KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Algoritmų sudarymas ir analizė

Laboratorinis darbas nr. 2

Atliko:
IFF-8/2 gr. studentas
Nedas Šimoliūnas
2020 m. gegužės 12 d.
Priėmė:
Doc. Dalius Makackas

KAUNAS 2020

TURINYS

Algoritmų sudarymas ir analizė..... 1

1. Uždavinys (nr. 1) 3

1.1. Programos kodo sudarymas ir sudėtingumo apskaičiavimas 3

1.2. Eksperimentinis algoritmų sudėtingumo įvertinimas 5

2. Užduotis 6

2.1. Rekurentinės formulės sudarymas 6

2.2. Sprendimo iliustravimas ir sudėtingumo skaičiavimas 6

2.3. Dinaminio programavimo uždavinio programos kodo sudarymas ir sudėtingumo apskaičiavimas 8

2.4. Eksperimentinis algoritmų sudėtingumų įvertinimas 9

3. Užduotis 10

4. Išvados 11

1. Uždavinys (nr. 1)

Duota: $x_1, x_2, \ldots, x_m$, ir $y_1, y_2, \ldots, y_n$.

$$F(m,n) = \begin{cases} m, & jei\ n = 0; \\ n, & jei\ m = 0, n > 0; \\ \min\{1 + F(m-1,n), 1 + F(m,n-1), D(m,n) + F(m-1,n-1)\} & kitais\ atvejais. \end{cases}$$

kur $D(i,j) = \begin{cases} 1 & jei\ x_i = y_j; \\ 0 & kitais\ atvejais. \end{cases}$

1.1. Programos kodo sudarymas ir sudėtingumo apskaičiavimas

Rekursinis atvejis:

```
                                                    Kaina       Kiekis
static int F(int m, int n)                          T(m,n)
    {
      int returnValue;                              c1            1
      if (n == 0)                                   c2            1
      {
        returnValue = m;                            c3            1
      }
      else if (m == 0 && n >
0)                                                  c4            1
      {
        returnValue = n;                            c5            1
      }
      else
      {
        int value1 = 1 + F(m - 1, n);               T(m-1,n)      1
        int value2 = 1 + F(m, n - 1);               T(m,n-1)      1
        int value3 = D(m, n) + F(m - 1, n - 1);     T(m-1,n-1)    1
        returnValue = Least(value1, value2,
value3);                                            c6            1
      }
      return returnValue;                           c7            1
    }
```

T(m,n) = T(m-1,n) + T(m,n-1) + T(m-1,n-1) + C

T(m,n) = O(3^min(m,n))

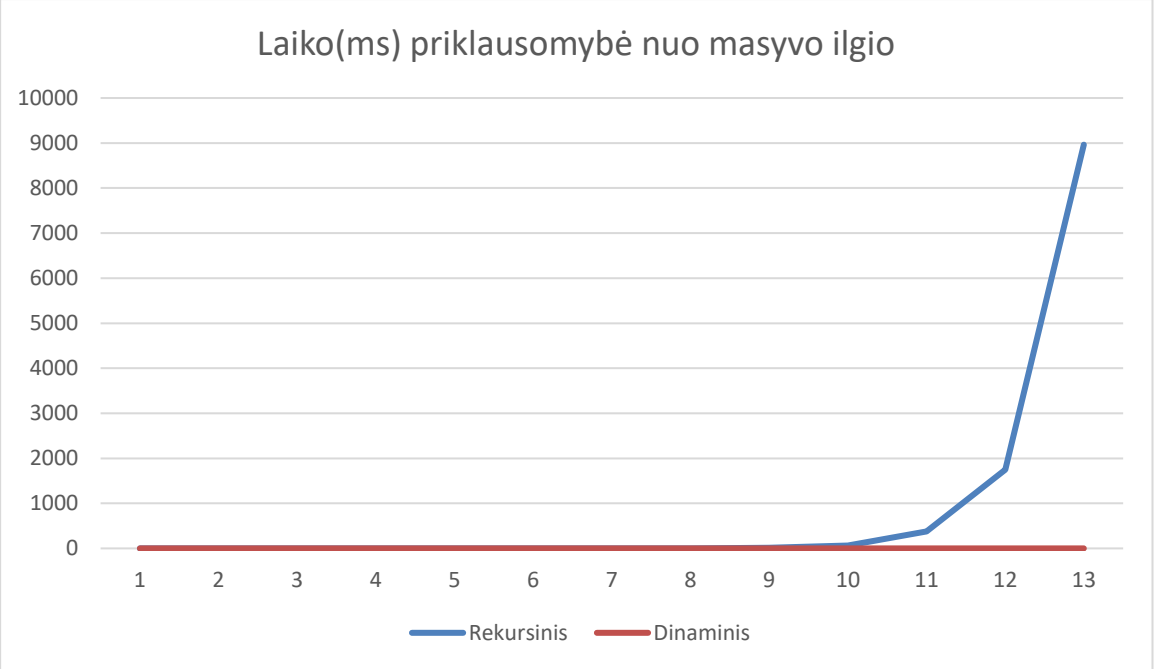
Dinaminio programavimo atvejis:

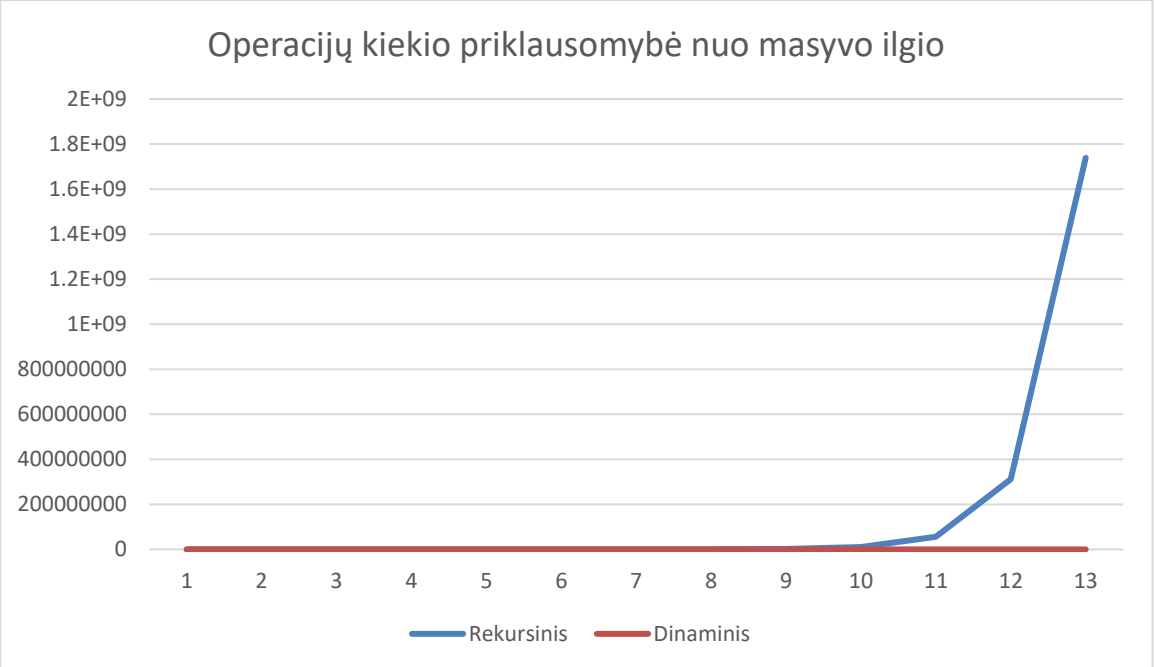
```
                                                                  Kaina        Kiekis
static int FDin(int m, int n)                                     T(m,n)
    {
      for (int i = 0; i <= m;
i++)                                                              c1           m+2
      {
        din[0, i] =
i;                                                                c2           m+1
      }
      for (int i = 0; i <= n; i++)                                c3           n+2
      {
        din[i, 0] =
i;                                                                c4           n+1
      }
      for (int i = 1; i <= m;
i++)                                                              c5           m+1
      {
        for (int j = 1; j <= n;
j++)                                                              c6           m(n+1)

{
            int value1 = din[i - 1, j] + 1;                       c7           mn
            int value2 = din[i, j - 1] + 1;                       c8           mn
            int value3 = din[i - 1, j - 1] + D(i, j);             c9           mn
            din[i, j] = Least(value1, value2, value3);            c10          mn

}
      }
      return din[m, n];                                           c11                1
    }
```

T(m,n) = C1*mn + C2*m + C3*n + C4

T(m,n) = O(mn)

1.2. Eksperimentinis algoritmų sudėtingumo įvertinimas





2. Užduotis

Turime ***n*** daiktų, kurių svoriai yra $s_1, s_2, \ldots, s_n$, o kaina $p_1, p_2, \ldots, p_n$. Reikia rasti daiktų rinkinio didžiausią vertę, kad rinkinio svoris neviršytų ***W***

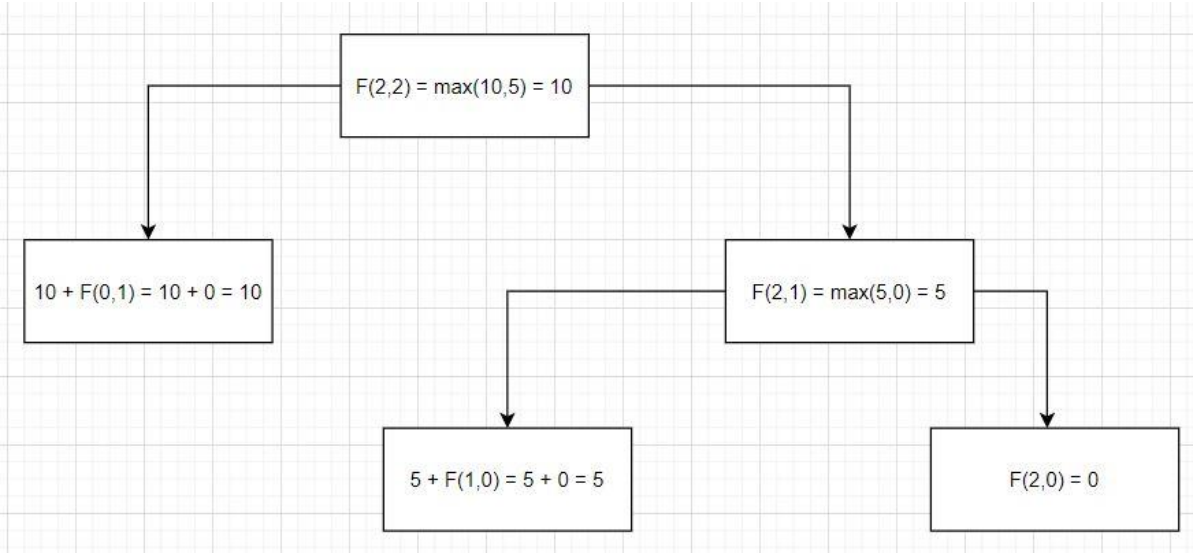
2.1. Rekurentinės formulės sudarymas

Turime svorių masyvą *s* ir kainų masyvą *p*.

$$F(w,n) = \begin{cases} 0, kai\ n = 0\ arba\ w = 0 \\ F(w, n-1), kai\ s[n-1] > w \\ \max\big(p[n-1] + F(w - s[n-1], n-1)\ , F(w, n-1)\big), kitais\ atvejais \end{cases}$$

2.2. Sprendimo iliustravimas ir sudėtingumo skaičiavimas

s = [1,2] , p = [5,10] , w = 2, n = 2



	Kaina	Kiekis
static int F(int w, int[] s, int[] p, int n)	T(w,n)	
{		
if(n == 0 || w == 0)	c1	1
{		
return 0;	c2	1
}		
if (s[n-1] > w)	c3	1
{		
return F(w, s, p, n - 1);	T(w,n-1)	1
}		
else		
{		
return max(p[n - 1] + F(w - s[n - 1], s, p, n - 1), F(w, s, p, n - 1));	c5 + T(w-s[n-1], n-1) T(w, n-1)	1
}		
}		

T(w,n) = Ω(1)

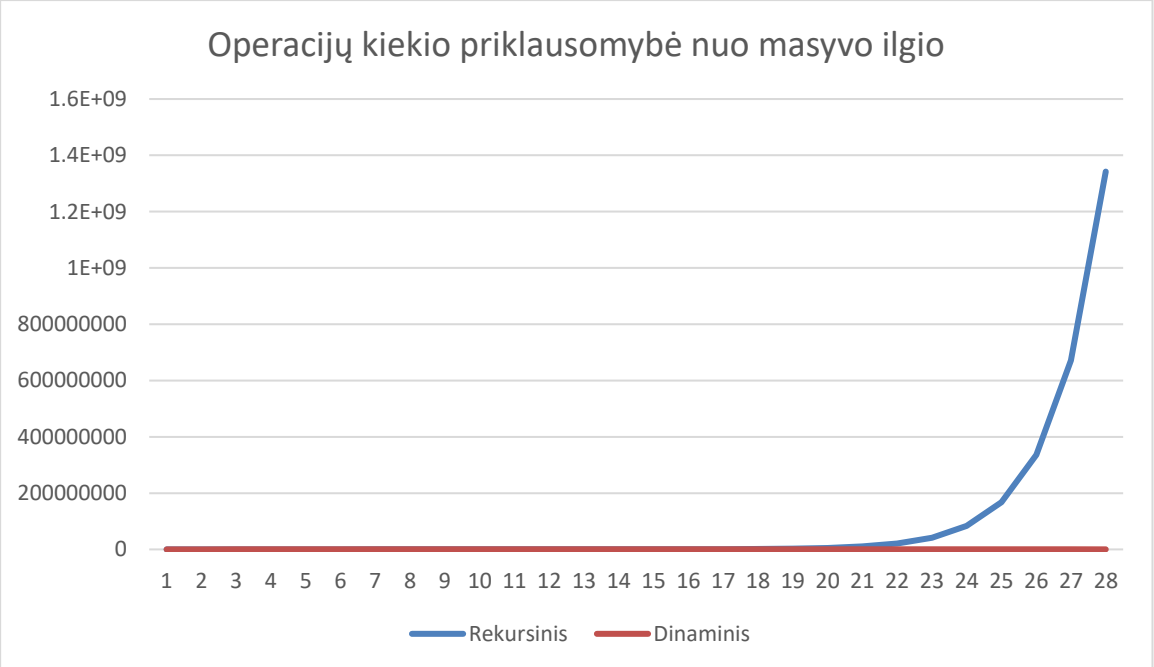
T(w,n) = O(2^n)

2.3. Dinaminio programavimo uždavinio programos kodo sudarymas ir sudėtingumo apskaičiavimas

	Kaina	Kiekis
static int FDin(int w, int[] s, int[] p, int n)	T(w,n)	
{		
int[,] din = new int[n + 1, w + 1];	c1	1
for (int i = 0; i <= n; i++)	c2	n+2
{		
for (int j = 0; j <= w; j++)	c3	(n+1)(w+2)
{		
if (i == 0 || j == 0)	c4	(n+1)(w+1)
{		
din[i, j] = 0;	c5	n+w+1
}		
else if (s[i-1] > j)	c6	(n+1)(w+1)
{		
din[i, j] = din[i - 1, j];	c7	O(nw)
}		
else		
{		
int value1 = p[i - 1] + din[i - 1, j - s[i - 1]];	c8	O(nw)
int value2 = din[i - 1, j];	c9	O(nw)
din[i, j] = max(value1, value2);	c10	O(nw)
}		
}		
}		
return din[n, w];	c11	1
}		

T(w,n) = O(nw)

2.4. Eksperimentinis algoritmų sudėtingumų įvertinimas



3. Užduotis

Panaudojus 1 uždavinyje duotą rekurentinę formulę realizuoti jai algoritmą tiesiogiai panaudojant rekursiją bei lygiagretų programavimą.

```
static int F2(int m, int n)
{
    int returnValue;
    if (n == 0)
        return m;
    if (m == 0 && n > 0)
        return n;

    int countCPU = 3;
    Task[] tasks = new Task[countCPU];
    for (int j = 0; j < countCPU; j++)
    {
        tasks[j] = Task.Factory.StartNew(
            (Object p) =>
            {
                var data = p as CustomData; if (data == null) return;
                data.TResult = F1(j == 1 ? m : m - 1,
                                  j == 0 ? n : n - 1);
            },
            new CustomData() { TNum = j });
    }

    Task.WaitAll(tasks);
    returnValue =
        Least((tasks[0].AsyncState as CustomData).TResult + 1,
              (tasks[1].AsyncState as CustomData).TResult + 1,
              (tasks[2].AsyncState as CustomData).TResult + D(m, n));

    return returnValue;
}
```

4. Išvados

Įvykdytos trys užduotys: dviejų uždavinių realizacija rekursiniu ir dinaminiu metodais, bei vieno uždavinio įgyvendinimas lygiagretaus programavimo būdu. Rekursiniais atvejais algoritmų sudėtingumai tampa eksponentiniais, o dinaminių atvejų sudėtingumai – polinominiai. Tai lėmė, kad dinaminių metodų atveju užduotys atliekamos daug greičiau nei rekursiniais, tačiau pastarajame papildomai nenaudojama atmintis įsiminti dalinius rezultatus. Lygiagretaus programavimo atveju rekursinį uždavinį pavyko išskaidyti į tris dalis, kurias programa atliko vienu metu. Tai lėmė greitesnį užduoties atlikimą nei paprastu rekursiniu atveju.