Анализ на задача ntsort

Тагове: сортиране, теория на числата

В повечето алгоритми за сортиране (поне тези, базирани на разменяне на съседни числа) можем да броим инверсиите лесно, просто броейки размените, които извършваме – това е така, защото всяка размяна на два съседни елемента променя единствено състоянието на инверсията между тях. Заради това в остатъка на анализа ще разгледаме проблема да сортираме редицата, а не да броим инверсиите.

Очевидно добра стъпка е да започнем с модификация на познат алгоритъм на темата (а такива има много).

Най-интуитивно е да започнем със `selection sort` – на всяка итерация, работата ни е да намерим най-малкото число, което не си е на мястото и да го преместим. Докато в стандартния алгоритъм единствената трудност е да намерим най-малкото число за всяка итерация на алгоритъма, тук допълнително трябва да проверим, че този елемент ще е делител на всички вляво от него, които трябва да "прескочи". Понеже така или иначе алгоритъма е $O(n^2)$, няма нужда да мислим допълнително – на всяка стъпка от `selection sort`-а, намираме най-лявото най-малко число и след това допълнително проверяваме дали то дели всички числа вдясно от него (освен тези във вече сортирания префикс).

Допълнително наблюдение, което дотук не ни помага, но ще е полезно за пълното решение е вместо първо (1) да намерим най-лявото най-малко число и после (2) да проверим дали то ще може да се размени с всички по-големи в ляво от него, по време на първото обхождане в (1) да поддържаме като допълнителна информация най-големия общ делител на всички числа, които са останали вдясно от текущата ни позиция. Така, ако едно число дели всички числа вдясно, то ще бъде техен общ делител, а следователно трябва и да дели техния най-голям общ делител.

Стандарт при броенето на инверсии е ползването на `merge sort` (Есенен турнир 2013 B invcnt) – за да сортираме интервал $[l, r]$ от масива, първо ще го разделим на две половини, които ще сортираме, а след това сортирането на целия интервал се свежда до значително по-лесната задача да се "слеят" два сортирани списъка (как точно е упражнение за читателя).

Решението на задачата става почти шаблонно, с единствения детайл, че трябва да проверим особеното свойство за делимост между две съседни числа. За тази цел всъщност е достатъчно да добавим единствена проверка: по време на сливането на двете сортирани половини, ако в даден момент най-малкия елемент е в дясната половина, той ще трябва да "прескочи" всички оставащи елементи от лявата половина. Тези оставащи елементи образуват някакъв суфикс на оригиналния списък – тоест проверката ни се свежда до това текущия елемент да дели суфиксния най-голям общ делител в левия списък. Финалното решение възлиза на $O(n \log n \log A)$.

Друг ъгъл на атака при задача за броене на инверсии е сканиране на масива със сегментно дърво – в случая допълнителната проверка ще ни принуди да ползваме `merge sort` дърво. Ще сканираме редицата от ляво надясно и за всяко число ще изпълним заявка "колко числа вдясно от мен са по-големи от мен и какъв е техния най-голям общ делител?". Ако текущото число дели намерения най-голям общ делител, значи дотук можем да сортираме редицата, а в противен случай въобще няма да можем и ще отпечатаме -1.

Автор: Иван Лупов