Санкт-Петербургский
государственный
университет

Сортировки

Метод разделяй и властвуй, сортировка слиянием

Никита Гребень

Факультет прикладной математики - процессов управления

Обзор

Санкт-Петербургский государственный университет

Обзор

Санкт-Петербургский государственный университет

Декомпозиция

- Многие алгоритмы имеют рекурсивную структуру: для решения поставленной задачи они вызывают сами себя несколько раз, решая подзадачи меньшего размера
- Такие алгоритмы используют метод **декомпозиции** или метод **разделяй и властвуй**
- Парадигма, лежащая в основе метода *разделяй и властвуй*
  - **Разделение** задачи на несколько подзадач, которые представляют собой меньшие экземпляры той же задачи.
  - **Властвование** над подзадачами путем их рекурсивного решения. Если параметры размер малы, то такие подзадачи могут быть решены напрямую.
  - **Комбинирование** решений подзадач в решении исходной задачи.

Принцип работы

Алгоритм сортировки слиянием следует парадигме **разделяй и властвуй** и интуитивно работает следующим образом:

- Делим n-элементную сортируемую последовательность на две последовательности по $n/2$ элементов
- Рекурсивно сортируем эти две подпоследовательности с использованием сортировки слиянием
- Соединяем две отсортированные подпоследовательности для получения окончательного отсортированного варианта

Принцип работы

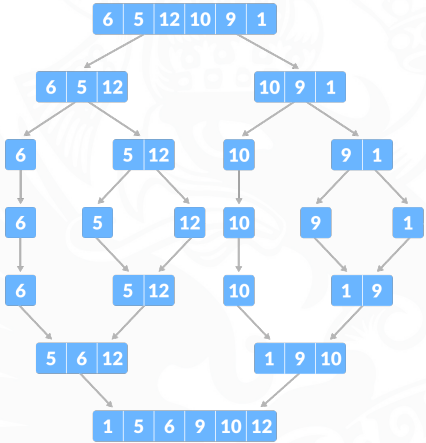

Рис. 1: Пример сортировки слиянием

Принцип работы

Рис. 2: Процесс слияния двух отсортированных массивов

Принцип работы

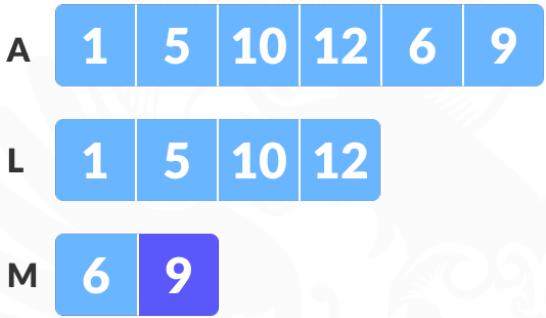

Рис. 3: Создаем копии двух подмассивов

Принцип работы

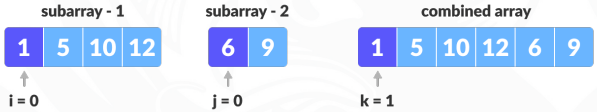

Рис. 4: Поддерживаем индексы двух подмассивов

Принцип работы

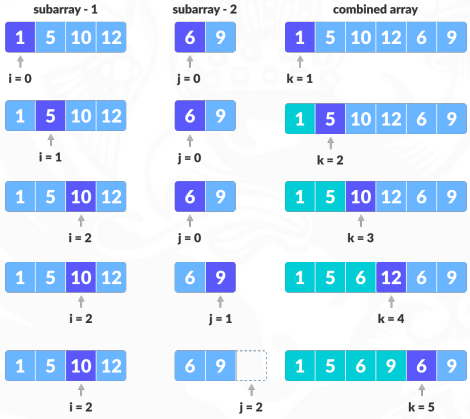

Рис. 5: Попарно сравниваем элементы подмассивов, пока не достигнем конца

Принцип работы

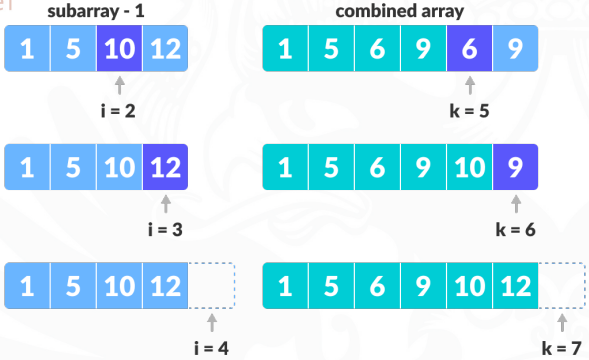

Рис. 6: Копируем оставшиеся элементы из первого подмассива в исходный массив

Принцип работы

Рис. 7: Копируем оставшиеся элементы из второго подмассива в исходный массив

Алгоритм процесса слияния

- Описанная идея реализована в представленном ниже псевдокоде, однако в нем присутствует некоторое ухищрение, благодаря чему не нужно в конце проверять, что мы дошли до конца каждого из подмассивов.
- Мы помещаем в конец так называемые **сигнальный элемент** в качестве ограничителя.
- В качестве сигнального элемента выбрана ∞, так что, когда мы встречаем элемент равный ∞, меньшего элемента мы уже не встреим до полного исчерпания обоих массивов.

Алгоритм процесса слияния I

1: **function** MERGE(A, p, q, r)
2: $n_1 = q - p + 1$
3: $n_2 = r - q$
4: Пусть $L[1 \dots n_1 + 1]$ и $R[1 \dots n_2 + 1]$ новые массивы
5: **for** $i = 1$ to n_1 **do**
6: $L[i] = A[p + i - 1]$
7: **end for**
8: **for** $j = 1$ to n_2 **do**
9: $R[i] = A[q + j]$
10: **end for**
11: $L[n_1 + 1] = \infty$
12: $R[n_2 + 1] = \infty$
13: $i = 1$
14: $j = 1$

Алгоритм процесса слияния II


```
15:     for k = p to r do
16:         if L[i] < R[j] then
17:             A[k] = L[i]
18:             i = i + 1
19:         else if L[i] > R[j] then
20:             A[k] = R[j]
21:             j = j + 1
22:         end if
23:     end for
24: end function
```

Алгоритм сортировки I


```
1: function MERGE-SORT(A, p, r)
2:     if p < r then
3:         q = ⌊(p + r)/2⌋
4:         MERGE-SORT(A, p, q)
5:         MERGE-SORT(A, q + 1, r)
6:         MERGE(A, p, q, r)
7:     end if
8: end function
```

Задание (0 баллов)

- Реализуйте сортировку слиянием без **сигнального** элемента, как это было представлено на Рис. 6-7.
- Т.е. пока не дойдете до конца либо L либо R, выбираете наибольший из текущих элементов в L и в R и помещаете его на корректную позицию в $A[p..r]$.
- Когда закончились элементы либо из L либо из R, помещаете оставшиеся элементы в $A[p..r]$.

Решение


```
1   while (i < n1 && j < n2) {
2     if (L[i] <= R[j]) {
3       A[k] = L[i];
4       i++;
5     } else {
6       A[k] = R[j];
7       j++;
8     }
9     k++;
10  }
11  while (i < n1) {
12    A[k] = L[i];
13    i++;
14    k++;
15  }
16  while (j < n2) {
17    A[k] = R[j];
18    j++;
19    k++;
20  }
```

Обзор

Санкт-Петербургский государственный университет

Вывод рекурентного отношения

- **Разделение.** В ходе разбиения определяется где находится середина массива. Эта операция выполняется за константное время, поэтому $D(n) = \Theta(1)$.
- **Властвование.** Рекурсивно выполняются две подзадачи, размер которых составляет $n/2$. Время решения этих подзадач равно $2T(n/2)$
- **Комбинирование.** Как уже упоминалось, процедура MERGE при работе с n-элементным массивом выполняется за время $\Theta(n)$, таким образом $C(n) = \Theta(n)$.

Вывод рекурентного отношения

- Складывая функции $D(n)$ и $C(n)$ на определенной глубине рекурсии, мы складываем функции, который представляют собой $\Theta(1)$ и $\Theta(n)$.
- Таким образом, выполнение операция на уровне n является линейной функцией $\Theta(n)$.
- Добавление ее к члену $2T(n/2)$ из шага "Властвование" дает следующее соотношение для времени работы в худшем случае:

$$T(n) = \begin{cases} \Theta(1), \text{если n} = 1 \\ 2T(n/2) + \Theta(n), \text{если n} > 1 \end{cases} \quad (1)$$

Интуиция

Санкт-Петербургский государственный университет

- Опираясь на **Мастер-теорему** показывается, что $T(n) = \Theta(n \log n)$
- Однако можно и без этого понять, почему решением рекурентного соотношения 1 является $\Theta(n \log n)$
- Перепишем рекурентное соотношение:

$$T(n) = \begin{cases} c, \text{если n} = 1 \\ 2T(n/2) + cn, \text{если n} > 1 \end{cases} \quad (2)$$

Интуиция

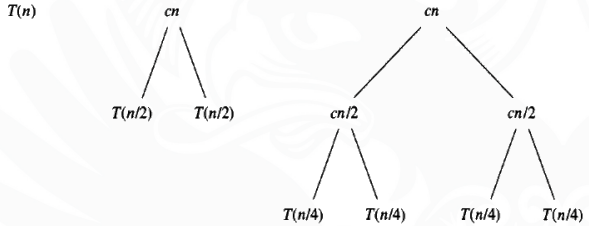

Рис. 8: Построение дерева рекурсии для $T(n) = 2T(n/2) + cn$

Интуиция

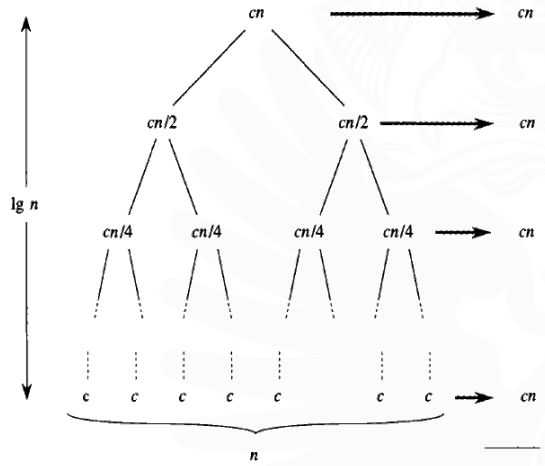

Рис. 9: Полностью раскрытое дерево рекурсии, имеющее $\log n + 1$ уровней с вкладом каждого равным cn. Таким образом, общая стоимость равна $cn \log n + cn$, что представляет собой $\Theta(n \log n)$

Обзор

Санкт-Петербургский государственный университет

Merge sort - модификация вставками (2 балла)

- Будем рассматривать модификацию сортировки слиянием, где n/m подмассивов длины m нужно отсортировать сортировкой вставками.
- Докажите, что сортировка вставкой сортирует эти n/m последовательностей за время $\Theta(nm)$.
- Вам необходимо выполнить слияния этих отсортированных последовательностей за $\Theta(n \log(n/m))$ в наихудшем случае.
- Таким образом, модифицированный алгоритм выполняется за время $\Theta(nm + n \log(n/m))$ в наихудшем случае. Чему равно наибольшее значение m как функции от n, при котором модифицированный алгоритм выполняется за столько, за сколько и стандартная сортировка слиянием?
- Саму величину m необходимо найти в процессе задачи. Какое m стоит выбирать на практике? Какое m будет наилучшим для массивов размера $10^3 \dots 10^7$?

Количество интересных пар (2 балла)

- Пусть $A[1..n]$ представляет собой массив из n целых чисел. Если $i < j$ и $A[i] > A[J]$, то пару (i, j) будем называть интересной.
- Разработайте алгоритм, находящий количество таких пар в произвольном массиве за $\Theta(n \log n)$ в наихудшем случае.
- *Пользоваться методом “разделяй и властвуй”, внимательно изучить реализацию сортировки слиянием.

Найти сумму (3 балла)*

- Пусть $A[1..n]$ также представляет собой массив из **различных** n целых чисел.
- Вам задано какое-то беззнаковое 32-битное число число S.
- Разработайте алгоритм, находящий за $\Theta(n \log n)$ среди массива A все такие пары i, j, что $a[i] + a[j] = S$. Если такие пары отсутствуют, сообщить об этом.
- Запрещено пользоваться встроенными контейнерами *map*, *set*!!!
- Как и в прошлой задаче, использовать метод "разделяй и властвуй".