TimSort Теория

© И. И. Гридасов

23 октября 2016

Покажем сначала, что первый и третий этап работают за $O(n \log n)$.

- Первый этап работает за время $O(n)$. Так как число minRun можно считать константой, то сортировка каждого рана выполняется за $O(1)$. Всего Run-ов $O(n)$, поэтому суммарное время $O(n)$.

- Третий этап - слияние всех Run-ов в стеке. Заметим, что в начале третьего этапа в стеке находится $O(\log n)$ Run-ов. Так как в самом верхнем Run-е кол-во эл-тов $\geqslant F_1$, во втором сверху $\geqslant F_2$. Далее по индукции получаем, что в k-ом сверху Run-е, хотя бы F_k эл-ов. Так как в каждом Run-е эл-ов больше, чем сумма двух выше его. Но в любом Run-е, кол-во эл-во $\leqslant n$, поэтому $F_k \leqslant n$. Но так как F_k можно оценить снизу как $2^{\frac{k}{2}}$, то $k \leqslant 2 * \log n$, следовательно кол-во эл-ов в стеке есть $O(\log n)$. А слияние двух Run-ов выполняется за $O(n)$, значит суммарная асимптотика есть $O(n \log n)$.

Теперь разберёмся со вторым этапом: Будем в стеке поддерживать следующий инвариант: для любой тройки подряд идущих Run-ов в стеке с длинами l_1, l_2, l_3, записанных от вершины стека, верно, что $l_2 > l_3$ и $l_1 > l_2 + l_3$. Будем при добавлении Run-а в стек накидывать на него $P * l * h$ монет, где l - длина Run-а, а h - его уровень в стеке. Рассмотрим операции проталкивания, которые мы совершаем, чтобы сохранить инвариант стека при добавлении сверху нового Run-а. Будем каждый раз рассматривать верхнюю тройку Run-ов и проверять соблюдается ли инвариант, если нет то как-то сливать 2 Run-а из верхних трёх и проверять инвариант дальше.

Есть два случая, когда инвариант не выполняется:

$l_3 \geqslant l_2$: Тогда сливаем Run-ы 2 и 3. Пусть второй Run находится на уровне h, тогда 3 на уровне $h+1$. Рассмотрим какое кол-во монеток освободилось при совершении данной операции. Было: $P * l_3 * (h+1) + P * l_2 * h$ - стало: $P * (l_2 + l_3) * h$, тогда освободилось $P * l_3$ монет. Тогда при $P > 2$, имеем $P * l_3 > 2 * l_3 \geqslant l_2 + l_3$, значит этих монеток хватит на слияние Run-ов 2 и 3.

$l_3 < l_2$ **и** $l_1 \leqslant l_2 + l_3$: Тогда сливаем Run-ы 1 и 2. Пусть первый Run находится на уровне h, тогда второй на уровне $h+1$. Аналогично смотрим на кол-во освободившихся монеток: $P*l_1*h+P*l_2*(h+1)-P*(l_2+l_1)*h = P*l_2$. Тогда при $P \geqslant 3$, $P*l_2 \geqslant 3*l_2 > 2*l_2+l_3 \geqslant l_1+l_3$. Значит этих монеток хватит на слияние Run-ов 1 и 2.

Значит при $P \geqslant 3$, нам хватит монеток, чтобы выполнить все добавления Run-ов в стек и все слияния, чтобы сохранить инвариант стека. Осталось понять, что кол-во монеток, которое мы использовали есть $O(n \log n)$. Для этого заметим, что в любой момент времени кол-во Run-ов в стеке есть $O(\log n)$(из док-ства в этапе 3). Тогда при накидывании на Run $P*l*h$ монет, то это тоже самое, что на каждый эл-т Run-а накинуть $P * h$ монет. Так как $h = O(\log n)$ и $P = 3 = O(1)$, то на каждый эл-т мы накинули $O(\log n)$ монет. Значит всего монет поступило $O(n \log n)$ монет. Поэтому суммарно второй этап выполняется за $O(n \log n)$.

Наконец, так как все три этапа Алгоритма TimSort выполняются за $O(n \log n)$, то и весь алгоритм выполняется за $O(n \log n)$. ч.т.д.