Слияние за $O(N)$ без использования дополнительной памяти

Базовые факты, идеи и алгоритмы:

— для слияния упорядоченных массивов длины N и M достаточно буфера длины $\min(N, M)$;

— буфер слияния может сохранять исходное множество значений своих элементов (но как-то переставленных), если вместо присваивания использовать процедуру обмена swap;

— существует метод сортировки с линейной трудоемкостью по перестановкам (метод перестановки максимума);

— циклический сдвиг массива длины N на K позиций осуществляется за $O(N)$ операций.

Алгоритм слияния.

Пусть дан массив длины N, который разбит на две упорядоченных по возрастанию части: начальный участок A длины NA и конечный участок B длины NB, $N = NA + NB$. Требуется получить упорядоченное по возрастанию состояние всего массива.

Пусть $K = \sqrt{N}$

1. Будем разбивать массив на подотрезки длины K. Определим K наибольших элементов отсортированного массива. Для этого запустим алгоритм слияния двух имеющихся частей со стороны их максимумов, но не будем реально переписывать элементы массива, а будем только двигать указатели, пока "накопленная" часть (наибольших элементов) не достигнет длины K. Позиции указателей на частях A и B покажут какие элементы пойдут в набор K максимальных из каждой отсортированной части.

= сложность шага $O(K)$ операций

2. Переместим эти "старшие" элементы части в конец массива при помощи циклического сдвига (старшие элементы части B и так находятся в конце массива)

= сложность шага $O(N)$ операций

Итак, K максимальных элементов находятся в конце массива, и этот блок K элементов будет у нас буфером слияния (с сохранением значений).

3. Теперь разобъем оставшуюся часть на отрезки длины K, отсчитывая их от точки контакта частей A и B. В результате наш массив в общем случае разобъется на 5 зон:

A1 — начальная часть зоны A, которая оказалась длиной меньше K,

AB — последовательность отрезков одинаковой длины K, сначала из части A, потом из части B,

B1 — конечная часть зоны B, которая оказалась длиной меньше K,

C — отрезок длины K с K наибольшими элементами всего массива.

= сложность шага $O(1)$ операций (определить начальные позиции и длины этих зон)

4. Отсортируем зону AB по возрастанию как массив отрезков размера K, взяв в качестве функции сравнения сравнение первых элементов отрезков и используя метод сортировки с линейным количеством перестановок.

= сложность шага: сравнений $O(K^2) = O(N)$, перестановок $O(K)$, каждая перестановка $O(K)$, итого $O(N)$.

5. Проведем слияние двух первых отрезков части AB с использованием буфера C. Заметим, что при этом первый отрезок будет уже соответствовать отсортированному состоянию части AB. Действительно, никакие элементы третьего и последующих отрезков не могут при последующей сортировке сдвинуться в первый отрезок так как они заведомо не меньше либо всех элементов первого отрезка, либо всех элементов второго отрезка (так как из трех отрезков как минимум два происходят из одной и той же части A или B). Двлее проведем слияние 2 и 3 отрезков части AB, и т.д. слияние отрезков i, $i+1$ до конца части AB. В результате часть AB окажется отсортированной.

= сложность шага: K слияний со сложностью $O(K)$, т.е. всего $O(N)$.

6. Итак, часть AB отсортирована. Проведем ее слияние с частями A1 и B1 (они уже изначально отсортированы) с использованием буфера C.

= сложность шага: два слияния со сложностью $O(N)$ каждое.

7. Теперь весь массив отсортирован за исключением части C, которая содержит K наибольших элементов и находится в конце массива. Отсортируем ее произвольным алгоритмом (пузырьком). Массив полностью упорядочен, т.е. слияние окончено.

= сложность шага: сортировка $O(K^2) = O(N)$.

Итого полная трудоемкость есть $O(N)$, дополнительной памяти не потребовалось.

Интересно, насколько это будет медленее слияния с дополнительной памятью :))) Кстати, раз уж мы взялись экономить память, то и саму процедуру сортировки слиянием надо реализовывать нерекурсивно, т.е. по “восходящему” варианту. А это тоже не самый прозрачный алгоритм :)))