



Data Structures and Algorithms

Алгоритмы.

Быстрая сортировка. Разбиение Хоара



Сведение о алгоритме

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



Сведение о алгоритме и его авторе

Быстрая сортировка была разработана английским ученым Тони Хоаром в 1960 году. Этот алгоритм также как и алгоритм сортировки слиянием использует подход «разделяй и властвуй».

Сэр Чарльз Энтони Ричард Хоар (1934 г. р.) английский ученый работающий в области информатики и вычислительной техники известен своими работами в области разработки алгоритмов и средств для определения корректности программ.



Charles Antony Richard Hoare
(1934)



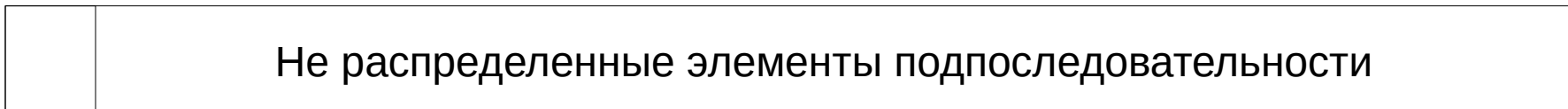
Описание алгоритма

- 1) Выбрать в последовательности элемент, называемым опорным. Этот элемент выбирается произвольным образом (автор алгоритма предлагает выбирать первый элемент последовательности).
- 2) В последовательности сравнить все остальные элементы с опорным и переставить их в последовательности так, чтобы разбить ее на две непрерывных области, следующих друг за другом: «элементы меньше опорного» и «большие». Рекурсивно вызвать пункт 1 сначала для подпоследовательности где хранятся элементы меньше опорного, после для подпоследовательности с элементами больше опорного. Условие прекращения рекурсии это размер просматриваемой подпоследовательности равный 1.



Объяснение принципа разбиения подпоследовательности

Опорный элемент



Опорный элемент





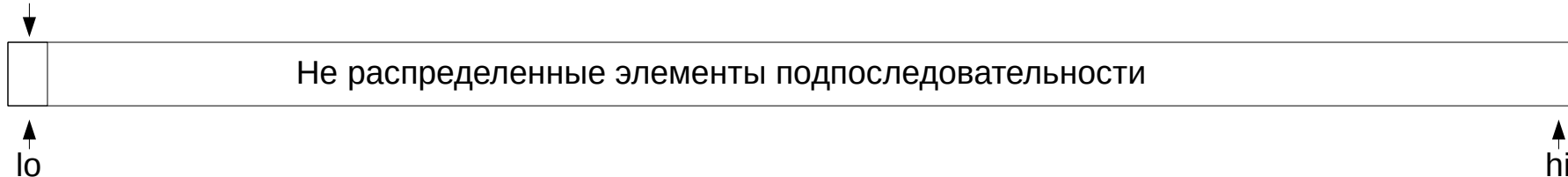
Разбиение Хоара

- 1) В качестве опорного элемента выбирается первый элемент последовательности (supportElement). Объявляются две переменных для хранения индексов (в дальнейшем i и j). Переходим к пункту **1**.
- 2) Начиная от начала последовательности проводится поиск элемента для которого $sequence[i] > supportElement$. Начиная от конца последовательности проводится поиск элемента для которого $sequence[j] < supportElement$. Выполняем проверку если $i \geq j$ то переходим к пункту **3**. В противном случае проводим обмен элементов на i и j индексе. Увеличиваем i на единицу. Уменьшаем j на единицу и продолжаем поиск.
- 3) Проводим обмен первого элемента последовательности и элемента на индексе j . Возвращаем значение индекса j и заканчиваем текущее разбиение.

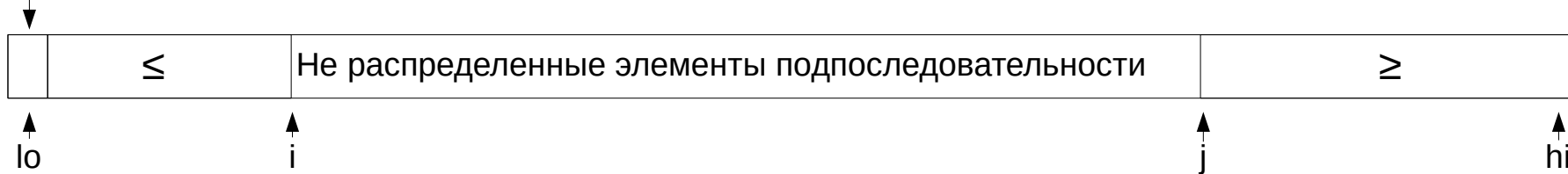


Объяснение принципа разбиения подпоследовательности

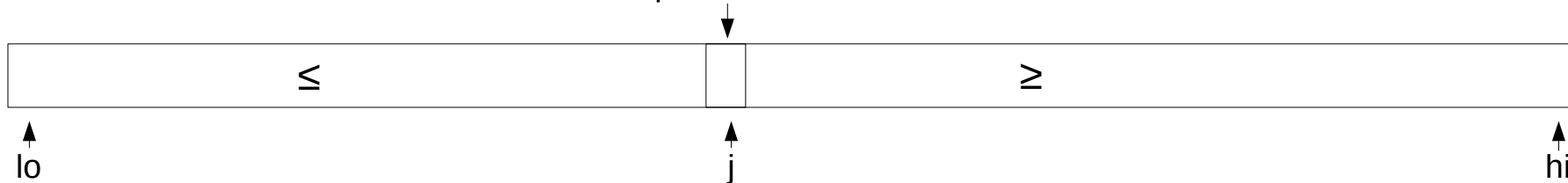
Опорный элемент



Опорный элемент



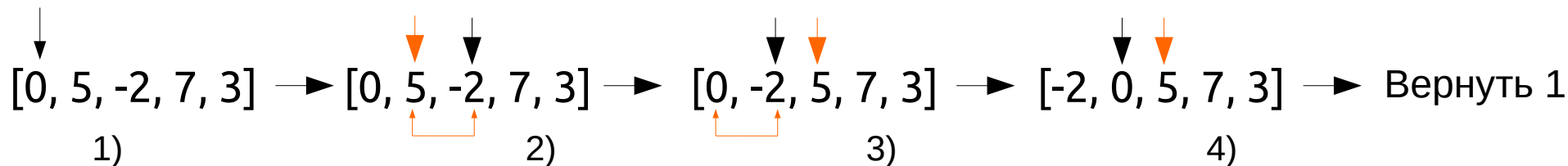
Опорный элемент








Графическое пояснение алгоритма разбиения подпоследовательностей Хоара

supportElement=0



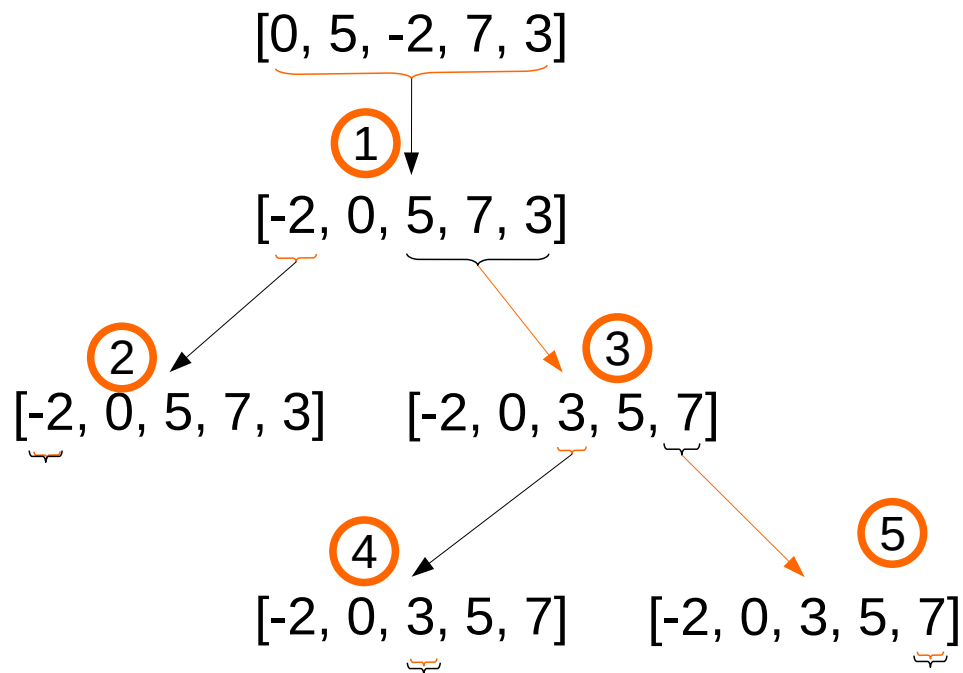
- 1) Выбираем в качестве опорный элемент первый.
- 2) Начиная с начала последовательности ищем элемент который больше опорного ($i=1$) и с конца элемент меньше опорного ($j=2$). Меняем их местами. Увеличиваем i на единицу. Уменьшаем j на единицу.
- 3) $i \geq j$ меняем местами j и опорный элемент
- 4) Заканчиваем возвращаем $j=1$

Обозначения

-  Индекс i
-  Индекс j
-  Обмен элементов



Графическая иллюстрация работы алгоритма



Обозначения



Сливаемые подпоследовательности

[-2, 0, 5, 3, 7] Сортируемая последовательность



Реализация алгоритма на Python



Функция для разбиения подпоследовательностей предложенная Хоаром

```
def partition(sequence, lo_index, hi_index):
    support_element = sequence[lo_index]
    i = lo_index + 1
    j = hi_index
    while True:
        while i < hi_index and sequence[i] < support_element:
            i += 1
        while sequence[j] > support_element:
            j -= 1
        if i >= j:
            break
        sequence[i], sequence[j] = sequence[j], sequence[i]
        i += 1
        j -= 1
    sequence[lo_index], sequence[j] = sequence[j], sequence[lo_index]
    return
```



Python

Реализация алгоритма быстрой сортировки

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



Java

Реализация алгоритма на Java



Метод для разбиения подмассивов предложенный Хоаром

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



Метод 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);  
}
```

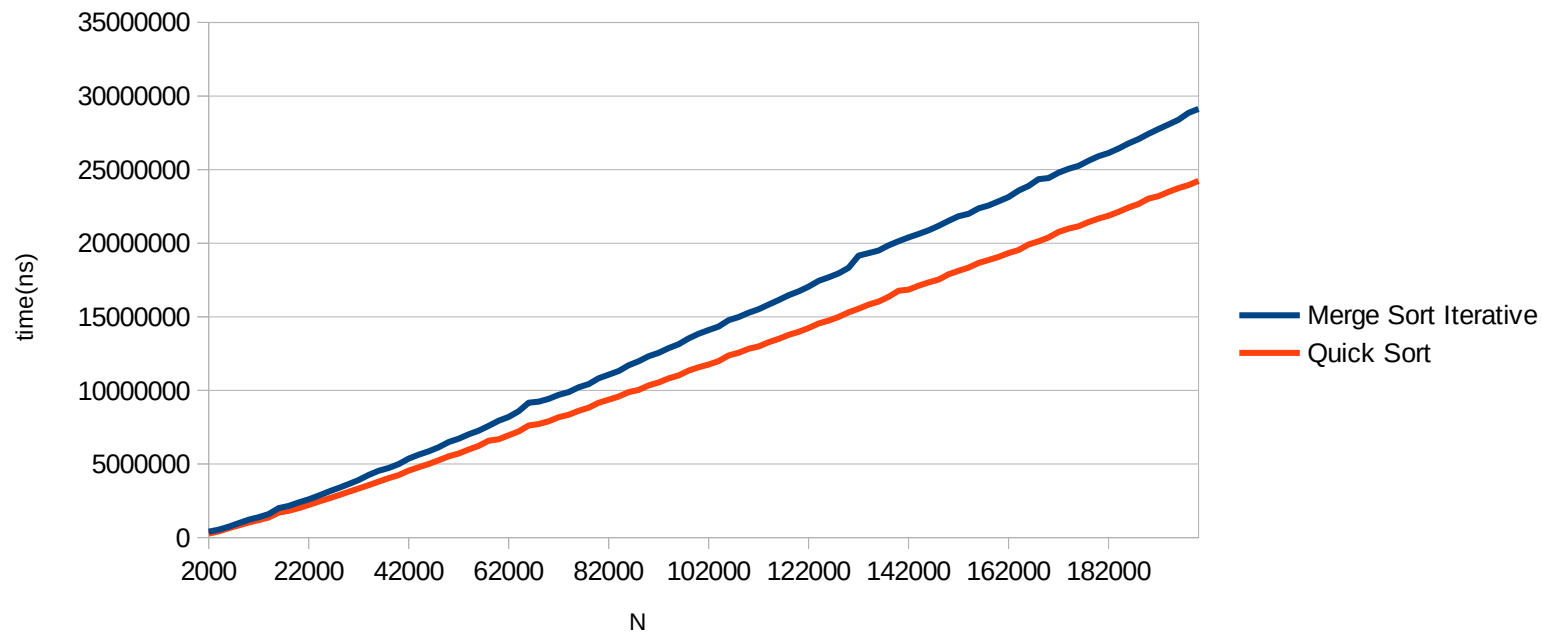
Рекурсивный метод реализующий быструю сортировку

```
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

Для проверки эффективности алгоритма быстрой сортировки был проведен вычислительный эксперимент. Провели сравнение скоростей сортировки массивов разного размера для алгоритма быстрой сортировки и алгоритма сортировки слиянием. Оба алгоритма были реализованы на Java. На графике ниже вы можете видеть зависимость времени сортировки от размера массива.





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.