Лабораторная работа 2

ИССЛЕДОВАНИЕ И ОЦЕНКА АЛГОРИТМОВ ПОИСКА

Цель работы: разработка программ, реализующих различные алгоритмы поиска, и оценка их временной и пространственной сложности.

Вариант: 4.

Условие: разработать алгоритм и программу дихотомического поискаю В качестве исходных данных использовать массив целых чисел, который формируется с помощью датчика случайных чисел. Аргумент поиска - число.

Асимтотическая сложность алгоритма

1 = n / 2^x - Примечание: сколько раз нужно разделить n на 2, чтобы получить 1

$2^x = n$

$Log2(2^x) = log2(n)$

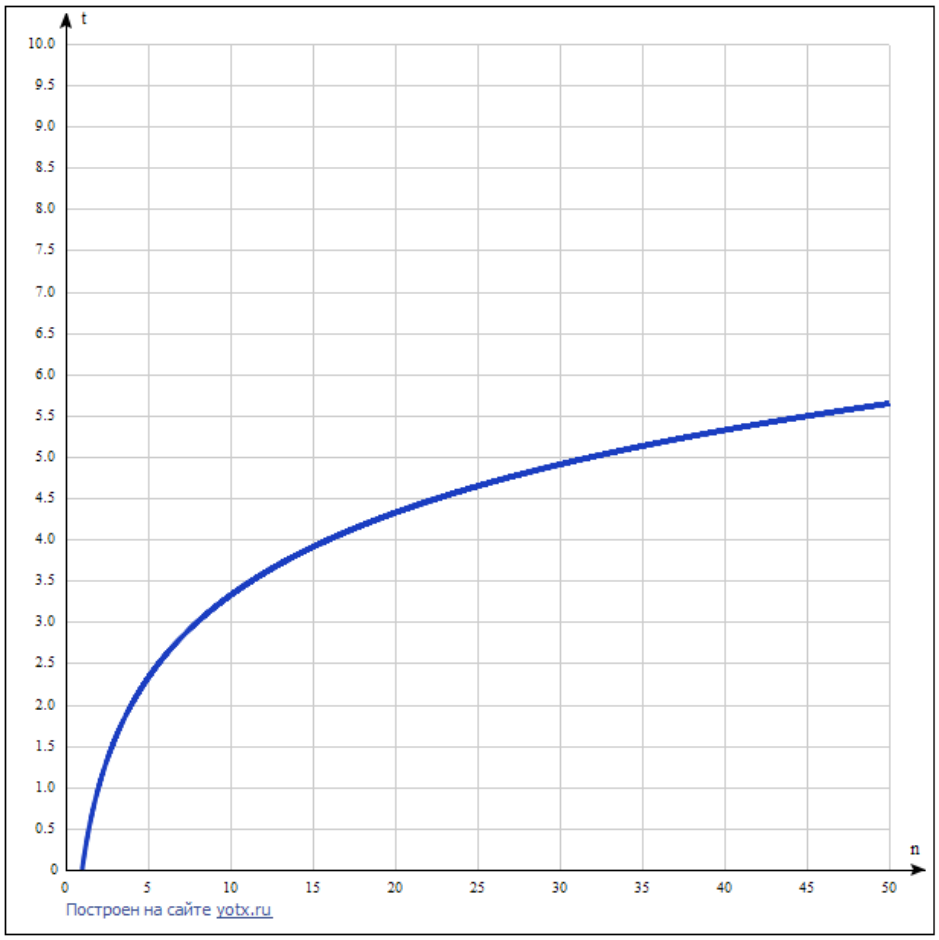
$X = log2(n)$

Асимтотическая сложность - O(n)

Временная оценка алгоритма

Размерность массива	Время работы алгоритма(в миллисекундах)
100000	3.183
75000	2.1843
50000	1.183
25000	0.7133
10000	0.4696

График зависимости времени выполнения алгоритма от объема исзодных данных



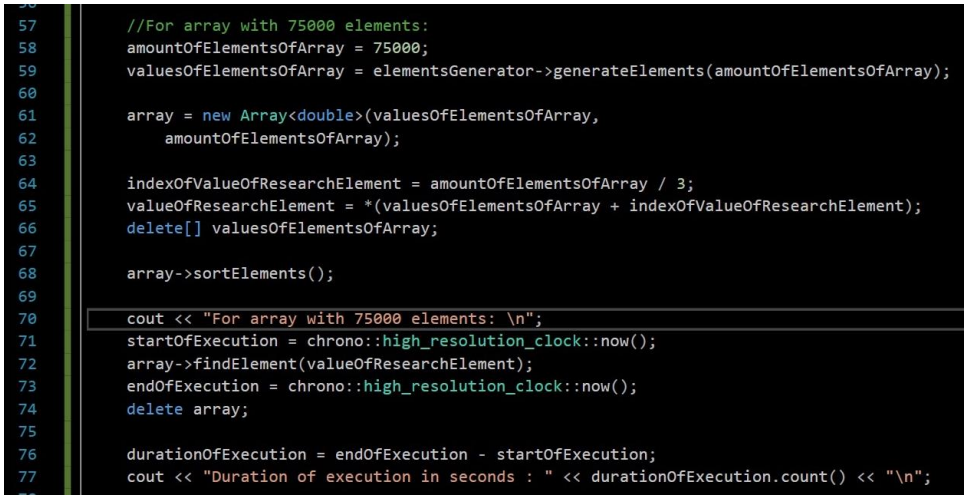
Код алгоритма

```cpp
//*************************************************************
template<class ElementType>
ElementType& BinarySearchElementMethod<ElementType>::findElement(
    const ElementType &valueOfElement, const Array<ElementType> &array) const
{
    const int startIndexOfSearching = 0;
    const int endIndexOfSearching = array.getAmountOfElements() - 1;
    return this->findElement(valueOfElement, array, startIndexOfSearching, endIndexOfSearching);
}
//*************************************************************
template<class ElementType>
ElementType& BinarySearchElementMethod<ElementType>::findElement(
    const ElementType &valueOfElement, const Array<ElementType> &array,
    const int startIndexOfSearching, const int endIndexOfSearching) const
{
    const int currentIndexOfSearching = (startIndexOfSearching + endIndexOfSearching) / 2;
    if (array[currentIndexOfSearching] == valueOfElement)
    {
        return array[currentIndexOfSearching];
    }
    else if (startIndexOfSearching >= endIndexOfSearching)
    {
        throw new BinarySearchingElementException("Impossible to find element with"
            " given value: " + std::to_string(valueOfElement) + ".");
    }
    else
    {
        return (valueOfElement < array[currentIndexOfSearching])
            ? this->findElement(valueOfElement, array, startIndexOfSearching,
                currentIndexOfSearching - 1)
            : this->findElement(valueOfElement, array, currentIndexOfSearching + 1,
                endIndexOfSearching);
    }
}
//*************************************************************
```

Исходный код программы эксперементальной оценки временной сложности алгоритма для массива с 100000 элементами

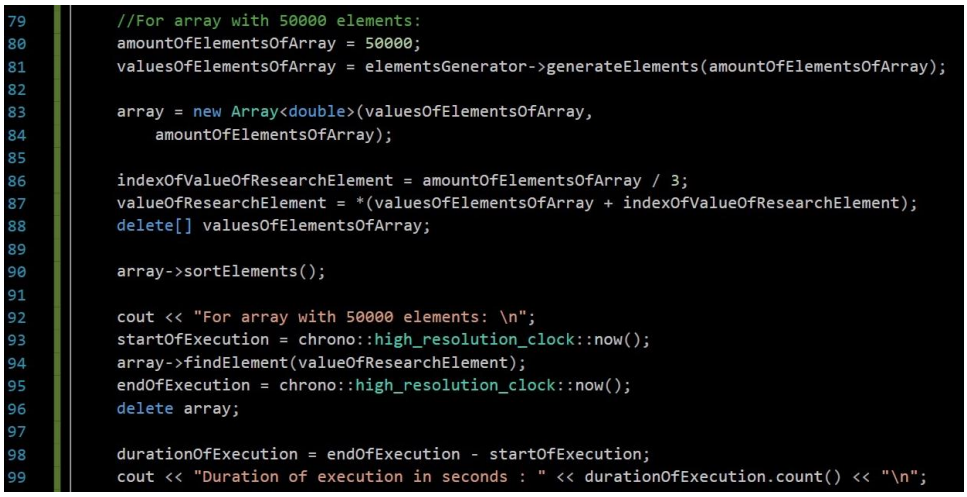
```cpp
    //For array with 100000 elements:
    int amountOfElementsOfArray = 100000;

    const double *valuesOfElementsOfArray
        = elementsGenerator->generateElements(amountOfElementsOfArray);

    Array<double> *array = new Array<double>(valuesOfElementsOfArray,
        amountOfElementsOfArray);

    int indexOfValueOfResearchElement = amountOfElementsOfArray / 3;
    double valueOfResearchElement
        = *(valuesOfElementsOfArray + indexOfValueOfResearchElement);
    delete[] valuesOfElementsOfArray;

    array->sortElements();

    MethodOfSearchingElement<double> * const methodOfSearchingElement
        = new BinarySearchElementMethod<double>();
    array->setMethodOfSearchingElement(methodOfSearchingElement);

    cout << "For array with 100000 elements: \n";
    chrono::steady_clock::time_point startOfExecution
        = chrono::high_resolution_clock::now();
    array->findElement(valueOfResearchElement);
    chrono::steady_clock::time_point endOfExecution
        = chrono::high_resolution_clock::now();
    delete array;

    chrono::duration<double> durationOfExecution = endOfExecution - startOfExecution;
    cout << "Duration of execution in seconds : " << durationOfExecution.count() << "\n";
```

Исходный код программы эксперементальной оценки временной сложности алгоритма для массива с 75000 элементами



```
//For array with 75000 elements:
amountOfElementsOfArray = 75000;
valuesOfElementsOfArray = elementsGenerator->generateElements(amountOfElementsOfArray);

array = new Array<double>(valuesOfElementsOfArray,
    amountOfElementsOfArray);

indexOfValueOfResearchElement = amountOfElementsOfArray / 3;
valueOfResearchElement = *(valuesOfElementsOfArray + indexOfValueOfResearchElement);
delete[] valuesOfElementsOfArray;

array->sortElements();

cout << "For array with 75000 elements: \n";
startOfExecution = chrono::high_resolution_clock::now();
array->findElement(valueOfResearchElement);
endOfExecution = chrono::high_resolution_clock::now();
delete array;

durationOfExecution = endOfExecution - startOfExecution;
cout << "Duration of execution in seconds : " << durationOfExecution.count() << "\n";
```

Исходный код программы эксперементальной оценки временной сложности алгоритма для массива с 50000 элементами



```
//For array with 50000 elements:
amountOfElementsOfArray = 50000;
valuesOfElementsOfArray = elementsGenerator->generateElements(amountOfElementsOfArray);

array = new Array<double>(valuesOfElementsOfArray,
    amountOfElementsOfArray);

indexOfValueOfResearchElement = amountOfElementsOfArray / 3;
valueOfResearchElement = *(valuesOfElementsOfArray + indexOfValueOfResearchElement);
delete[] valuesOfElementsOfArray;

array->sortElements();

cout << "For array with 50000 elements: \n";
startOfExecution = chrono::high_resolution_clock::now();
array->findElement(valueOfResearchElement);
endOfExecution = chrono::high_resolution_clock::now();
delete array;

durationOfExecution = endOfExecution - startOfExecution;
cout << "Duration of execution in seconds : " << durationOfExecution.count() << "\n";
```

Исходный код программы эксперементальной оценки временной сложности алгоритма для массива с 25000 элементами

```
101     //For array with 25000 elements:
102     amountOfElementsOfArray = 25000;
103     valuesOfElementsOfArray = elementsGenerator->generateElements(amountOfElementsOfArray);
104
105     array = new Array<double>(valuesOfElementsOfArray,
106         amountOfElementsOfArray);
107
108     indexOfValueOfResearchElement = amountOfElementsOfArray / 3;
109     valueOfResearchElement = *(valuesOfElementsOfArray + indexOfValueOfResearchElement);
110     delete[] valuesOfElementsOfArray;
111
112     array->sortElements();
113
114     cout << "For array with 25000 elements: \n";
115     startOfExecution = chrono::high_resolution_clock::now();
116     array->findElement(valueOfResearchElement);
117     endOfExecution = chrono::high_resolution_clock::now();
118     delete array;
119
120     durationOfExecution = endOfExecution - startOfExecution;
121     cout << "Duration of execution in seconds : " << durationOfExecution.count() << "\n";
```

Исходный код программы эксперементальной оценки временной сложности алгоритма для массива с 10000 элементами

```
123     //For array with 10000 elements:
124     amountOfElementsOfArray = 75000;
125     valuesOfElementsOfArray = elementsGenerator->generateElements(amountOfElementsOfArray);
126
127     array = new Array<double>(valuesOfElementsOfArray,
128         amountOfElementsOfArray);
129
130     indexOfValueOfResearchElement = amountOfElementsOfArray / 3;
131     valueOfResearchElement = *(valuesOfElementsOfArray + indexOfValueOfResearchElement);
132     delete[] valuesOfElementsOfArray;
133
134     array->sortElements();
135
136     cout << "For array with 10000 elements: \n";
137     startOfExecution = chrono::high_resolution_clock::now();
138     array->findElement(valueOfResearchElement);
139     endOfExecution = chrono::high_resolution_clock::now();
140     delete array;
141
142     durationOfExecution = endOfExecution - startOfExecution;
143     cout << "Duration of execution in seconds : " << durationOfExecution.count() << "\n";
```

Полный проект: https://github.com/vladzuev00/search-methods-project-for-bsu