Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 9

АЛГОРИТМ СОРТИРОВКИ ИНДЕКСАМИ И ЗАДАЧА РАЦИОНАЛЬНОГО ВЫБОРА АЛГОРИТМОВ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

Алгоритм сортировки массива методом индексов

Этот оригинальный метод работает не для всех возможных массивов. Его применение предполагает, что исходные числа являются целыми, положительными (натуральными), и не содержат повторений. (Задача для модификации — как сделать его пригодным для целых чисел и повторений?).

Основная идея алгоритма заключается в использовании дополнительного массива, который изначально заполнен нулями, и, количество элементов в котором равно максимальному значению в исходном массиве. (А как можно сократить размер дополнительного массива?).

В предположении, что в исходном массиве нет повторяющихся элементов (а если есть?), можно использовать значение элемента из исходного массива в качестве индекса дополнительного массива — B[A[j]]=1.

Таким образом, если некоторый элемент исходного массива A равен 17, то 17-ый элемент дополнительного массива B устанавливается в единицу. Остается только убрать «лишние» нули, и мы получим исходный массив, отсортированный по возрастанию. Заметим, что в этом алгоритме мы не сравниваем числа друг с другом!

Более детальное рассмотрение этой идеи позволяет уменьшить затраты дополнительной памяти. На самом деле, если минимальный элемент исходного массива равен 1000, то, очевидно, что первые 1000 элементов дополнительного массива не будут использоваться.

Таким образом, размер дополнительного массива должен быть равен размаху варьирования сортируемых чисел, т.е. разницы между минимальным и максимальным элементом +1. Реализация такого алгоритма включает в себя следующие шаги:

— определение максимального и минимального значений в исходном массиве и размаха варьирования;

— обнуление дополнительного массива;

— заполнение дополнительного массива единицами методом индексов;

— получение сортированного массива путем сборки единиц в дополнительном массиве и сдвига ненулевых индексов на размах варьирования.

Запись этого алгоритма в принятом алгоритмическом базисе имеет вид:

```
Sort_Ind_A2 (A, n)
   Max ← A[1]                  2
   Min ← Max                   1
   For i ← 2 to n              1+3(n-1)
      If Max < A[i]            2(n-1)
         then
```

```
            Max ← A[i]          2m
      If Min > A[i]             2(n-1)
         then
            Min ← A[i]          2m
   end For
   d ← Min-1                    2 (Аффинный сдвиг)
   L ← Max-Min+1                3
! массив B статически резервирован
Обнуление массива B
   For i ← 1 to L               1+3L
      B[i] ← 0                  2L
   end For
```

```
Метод индексов!
   For i ← 1 to n          1+3n (по массиву A)
      k ← A[i] - d
      B[k] ← B[k] + 1        4n
   end For
Обратная сборка
   i ← 1                     1
   For j ← 1 to L          1+3L
      If B[j]> 0             2L
         then
            k ← B[j] (число повторений)
            for m ← 1 to k
```

```
            A[i] ← j + d                3n
            i ← i + 1                   2n
        end for m
    end for j
End.
```

Отметим, что трудоемкость этого алгоритма существенно зависит от размаха варьирования L, который и будет использован как дополнительный аргумент функции трудоемкости. Трудоемкость первого этапа алгоритма — поиска минимума и максимума уже была получена, и мы будем использовать результат для среднего случая.

С учетом введенной параметризации функция трудоемкости может быть легко получена на основе информации о числе базовых операций, указанных в

строках записи алгоритма. Отдельный вопрос — о числе повторений. Худший случай обратной сборки — все числа различны, т.е. все B[j]>0 таковы, что B[j]=1.

$$f(n, L) = 10L + 28n + 4\,lnn + 6$$

Отметим, что при большом размахе варьирования объем дополнительной памяти, требуемой данным алгоритмом, может быть значительным. Интересно отметить также, что при значениях $n \approx L$ данный алгоритм за счет дополнительный затрат памяти сортирует массив за линейное время!

Для данного алгоритма ресурсная сложность имеет вид

$$\overline{\Re}_{\mathrm{c}}(A) = \langle \Theta(n + L),\ \Theta(n + L) \rangle,$$

где второй компонент отражает суммарные затраты памяти, определяемые объемом исходного и дополнительного массивов. Дополнительные затраты памяти алгоритма составляют $\Theta(L)$ и могут быть значительны!

Определение областей рационального применения алгоритмов

Ответим на вопрос — при каких параметрах входа применение алгоритма сортировки индексами или вставками будет более рационально? Тем самым мы решаем задачу рационального выбора алгоритма на основе особенностей входа (адаптивный выбор!).

При определении областей рационального применения рассмотренных алгоритмов будем считать, что исходные размерности таковы, что компоненты порядка $\Theta(1), \Theta(ln\, n)$ не являются существенными. В этом случае, приравнивая трудоемкости, получаем уравнение

$$2{,}5n^2 + 11{,}5n = 10L + 28n,$$

решая которое относительно L, получаем

$$L(n) = \frac{1}{4}n^2 - \frac{33}{20}n.$$

Таким образом, в координатах аргументов функции трудоемкости (L, n) мы получили уравнение параболы, разграничивающей области рационального применения рассматриваемых алгоритмов, что проиллюстрировано на рисунке

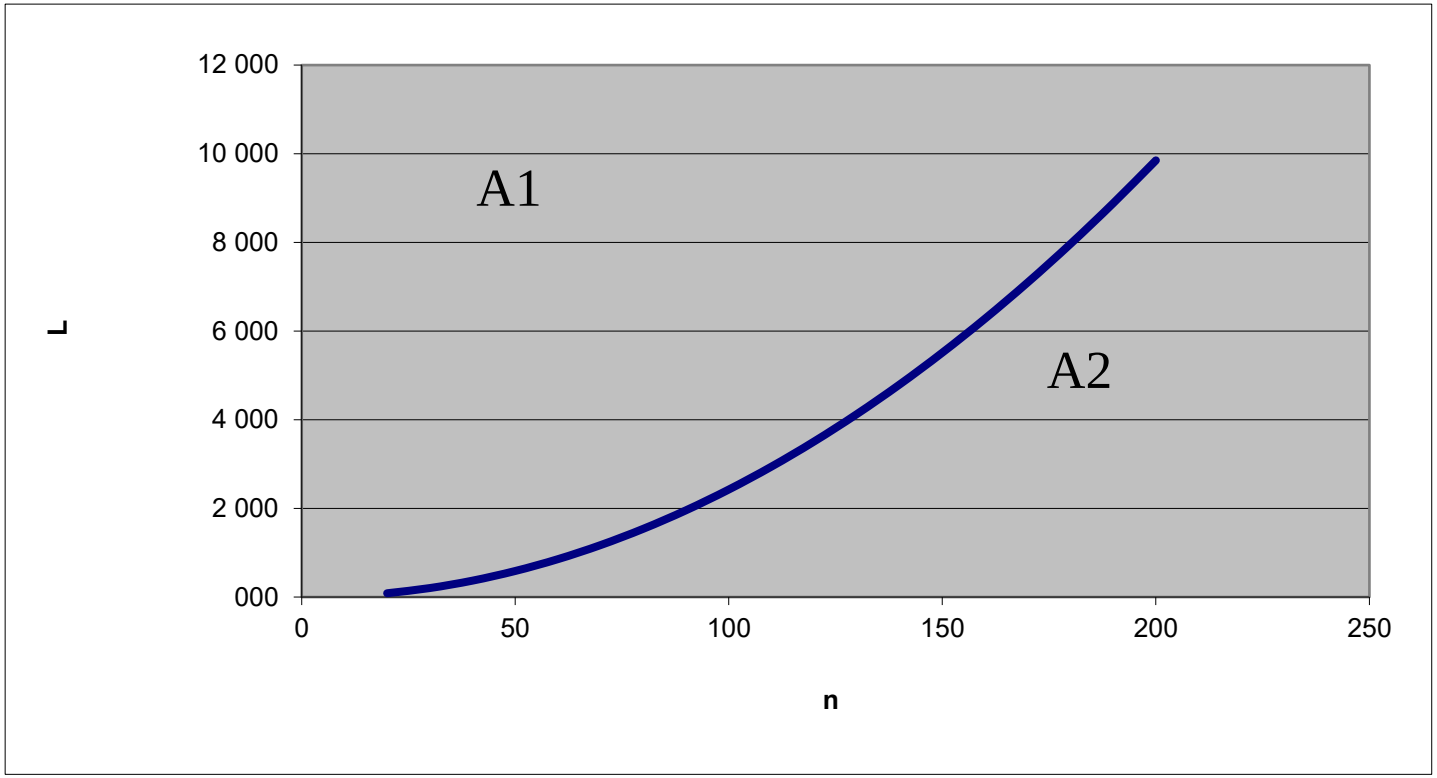

Рисунок 1. Разграничение областей рационального применения алгоритмов сортировки методом индексов A2 и методом вставки A1.

Для случаев конкретных входов можно вычислить пороговое значение L по формуле и сравнить с реальным размахом варьирования значений исходного массива. Например, при $n = 100$ пороговое значение L равно 2425, т. е. если размах варьирования в исходном массиве меньше, чем 2425, то более эффективным по трудоемкости будет алгоритм сортировка индексами (очевидно в рамках условий его применения), при больших размахах варьирования — алгоритм сортировки методом вставки.

Мы можем в рамках создания комбинированного алгоритмического решения написать монитор, который по результатам анализа текущего входа определяет более эффективный алгоритм (с учетом трудоемкости и ёмкостной эффективности, поскольку не надо забывать про дополнительный массив длины L для сортировки индексами). Мы получаем решение, адаптивное к текущему входу!

Литература

Головешкин В.А., Пономарев А.В., Ульянов М.В., Жукова Г.Н. Аналитическая функция трудоемкости в среднем алгоритма сортировки индексами на основе распределения размаха варьирования // Автоматизация и современные технологии. 2014. № 6. С. 11–17.