno apzvalgos aikstele
3013040135: Nemuno apzvalgos aikstele
280777081: Rambynas
5028700987: Bardenu apzvalgos bokstas
5028700967: Jogaudu apzvalgos bokstas
472455729: Paplateles tako apzvalgos bokstas
5028699371: Paplateles apzvalgos aikstele
5028700986: Bokstas prie Barstytaliu

5732083226: Vaizdas i Plateliu ezera
5732083281: Vaizdas i Piktezeri
1930144634: Plateliu ezero apzvalgos aikstele
2435263457: Plunges Lurdas
2993013534: Salantu parkas
2349902660: Zvainiu Gaidzio koplycia
1316308655: Kalnalio apzvalgos bokstas
4952741196: Dauginciu atodanga
2068513657: Aukstumalos pazintinio tako platforma
5542271251: Apzvalgos aikstele ant Tulkiarages siurblines stogo
5549887839: Bokstelis su vaizdu i Kroku lanka
2068544803: Kroku lankos stebykla
3010827377: Rusnes regykla
474708837: Rusnes apzvalgos bokstas
4485539290: Polderio bokstelis
4485532803: Kintu apzvalgos bokstelis
3453976476: Drevernos apzvalgos bokstas
2068593691: Apzvalgos bokstelis
976164581: Ventes rago svyturys
5028214988: Juodkrantes apzvalgos aikstele
303908102: Karvaiciu apzvalgos aikstele
2704421297: Avikalnio ragas
2756927419: Nagliu ragas
475255988: Pilkieji garniai ir kormoranai
1118427594: Avikalnio apzvalgos aikstele
6644356767: Vaizdas i Kursiu marias
2756825220: Meskos galva
2756825218: Neringos kalnas
5178044391: Apzvalgos aikstele senojoje perkeloje
4320515370: Sky21 apzvalgos aikstele
2320296301: Jono kalnelis
5689446695: Vaizdas i Baltijos jura
4319222673: Olando kepures skardis
475305601: Plocio ezero pauksciu stebejimo namelis
3800898681: Nemirsetos kopu ir smiltpieviu apzvalgos aikstele
5844602772: Palangos baznycios bokstas
1251739821: Palangos tiltas
288085010: Negyvosios kopos
5028700205: Pervalkos ragas
4285830489: Pauksciu stebejimo taskas
2756927416: Birstvyno ragas
5028700609: Zirgu ragas
3343740219: Ozku ragas
3343687924: Menininku kopa
5776808483: Vaizdas i Vecekrugo kalna
2111293780: Vecekrugo kalnas

5793598068: Vecekrugo apzvalgos bokstelis
2756927300: Parnidzio ragas
2756927304: Mariu Mergelio atodanga
5839452986: Vaizdas i Parnidzio kopa
5834155488: Urbo kalnas
4692081010: Apzvalgos bokstas
2837519165: Parnidzio kopa
1444952227: Lurdas
1997471637: Ipilties saltinio akmuo
5588109230: Artosios aukstapelkes apzvalgos aikstele
1685048533: Satrijos kalnas
4964546709: Svirkanciu atodanga
3491500010: Juodeikiu apzvalgos aikstele
1316303149: Ventos ir Virvytes santakos apzvalgos bokstas
3498985419: Vaizdas i Virvycios upe
1631476416: Avizlio atragis
2621993177: Didysis akmuo
1628529558: Jurakalnio griovos atodanga ir atragis
3446718472: Jurakalnio apzvalgos bokstas
5216114518: Raudonskardzio atodanga
2103856550: Sviles kryziu kryzkele
3373025183: Apzvalgos bokstelis
4113995490: Dubysos-Ventos perkaso regykla
5028701026: Rimo kalnelis
5028699376: Naisiu kaimo apzvalgos bokstelis
5028214989: Apusio apzvalgos aikstele
6646672505: Giliaus ezero panorama
3372998255: Tytuvenu (Kokmaniskes / Majako kln.) apzvalgos bokstas
5544859119: Apzvalgos aikstele salia Plokściu baznycios
5028701033: Regykla ‚Voras"
3491133889: Zvilgsnis i Rominta
5028701024: Skardis ir Vistycio vaizdas
3884679871: Plikakalnio atodanga
6700273919: Apzvalgos aikstele
5554620085: Miesto panorama
3169074463: Riverfront terasa
2022625859: Terasa
3651935480: Observatorijos kiemas
2347023774: Mikalojaus Dauksos kiemas
3651935478: Petro Skargos kiemas
2837707104: Gedimino pilis
477399208: Subaciaus g. apzvalgos aikstele
766866892: Laimio kalnas
2484428163: Zaliuju regykla
3068159618: Rokantiskiu regykla
4880667870: Panorama i miesta ir Puckoriu sleni

1718008073: Sakiskiu skardis
1758423764: Papiskiu piliakalnis
3603784446: Vitku seimos koplycia
866079049: Vosgeliu piliakalnis
4397056176: Ilzenbergo dvaro Apzvalgos aikstele
1690325421: Liesenu piliakalnis
831207869: Rubikiu apzvalgos bokstas
3296721649: Gaujos mokomojo pazintinio tako bokstelis
1670019641: Atodanga
5910951648: Sartu apzvalgos bokstas
3751956238: Ladakalnis
2709657798: Lygumu apzvalgos bokstas
1309957971: Svencioniu apzvalgos bokstas
4258095517: Apzvalgos ratas
4851615022: Pusu aleja
4996536359: Salako pauksciu stebejimo bokstelis
3153759720: Kuckuriskiu piliakalnis
1731566102: Vilniaus senamiescio vaizdas
4411249001: Nuostabus saulelydziai ir Pagojes tvenkinys
2513516321: Vaizdas i Piliskiu ezera
1131223985: Traku pilies panorama
1309875494: Cepkeliu raisto apzvalgos bokstas
2803225205: Ziegzdriu atodanga II
1907459532: Vaizdas i Nemuna
5177967378: Vosbuciu atodanga
5178044390: Siponiu atodanga
2659499587: Sirvetos apzvalgos bokstas
2756825216: Ledu ragas
3343662439: Bulvikio ragas
2626352567: Upes slenkstis
2626282736: Vaizdas i miskus
2590491615: Stalo kalno regykla
308878465: Triju kryziu apzvalgos aikstele
2045876096: Paneriskiu skardis
4966875764: Andruskoniu atodanga
2501859774: Raudones pilies bokstas
4964546626: Skaudviles atodanga
2370221555: Akmenos - Juros upiu santaka
1316287393: Aukstagires apzvalgos bokstas
3872499795: Apzvalgos aikstele salia Telsiu baznycios
2638244459: Siberijos (Cidabro kalno) apzvalgos bokstas
2821328850: Kartenos piliakalnis
5028701025: Sereiklaukio misko apzvalgos bokstas
5028699395: Jociu-Alangos atodangu regykla
3713676194: Akmenine uztvanka

Trukme: 0.3837755

1.6. Programos kodas

Location.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IP
{
    internal class Location
    {
        public string Name { get; set; }

        public long Id { get; set; }

        public double X { get; set; }

        public double Y { get; set; }

    }
}
```

InOutUtils.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IP
{
    internal class InOutUtils
    {
        public static List<Location> ReadPlacesFromTextFile(string fileName)
        {
            List<Location> places = new List<Location>();
            using (StreamReader reader = new StreamReader(fileName))
            {
                reader.ReadLine();
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] values = line.Split('\t');
                    string name = values[0];
                    long id = long.Parse(values[1]);
```

```
                    double x = double.Parse(values[2]);
                    double y = double.Parse(values[3]);
                    places.Add(new Location { Name = name, Id = id, X = x, Y = y });
                }
            }
            return places;
        }

        public static void PrintResults(List<Location> places, List<long>
visitedIds, double totalCost)
        {
            Console.WriteLine($"Aplankytu vietu skaicius: {visitedIds.Count}, Pilna
kaina: {totalCost}");
            foreach (long id in visitedIds)
            {
                Location place = places.First(p => p.Id == id);
                if (place != null)
                {
                    Console.WriteLine($"{id}: {place.Name}");
                }
                else
                {
                    Console.WriteLine($"Negalima surasti vietos pagal
identifikatoriu {id}");
                }
            }
            Console.WriteLine();
        }
    }
}
```

TaskUtils.cs:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace IP
{
    internal class TaskUtils
    {
             public static double CalculateCost(List<Location> places, List<long>
visitedIds)
        {
            double cost = 0.0;
            for (int i = 0; i < visitedIds.Count - 1; i++)
            {
                long id1 = visitedIds[i];
                long id2 = visitedIds[i + 1];
                Location place1 = places.First(p => p.Id == id1);
                Location place2 = places.First(p => p.Id == id2);
                if (place1 != null && place2 != null)
                {
```

```
                double dx = place2.X - place1.X;
                double dy = place2.Y - place1.Y;
                cost += Math.Sqrt(dx * dx + dy * dy);
            }
        }
        return cost;
    }

    public static void Solve(List<Location> places, long nameOfLocation1, long
nameOfLocation2, List<long> visitedIds1, List<long> visitedIds2)
    {
        double[,] distances = new double[places.Count, places.Count];
        for (int i = 0; i < places.Count; i++)
        {
            for (int j = 0; j < places.Count; j++)
            {
                double dx = places[j].X - places[i].X;
                double dy = places[j].Y - places[i].Y;
                distances[i, j] = Math.Sqrt(dx * dx + dy * dy);
            }
        }

        int placeId1 = places.FindIndex(p => p.Id == nameOfLocation1);
        visitedIds1.Add(places[placeId1].Id);

        int placeId2 = places.FindIndex(p => p.Id == nameOfLocation2);
        visitedIds2.Add(places[placeId2].Id);

        while (visitedIds1.Count + visitedIds2.Count < places.Count)
        {

            long nextId1 = NextId(visitedIds1, visitedIds2, places, distances);
            if (nextId1 != -1)
            {
                visitedIds1.Add(nextId1);
            }

            long nextId2 = NextId(visitedIds2, visitedIds1, places, distances);
            if (nextId2 != -1)
            {
                visitedIds2.Add(nextId2);
            }

        }
        visitedIds1.Add(places[placeId1].Id);
        visitedIds2.Add(places[placeId2].Id);
    }

    private static long NextId(List<long> visitedIds, List<long>
otherVisitedIds, List<Location> places, double[,] distances)
    {
        long nextId = -1;
        double minCost = double.MaxValue;
        long lastId = visitedIds.Last();
        int lastIdx = places.FindIndex(p => p.Id == lastId);

        for (int i = 0; i < places.Count; i++)
        {
```

```
                if (visitedIds.Contains(places[i].Id) ||
otherVisitedIds.Contains(places[i].Id))
                {
                    continue;
                }
                double cost = distances[lastIdx, i];
                if (cost < minCost)
                {
                    nextId = places[i].Id;
                    minCost = cost;
                }

            }

            return nextId;
        }
    }
}
```

Program.cs:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IP
{
    internal class Program
    {
        const string InputFile = "IP_duomenys.txt";
        public static void Main(string[] args)
        {
            Stopwatch sw = Stopwatch.StartNew();
            sw.Start();
            List<Location> places = InOutUtils.ReadPlacesFromTextFile(InputFile);

            List<long> visitedIds1 = new List<long>();
            List<long> visitedIds2 = new List<long>();
            long firstName = 4314994364;
            long secondName = 3713676194;

            TaskUtils.Solve(places, firstName, secondName, visitedIds1,
visitedIds2);

            Console.Write($"Pradžios vietovė: {places[places.FindIndex(p => p.Id ==
firstName)].Name}, ");
            InOutUtils.PrintResults(places, visitedIds1,
TaskUtils.CalculateCost(places, visitedIds1));

            Console.Write($"Pradžios vietovė: {places[places.FindIndex(p => p.Id ==
secondName)].Name}, ");
            InOutUtils.PrintResults(places, visitedIds2,
TaskUtils.CalculateCost(places, visitedIds2));
```

```
            sw.Stop(); Console.WriteLine($"Trukmė: {sw.Elapsed.TotalSeconds}");
        }
    }
}
```