Kauno technologijos universitetas

Informatikos Fakultetas

Laboratorinis darbas 1

Laboratorinio darbo ataskaita

P170B400 Algoritmų sudarymas ir analizė

Ataskaitą parengė

Lukas Dargevičius

Laboratoriniam darbui vadovavo

dr. Dalius Makackas

Kaunas, 2026

Turinys

Įvadas

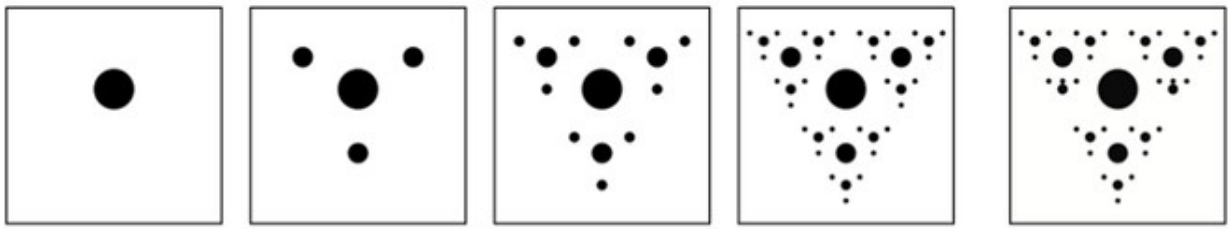
Šis laboratorinis darbas skirtas efektyvių rekursinių procedūrų implementacijų kūrimui duotai problemai spręsti, eksperimentiniam jų našumo įvertinimui bei atitinkamų rekurencinių lygčių formulavimui ir sprendimui.

1. Užduotis

Užduoties aprašymas: sukurti fraktalo generavimo rekursinį algoritmą, šį nupiešta fractalą įrašyti į BMP failą 16 bitų spalvos gyliu.

Programos reikalavimai:

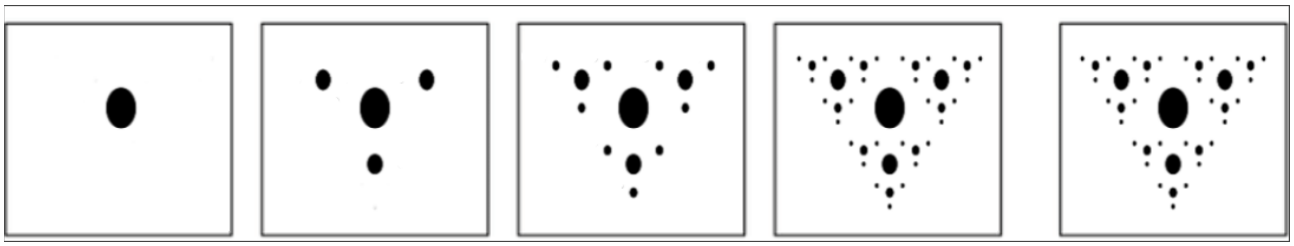
- Du realizacijos atvejai: 1) Nurodomas rekursijos gylis; 2) Generuojamas maksimalaus detalumo paveikslėlis, pasirinktiems paveikslėlio matmenims.
- Programų rezultatas BMP formato bylos demonstruojančios programų veikimą.
- Eksperimentiškai nustatykite darbo laiko ir veiksmų skaičiaus priklausomybę nuo rekursijos gylio ar generuojamo paveikslėlio dydžio (taškų skaičiaus). Gautus rezultatus atvaizduokite grafikais (4 grafikai). Grafiką turi sudaryti nemažiau kaip 5 taškai ir paveikslėlio taškų skaičius turi didėti proporcingai (kartais). Paskutinis eksperimento grafiko taškas turi atitikti maksimalų galimą paveikslėlio raišką (apie 20000x20000 taškų).
- Analitiškai įvertinkite procedūrų, kurios generuoja paveikslėlius, veiksmų skaičių (vertindami skirtingas operacijas) ir laiką, sudarydami rekurentines lygtis ir jas išspręskite. Gauti rezultatai turi patvirtinti eksperimentinius rezultatus (našumo testus: vykdymo laiką ir veiksmų skaičių).
- Paruoškite detalią ataskaitą su atliktais skaičiavimais. Ataskaita privalo tenkinti rašto darbų reikalavimus (visos formulės privalo būti rašomos pagal matematikoje priimtus kanonus, programos fragmentai neturėtų būti tiesiog „blynai“ (ekrano kopijos...) Rekomenduojama naudoti Latex ar Word redaktorius matematinių išraiškų rašymui. Už neatliktas užduotis balas proporcingai mažinamas.



pav. 1: užduotis

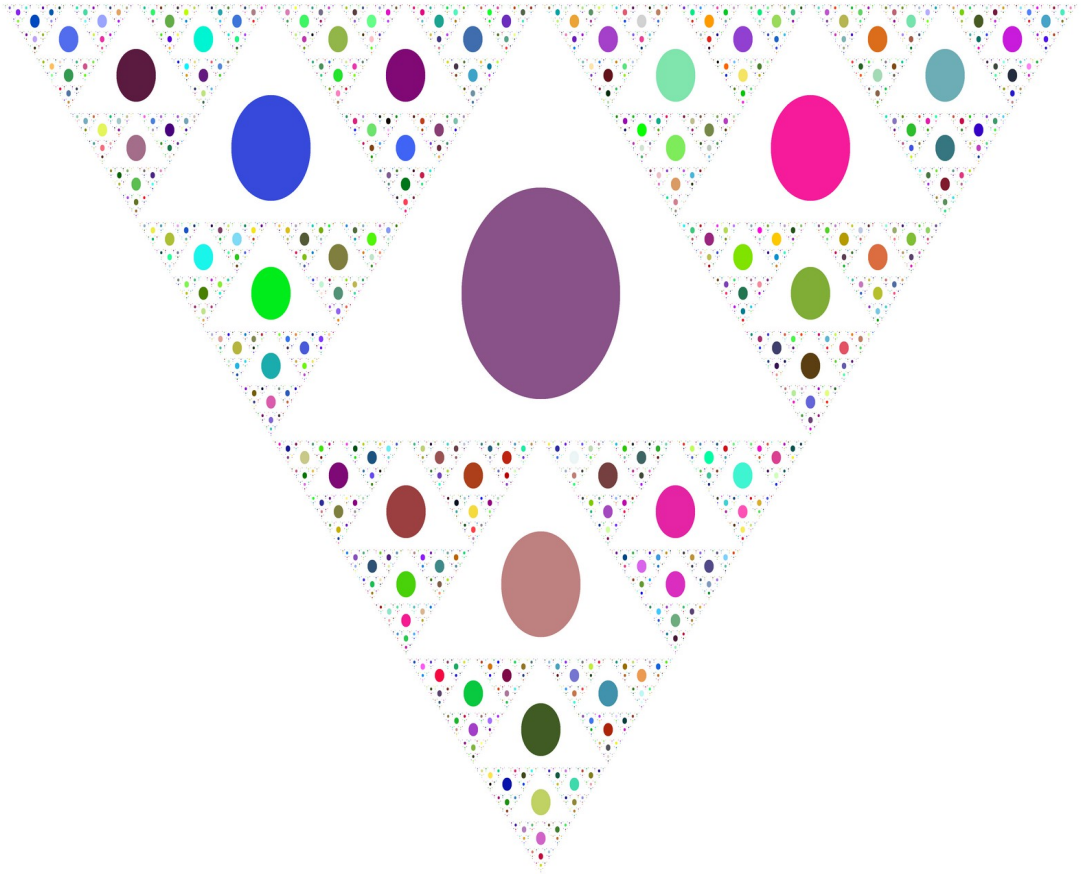
2. Resultatai

Bandant pakartoti pavyzdį, programa padarė:



pav. 2: rezultatas

Rezultatas programai dirbant su 16 bitų spalvų gyliu:



pav. 3: rezultatas spalvotas

3. Rekurentinės lygtys

Pazymekime, kad Recursion sudetingumas bus $T(p,d)$

Pazymekime, kad drawFilledEllipse sudetingumas bus $T_d(radiusX, radiusY)$

Pazymekime, kad writePoint sudetingumas bus $T_p()$

Pazymekime, kad randomColor sudetingumas bus $T_r()$

C_1 - priskyrimas

C_2 - Loginiai

C_3 - Sudetis/Atimtis

C_4 - Daugyba/Dalyba

C_5 - Sin/Cos

C_6 - Rekurcinis iskvietimas

C_7 - return

C_8 - byteAnd

C_9 - byteShift

C_{10} - Cast

C_{11} - random

Utils:

piesti:

`public void writePixel(int x, int y, ushort color) {`	Laikas	Kiekis
`int byteLoc = (x * 2) + (y * bytesInRow);`	$C_1+C_3+2\cdot C_4$	χ
`mapArray[byteLoc] = (byte)(color & 0×FF);`	$C_{10}+C_8$	χ
`mapArray[byteLoc + 1] = (byte)((color >> 8) & 0×FF);`	$C_{10}+C_8+C_9$	χ
`}`		

$T_p()=C_1+C_3+2C_4+C_{10}+C_8+C_{10}+C_8+C_9=C_1+C_3+2C_4+2C_8+C_9+C_{10}$

$T_p()=C_1+C_3+2C_4+2C_8+C_9+C_{10}$

$$\chi_1=\begin{cases}1 \to \frac{x^2}{radX^2}+\frac{y^2}{radY^2}\le 1\\ 0 \to \frac{x^2}{radX^2}+\frac{y^2}{radY^2}>1\end{cases}$$

$p_1=radY-(-radY)+1$

$p_2=radX-(-radX)+1$

piestiElipse:

`public void drawFilledEllipse(Coordinate center, int radX, int radY, ushort color) {`	Laikas	Kiekis
`for (int y = -radY; y ⩽ radY; y++) {`	$C_1+C_4+C_2$	1
	C_2+C_3	p_1
`for (int x = -radX; x ⩽ radX; x++) {`	$C_1+C_4+C_2$	p_1
	C_2+C_3	$p_1 \cdot p_2$
`if ((x * x) / (double)(radX * radX) + (y * y) / (double)(radY * radY) ⩽ 1) {`	$C_2+6C_4+2C_{10}$	$p_1 \cdot p_2$
`writePixel(center.X + x, center.Y + y, color);`	$T_p()$	$p_1 \cdot p_2 \cdot \chi_1$
`}`		
`}`		
`}`		
`}`		

$T_d(radX,radY)=C_1+C_4+C_2+p_1\cdot(C_2+C_3+C_1+C_4+C_3+p_2(C_2+C_3+C_2+6C_4+2C_{10}+T_p()))$

$T_d(radX,radY)=C_1+C_2+C_4+p_1\cdot(C_1+C_2+2C_3+C_4+p_2(2C_2+C_3+6C_4+2C_{10}+T_p()))$

$T_d(radX,radY)=C_1+C_2+C_4+p_1\cdot(C_1+C_2+2C_3+C_4+p_2\cdot 2C_2+p_2C_3+p_2 6C_4+p_2 2C_{10}+p_2T_p())$

$T_d(radX,radY)=C_1+C_2+C_4+p_1C_1+p_1C_2+p_1 2C_3+p_1C_4+p_1p_2 2C_2+p_1p_2C_3+p_1p_2 6C_4+p_1p_2 2C_{10}+p$

$T_d(radX,radY)=C_1(1+p_1)+C_2(1+p_1+2p_1p_2)+C_3(2p_1+p_1p_2)+C_4(1+p_1+6p_1p_2)+2p_1p_2C_{10}+p_1p_2T_p$

$1+p_1=1+radY-(-radY)+1=2radY+2$

$p_1p_2=(radY-(-radY)+1)\cdot(radX-(-radX)+1)=(2radY+1)\cdot(2radX+1)$

$$T_d(radX,radY)=C_1(2radY+2)+C_2(2radY+2+2(2radY+1)(2radX+1))+$$
$$+C_3(4radY+2+(2radY+1)(2radX+1))+C_4(2radY+2+6(2radY+1)(2radX+1))+$$
$$+C_{10}2(2radY+1)(2radX+1)+T_p()(2radY+1)(2radX+1)\chi_1$$

Random:

	Laikas	Kiekis
`static ushort GetRandomColor565()`		
`int r = rand.Next(0, 32);`	C_{11}	1
`int g = rand.Next(0, 64);`	C_{11}	1
`int b = rand.Next(0, 32);`	C_{11}	1
`return (ushort)((r << 11) | (g << 5) | b);` `}`	$C_8+2C_9+C_{10}$	1

$T_r()=C_8+2C_9+C_{10}+3C_{11}$

$\chi_2=\begin{cases}1 \rightarrow d<maxDepth\\0 \rightarrow d>maxDepth\end{cases}$

$r=p*0.2/2^d$

Rekurcija:

	Laikas	Kiekis
`static int Recursion(BMPfile BMP, int PictureSize, Coordinate Pos, int maxDepth, int curDepth){`		
`if (maxDepth < curDepth) {`	C_2	1
`return;` `}`	C_7	χ_2
`int length = (int)(PictureSize * 0.55 / Math.Pow(2, curDepth));`	$C_{10}+2C_4+C_5$	χ_2
`double radius = PictureSize * 0.2 / Math.Pow(2, curDepth);`	$2C_4+C_5$	χ_2
`BMP.drawFilledEllipse(Pos, (int)(radius*0.7), (int)radius, GetRandomColor565());`	$2C_{10}+C_4+T_r()+T_d(r*0.3,r)$	χ_2
`double height = Math.Sin(30 * (Math.PI / 180)) * length;`	$3C_4+C_5$	χ_2
`double width = Math.Cos(30 * (Math.PI / 180)) * length;`	$3C_4+C_5$	χ_2
`Recursion(BMP, PictureSize, new Coordinate((int)(Pos.X + width), (int)(Pos.Y + height)), maxDepth, curDepth + 1);`	$T(p,d-1)+3C_3+2C_{10}$	χ_2
`Recursion(BMP, PictureSize, new Coordinate((int)(Pos.X - width), (int)(Pos.Y + height)), maxDepth, curDepth + 1);`	$T(p,d-1)+3C_3+2C_{10}$	χ_2
`Recursion(BMP, PictureSize, new Coordinate((int)(Pos.X), (int)(Pos.Y - length)), maxDepth, curDepth + 1);`	$T(p,d-1)+3C_3+2C_{10}$	χ_2

$$T(p,d)=C_2+9C_3+11C_4+4C_5+C_7+9C_{10}+T_r+T_d(\frac{p*3}{5\cdot 2^{d-1}},\frac{p*1}{5\cdot 2^{d-1}})+3T(P,d-1)$$

$$T(p,d)=C_2+9C_3+11C_4+4C_5+C_7+9C_{10}+T_r+\frac{p}{5\cdot 2^{d-1}}T_d(3,1)+3T(P,d-1)$$

$$T_d(3,1)=C_1(2\cdot 1+2)+C_2(2\cdot 1+2+2(2\cdot 1+1)(2\cdot 3+1))+$$
$$+C_3(4\cdot 3+2+(2\cdot 3+1)(2\cdot 1+1))+C_4(2\cdot 3+2+6(2\cdot 3+1)(2\cdot 1+1))+$$
$$+C_{10}2(2\cdot 3+1)(2\cdot 1+1)+T_p()(2\cdot 3+1)(2\cdot 1+1)$$

$$T_d(3,1)=4C_1+46C_2+35C_3+132C_4+42C_{10}+21T_p()$$

$$T_p()=C_1+C_3+2C_4+2C_8+C_9+C_{10}$$

$$T_d(3,1)=4C_1+46C_2+35C_3+132C_4+42C_{10}+21(C_1+C_3+2C_4+2C_8+C_9+C_{10})$$

$$T_d(3,1)=25C_1+46C_2+56C_3+174C_4+42C_8+21C_9+63C_{10}$$

Geometrine formule $S=\frac{r^n-1}{r-1}$

$$T(p,d)=C_2+9C_3+11C_4+4C_5+C_7+9C_{10}+T_r+\frac{p}{5\cdot 2^d-5}\cdot T_d(3,1)+3T(P,d-1)$$

$$T(p,d)=C_2+9C_3+11C_4+4C_5+C_7+9C_{10}+T_r+\frac{p}{5\cdot 2^d-5}\cdot T_d(3,1)$$

$$T(p,d)=C_2+9C_3+11C_4+4C_5+C_7+C_8+2C_9+9C_{10}+3C_{11}+\frac{p}{5\cdot 2^d-5}\cdot$$
$$\cdot(25C_1+46C_2+56C_3+174C_4+42C_8+21C_9+63C_{10})$$

$$T_r()=7$$

$$T_p()=C_1+C_3+2C_4+2C_8+C_9+C_{10}=8$$

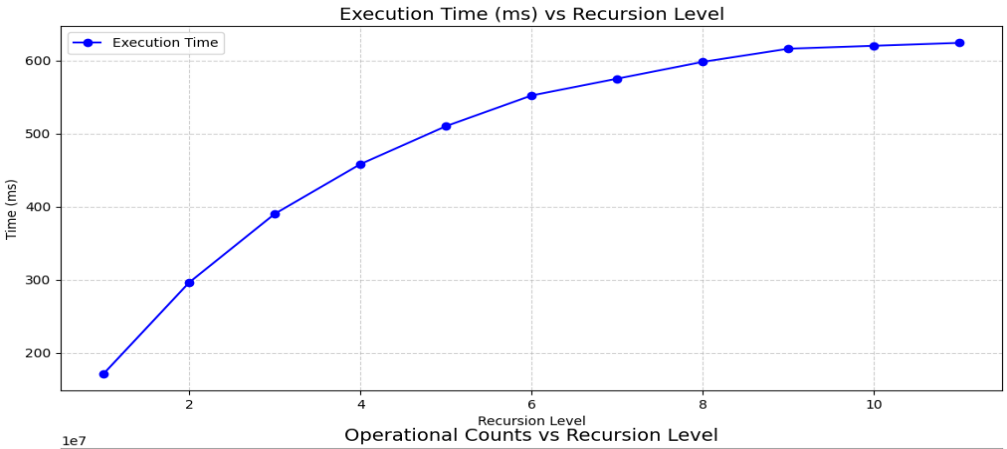
$$T_d(radX,radY)=76radXradY+48radX+38radY+27$$

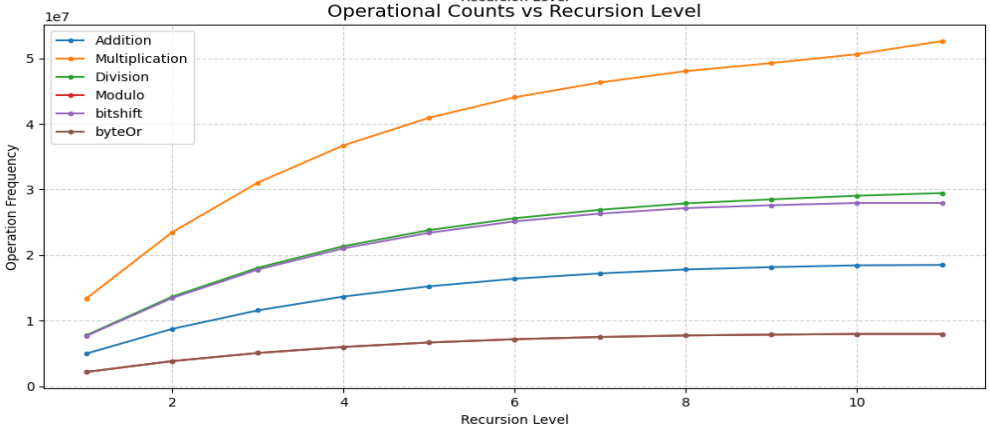
$$T_d(x,y)=178(\frac{p\cdot 0.2}{2^d})+27$$

$$T(p,d)=42+(178*(\frac{p*0.2}{2^d})+27)+3T(P,d-1)$$

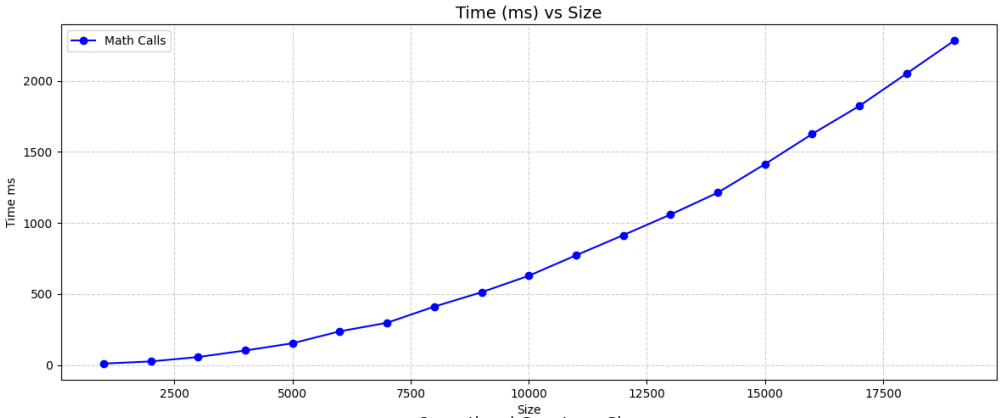
4. Programos greičio testavimas

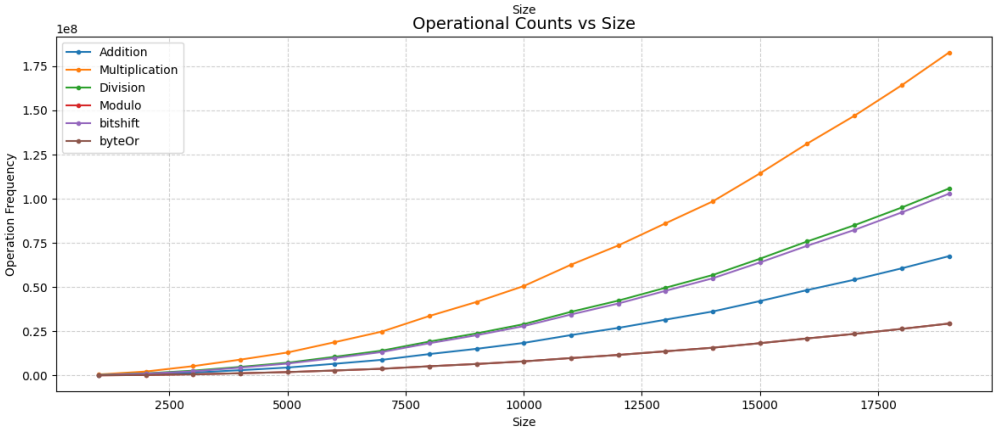
Buvo naudojamas 10000x10000 dydžio BPM failas:





Šiam testui buvo naudojama 10 rekursijos gylių:





Išvados

Šio laboratorinio darbo metu buvo išmokta dirbti su BMP, Suprasta, kaip Suprasta, kaip BMP formate spalvų informacija yra saugoma pikselių masyve. Taipogi buvo atliktas analitinis algoritmo skaičiavimas