****

**Kauno technologijos universitetas**

Informatikos fakultetas

**Antro laboratorinio darbo ataskaita**

2LD individuali užduotis

|  |  |
| --- | --- |
|  |  |
| **Aistis Jakutonis**  Studentas | (parašas) (data) |
|  |  |
| **Dalius Makackas**  **Andrius Kriščiūnas**  **Tadas Kraujalis**  **Vidmantas Rimavičius**  Dėstytojai | (parašas) (data) |
|  |  |

**Kaunas, 2025**

Turinys

[1. Pateikto programinio kodo metodo analizė 3](#_Toc193904141)

[1.1. Apskaičiuojamas metodo asimptotinis sudėtingumas 3](#_Toc193904142)

[1.2. Atlikti eksperimentinį tyrimą 3](#_Toc193904143)

[1.2.1. Vykdymo laiko priklausomybė nuo masyvo dydžio 3](#_Toc193904144)

[1.2.2. Veiksmų skaičiaus priklausomybė nuo masyvo dydžio 4](#_Toc193904145)

[1.3. Patikrinti ar apskaičiuotas asimptotinis sudėtingumas atitinka eksperimentinius rezultatus 5](#_Toc193904146)

[2. Užduotys su pateikta rekurentine lygtimi 6](#_Toc193904147)

[2.1. Rasti tikslų duotos lygties sprendinį ir patikrinti ar jis tenkina duotą lygtį 6](#_Toc193904148)

[2.2. Parašyti programinį kodą 7](#_Toc193904149)

[2.3. Rasti tikslų programinio kodo sudėtingumą, skaičiuojant vykdomų eilučių skaičių 8](#_Toc193904150)

[3. Uždavinių sprendimas 9](#_Toc193904151)

[3.1. Pirma užduotis. Palyginti funkcijas 9](#_Toc193904152)

[3.2. Antra užduotis. Išspręsti rekurentines lygtis 11](#_Toc193904153)

[3.3. Trečia užduotis. Suprastinti funkcionalus: 13](#_Toc193904154)

[3.4. Ketvirta užduotis. Įvertinti programinio kodo sudėtingumą geriausiu ir blogiausiu atvejais 14](#_Toc193904155)

# Pateikto programinio kodo metodo analizė

## Apskaičiuojamas metodo asimptotinis sudėtingumas

1 lentelė Suskaičiuojamas kodo asimptotinis sudėtingumas

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Kodas | Laikas | Kartai |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **public** **static** **long** **methodToAnalysis** (**int**[] arr)  {  **long** n = arr.Length;  **long** k = n;  **for** (**int** i = **0**; i < n\***2**; i++)  {  **for** (**int** j = **0**; j < n/**2**; j++)  {  k -= **2**;  }  }  **return** k;  } | c1  c2  c3  c4  c5  c6 | 1  1  1 |
|  | | | |

## Atlikti eksperimentinį tyrimą

### 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

### 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

## 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 (). Taigi, galiu teigti, kad gauti rezultatai patvirtina, kad asimptotinis metodo sudėtingumas buvo apskaičiuotas teisingai.

# Užduotys su pateikta rekurentine lygtimi

Pateikta rekurentinė lygtis:

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

Naudojant pagrindinę teoremą:

Kadangi , tai šis variantas netinka.

2.

Kadangi nelygu tai šis variantas taip pat netinka.

3.

Kadangi ir bei , tai šis variantas atitinka reikalavimus.

Gautas sprendinys:

Taikant apibrėžimą:

Rezultatas:

; bei

## Parašyti programinį kodą

5 lentelė Parašytas programini kodas

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | **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;  } | |

## Rasti tikslų programinio kodo sudėtingumą, skaičiuojant vykdomų eilučių skaičių

6 lentelė Tikslus programinio kodo sudėtingumas

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Kodas | Laikas | Kartai |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | |  | | --- | | **public** **static** **long** **Method**(**int** n)  {  **if** (n <= **1**)  {  **return** **1**;  }  **for** (**int** i = **0**; i < n; i++)  {  actionCount++;  }  Method(n / **9**);  Method(n / **9**);  **return** actionCount;  } | | c1  c2  c3  c4  c5 | 1,  n |
| Kai blogesnis variantas: | | | |

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

# Uždavinių sprendimas

## Pirma užduotis. Palyginti funkcijas

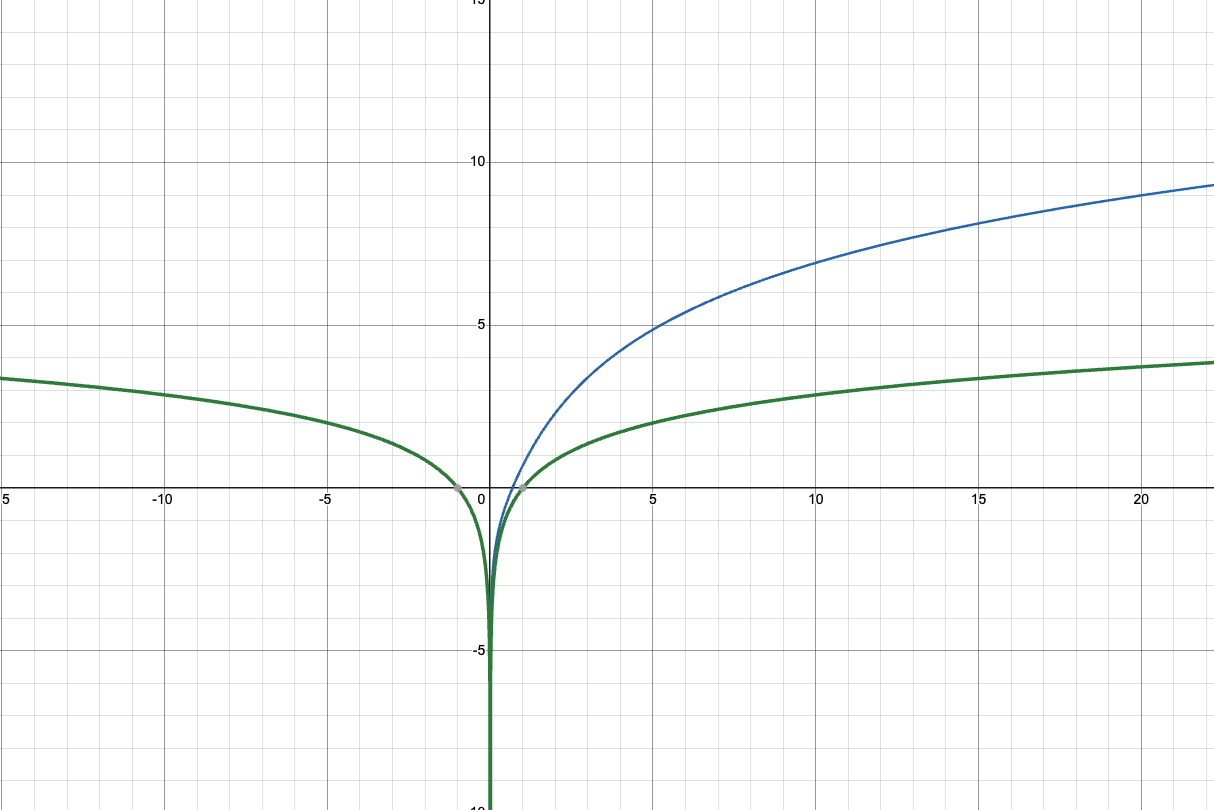
1. ;

Kai n artėja link begalybės:

Tuomet:

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

Apie 2.414 karto greičiau augs f(n) funkcija.



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

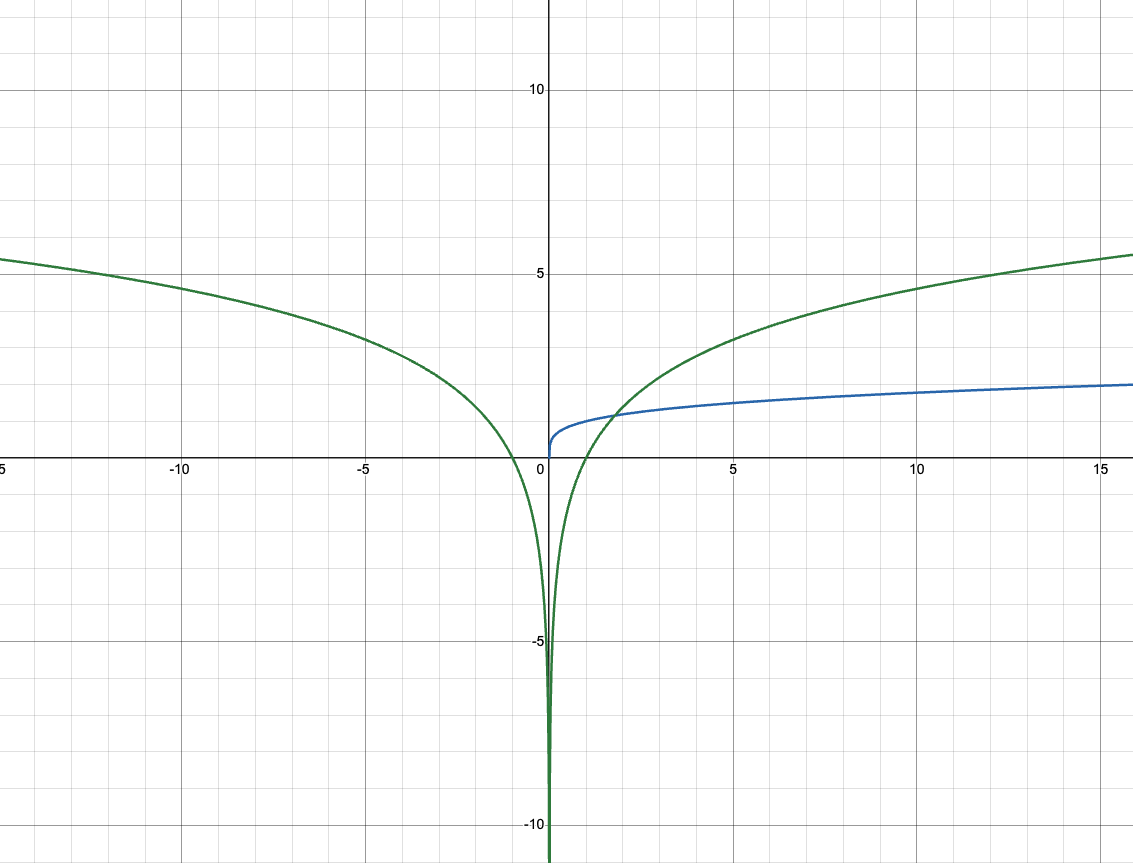
1. ;

Palyginimas:

Kai žinome, kad: . Galime atlikti tokius veiksmus:

Imame kitą funkciją:

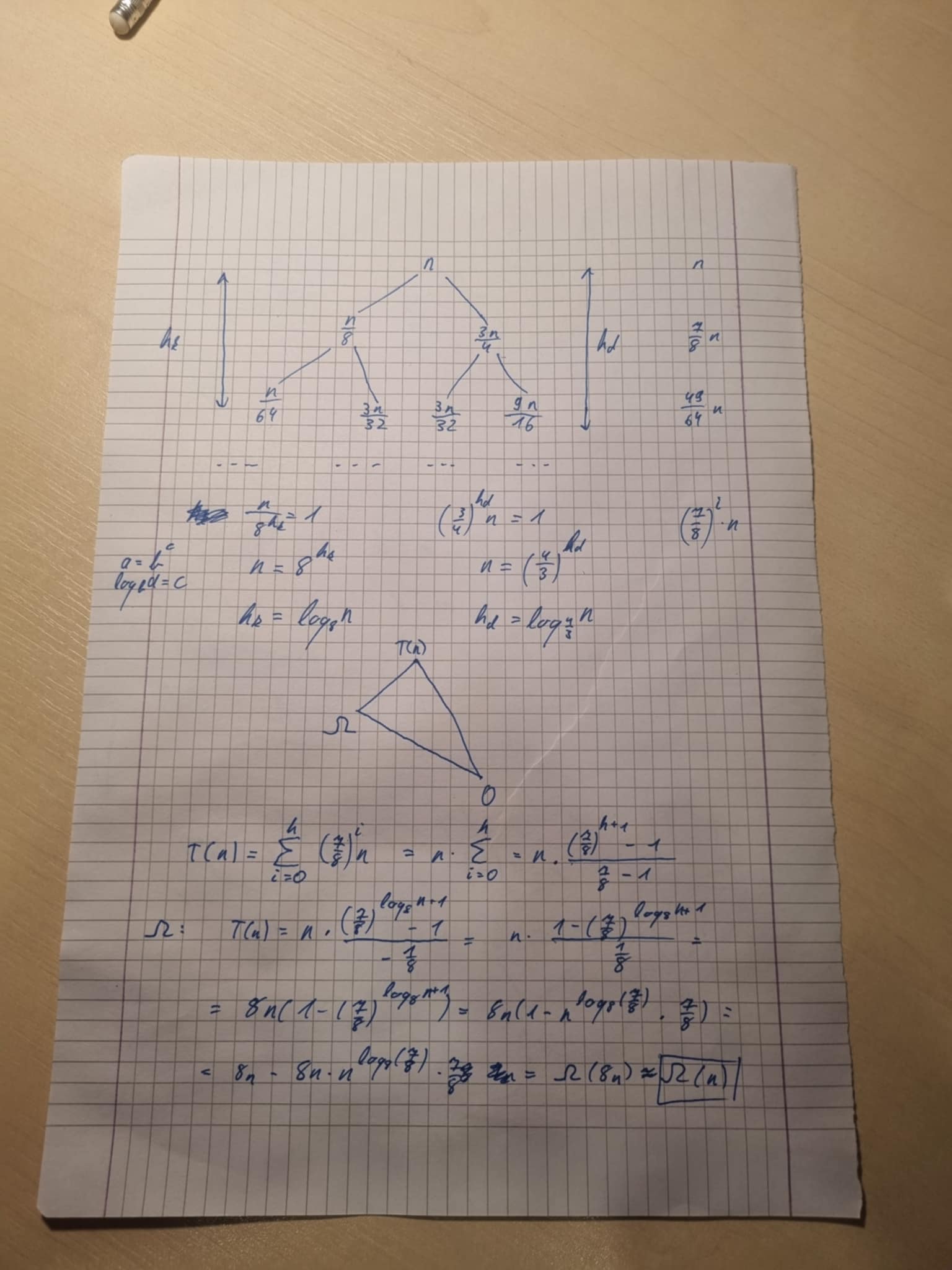
Kadangi f(n) gautas rezultatas yra polinominis, o g(n) funkcijos logaritminis, tai f(n) funkcija augs greičiau. Tačiau šiuo atveju kadangi f(n) rezultatas gavosi pakeltas mažu laipsniu jos didesnį paaugimą didėjant n galima bus pamatyti ne iš pat pradžių.



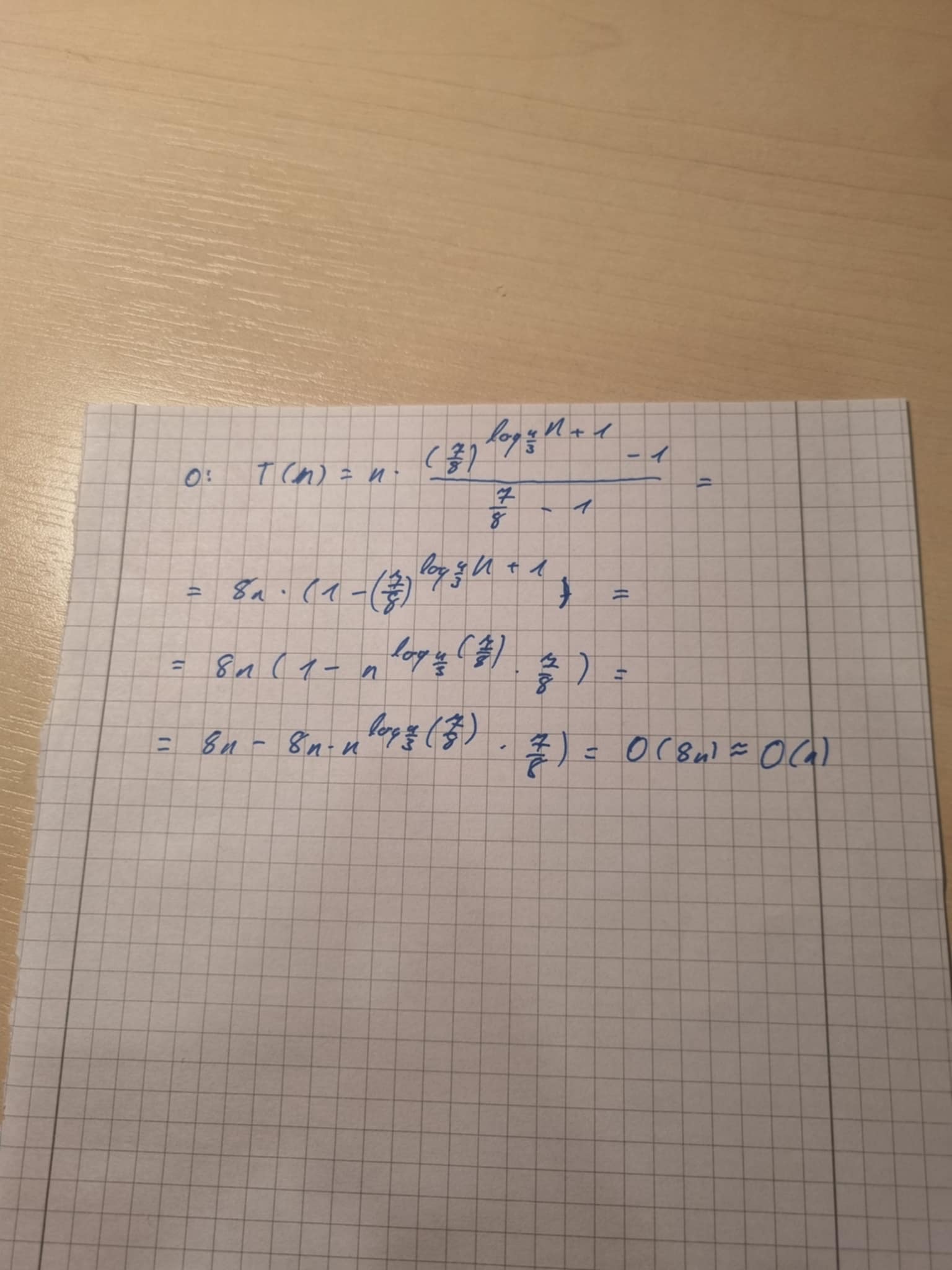
2 pav. Nubraižytos f(n) – mėlyna ir g(n) – žalia funkcijos

## Antra užduotis. Išspręsti rekurentines lygtis

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 užtruks ilgiau nei .

Kai žinome, kad , tai atliekame šiuos veiksmus:

Šis variantas tinka, nes .

Taigi, šios funkcijos rezultatas bus:

­

## Trečia užduotis. Suprastinti funkcionalus:

Atmetame konstantą:

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

Taigi rezultatas bus:

Atmetame konstantą:

Kadangi auga lėčiau nei , todėl galime suprastinti:

Taigi, gavome rezultatą:

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

7 lentelė Programinio kodo sudėtingumo įvertinimas (a dalis)

|  |  |  |
| --- | --- | --- |
| Kodas | Laikas | Kartai |
| |  | | --- | | **1.** **static** **int**[] AA(**int**[] A, **int** m, **int** n){  **2.** **int** p = (n - m + **1**) / **3**;  **3.** **if** (p > **3**) {  **4.** **for** (**int** i = **1**; i < **4**; i++)  **5.** A = AA(A, m, m + p - **1**);  **6.** **for** (**int** i = m; i <= n;i++)  **7.** A[i] = i + p;  **8.** }  **9.** **return** A;  **10.** } | | c1  c2  c3  c4  c5  c6  c7 | 1  1,  1 |
| Geriausias atvejis, kai , tai:  Blogiausias atvejis, kai , tai:  Kadangi , tai:  Taigi, gavome rezultatus:  Geriausias atvejis:  Blogiausias atvejis: | | |

8 lentelė Programinio kodo sudėtingumo įvertinimas (b dalis)

|  |  |  |
| --- | --- | --- |
| Kodas | Laikas | Kartai |
| |  | | --- | | **11.** **static** **int**[] CC(**int**[] C, **int** n){  **12.** **if** (C[**0**] > **10**)  **13.** **for**(**int** i = **0**; i < n; i++)  **14.** C[i] = C[i] + **1**;  **15.** **for**(**int** j = **1**; j <= n; j = j \* **2**)  **16.** C[j] = C[j] + **1**;  **17.** **return** C;  **18.** } | | c1  c2  c3  c4  c5  c6 | 1,  1 |
| Geriausias atvejis kai , tai:  Blogiausias atvejis kai , tai:  Taigi, gavome rezultatus:  Geriausias atvejis:  Blogiausias atvejis: | | |