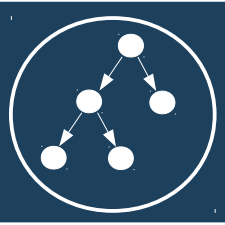


Data Structures and Algorithms

Алгоритмы.
Сортировка слиянием. Итерационный алгоритм

Сведение о алгоритме

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

Краткие сведения о алгоритме и авторе

Алгоритм был разработан в 1945 г Джоном Фон Нейманом. Этот алгоритм использует подход «разделяй и властвуй» и может быть использован для сортировки структур данных, доступ к элементам которых можно получать только последовательно (например внешние файлы).

Джон фон Нейман (1903-1957) венгеро-американский математик, физик и педагог еврейского происхождения, сделавший важный вклад в квантовую физику, квантовую логику, функциональный анализ, теорию множеств, информатику, экономику и другие отрасли науки. Наиболее известен как человек, с именем которого связывают архитектуру большинства современных компьютеров (так называемая архитектура фон Неймана).

Описание алгоритма

1) Создается дополнительная последовательность размер которой равен сортируемой последовательности. Перейти в **2**.
2) Устанавливается начальный размер сливаемых последовательностей равный 1. Выполняем попарное слияние соседних подпоследовательностей указанного размера начиная с начала последовательности. В случае если для подпоследовательности нет пары, то слияние производить не нужно (для последовательности нечетной длины). Перейти к **3**.
3) Увеличить значение размера в два раза. Если размер больше длинны последовательности **закончить алгоритм**. В противном случае перейти к **2**.

Графическая иллюстрация работы алгоритма

[5, 0, -2, 7, 3] → [0, 5, -2, 7, 3] → [0, 5, -2, 7, 3] →

[0, 5, -2, 7, 3] → [-2, 0, 5, 7, 3] → [-2, 0, 3, 5, 7]

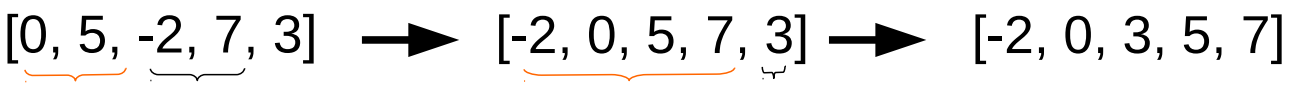
Обозначения

Сливаемые подпоследовательности

Графическое пояснение алгоритма

В начале алгоритма производят попарное слияние подпоследовательностей длинной 1. По сути это означает упорядочивание пары соседних элементов. Для последнего элемента пары нет, поэтому и слияние для него не производится.

Графическое пояснение алгоритма



После чего производим попарное слияние подпоследовательностей размером 2. После чего слияние подпоследовательностей длинной 4, таких подпоследовательностей 2 (вторая из одного элемента). После этого оканчиваем алгоритм.

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
```

Python

Реализация алгоритма сортировки

```
def merge_sort(sequince):
    support = sequince[::]
    n = len(support)
    size = 1
    while size < n:
        j = 0
        while j < n-size:
            merge(sequince, support, j, j+size-1, j+size, min(j+2*size-1, n-1))
            j += 2*size
        size = size * 2
    return None
```

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

```
public static void mergeSort(int[] array) {
     int[] supportArray = Arrays.copyOf(array, array.length);
     int n = array.length;
     for (int size = 1; size < n; size *= 2) {
          for (int j = 0; j < n - size; j += 2 * size) {
               merge(array, supportArray, j, j + size - 1, j + size, Math.min(j + 2 * size - 1, n - 1));
          }
     }
}
```

Обобщенная реализация алгоритма

Метод для слияния подпоследовательностей

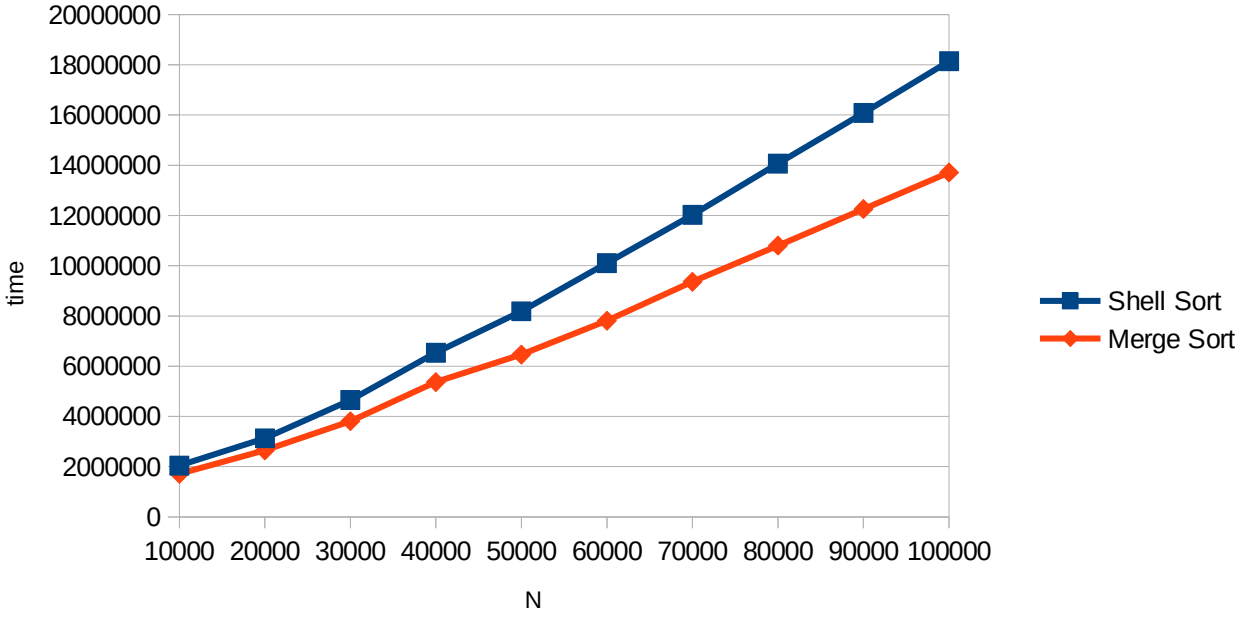
```
public static <T> void merge(T[] array, T[] support, Comparator<T> comp, int ls, int le, int rs, int re) {
     for (int i = ls; i <= re; i++) {
          support[i] = array[i];
     }
     int l = ls;
     int r = rs;
     for (int i = ls; i <= re; i++) {
          if (l > le) {
               array[i]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

```
public static <T> void mergeSort(T[] array, Comparator<T> comp) {
     T[] support = Arrays.copyOf(array, array.length);
     int n = array.length;
     for (int size = 1; size < n; size *= 2) {
          for (int j = 0; j < n - size; j += 2 * size) {
               merge(array, support, comp, j, j + size - 1, j + size, Math.min(j + 2 * size - 1, n - 1));
          }
     }
}
```

Вычислительный эксперимент

Для проверки эффективности данного алгоритма было проведено сравнение зависимости времени сортировки массива от его размера. Сортировка проводилась с помощью алгоритмов: сортировка слиянием и сортировка Шелла. Как можно видеть из графика сортировка слиянием более оптимальна по времени (но затратна по памяти). Но сортировку слиянием можно модифицировать для более высокого быстродействия (в дальнейших лекциях).



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.