Kauno technologijos universitetas

Informatikos fakultetas

Antro laboratorinio darbo ataskaita

2LD individuali užduotis

Aistis Jakutonis

Studentas

Dalius Makackas
Andrius Kriščiūnas
Tadas Kraujalis
Vidmantas Rimavičius

Dėstytojai

Kaunas, 2025

Turinys

1. Pateikto programinio kodo metodo analizė

1.1. Apskaičiuojamas metodo asimptotinis sudėtingumas

1 lentelė Suskaičiuojamas kodo asimptotinis sudėtingumas

Kodas	Laikas	Kartai
`public static long methodToAnalysis (int[] arr)`		
`{`		
`    long n = arr.Length;`	c_1	1
`    long k = n;`	c_2	1
`    for (int i = 0; i < n*2; i++)`	c_3	$2n-0+1=2n+1$
`    {`		
`        for (int j = 0; j < n/2; j++)`	c_4	$2n*\left(\frac{n}{2}-0+1\right)=n^2+1$
`        {`		
`            k -= 2;`	c_5	$2n*\frac{n}{2}=n^2$
`        }`		
`    }`		
`    return k;`	c_6	1
`}`		

$$T_{methodToAnalysis}(arr)=c_1+c_2+c_3*(2n+1)+c_4*(n^2+1)+c_5*(n^2)+c_6=C+c_3 2n+c_4 n^2+c_5 n^2=2n^2+c_3 2n+C$$

$$T_{methodToAnalysis}(n)=O(n^2)$$

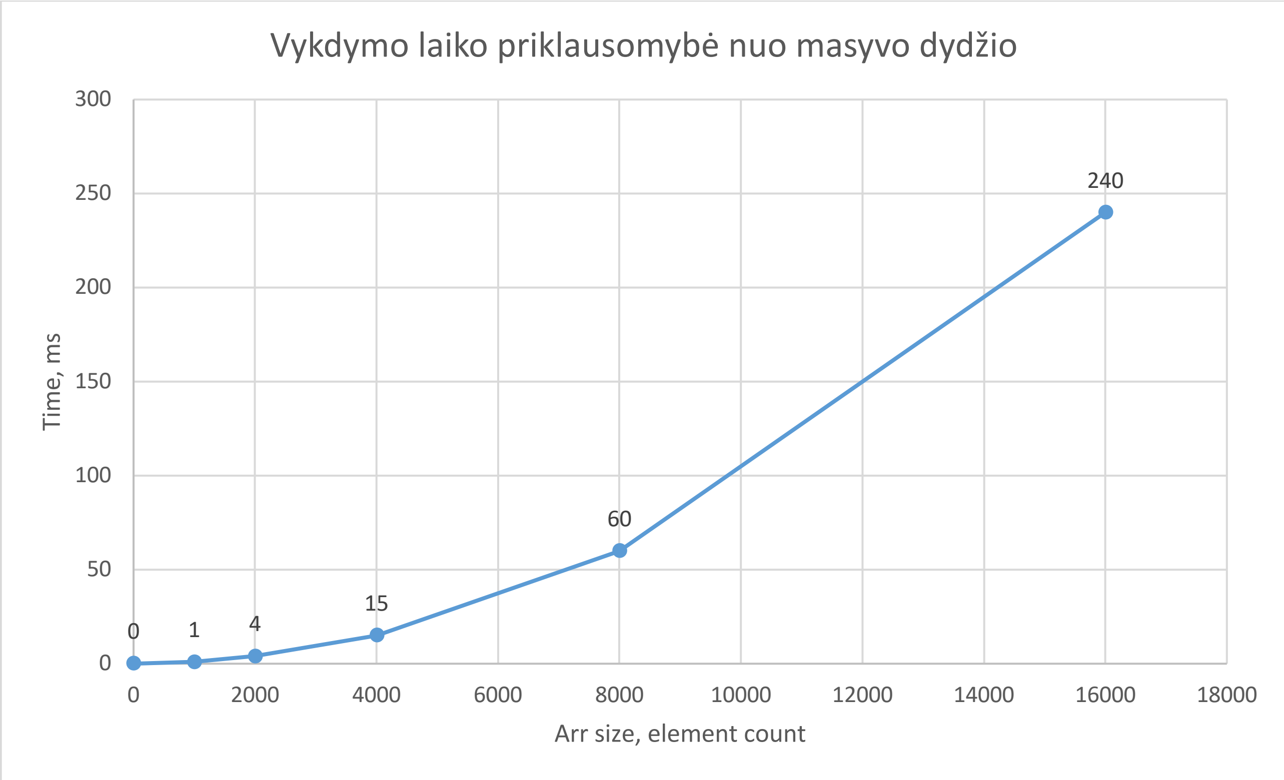
1.2. Atlikti eksperimentinį tyrimą

1.2.1. Vykdymo laiko priklausomybė nuo masyvo dydžio

2 lentelė Vykdymo laiko priklausomybės nuo masyvo dydžio grafiko duomenys

Arr size, element count	Time, ms
0	0
1000	1
2000	4
4000	15
8000	60
16000	240

1 grafikas Vykdymo laiko priklausomybė nuo masyvo dydžio

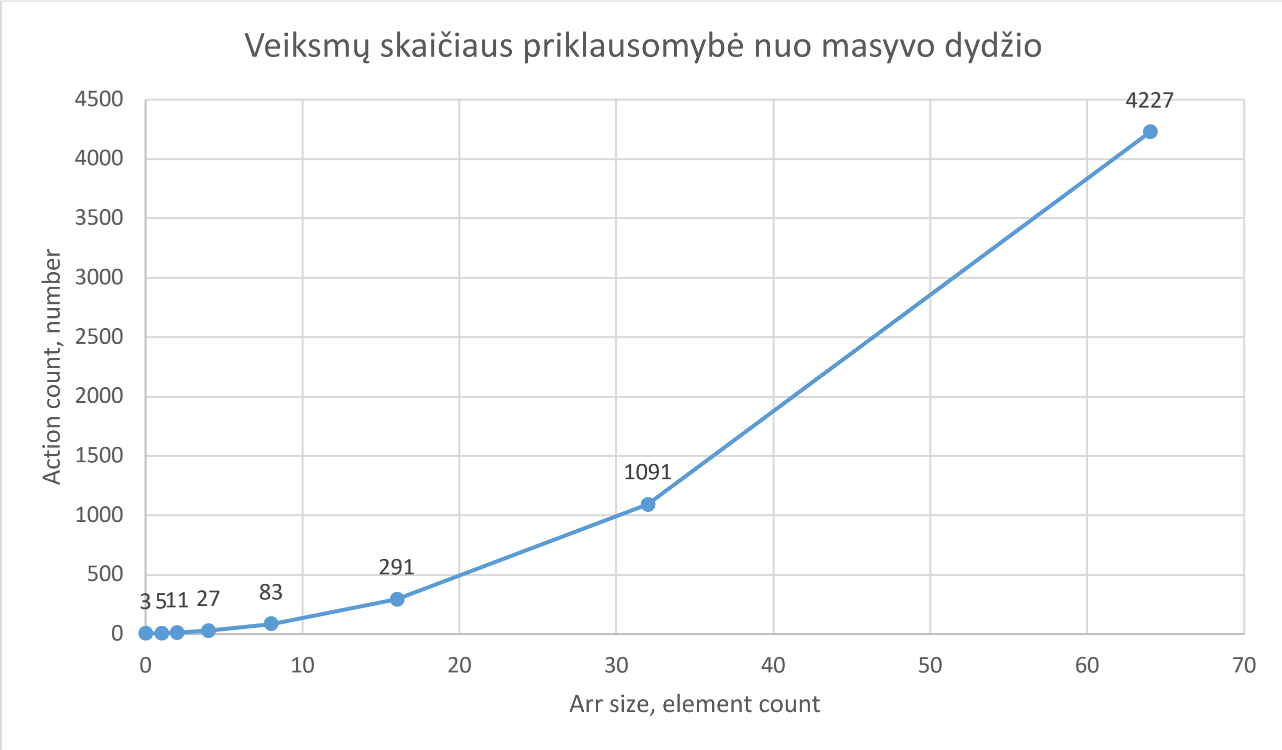


1.2.2. Veiksmų skaičiaus priklausomybė nuo masyvo dydžio

3 lentelė Veiksmų skaičiaus priklausomybės nuo masyvo dydžio grafiko duomenys

Arr size, element count	Action count, number
0	3
1	5
2	11
4	27
8	83
16	291
32	1091
64	4227

2 grafikas Veiksmų skaičiaus priklausomybė nuo masyvo dydžio



1.3. Patikrinti ar apskaičiuotas asimptotinis sudėtingumas atitinka eksperimentinius rezultatus

4 lentelė Masyvo dydžio ir vykdymo laiko duomenys

Arr size, element count	Time, ms
0	0
1000	1
2000	4
4000	15
8000	60
16000	240

Iš šios lentelės galime sužinoti ar mūsų gauti eksperimentiniai rezultatai sutampa su apskaičiuotu asimptotiniu sudėtingumu.
Matome, jog padidėjus elementų skaičiui du kartus jo laikas padidėja keturis. Taip ir turėtų būti pagal gautus asimptotinius skaičiavimus ($O(n^2)$). Taigi, galiu teigti, kad gauti rezultatai patvirtina, kad asimptotinis metodo sudėtingumas buvo apskaičiuotas teisingai.

2. Užduotys su pateikta rekurentine lygtimi

Pateikta rekurentinė lygtis:

$$T(n) = 2 * T\left(\frac{n}{9}\right) + n$$

2.1. Rasti tikslų duotos lygties sprendinį ir patikrinti ar jis tenkina duotą lygtį

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a = 2$
$b = 9$
$f(n) = n$

Naudojant pagrindinę teoremą:

1. $\boldsymbol{f(n) = \mathrm{O}\left(n^{\log_b a - \varepsilon}\right)}$

$$n^1 = n^{\log_9 2 - \varepsilon}$$

$$1 = \log_9 2 - \varepsilon$$

$$\varepsilon = \log_9 2 - 1 \approx -0.6845$$

Kadangi $\varepsilon < 0$, tai šis variantas netinka.

2. $f(n) = \Theta\left(n^{\log_b a}\right)$

$$n^1 = n^{\log_9 2}$$

$$1 \neq \log_9 2$$

Kadangi nelygu tai šis variantas taip pat netinka.

3. $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$

$$n^1 = n^{\log_9 2 + \varepsilon}$$

$$1 = \log_9 2 + \varepsilon$$

$$\varepsilon = 1 - \log_9 2 \approx 0.6845$$

$\varepsilon > 0$

$$af\left(\frac{n}{b}\right) \leq cf(n)$$

$$\frac{2n}{9} \leq cn$$

$\frac{2}{9} \leq c$

Kadangi $\varepsilon > 0$ ir $c > 0$ bei $c < 1$, tai šis variantas atitinka reikalavimus.

Gautas sprendinys:

$$T(n) = \Theta\big(f(n)\big) = \Theta(\mathrm{n})$$

Taikant apibrėžimą:

$$T(n) = 2 * T\left(\frac{n}{9}\right) + n$$

$$c_1 n \le T(n) \le c_2 n$$

$$c_1 \frac{n}{9} \le T\left(\frac{n}{9}\right) \le c_2 \frac{n}{9}$$

1. $T(n) \ge 2\left(c_1 \frac{n}{9}\right) + n = \left(\frac{2c_1}{9} + 1\right) n$

$$\left(\frac{2c_1}{9} + 1\right) n \ge c_1 n$$

$$\frac{2c_1}{9} + 1 \ge c_1$$

$$c_1\left(1 - \frac{2}{9}\right) \le 1$$

$$c_1 \frac{7}{9} \le 1$$

$$c_1 \le \frac{9}{7}$$

2. $T(n) \le 2\left(c_2 \frac{n}{9}\right) + n = \left(\frac{2c_2}{9} + 1\right) n$

$$\left(\frac{2c_2}{9} + 1\right) n \le c_2 n$$

$$c_2\left(1 - \frac{2}{9}\right) \le 1$$

$$c_2 \ge \frac{9}{7}$$

Rezultatas:

$c_1 \le \frac{9}{7}$; $c_2 \ge \frac{9}{7}$ bei $T(n) = \Theta\big(f(n)\big) = \Theta(\mathrm{n})$

2.2. Parašyti programinį kodą

5 lentelė Parašytas programini kodas

```
public static long Method(int n)
    {
        if (n <= 1)
        {
            return 1;
        }

        for (int i = 0; i < (n - 1); i++)
        {
            actionCount++;
        }

        Method(n / 9);
        Method(n / 9);

        return actionCount;
    }
```

2.3. Rasti tikslų programinio kodo sudėtingumą, skaičiuojant vykdomų eilučių skaičių

6 lentelė Tikslus programinio kodo sudėtingumas

Nr.	Kodas	Laikas	Kartai
1	`public static long Method(int n)`		
2	`{`		
3	`if (n <= 1)`	c_1	$1, X_1 \in \{0,1\}$
4	`{`		
5	`return 1;`	c_2	X_1
6	`}`		
7			
8	`for (int i = 0; i < n; i++)`	c_3	$(1-X_1)(n-0+1) = (1-X_1)(n+1)$
9	`{`		
10	`actionCount++;`	c_4	$(1-X_1)n$
11	`}`		
12			
13	`Method(n / 9);`	$T\left(\frac{n}{9}\right)$	$1-X_1$
14	`Method(n / 9);`	$T\left(\frac{n}{9}\right)$	$1-X_1$
15			
16	`return actionCount;`	c_5	$(1-X_1)$
17	`}`		

$$T_{Method}(n) = c_1 + c_2X_1 + c_3(1-X_1)(n+1) + c_4(1-X_1)n + 2 * T\left(\frac{n}{9}\right) * (1-X_1) + c_5(1-X_1)$$

Kai $X_1 = 0$ blogesnis variantas:

$$T_{Method}(n) = c_1 + c_3 + c_3n + c_4n + 2 * T\left(\frac{n}{9}\right) + c_5 = 2 * T\left(\frac{n}{9}\right) + n(c_3 + c_4) + c_1 + c_3 + c_5$$

Norint, kad sutaptų gauta programinio kodo lygtis su pateikta lygtimi turime duotą lygtį papildyti žemesnio laipsnio nariais.

$$T_{Method}(n) = 2 * T\left(\frac{n}{9}\right) + n(c_3 + c_4) + c_1 + c_3 + c_5$$

3. Uždavinių sprendimas

3.1. Pirma užduotis. Palyginti funkcijas

a) $f(n) = \ln(n^3 + n)$; $g(n) = \log_5 n^2$

$$f(n) = \ln(n^3 + n) = \ln\left(n^3\left(1 + \frac{1}{n^2}\right)\right)$$

Kai n artėja link begalybės:

$$\lim_{n\to\infty}\left(1 + \frac{1}{n^2}\right) = 1 + 0 = 1$$

Tuomet:

$$f(n) \approx \ln(n^3) = 3\ln n$$

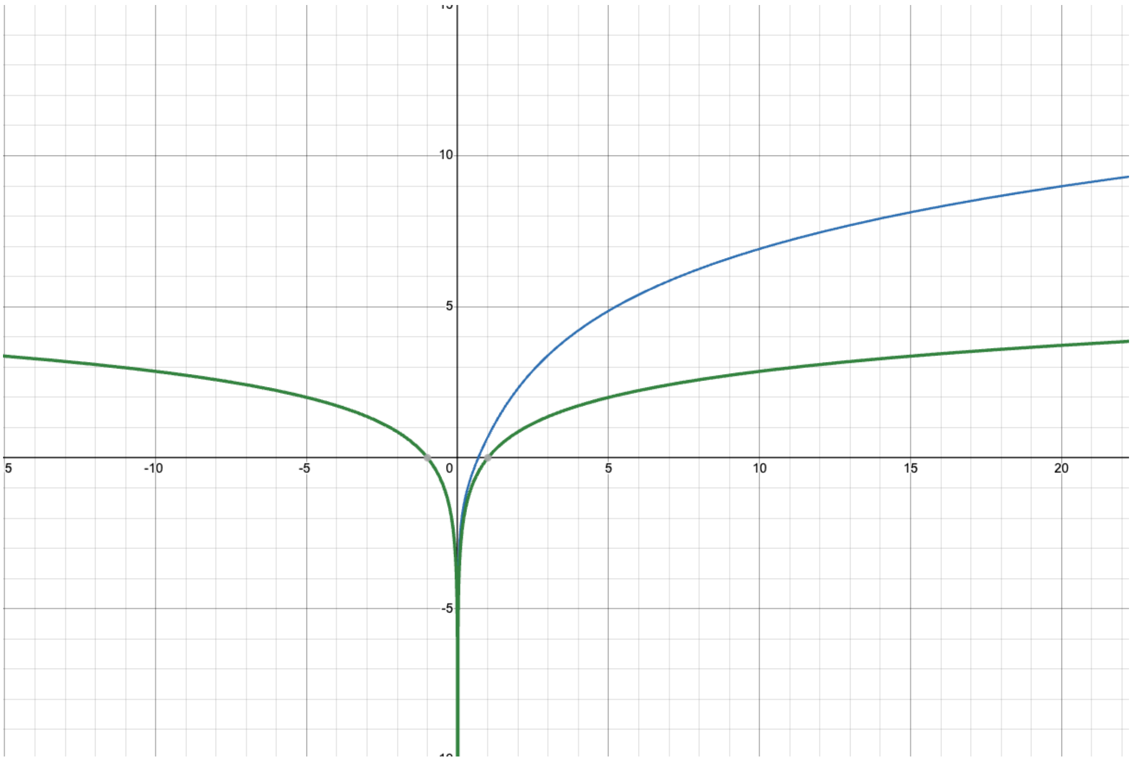
$$g(n) = 2\log_5 n = 2\frac{\ln n}{\ln 5} = \frac{2}{\ln 5}\ln n$$

$$\frac{2}{\ln 5} \approx 1.243$$

Vadinasi, pagal gautus rezultatus matome, kad g(n) funkcija augs lėčiau nei f(n).

$$\frac{f(n)}{g(n)} = \frac{3\ln n}{\frac{2}{\ln 5}\ln n} = \frac{3}{\frac{2}{\ln 5}} = \frac{3\ln 5}{2} \approx 2.414$$

Apie 2.414 karto greičiau augs f(n) funkcija.



1 pav. Nubraižytos f(n) – mėlyna ir g(n) – žalia funkcijos

b) $f(n) = 2^{\frac{\log_4 n}{2}}; g(n) = \ln(n^2)$

Palyginimas:

$$\log_4 n = \frac{\ln n}{\ln 4}$$

$$f(n) = 2^{\frac{\ln n}{2 \ln 4}}$$

Kai žinome, kad: $a^{\log_b x} = x^{\log_b a}$. Galime atlikti tokius veiksmus:

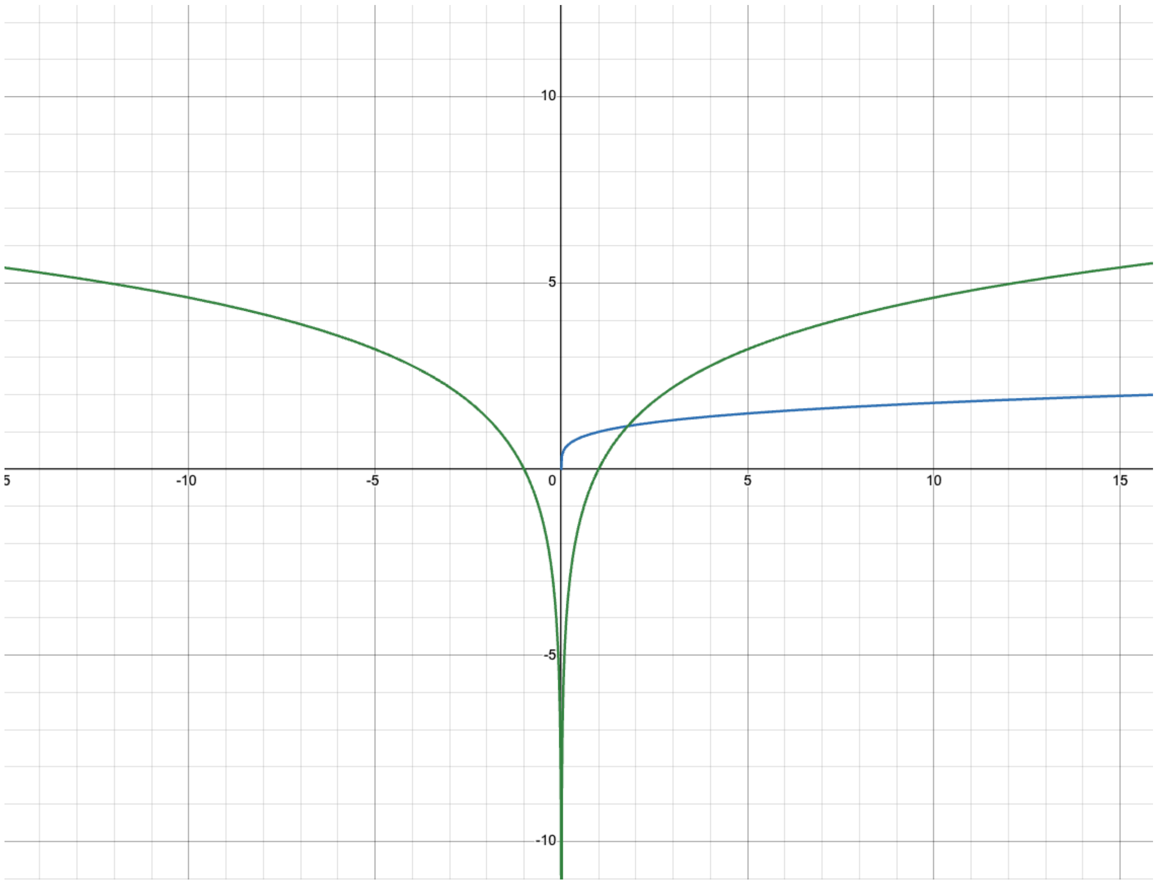
$$2^{\frac{\ln n}{2 \ln 4}} = n^{\frac{\ln 2}{2 \ln 4}} = n^{\frac{\ln 2}{2(2 \ln 2)}} = n^{\frac{1}{4}}$$

$$f(n) = n^{\frac{1}{4}}$$

Imame kitą funkciją:

$$g(n) = \ln(n^2) = 2 \ln n$$

Kadangi f(n) gautas rezultatas yra polinominis, o g(n) funkcijos logaritminis, tai f(n) funkcija augs greičiau. Tačiau šiuo atveju kadangi f(n) rezultatas gavosi pakeltas mažu laipsniu jos didesnį paaugimą didėjant n galima bus pamatyti ne iš pat pradžių.

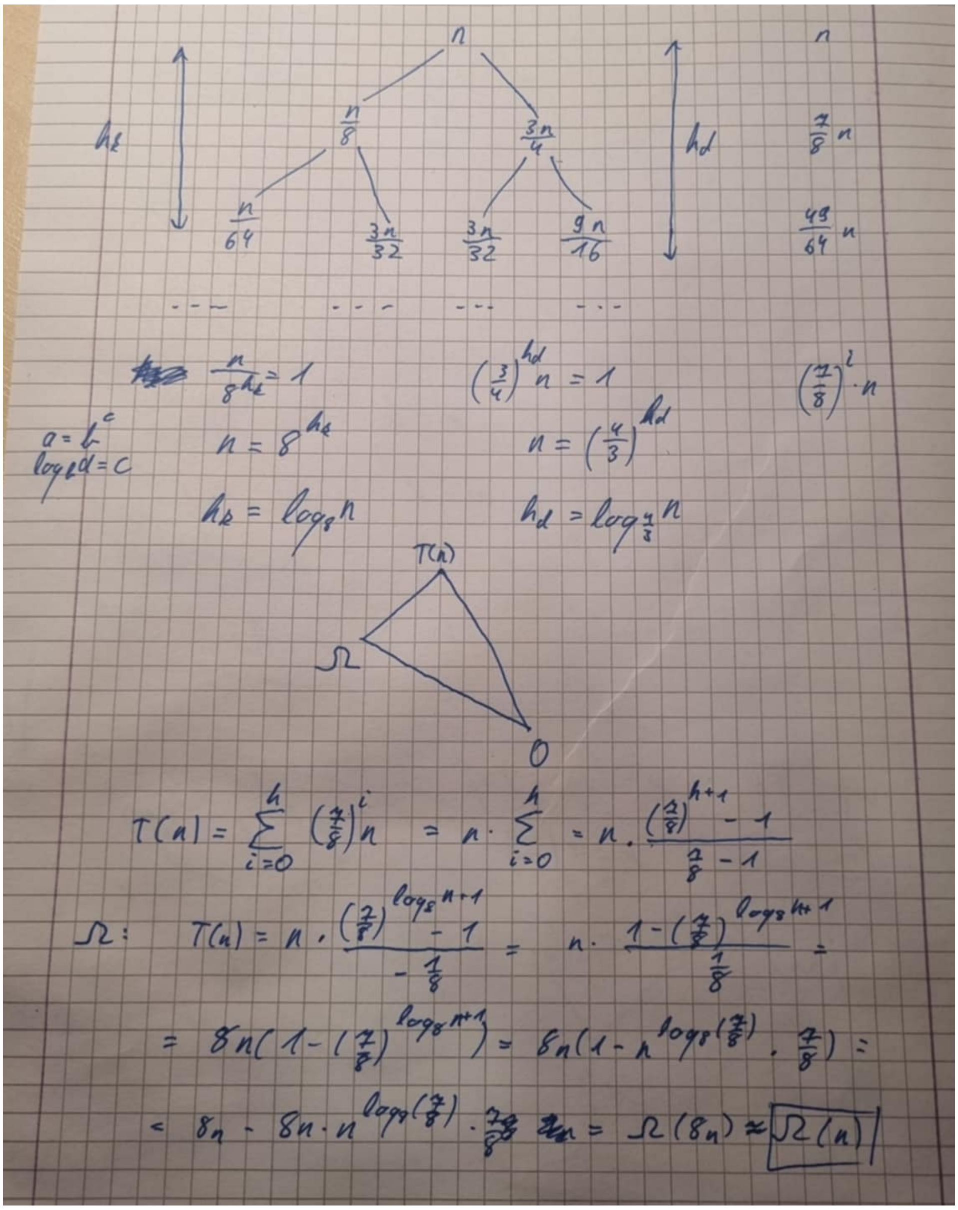


2 pav. Nubraižytos f(n) – mėlyna ir g(n) – žalia funkcijos

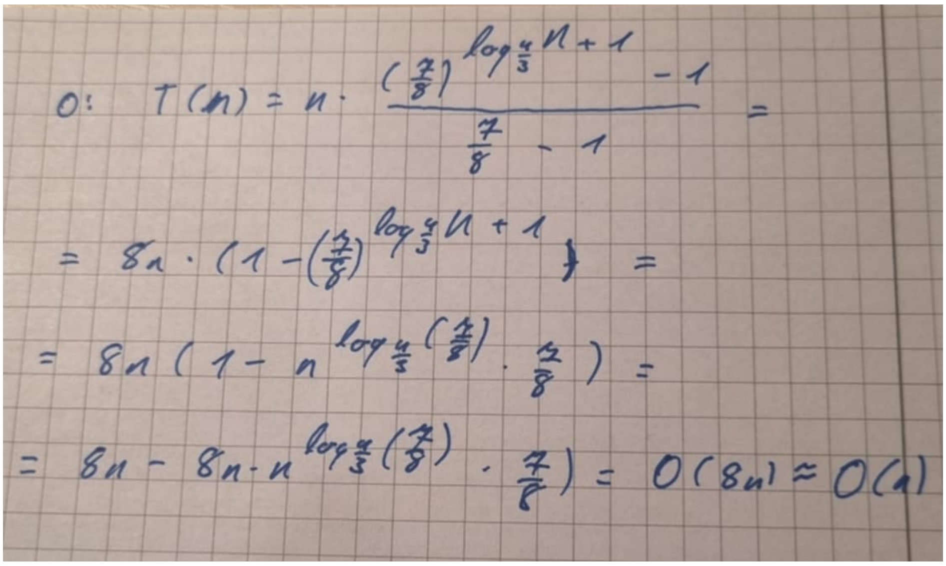
3.2. Antra užduotis. Išspręsti rekurentines lygtis

a) $T(n) = T\left(\frac{n}{8}\right) + T\left(\frac{3n}{4}\right) + n$

Sprendimas naudojant medį:



3 pav. Lygties sprendimas medžio metodu



4 pav. Lygties sprendimo tęsinys naudojant medžio metodą

Nors gauname ir geriausią ir blogiausią atvejų sudėtingumą vienodą, jie nėra vienodi. Šie rezultatai yra suapvalinti tačiau pagal gautą aukštį galime akivaizdžiai pastebėti, jog $T\left(\frac{3n}{4}\right)$ užtruks ilgiau nei $T\left(\frac{n}{8}\right)$.

b) $T(n) = 8T\left(\frac{n}{2}\right) + n^2$

Kai žinome, kad $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, tai atliekame šiuos veiksmus:

1) $n^2 = O\left(n^{\log_2 8-\varepsilon}\right)$

$$2 = \log_2 8 - \varepsilon$$

$$2 = 3 - \varepsilon$$

$$\varepsilon = 1$$

Šis variantas tinka, nes $\varepsilon > 0$.
Taigi, šios funkcijos rezultatas bus:

$$\mathrm{T(n)} = \Theta\left(\mathrm{n}^{\log_2 8}\right) = \Theta(n^3)$$

3.3. Trečia užduotis. Suprastinti funkcionalus:

a) $O\left(k^3 + \sum_{i=0}^{k} e^i\right)$

$$\sum_{i=0}^{k} e^i = \frac{e^{k+1}-1}{e-1}$$

$$O\left(k^3 + \frac{e^{k+1}-1}{e-1}\right)$$

Atmetame konstantą:

$$O(k^3 + e^k)$$

Norint dar suprastinti: atsižvelgiame, kad k^3 auga lėčiau, nes čia augimo greitis yra kubinis, o funkcija e^k auga daug greičiau, nes kai k artėja link begalybės jos augimo greitis yra žymiai didesnis nei kubinis.

Taigi rezultatas bus:

$$O(e^k)$$

b) $O\left(\log_3^2 n + \sqrt{n} * \ln n\right)$

$$\log_3 n = \frac{\ln n}{\ln 3}$$

$$O\left(\frac{(\ln n)^2}{(\ln 3)^2} + \sqrt{n} * \ln n\right)$$

Atmetame konstantą:

$$O\left((\ln n)^2 + \sqrt{n} * \ln n\right)$$

Kadangi $(\ln n)^2$ auga lėčiau nei $\sqrt{n} * \ln n$, todėl galime suprastinti:

$$O(\sqrt{n} * \ln n)$$

Taigi, gavome rezultatą:

$$O(\sqrt{n} * \ln n)$$

3.4. Ketvirta užduotis. Įvertinti programinio kodo sudėtingumą geriausiu ir blogiausiu atvejais

a)

7 lentelė Programinio kodo sudėtingumo įvertinimas (a dalis)

Kodas	Laikas	Kartai
`static int[] AA(int[] A, int m, int n){`		
`    int p = (n - m + 1) / 3;`	c1	1
`    if (p > 3) {`	c2	1, $X_1 \in \{0, 1\}$
`        for (int i = 1; i < 4; i++)`	c3	$X_1(4 - 1 + 1) = X_1 4$
`            A = AA(A, m, m + p - 1);`	c4	$X_1 * 3 * T(A, m, m + p - 1)$
`        for (int i = m; i <= n;i++)`	c5	$X_1(n + 1 - m + 1) = X_1(n - m + 2)$
`            A[i] = i + p;`	c6	$X_1(n + 1 - m) = X_1(n - m + 1)$
`    }`		
`    return A;`	c7	1
`}`		

$$T_{AA}(A, m, n) = c_1 + c_2 + c_3 X_1 4 + c_4 X_1 3T\left(\frac{n}{3}\right) + c_5 X_1(n - m + 2) + c_6 X_1(n - m + 1) + c_7$$

Geriausias atvejis, kai $X_1 = 0$, tai:

$$T_{AA}(A, m, n) = c_1 + c_2 + c_3 = \Omega(1)$$

Blogiausias atvejis, kai $X_1 = 1$, tai:

$$T_{AA}(A, m, n) = c_1 + c_2 + 4c_3 + c_7 + c_4 3T\left(\frac{n}{3}\right) + c_5(n - m + 2) + c_6(n - m + 1)$$

$$= 3T\left(\frac{n}{3}\right) + c_5(n - m + 2) + c_6(n - m + 1) + C = 3T\left(\frac{n}{3}\right) + n$$

Kadangi $3T\left(\frac{n}{3}\right) + n = \Theta(n * \log n)$, tai:

$$T_{AA}(A, m, n) = O(n \log n)$$

Taigi, gavome rezultatus:

Geriausias atvejis: $\Omega(1)$

Blogiausias atvejis: $O(n\log n)$

b)

8 lentelė Programinio kodo sudėtingumo įvertinimas (b dalis)

Kodas	Laikas	Kartai
`static int[] CC(int[] C, int n){`		
`    if (C[0] > 10)`	c1	1, $X_1 \in \{0,1\}$
`        for(int i = 0; i < n; i++)`	c2	$X_1(n-0+1) = X_1(n+1)$
`            C[i] = C[i] + 1;`	c3	$X_1 n$
`    for(int j = 1; j <= n; j = j * 2)`	c4	$\log_2 n + 2$
`        C[j] = C[j] + 1;`	c5	$\log_2 n + 1$
`    return C;`	c6	1
`}`		

$$T_{CC}(C,n) = c_1 + c_2 X_1(n+1) + c_3 X_1 n + c_4(\log_2 n + 2) + c_5(\log_2 n + 1) + c_6$$

Geriausias atvejis kai $X_1 = 0$, tai:

$$T_{CC}(C,n) = c_1 + c_4 \log_2 n + c_4 2 + c_5 \log_2 n + c_5 + c_6 = \log_2 n\,(c_4 + c_5) + c_1 + c_5 + c_6 + 2c_4 = \Omega(\log_2 n) = \Omega(\log n)$$

Blogiausias atvejis kai $X_1 = 1$, tai:

$$\begin{aligned} T_{CC}(C,n) &= c_1 + c_2(n+1) + c_3 n + c_4(\log_2 n + 2) + c_5(\log_2 n + 1) + c_6 \\ &= c_2 n + c_3 n + c_4 \log_2 n + c_5 \log_2 n + c_1 + c_2 + 2c_4 + c_5 + c_6 \\ &= n(c_2 + c_3) + \log_2 n\,(c_4 + c_5) + c_1 + c_2 + 2c_4 + c_5 + c_6 = n + \log_2 n = O(n) \end{aligned}$$

Taigi, gavome rezultatus:

Geriausias atvejis: $\Omega(\log n)$

Blogiausias atvejis: $O(n)$