This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.

Kaunas University of Technology
Department of Applied Informatics

Design and Analysis of Computer Algorithms

Practical work 1

Student:

Lecturers:

Kaunas, 2023

This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.

Practical Tasks:

First task: Write a program for each recurrent equation (obtained from the choice test of the task) and perform experimental and theoretical analysis:

Describe the steps of task implementation.

Recurrence expressions obtained from test:
1.
2.
3.

Second task: Using recursion and not using graphic libraries, create the BMP format of the specified structure (obtained after completing the task selection test):

Describe the steps of task implementation.

This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.

1. First task

1.1. First recurrence

1.1.1. Analysis of the code of the fist recurrence

Recurrence equation: T(n) = 2 * T($\frac{n}{9}$) + n^5

```
public static void R1(ref int[] a, int k)  //                Costs     |    Times
{
    if (k >= 1)                            //                  c1      |       1
    {
        R1(ref a, k / 9);                  //                 T(k/9)   |       1
        R1(ref a, k / 9);                  //                 T(k/9)   |       1

        for (int i = 0; i < k; i++)        //                c2 + c3   |       1
        {                                  //                c4 + c3   |       k
            for (int j = 0; j < k; j++)    //                c2 + c3   |       k
            {                              //                c4 + c3   |      k^2
                for (int l = 0; l < k; l++)    //            c2 + c3   |      k^2
                {                              //            c4 + c3   |      k^3
                    for (int m = 0; m < k; m++)     //       c2 + c3   |      k^3
                    {                               //       c4 + c3   |      k^4
                        for (int p = 0; p < k; p++)     //   c2 + c3   |      k^4
                        {                               //   c4 + c3   |      k^5
                            a[p]++;                     //      c5     |      k^5
                            s++;                        //      c6     |      k^5
                        }
                    }
                }
            }
        }
    }
    else                                   //                  c7      |       1
    {
        return;                            //                  c8      |       1
    }

}
```

We add up the costs of all code lines to determine the cost of entire program:

$$T_{R1}(k) = c_1 + T\left(\frac{k}{9}\right) + T\left(\frac{k}{9}\right) + c_2 + c_3 + k(c_4 + c_3) + k(c_2 + c_3) + k^2(c_4 + c_3) + k^2(c_2 + c_3) + k^3(c_4 + c_3) + k^3(c_2 + c_3) + k^4(c_4 + c_3) + k^4(c_2 + c_3) + k^5(c_4 + c_3) + k^5(c_2 + c_3) + k^5(c_5 + c_6) + c_7 + c_8$$

Simplified expression:

$$T_{R1}(k) = 2 * T\left(\frac{k}{9}\right) + k^5(c_4 + c_3 + c_5 + c_6) + k^4(c_2 + 2 * c_3 + c_2) + k^3(c_2 + 2 * c_3 + c_2) + k^2(c_2 + 2 * c_3 + c_2) + k(c_2 + 2 * c_3 + c_2) + c_1 + c_2 + c_3 + c_7 + c_8$$

1.1.2. Application of Master method to solve T(n) = 2 * T($\frac{n}{9}$) + n^5 *recurrence*

We have: $a = 2, b = 9, f(n) = n^5$

We can apply case 3 of the master theorem*:*

$$\lim_{n\to\infty} \frac{f(n)}{n^{\log_b a+\varepsilon}} = \infty$$

$$\lim_{n\to\infty} \frac{n^5}{n^{\log_9 2+\varepsilon}} = \lim_{n\to\infty} n^{5-\log_9 2-\varepsilon} = \infty$$

$$0 < \varepsilon < \log_9 \frac{59049}{2}$$

Case 3 applies if we can show that the regularity condition holds for *f(n)*. For sufficiently large *n*, we have that

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

$$2 * \frac{n^5}{9^5} \le cn^5$$

$$\frac{2}{9^5} \le c$$

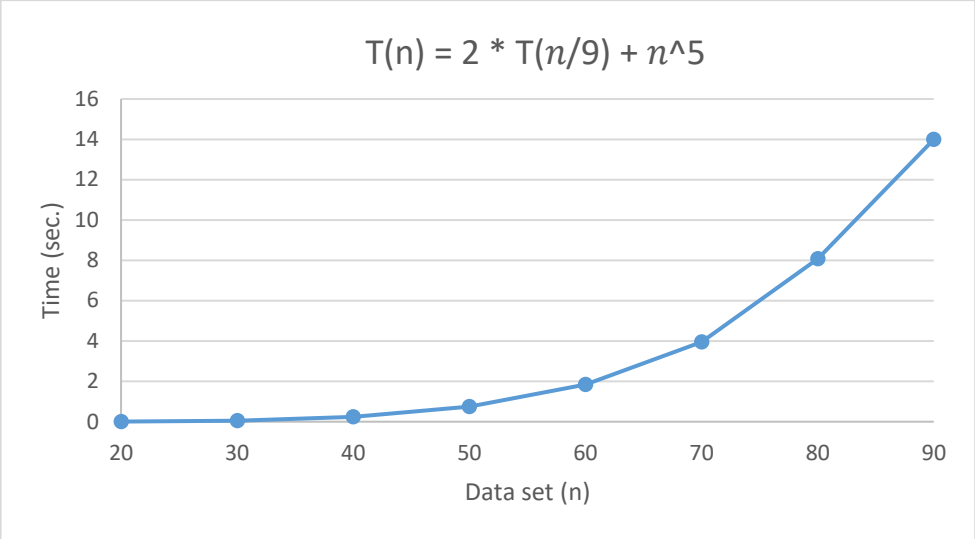
Regularity condition is satisfied. Consequently, by case 3, the solution to the recurrence is $T(n) = \theta(n^5)$.

1.1.3. Experimental analysis of the program

Data set (n)	Time	Number of code lines
20	0.007	3200064
30	0.056	24300486
40	0.236	102402048
50	0.754	312506250
60	1.835	777615552
70	3.955	1680733614
80	8.073	3276865536
90	13.995	5905100004

Table No. 1.1 Calculation time and the number of code lines of program

This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.



1.1 Experimental results of the first recurrence

1.2. Second recurrence

1.2.1. Analysis of the code of the recurrence $T(n) = T(\frac{n}{6}) + T(\frac{n}{4}) + n^2$

```
public static void R2(ref int[] a, int k)  //          Costs      |    Times
{
    if (k >= 1)                            //            c1       |      1
    {
        R2(ref a, k / 6);                  //          T(k/6)     |      1
        R2(ref a, k / 4);                  //          T(k/4)     |      1

        for (int i = 0; i < k; i++)        //          c2 + c3    |      1
        {                                  //          c4 + c3    |      k
            for (int j = 0; j < k; j++)    //          c2 + c3    |      k
            {                              //          c4 + c3    |     k^2
                a[j]++;                    //            c5       |     k^2
                s++;                       //            c6       |     k^2
            }
        }
    }
    else                                   //            c7       |      1
    {
        return;                            //            c8       |      1
    }
}
```

Sum the cost of each line to determine the total cost of the program.

$$T_{R2}(k) = c_1 + T\left(\frac{k}{6}\right) + T\left(\frac{k}{4}\right) + c_2 + c_3 + k(c_4 + c_3) + k(c_2 + c_3) + k^2(c_4 + c_3) + k^2(c_5 + c_6) + c_7 + c_8$$

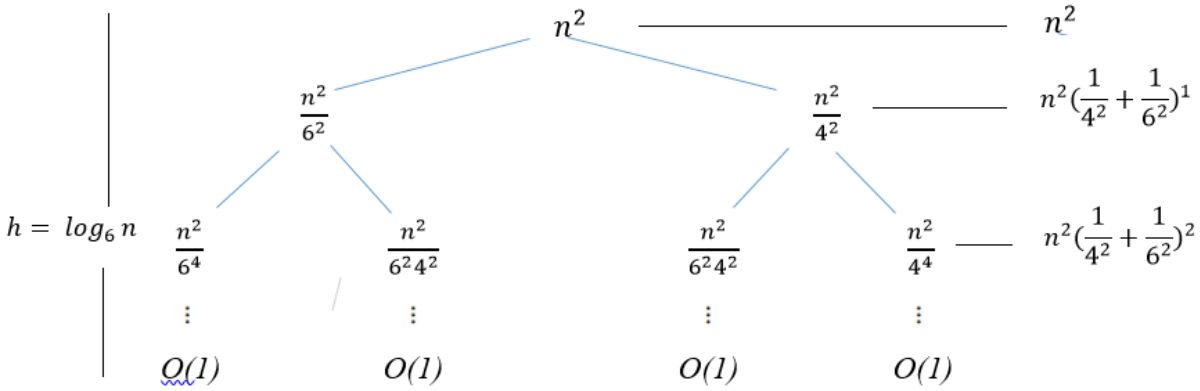
Simplified expression:

This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.

$$T_{R2}(\mathrm{k}) = \mathrm{T}\left(\frac{k}{6}\right) + \mathrm{T}\left(\frac{k}{4}\right) + k^2(c_4 + c_3 + c_5 + c_6) + k(c_2 + 2 * c_3 + c_4) + c_1 + c_2 + c_3 + c_7 + c_8$$

1.2.2. Application of recursion-tree method for solving $\mathrm{T(n)} = \mathrm{T}(\frac{n}{6}) + \mathrm{T}(\frac{n}{4}) + n^2$ recurrence

In a recursion tree, each node represents the cost of a single subproblem. Next we determine the cost at each level of the tree.



The subproblem size for a node at depth i is $n/6i$, or $i = log_6 n$. Thus, the tree has $log_6 n + 1$ levels (at depths 0; 1; 2,..., $log_6 n$), $h = log_6 n$.

We sum the costs within each level of the tree to obtain a set of per-level costs, and then we sum all the per-level costs to determine the total cost of all levels of the recursion.

$T(n) = n^2 + n^2(\frac{1}{4^2}+\frac{1}{6^2})^1 + n^2(\frac{1}{4^2}+\frac{1}{6^2})^2 + \ldots + n^2(\frac{1}{4^2}+\frac{1}{6^2})^i = n^2 \sum_{i=0}^{h}(\frac{1}{4^2}+\frac{1}{6^2})^i = n^2 \sum_{i=0}^{log_6 n}(\frac{13}{144})^i$

We can apply decreasing geometric series as an upper bound. Thus, we have:

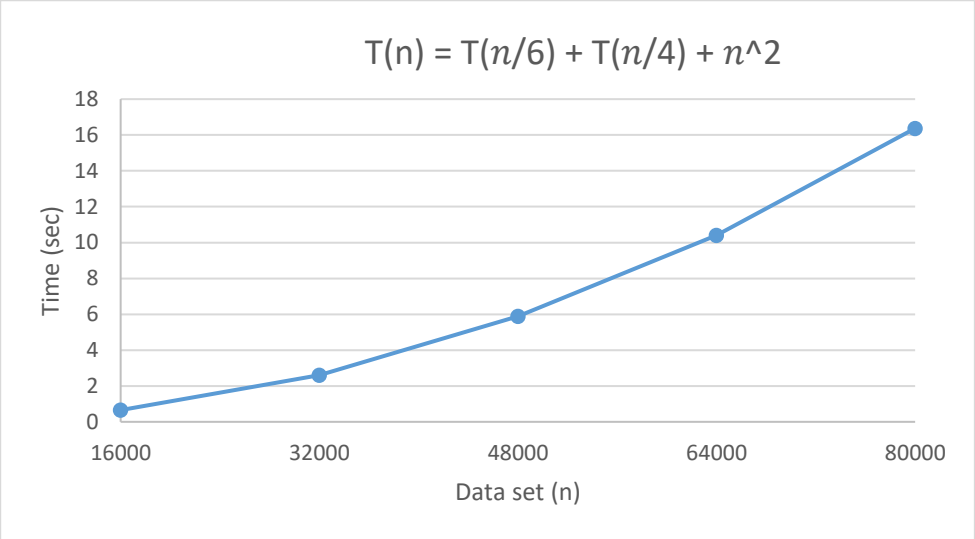
$$T(n) = n^2 \sum_{i=0}^{log_6 n}(\frac{13}{144})^i < n^2 \sum_{i=0}^{\infty}(\frac{13}{144})^i = \frac{n^2}{1-\frac{13}{144}} = \frac{144}{131}n^2 = O(n^2)$$

1.2.3. Experimental analysis of the program

Data set (n)	Time	Number of code lines

16000	0.655	281397155
32000	2.595	1125609117
48000	5.894	2532637733
64000	10.406	4502442545
80000	16.371	7035092207

Table No. 2.2 Calculation time and the number of code lines of program



1.2 Experimental results of the second recurrence

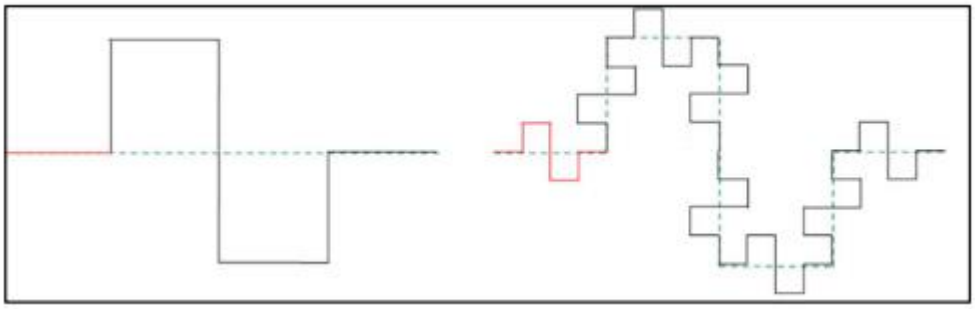
1.3. Third recurrence

Analysis of the third recurrence is similar to first and second recurrences.

This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.

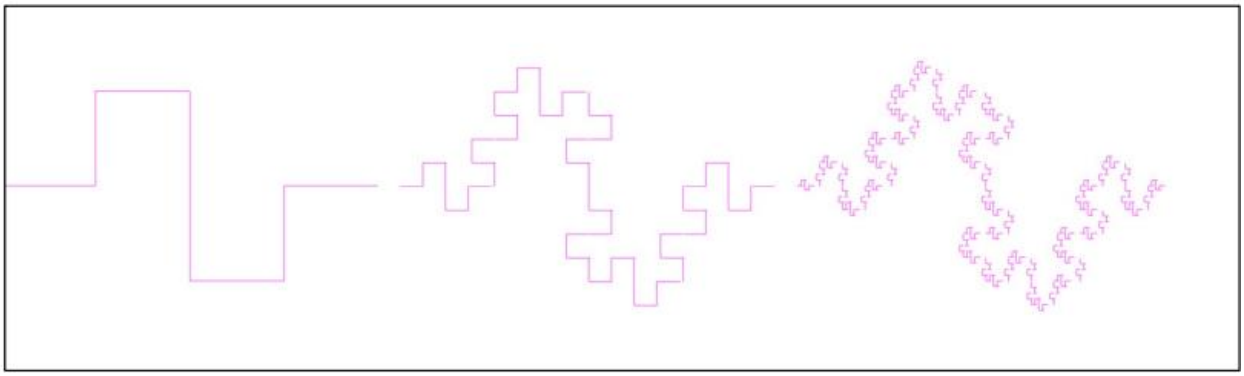
2. Second task

2.1. The task is to create a BMP format picture according to the figure presented below.



2.1 Figure of the given task

Results of the task presented below.



2.2 Results of the given task

This is a sample report outlining the step-by-step instructions that needs to be followed. Please note that in some cases, calculations have been simplified and some mathematical operations may have been omitted.

2.2. Analysis of the code

```
private static void curve (int depth, int beginning, int length, int direction)
{
   if (depth == 1)                                                                  c1   1
   {
        if (direction == 0) {                                                       c2   1
            PaintHorizontalLine(beginning, length);                                 cp   1
        }
        else if (direction == 1) {                                                  c2   1
            PaintVerticalLine(beginning, length);                                   cp   1
        }
   }
   if (depth <= 0)
   {                                                                                c3   1
        return;
   }                                                                                c4   1
   else
   {                                                                                c5   1
        length = length / 4;

        if (length == 0)                                                            c6   1
           return;                                                                  c4   1

        int divisor = Convert.ToInt32(1000 / Convert.ToDouble(length));             c7   1
        int gap = l / divisor;                                                      c8   1

        if(direction == 0) {                                                        c9   1
            curve(depth - 1, beginning, length, 0);                                 T(n/4) 1
            curve(depth - 1, beginning + gap, length, 1);                           T(n/4) 1
            curve(depth - 1, beginning + l*length + gap, length, 0);                T(n/4) 1
            curve(depth - 1, beginning + gap*2, length, 1);                         T(n/4) 1
            curve(depth - 1, beginning + gap * 2 - l*length, length, 1);            T(n/4) 1
            curve(depth - 1, beginning + gap*2 - l*length, length, 0);              T(n/4) 1
            curve(depth - 1, beginning + gap*3 - l*length, length, 1);              T(n/4) 1
            curve(depth - 1, beginning + gap*3, length, 0);                         T(n/4) 1
        }
        else if (direction == 1) {
            curve(depth - 1, beginning, length, 1);                                 c9   1
            curve(depth - 1, beginning + l * length - gap, length, 0);              T(n/4) 1
            curve(depth - 1, beginning + l * length - gap, length, 1);              T(n/4) 1
            curve(depth - 1, beginning + l * 2 * length - gap, length, 0);          T(n/4) 1
            curve(depth - 1, beginning + l * 2 * length, length, 0);                T(n/4) 1
            curve(depth - 1, beginning + l * 2 * length + gap , length, 1);         T(n/4) 1
            curve(depth - 1, beginning + l * 3 * length, length, 0);                T(n/4) 1
            curve(depth - 1, beginning + l * 3 * length, length, 1);                T(n/4) 1
        }
   }}
```

Summed up the costs of all code lines we get a recursive equation:

$$T(n) = \begin{cases} c_1 + c_4 + c_5, & n \leq 0 \\ 8T\left(\frac{n}{2}\right) + c_1 + c_2 + c_p + c_5 + c_6 + c_7 + c_8 + c_9, n > 0 \end{cases}$$

2.2.1. To sole the recurrence we can apply a Master method.

We have: $a = 8, b = 2, f(n) = c$, c – is a constant.

We can apply case 1 of the master theorem*:*

Consequently, by case 1, the solution of the recurrence is:

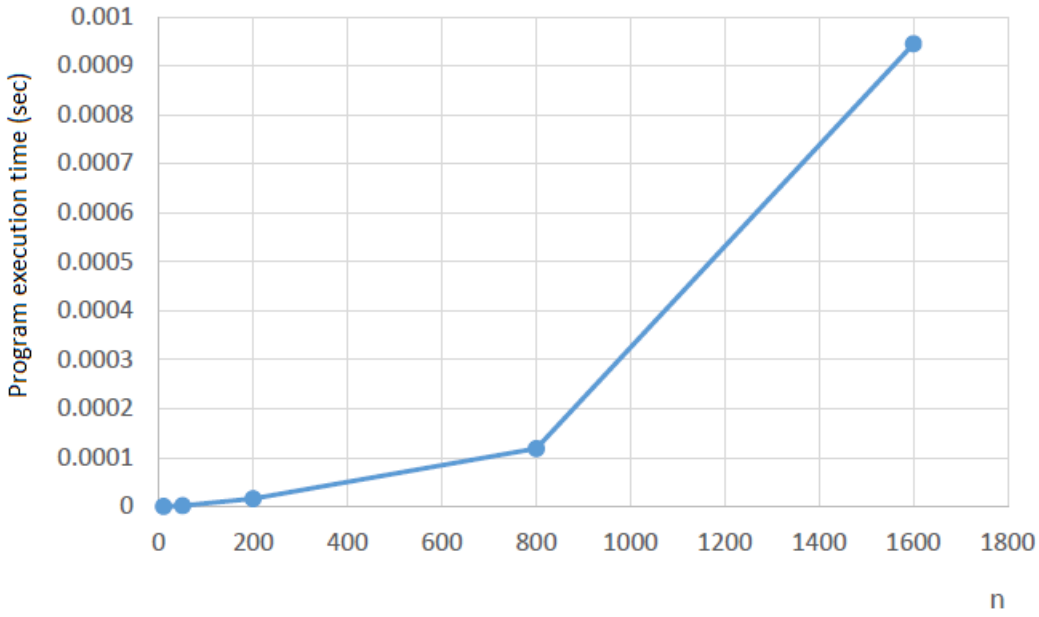
$$T(n) = \Theta(n^{\log_2 8}) = \Theta(n^3)$$

2.3. Experimental analysis

Experimental analysis were performed under a different numbers of *n* and calculation time of each execution were evaluated.

n	Program execution time
5200	0.006626
1600	0.0009451
800	0.0001184
200	0.0000166
50	0.0000022
10	0.0000006

Dependency of the program execution time and *n* is depicted in figure below.



2.3 experimental analysis