

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Algoritmų sudarymas ir analizė

2 Laboratorinis darbas

Atliko:

IF 8/1 grupės stud.

Tomas Odinas

Priėmė

doc. Dalius Makackas

KAUNAS, 2020

TURINYS

[1. Užduotis 2](#_Toc38655443)

[2. Rekurentinės lygties algoritmas realizuotas panaudojant rekursiją 2](#_Toc38655444)

[4. Eksperimentinis algoritmų sudėtingumo įvertinimas 3](#_Toc38655445)

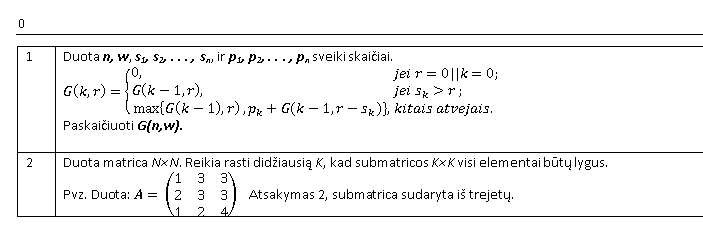
[Kai elementai saugomi operatyvioje atmintyje: 3](#_Toc38655446)

[Kai elementai saugomi diskinėje atmintyje: 4](#_Toc38655447)

[5. Išvados 5](#_Toc38655448)

[6. Priedas 6](#_Toc38655449)

# 1 Uždavinys



# Rekurentinės lygties algoritmas realizuotas panaudojant rekursiją:

**static** **int** **Ga**(**int** k, **int** r)

{

**if** (r == **0** || k == **0**) **return** **0**;

**if** (s[k - **1**] > r) **return** Ga(k - **1**, r);

**else** **return** Math.Max(Ga(k - **1**, r), p[k - **1**] + Ga(k - **1**, r - s[k - **1**]));

}

# Algoritmo sudėtingumas:

Geriausiu atveju: kai r = arba k = 0, tada

Blogiausiu atveju:

Sprendžiant medžio metodu gauname

Rekurentinės lygties algoritmas realizuotas panaudojant savybę, kad galime įsiminti dalinių sprendinių vertes

**static** **int** **Gb**(**int** k, **int** r)

{

**if** (r == **0** || k == **0**) **return** **0**;

**if** (cache[k - **1**, r - **1**] != -**1**) **return** cache[k - **1**, r - **1**];

**if** (s[k - **1**] > r)

{

cache[k - **1**, r - **1**] = Gb(k - **1**, r);

**return** cache[k - **1**, r - **1**];

}

**else**

{

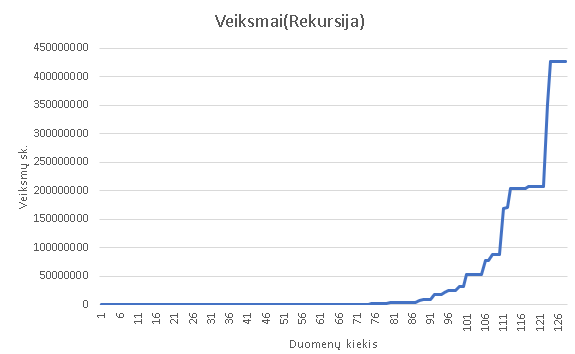
cache[k - **1**, r - **1**] = Math.Max(Gb(k - **1**, r), p[k - **1**] + Gb(k - **1**, r - s[k - **1**]));

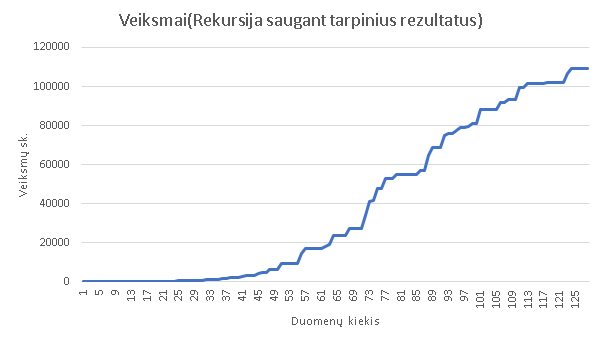
**return** cache[k - **1**, r - **1**];

}

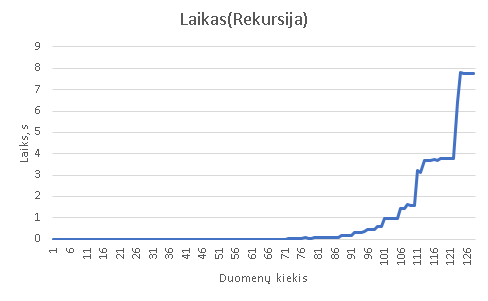
}

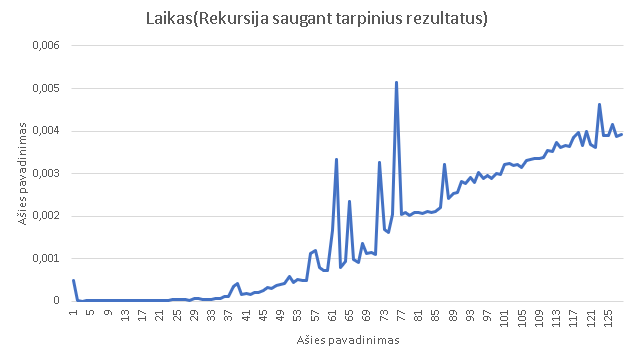
## Eksperimentinis algoritmų sudėtingumo įvertinimas





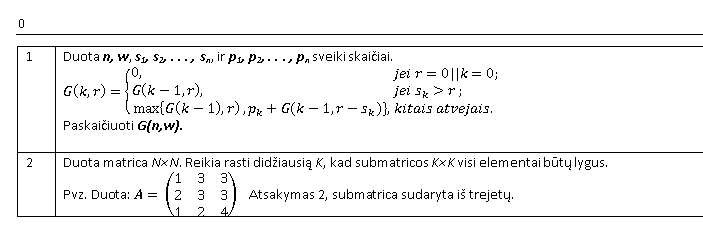
|  |  |  |
| --- | --- | --- |
| Kiekis | Rekursija (veiksmai) | Rekursija įsimenant reikšmes (veiksmai) |
| 2 | 3 | 3 |
| 4 | 9 | 9 |
| 8 | 80 | 80 |
| 16 | 88 | 88 |
| 32 | 912 | 888 |
| 64 | 95244 | 23476 |
| 128 | 426710369 | 109108 |



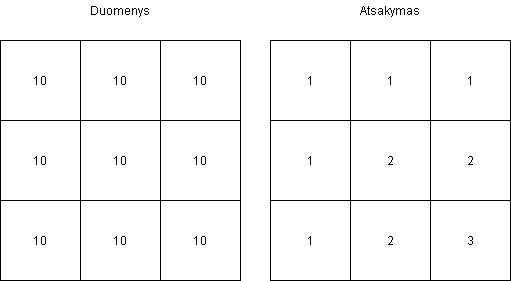


|  |  |  |
| --- | --- | --- |
| Kiekis | Rekursija (sekundės) | Rekursija įsimenant reikšmes (sekundės) |
| 2 | 0,0000082 | 0,0000066 |
| 4 | 0,0000126 | 0,0000089 |
| 8 | 0,0000148 | 0,0000162 |
| 16 | 0,0000086 | 0,0000085 |
| 32 | 0,0000346 | 0,0000445 |
| 64 | 0,0016873 | 0,0009221 |
| 128 | 7,7441405 | 0,003911 |

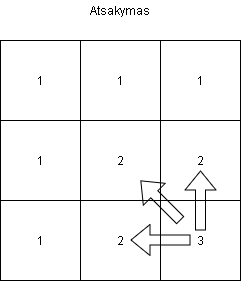
# 2 Uždavinys



Atsakymui formuoti išsiskirsime dydžio matricą, kur kiekvienas elementas saugos K reikšmę tokią, kad matricos visi elementai vienodi ir šis elementas bus apatinis dešinys. Pvz.:



Atsakymo (x,y) elementas priklausys nuo Ats(x-1,y), Ats(x,y-1), Ats(x-1,y-1) elementų



Uždavinio sprendinys ir bus didžiausias Ats matricos elementas.

Rekurentinė lygtis atsakymui formuoti:

Programos pseudokodas:

RastiK(m[])

{

// pirmos eilutes ar stulpelio elementai

// sudarys tik 1x1 kvadratus

Ats[**0**,] = Ats[,**0**] = **1**;

**for**(eilutes i **in** m)

**for**(stulpeliai j **in** m)

{

// jei 4 vienodi

**if**(m[i, j] = m[i - **1**, j] = m[i, j - **1**] = m[i - **1**, j - **1**])

Ats[i, j] = min(Ats[i - **1**, j], Ats[i, j - **1**], Ats[i - **1**, j - **1**]) + **1**;

}

**return** max(Ats[]);

}

Algoritmo sudėtingumas:

# 5. Išvados

# 6. Priedas

Nuoroda į programinio kodo saugyklą: <https://github.com/algoritmu-sudarymas-ir-analize/L2>