

# Алгоритмы и структуры данных

Семинар 2  
Алгоритмы сортировки









Вопросы по лекции?





## Что будет на уроке сегодня

-  Задача 1: Реализуем простой алгоритм сортировки
-  Задача 2: Реализуем быструю сортировку
-  Задача 3: Делаем сравнение времени выполнения
-  Задача 4: Реализуем алгоритм бинарного поиска



# Реализуем простой алгоритм сортировки



## Реализуем простой алгоритм сортировки

Необходимо написать один из простых алгоритмов сортировки, имеющий сложность  $O(N^2)$ .

- Сортировка пузырьком
- Сортировка выбором
- Сортировка вставками



**10 минут**



# Решение





## Сортировка пузырьком

```
public class BubbleSort {  
    public static void sort(int[] array) {  
        boolean needSort;  
        do {  
            needSort = false;  
            for (int i = 0; i < array.length - 1; i++) {  
                if (array[i] > array[i + 1]) {  
                    int temp = array[i];  
                    array[i] = array[i + 1];  
                    array[i + 1] = temp;  
                    needSort = true;  
                }  
            }  
        } while (needSort);  
    }  
}
```



## Сортировка выбором

```
public class DirectSort {  
    public static void sort(int[] array) {  
        for (int i = 0; i < array.length; i++) {  
            int minPosition = i;  
            for (int j = i + 1; j < array.length; j++) {  
                if (array[j] < array[minPosition]) {  
                    minPosition = j;  
                }  
            }  
            if (minPosition != i) {  
                int temp = array[i];  
                array[i] = array[minPosition];  
                array[minPosition] = temp;  
            }  
        }  
    }  
}
```





## Сортировка вставками

```
public class InsertionSort {  
    public static void sort(int[] array){  
        for (int i = 0; i < array.length; i++) {  
            for (int j = i + 1; j < array.length; j++) {  
                if (array[j] < array[i]) {  
                    int temp = array[i];  
                    array[i] = array[j];  
                    array[j] = temp;  
                }  
            }  
        }  
    }  
}
```



# Реализуем быструю сортировку



## Реализуем быструю сортировку

Необходимо написать алгоритм быстрой сортировки, имеющий сложность  $O(N \log(N))$ .

- Опорный элемент (pivot) берем за середину отрезка
- Решение рекурсивное (алгоритм “Разделяй и властвуй”)



**20 минут**



# Решение





## Алгоритм быстрой сортировки

```
public class QuickSort {
    public static void sort(int[] array) {
        sort(array, 0, array.length - 1);
    }
    public static void sort(int[] array, int startPosition, int endPosition) {
        int leftPosition = startPosition;
        int rightPosition = endPosition;
        int pivot = array[(startPosition + endPosition) / 2];
        do {
            while (array[leftPosition] < pivot) {
                leftPosition++;
            }
            while (array[rightPosition] < pivot) {
                rightPosition--;
            }
            if (leftPosition <= rightPosition) {
                if (leftPosition < rightPosition) {
                    int temp = array[leftPosition];
                    array[leftPosition] = array[rightPosition];
                    array[rightPosition] = temp;
                }
                leftPosition++;
                rightPosition--;
            }
        } while (leftPosition <= rightPosition);
        if (leftPosition < endPosition) {
            sort(array, leftPosition, endPosition);
        }
        if (startPosition < rightPosition) {
            sort(array, startPosition, rightPosition);
        }
    }
}
```



## Алгоритм сортировки слиянием

```
public class MergeSort {
    public static void main(String[] args){
        int[] a = {5,4,3,2,1};
        sort(a);
        for(int i=0; i<a.length; i++)
            System.out.print(a[i]+" ");
    }
    public static void sort(int[] array) {
        sort(array, 0, array.length - 1);
    }
    public static void sort(int[] array, int startPosition, int endPosition) {
        if(startPosition == endPosition)
            return ;
        int midPosition = (startPosition + endPosition) / 2;
        sort(array, startPosition, midPosition);
        sort(array, midPosition+1, endPosition);
        int[] buff = new int[array.length];
        int i = startPosition, j = midPosition+1, pos = startPosition;
        while(i <= midPosition && j<=endPosition){
            if(array[i] < array[j]){
                buff[pos] = array[i];
                i++;
                pos++;
            }else{
                buff[pos] = array[j];
                j++;
                pos++;
            }
        }
        while(i <= midPosition){
            buff[pos] = array[i];
            i++;
            pos++;
        }
        while(j <= endPosition){
            buff[pos] = array[j];
            j++;
            pos++;
        }
        for(int k = startPosition; k<=endPosition; k++){
            array[k] = buff[k];
        }
    }
}
```



Делаем сравнение  
времени выполнения



## Делаем сравнение времени выполнения

Пишем тесты для сравнения производительности сортировки больших массивов. Для наглядного результата стоит сравнивать массивы до 100 000 элементов. При таком подходе будет явно видно, какая из сортировок окажется быстрее.

- Создать массив из 100000 случайных чисел
- Определить время работы квадратичной сортировки
- Определить время работы быстрой сортировки



**15 минут**





# Решение





## Делаем сравнение времени выполнения

```
1 import java.util.Date;
2 public class Main {
3     public static void main(String[] args) {
4         for (int i = 10000; i <= 100000; i += 10000) {
5             int[] array = new int[i];
6             for (int j = 0; j < array.length; j++) {
7                 array[j] = (int) (Math.random() * 10000);
8             }
9             int[] array2 = new int[array.length];
10            for(int k=0; k<array.length; k++)
11                array2[k] = array[k];
12
13            Date startDate = new Date();
14            BubbleSort.sort(array);
15            Date endDate = new Date();
16            long bubbleSortDuration = endDate.getTime() - startDate.getTime();
17
18            startDate = new Date();
19            QuickSort.sort(array2);
20            endDate = new Date();
21            long quickSortDuration = endDate.getTime() - startDate.getTime();
22
23            System.out.printf("i: %s, bubble sort duration: %s, quick sort duration: %s\n",
24                i, bubbleSortDuration, quickSortDuration);
25        }
26    }
```



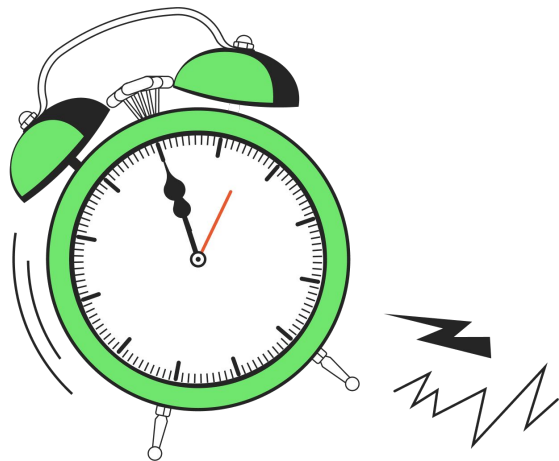
# Реализуем алгоритм бинарного поиска



## Реализуем алгоритм бинарного поиска

После успешной сортировки массива на нем можно использовать бинарный поиск. Необходимо реализовать алгоритм бинарного поиска по элементам.

- `public static int BinarySearch(int[] array, int value);`



**10 минут**



# Решение





## Бинарный поиск

```
1 public class BinarySearch {
2     public static int find(int[] array, int value){
3         int left = 0, right = array.length - 1;
4         while(right - left > 1){
5             int mid = (left + right) / 2;
6             if(array[mid] < value)
7                 left = mid;
8             else
9                 right = mid;
10        }
11        if(array[left] == value)
12            return left;
13        if(array[right] == value)
14            return right;
15        return -1;
16    }
17 }
```



# Домашнее задание





## Пирамидальная сортировка

Реализовать алгоритм пирамидальной сортировки (сортировка кучей).







Вопросы?





Спасибо за внимание