Алгоритмы сортировки

А. Г. Фенстер, fenster@fenster.name

26 марта 2010 г.

Конспект семинара №7 по программированию для студентов 1 курса ММФ НГУ. Помните, что чтение конспекта не делает посещение соответствующего семинара необязательным! О найденных ошибках и неточностях, пожалуйста, пишите мне по адресу fenster@fenster.name. Также буду рад получить любые комментарии по поводу этого текста.

На этом семинаре будет рассматриваться задача *сортировки* массива, т. е. упорядочения его элементов согласно некоторому порядку (например, в порядке неубывания). Мы рассмотрим несколько простых алгоритмов сортировки и два более сложных, но требующих меньшего времени для выполнения.

Во всех примерах мы рассматриваем массивы, состоящие из натуральных чисел, но приведённые алгоритмы применимы для массивов с элементами любого типа.

1 Простые методы сортировки

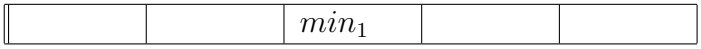
Перечисленные ниже методы достаточно просты для того, чтобы их можно было придумать самостоятельно. Все они имеют одну и ту же оценку сложности в худшем случае: $O(N^2)$.

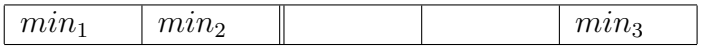
1.1 Сортировка выбором

Ищем в массиве минимальный элемент, меняем его с числом, стоящим на нулевой позиции. Затем в «хвосте» массива, начиная с первого элемента (считая от нуля), снова ищем минимальный элемент и меняем его с числом, стоящим на первой позиции. Всего эти действия выполним

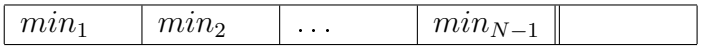
$N-1$, постоянно сдвигая начало диапазона поиска минимального элемента вправо — получим упорядоченный в порядке неубывания массив.

Здесь и далее будем обозначать минимальный элемент массива как min_1, минимальный из оставшихся — min_2 и так далее. Графически работу алгоритма можно представить следующим образом:





и так далее до тех пор, пока все элементы не встанут на свои места:

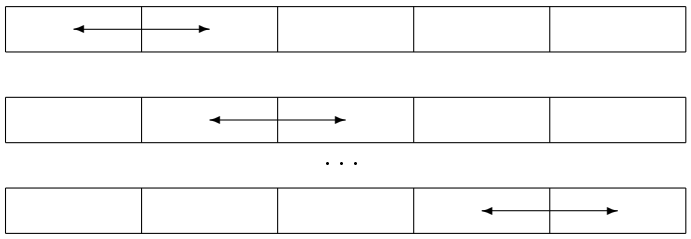


Последний элемент автоматически оказывается на своём месте.

Несложно подсчитать, что для такой сортировки необходимо выполнить $O(N^2)$ операций сравнения (по уже встречавшейся нам ранее формуле $(N-1)+(N-2)+\ldots+1$).

1.2 Сортировка пузырьком

Выполняем проход по массиву слева направо, сравнивая и при необходимости меняя местами соседние элементы:



После этих действий максимальный элемент max_1 окажется на последней позиции. Повторяя процесс $N-1$ раз (или до тех пор, пока массив не станет отсортированным), будем выводить в конец элементы max_2, max_3 и т. д.

Количество сравнений вычисляется аналогично, сложность алгоритма $O(N^2)$.

2 Улучшенные сортировки

Алгоритмы быстрой и пирамидальной сортировки подробно описаны в отдельном файле. Сложность пирамидальной сортировки в среднем и в худшем случае $O(N \log_2 N)$, сложность быстрой сортировки в худшем случае $O(N^2)$, но этот худший случай достигается достаточно редко; в среднем же её сложность также равна $O(N \log_2 N)$.

3 Проверочное задание («пятиминутка»)

Тема: алгоритм пирамидальной сортировки (см. задание).