ИТМО Кафедра Информатики и прокладной математики

Отчет по лабораторной работе №5 «Поиск – сравнение бинарного поиска и поиска с помощью двоичного дерева»

Выполнил: студент группы Р3117
Плюхин Дмитрий
Проверил: Симоненко З. Г.

2016 год

1. Задание

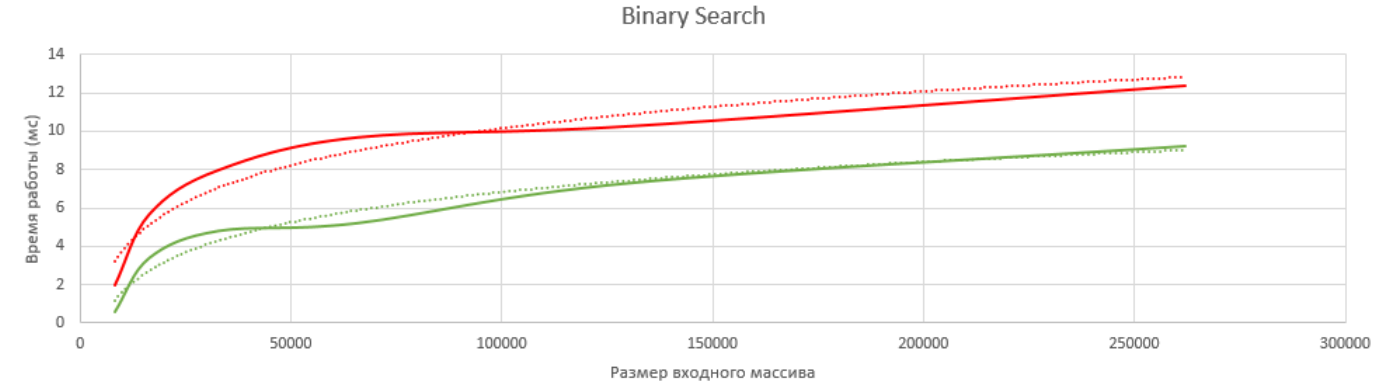
Реализовать на каком-либо языке программирования и сравнить между собой алгоритм бинарного поиска и алгоритм поиска с помощью двоичного дерева.

2. Выполнение

Для выполнения работы был выбран язык программирования C# по причине того, что алгоритмы поиска, реализуемые в работе, используют указатели, применение которых затруднительно в некоторых других языках программирования.

После реализации алгоритмов на языке программирования был проведен их запуск на различных исходных данных, в частности, на массивах разной длины. Для обоих алгоритмов представлено два графика, поскольку время работы каждого алгоритма возрастает, если элемент так и не был найден.

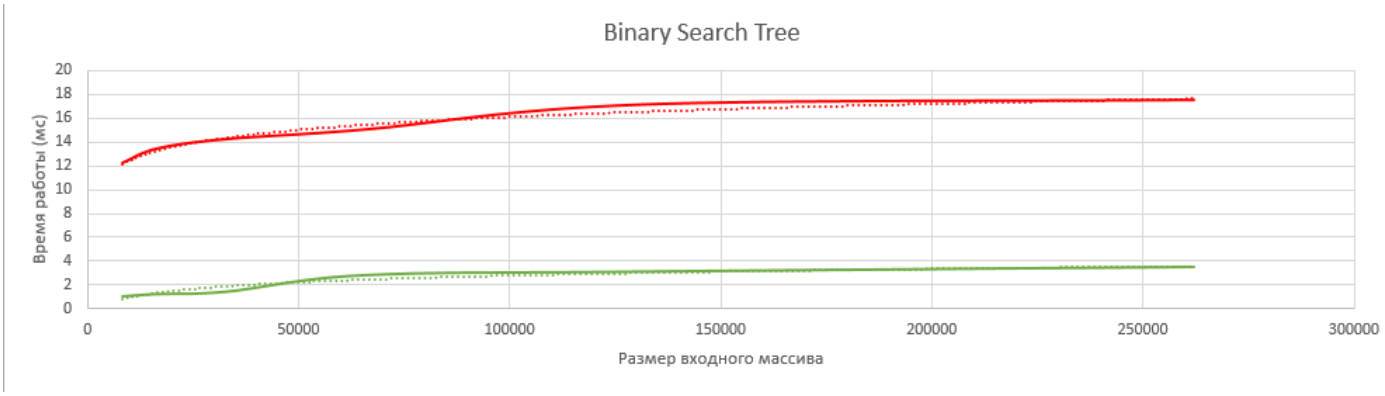
3. Результаты




```
static int BinarySearch(List<int> sortedarray, int key)
{
    int p = 0;
    int l = 0;
    int r = sortedarray.Count - 1;

    while (l <= r)
    {
        p = l + (r - l) / 2;
        if (sortedarray[p] > key)
        {
            r = p - 1;
        }
        else if (sortedarray[p] < key)
        {
            l = p + 1;

        }
        else
        {
            return p;
        }
    }
    throw new ArgumentException();
}
```



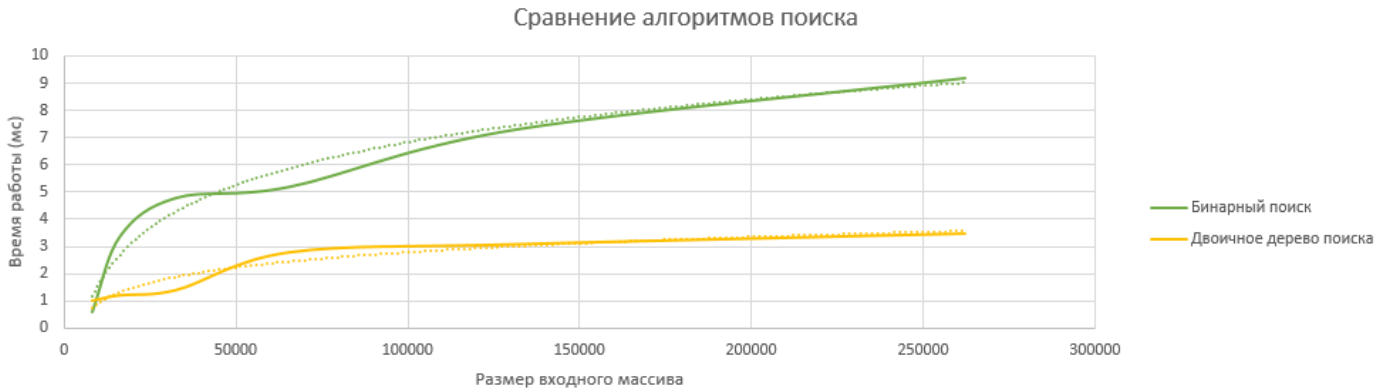
```
public static Node Find(Node root, int key)
{
    if (root == null)
    {
        throw new ArgumentException();
    }
    if (root.key == key)
    {
        return root;
    }
    if ((root.key > key) && (root.left != null))
    {
        return Find(root.left, key);
    }
    if ((root.key < key) && (root.right != null))
    {
        return Find(root.right, key);
    }
    throw new ArgumentException();
}
public static Node Insert(Node root, int key)
{
    if (root == null)
    {
        return new Node(null, null, key);
    }
    if (root.key == key)
    {
        return new Node(root.left, root.right, key);
    }
    if (root.key > key)
    {
        root.left = Insert(root.left, key);
    }
    if (root.key < key)
    {
        root.right = Insert(root.right, key);
    }
    return root;
}

public static Node FindMin(Node root)
{
    return (root.left == null) ? root.left : root;
}
public static Node RemoveKey(Node root, int key)
{
    if (root == null)
    {
        return null;
    }
    if (key < root.key)
    {
        root.left = RemoveKey(root.left, key);
    }
    else if (key > root.key)
    {
        root.right = RemoveKey(root.right, key);
    }
    else
    {
        Node l = root.left;
        Node r = root.right;
        if (r == null)
        {
            return l;
        }
        Node min = FindMin(r);
        min.right = RemoveMin(r);
        min.left = l;
        return min;
    }
    return root;
}
```

```
public static Node RemoveMin(Node root)
{
    if (root == null)
    {
        return null;
    }
    if (root.left == null)
    {
        return root.right;
    }
    root.left = RemoveMin(root.left);
    return root;
}
```

4. Анализ

Время работы (мс)		Размер массива
Бинарный поиск	Двоичное дерево поиска	
0,6	1	8192
3,4	1,2	16384
4,8	1,4	32768
5,2	2,8	65536
7,3	3,1	131072
9,2	3,5	262144



Так, бинарный поиск оказывается эффективнее поиска с использованием двоичного дерева только если объемы данных не велики. Если же требуется выполнять поиск на массивах очень больших размеров, то бинарный поиск заметно проигрывает по времени работы двоичному дереву поиска.