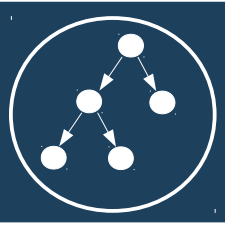


Data Structures and Algorithms

Алгоритмы.
Сортировка слиянием. Рекурсивный алгоритм

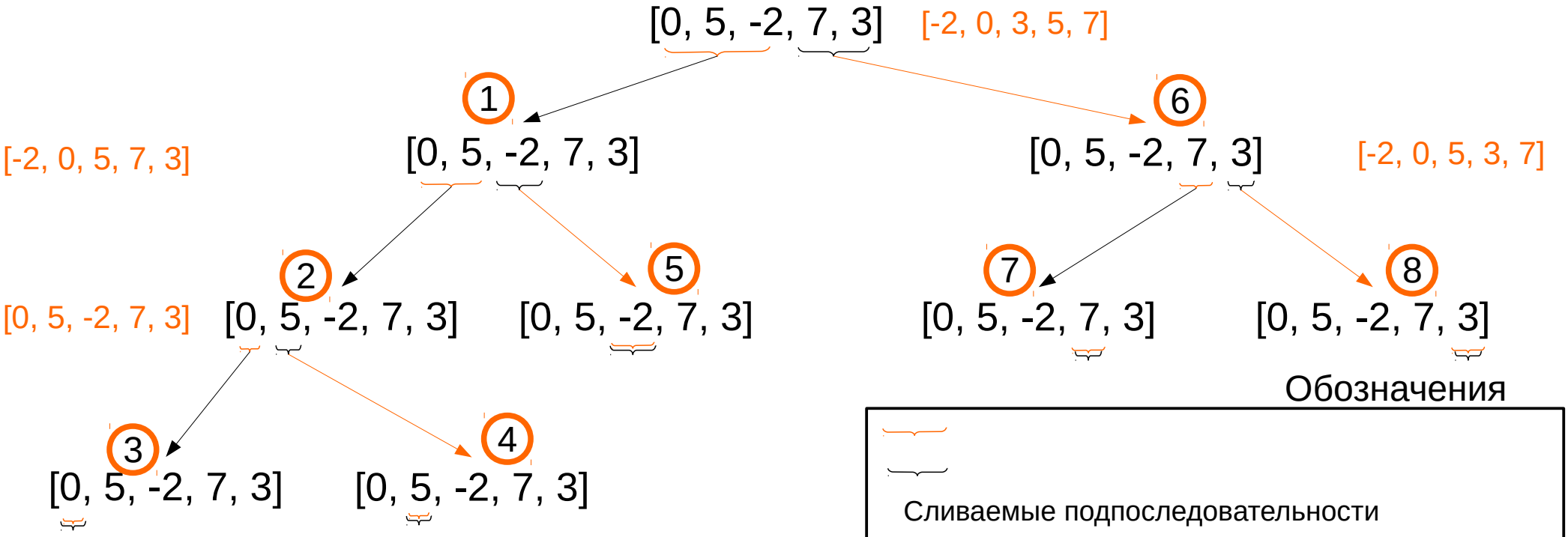
Сведение о алгоритме

Сложность по времени в наихудшем случае	O(n·ln(n))
Требует дополнительно памяти в размере	n

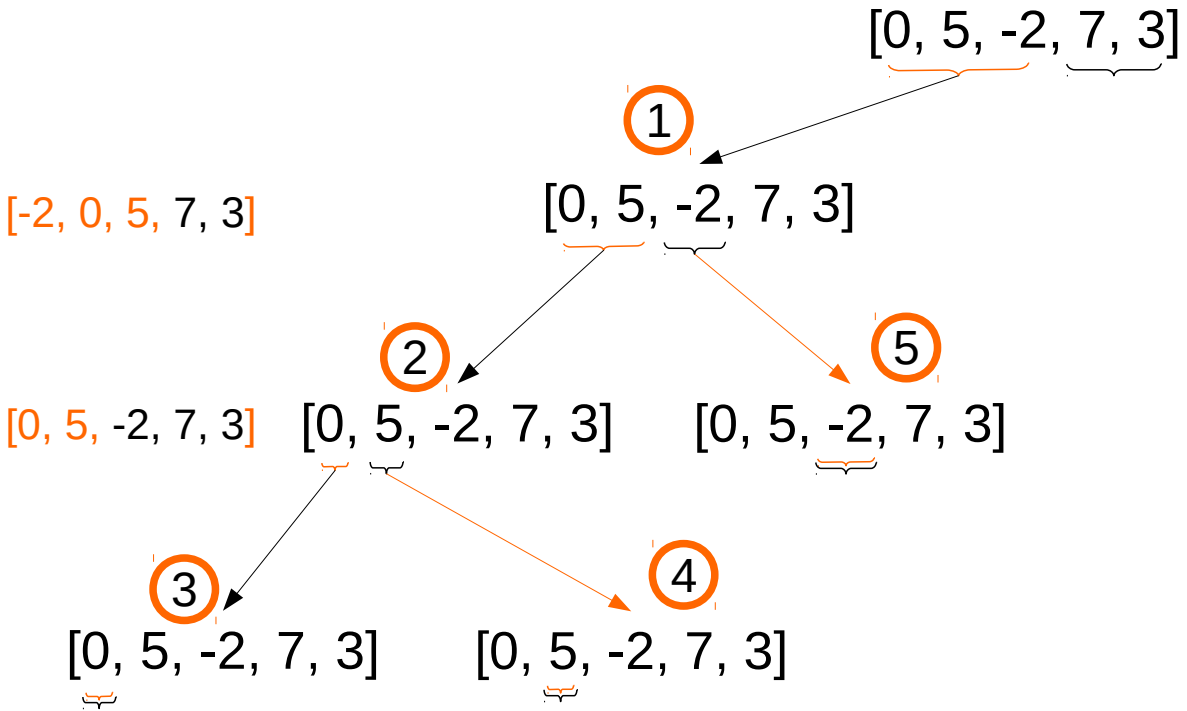
Описание алгоритма

1) Создается дополнительная последовательность размер которой равен сортируемой последовательности. Перейти в **2**.
2) Последовательность разбивается на две части и для каждой из частей рекурсивно запускается функция сортировки сначала для левой подпоследовательности, потом для правой. После чего проводят слияние отсортированных подпоследовательностей. Условием выхода из рекурсии является размер подпоследовательности равный нулю.

Графическая иллюстрация работы алгоритма

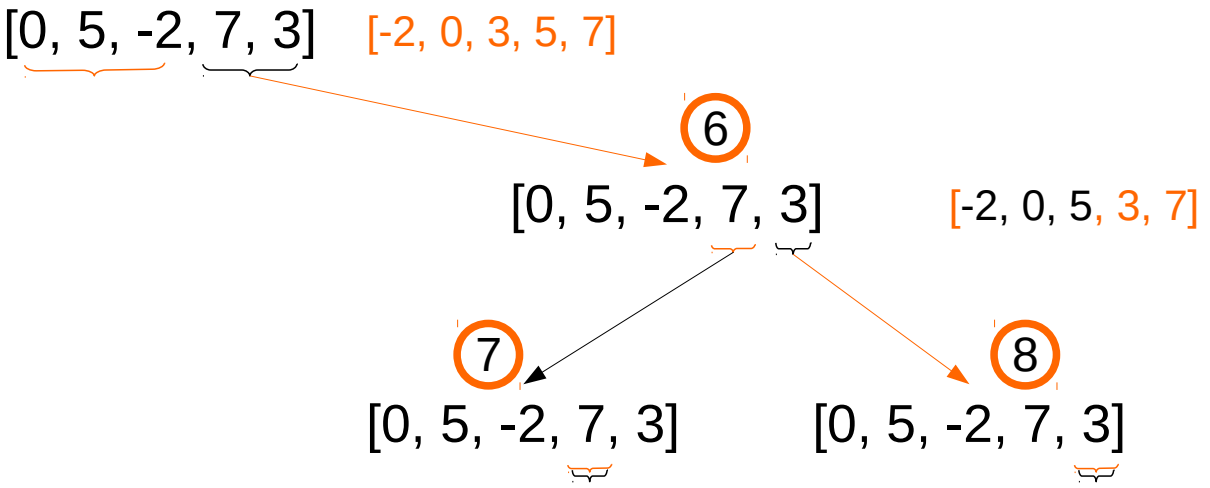


Графическое пояснение алгоритма



Сначала всегда вызывается сортировка левой подпоследовательности поэтому вызовы методов идут в том порядке как указанно. После окончания работы для правой подпоследовательности происходит слияние последовательностей на уровень выше. Сливаемая подпоследовательность выделена оранжевым.

Графическое пояснение алгоритма



Сначала всегда вызывается сортировка левой подпоследовательности поэтому вызовы методов идут в том порядке как указанно. После окончания работы для правой подпоследовательности происходит слияние последовательностей на уровень выше. Сливаемая подпоследовательность выделена оранжевым.

Python

Реализация алгоритма на Python

Python

Функция слияния подпоследовательностей

```
def merge(sequince, support, ls, le, rs, re):
    for i in range(ls, re+1):
        support[i] = sequince[i]
    l = ls
    r = rs
    for i in range(ls, re+1):
        if l > le:
            sequince[i] = support[r]
            r += 1
        elif r > re:
            sequince[i] = support[l]
            l += 1
        elif support[l] < support[r]:
            sequince[i] = support[l]
            l += 1
        else:
            sequince[i] = support[r]
            r += 1
    return None
```

Реализация алгоритма сортировки

```
def merge_sort(sequince, support=None, start_index=None, stop_index=None):
    if support is None:
        support = sequince[::]
    if start_index is None:
        start_index = 0
    if stop_index is None:
        stop_index = len(sequince)-1
    if stop_index <= start_index:
        return None
    h = start_index + (stop_index-start_index)//2
    merge_sort(sequince, support, start_index, h)
    merge_sort(sequince, support, h+1, stop_index)
    merge(sequince, support, start_index, h, h+1, stop_index)
```

Java

Реализация алгоритма на Java

Метод для слияния подпоследовательностей

```
public static void merge(int[] array, int[] supportArray, int ls, int le, int rs, int re) {
     for (int i = ls; i <= re; i++) {
          supportArray[i] = array[i];
     }
     int l = ls;
     int r = rs;
     for (int i = ls; i <= re; i++) {
          if (l > le) {
               array[i] = supportArray[r];
               r += 1;
          } else if (r > re) {
               array[i] = supportArray[l];
               l += 1;
          } else if (supportArray[l] < supportArray[r]) {
               array[i] = supportArray[l];
               l += 1;
          } else {
               array[i] = supportArray[r];
               r += 1;
          }
     }
}
```

Реализация алгоритма на Java

Метод для создания вспомогательного массива и запуска рекурсивного метода

```
public static void mergeSort(int[] array) {
     int[] support = Arrays.copyOf(array, array.length);
     int startIndex = 0;
     int stopIndex = support.length - 1;
     mergeSort(array, support, startIndex, stopIndex);
}
```

Рекурсивный метод реализующий сортировку слиянием

```
public static void mergeSort(int[] array, int[] support, int startIndex, int endIndex) {
     if (startIndex >= endIndex) {
          return;
     }
     int h = startIndex + (endIndex - startIndex) / 2;
     mergeSort(array, support, startIndex, h);
     mergeSort(array, support, h + 1, endIndex);
     merge(array, support, startIndex, h, h + 1, endIndex);
}
```

Обобщенная реализация алгоритма

Метод для слияния подпоследовательностей

```
public static <T> void merge(T[] array, T[] support, Comparator<T> comp, int ls, int le, int rs, int re) {
     for (int i = ls; i <= re; i++) {
          support[i] = array[i];
     }
     int l = ls;
     int r = rs;
     for (int i = ls; i <= re; i++) {
          if (l > le) {
               array[i] = support[r];
               r += 1;
          } else if (r > re) {
               array[i] = support[l];
               l += 1;
          } else if (comp.compare(support[l], support[r]) < 0) {
               array[i] = support[l];
               l += 1;
          } else {
               array[i] = support[r];
               r += 1;
          }
     }
}
```

Реализация алгоритма на Java

Метод для создания вспомогательного массива и запуска рекурсивного метода

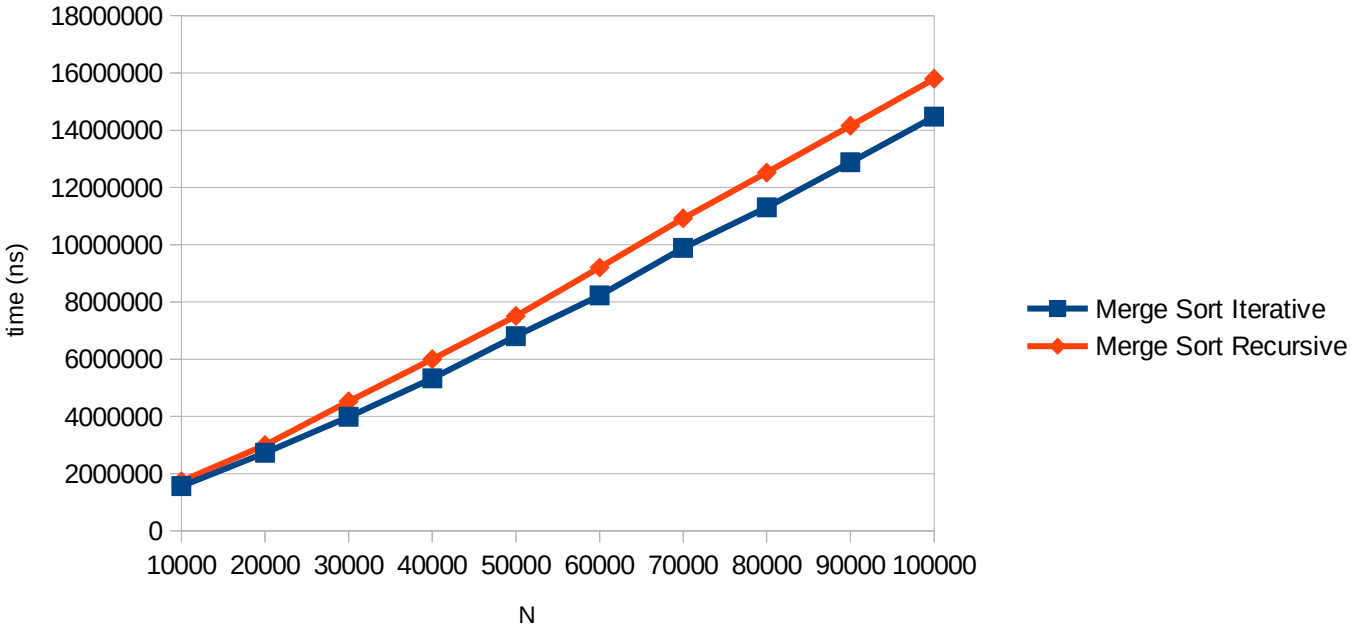
```
public static <T> void mergeSort(T[] array, Comparator<T> comp) {
     T[] support = Arrays.copyOf(array, array.length);
     int startIndex = 0;
     int endIndex = support.length - 1;
     mergeSort(array, support, comp, startIndex, endIndex);
}
```

Рекурсивный метод реализующий сортировку слиянием

```
public static <T> void mergeSort(T[] array,T[] support,Comparator<T> comp,int startIndex,int endIndex) {
     if (startIndex >= endIndex) {
          return;
     }
     int h = startIndex + (endIndex - startIndex) / 2;
     mergeSort(array, support, comp, startIndex, h);
     mergeSort(array, support, comp, h + 1, endIndex);
     merge(array, support, comp, startIndex, h, h + 1, endIndex);
}
```

Вычислительный эксперимент

Интересным вопросом является какой из алгоритмов (итерационный или рекурсивный) сортировки слиянием более эффективный. Для этого был проведен вычислительный эксперимент. Используя язык Java были реализованы оба этих алгоритма и было замерено среднее время необходимое для сортировки массивов с помощью этих реализаций. Такие замеры были проведены для массивов разных размеров. На графике ниже приведены зависимости времени сортировки от размера массива для разных реализаций.



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.