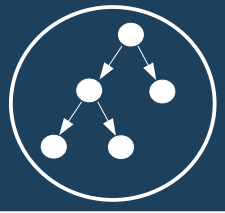


Data Structures and Algorithms

Алгоритмы.
Быстрая сортировка. Разбиение Ломуто

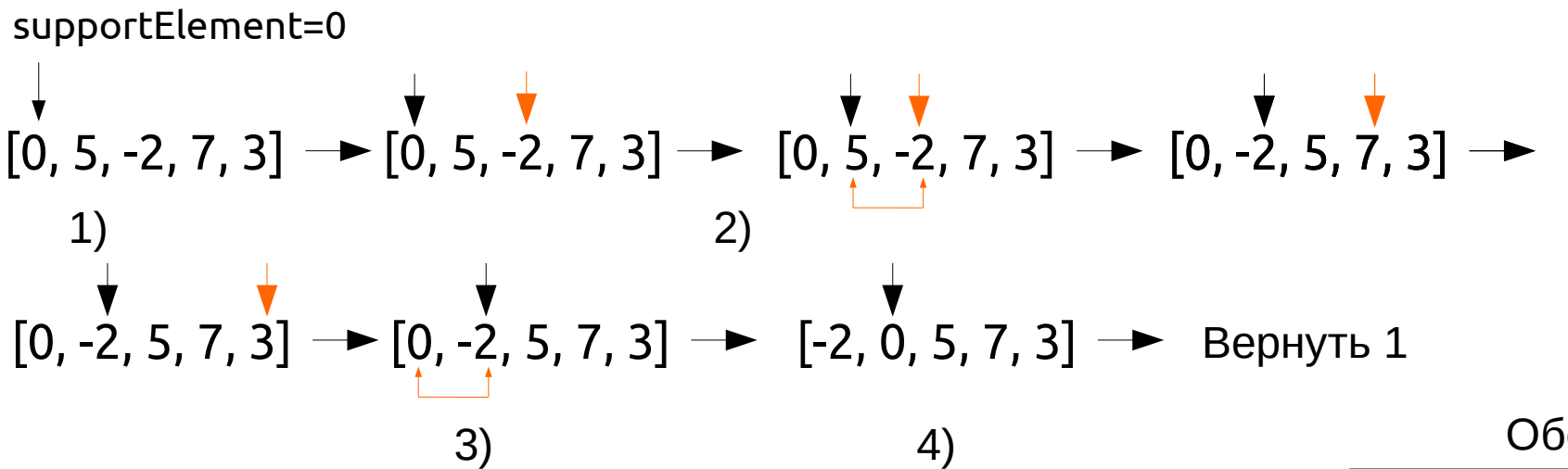
Сведение о алгоритме

Сложность по времени в наихудшем случае $O(n \cdot \ln(n))$

Разбиение Ломуто

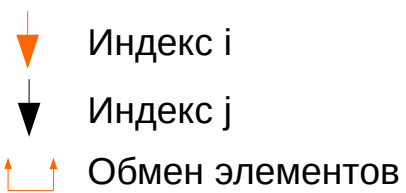
1) В качестве опорного элемента выбирается первый элемент последовательности (supportElement). Объявляется две переменных для хранения индексов (в дальнейшем i и j). Значение j равно первому индексу в подпоследовательности. Переходим к пункту **1**.
2) Начиная от начала последовательности проводиться поиск элемента для которого sequince[i]<supportElement. Если такой элемент найден, то увеличиваем значение j на одну единицу, производим обмен элементов которые стоят на i и j индексе.
3) Проводим обмен первого элемента последовательности и элемента на индексе j. Возвращаем значение индекса j и заканчиваем текущее разбиение.

Графическое пояснение алгоритма разбиения подпоследовательностей Ломуто



1) Выбираем в качестве опорный элемент первый.
2) Начиная с начала последовательности ищем элемент который меньше опорного (i=2) увеличиваем j а единицу, меняем их местами.
3) меняем местами j и опорный элемент
4) Заканчиваем возвращаем j=1

Обозначения



Python

Реализация алгоритма на Python

Функция для разбиения подпоследовательностей

```
def partition(sequince, lo_index, hi_index):
    support_element = sequince[lo_index]
    j = lo_index
    for i in range(lo_index + 1, hi_index + 1):
        if sequince[i] < support_element:
            j += 1
            sequince[i], sequince[j] = sequince[j], sequince[i]
    sequince[lo_index], sequince[j] = sequince[j], sequince[lo_index]
    return j
```

Реализация алгоритма быстрой сортировки

```
def quick_sort(sequince, lo_index=None, hi_index=None):
    if lo_index is None:
        lo_index = 0
    if hi_index is None:
        hi_index = len(sequince)-1
    if lo_index >= hi_index:
        return None
    h = partition(sequince, lo_index, hi_index)
    quick_sort(sequince, lo_index, h-1)
    quick_sort(sequince, h+1, hi_index)
```

Java

Реализация алгоритма на Java

Метод для разбиения подмассивов

```
public static int breakPartition(int[] array, int lo, int hi) {
    int j = lo;
    int supportElement = array[lo];
    for (int i = lo + 1; i <= hi; i++) {
        if (array[i] < supportElement) {
            j+=1;
            swap(array, i, j);
        }
    }
    swap(array, lo, j);
    return j;
}
```

Метод для для обмена местами элементов массива

```
public static void swap(int[] array, int i, int j) {
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
```

Реализация алгоритма на Java

Метод для запуска рекурсивного метода сортировки

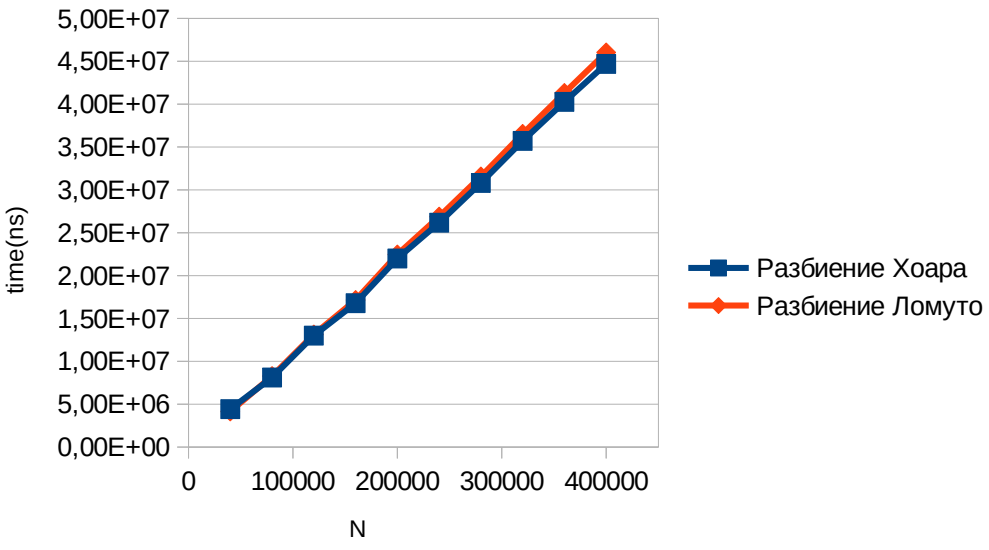
```
public static void quickSort(int[] array) {
    quickSort(array, 0, array.length - 1);
}
```

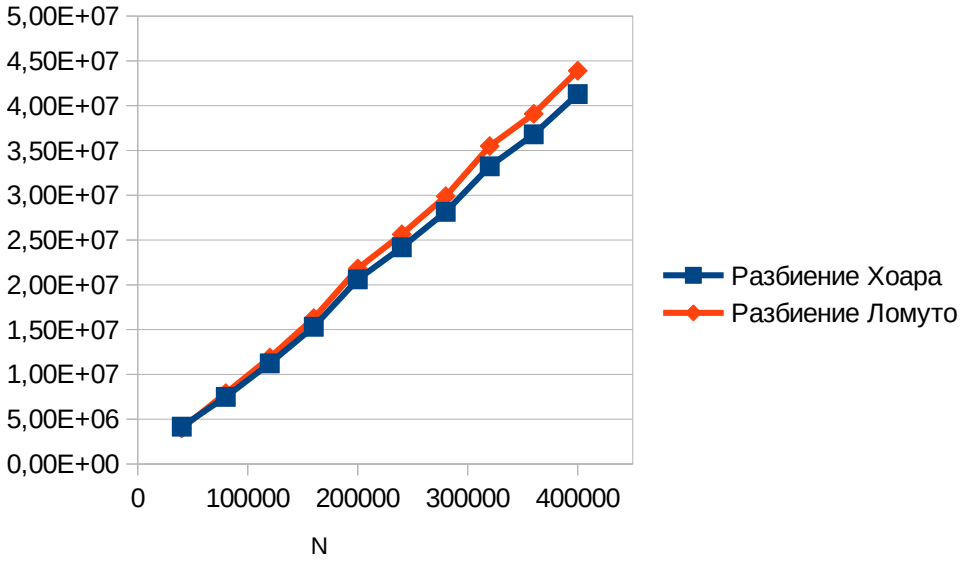
Рекурсивный метод реализующий быструю сортировку

```
public static void quickSort(int[] array, int lo, int hi) {
    if (lo >= hi) {
        return;
    }
    int h = breakPartition(array, lo, hi);
    quickSort(array, lo, h - 1);
    quickSort(array, h + 1, hi);
}
```

Вычислительный эксперимент

Интересным вопросом будет определение какой из предложенных алгоритмов разбиения оптимальнее. Для определения этого проведем вычислительный эксперимент. Реализуем оба алгоритма на Java и замерим среднее время сортировки массива для них. На рисунке вы видите зависимость среднего времени сортировки от размера массива.





Разбиение Ломуто показывает сравнимую эффективность если в сортируемой последовательности нет одинаковых элементов, но быстро деградирует до квадратичной сложности если в последовательности значительное количество равных элементов.

Список литературы

1) Д. Кнут. Искусство программирования. Том 3. «Сортировка и поиск», 2-е изд. ISBN 5-8459-0082-4
2) Роберт Седжвик, Кевин Уэйн «Алгоритмы на java 4-е издание» Пер. с англ. - М. : ООО "И.Д. Вильямс", 2013. ISBN 978-5-8459-1781-2.