Лекция 8

Лексикографическая сортировка

8.1 Сортировка целых чисел

Заметим, что n целых чисел от 1 до m отсортировать очень просто: организуем массив a из m списков и будем последовательно добавлять очередное число i к списку $a[i]$. После этого остается только объединить полученные списки: $a[1]$, затем $a[2]$, и т.д. Легко видеть, что благодаря операции `[·]` (то есть косвенной адресации РАМ) мы можем обойтись всего $O(n+m)$ операциями над целыми числами.

8.2 Лексикографическая сортировка

Задача: дано n строк общей длиной L, состоящих из чисел от 1 до k. Требуется отсортировать их в соответствии с лексикографическим упорядочением:

$$s \leq t \iff s \text{ — префикс } t \ \vee\ \exists i\, (s_i < t_i \wedge \forall j < i(s_j = t_j))$$

(как слова в словаре).

Алгоритм:

1. Отсортируем строки по убыванию длины, получим упорядоченный список A. (Так как длина — целое число, это можно осуществить за $O(n+l_{max})$ операций, где l_{max} — максимальная длина строки.)

2. Для всех i, начиная с l_{max} и заканчивая 1, мы будем сортировать строки по символу, стоящему на очередной i-й позиции.

   Для этого мы будем поддерживать массив B, состоящий из k упорядоченных списков строк. На очередном шаге в $B[c]$ будут попадать строки, имеющие символ c в позиции i.

   При этом строки мы будем обрабатывать в следующей последовательности: сначала строки длины i из A, затем строки из $B[1]$, $B[2]$, и т.д. Конечно, прежде чем начать записывать на очередном шаге в списки B, мы все упомянутые строки должны переместить (в этом порядке) в отдельный список Q — тем самым списки B очистятся, а из A будут удалены строки длины i.

   (Ясно, что для строк, которые прежде были в одном и том же списке $B[j]$, и после этого шага сохраняется «старое» упорядочение.)

Корректность этого алгоритма очевидна. Теперь будем «дорабатывать» его, чтобы ограничиться $O(L+k)$ операциями с символами, указателями на строки и длинами строк. Каждую строку мы перемещаем между списками столько раз, сколько в ней символов — всего $O(L)$ перемещений; сортировка строк по длине займет $O(L)$ на вычисление длины и $O(L)$ на сортировку (эта сортировка тоже состоит просто в том, что мы отправляем каждую строку длины l в список $A'[l]$ вспомогательного массива A', а потом сливаем все списки $A'[l]$ в список A); организация списков и их окончательное объединение — $O(k)$ операций. Проблема только в просматривании k списков на каждом шаге: так может получиться порядка kL операций; это много; но ведь списки-то зачастую — пустые (просмотр непустых списков мы, кстати, уже учли в перемещении строк!).

Чтобы не просматривать пустые списки, создадим заранее список пар (l, s_{il}), когда s_{il} действительно существует. Отсортируем этот список сначала по второй компоненте, а потом по первой (также «рассортировывая» соответственно по k или L упорядоченным спискам). После этого легко создать упорядоченные списки символов, встречающихся на позиции l. На все это уйдет $O(L+k)$ операций, зато даст возможность просматривать в исходном алгоритме только непустые списки.