

Лекция 7



Сортировка



- Сортировка – процесс перегруппировки заданного множества объектов в некотором определенном порядке, т.е. расположение объектов по возрастанию или убыванию согласно определенному отношению порядка. Например, отношение \leq для чисел.

Задача сортировки



- Задача сортировки стоит в упорядочивании последовательности записей таким образом, чтобы значения ключевого поля составляли неубывающую последовательность.
- Дано: записи r_1, r_2, \dots, r_n со значениями ключей k_1, k_2, \dots, k_n .
- Найти: перестановку $r_{i_1}, r_{i_2}, \dots, r_{i_n}$ такую, что $k_{i_1} \leq k_{i_2} \leq \dots \leq k_{i_n}$.

Устойчивость сортировки



- Метод сортировки называется *устойчивым*, если в процессе сортировки относительное положение элементов с одинаковыми ключами не меняется.
- Иначе метод сортировки называется *неустойчивым*.

Внутренняя и внешняя сортировка



- Внутренние алгоритмы сортировки – применяются, если количество данных так мало, что процесс сортировки можно провести в оперативной памяти компьютера.
- Внешние алгоритмы сортировки применяются, когда данные располагаются во внешней памяти компьютера.

Сортировка подсчетом



- Метод основан на идее, что j -й ключ в окончательно упорядоченной последовательности превышает ровно $j - 1$ остальных ключей.
- Т.е. если некоторый ключ превышает ровно 27 других, и никакие два ключа не равны, то после сортировки соответствующая запись должна занять 28 место.
- Таким образом, идея состоит в том, чтобы сравнить попарно все ключи и подсчитать, сколько из них меньше каждого отдельного ключа.

Сортировка подсчетом



- Очевидный способ:

((сравнить k_j с k_i) при $1 \leq j \leq N$) при $1 \leq i \leq N$

- Половина операций при таком сравнении излишни, т.к. не нужно сравнивать ключ сам с собой и после сравнения k_a с k_b нет необходимости сравнивать k_b с k_a .

- Поэтому достаточно

((сравнить k_j с k_i) при $1 \leq j \leq i$) при $1 \leq i \leq N$.

Подсчет сравнений



- **Подсчет сравнений.** Алгоритм сортирует записи r_1, r_2, \dots, r_n по ключам k_1, k_2, \dots, k_n , используя вспомогательный массив $\text{count}[N]$ для подсчета числа ключей, меньших данного. После завершения алгоритма $\text{count}[j] + 1$ определяет окончательное положение записи r_j .
- **A1.** [Сбросить счетчики count] Инициализировать count нулями.
- **A2.** [Цикл по i] Выполнить шаг A3 при $i = N, N - 1, \dots, 2$, затем завершить выполнение процедуры.
- **A3.** [Цикл по j] Выполнить шаг A4 при $j = i - 1, i - 2, \dots, 1$.
- **A4.** [Сравнить k_i, k_j] Если $k_i < k_j$, увеличить $\text{count}[j]$ на 1, в противном случае увеличить $\text{count}[i]$ на 1.

Подсчет сравнений



КЛЮЧИ:	503	087	512	061	908	170	897	275	653	42 6	154	509	612	677	765	703
COUNT (нач.):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COUNT ($i = N$):	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	12
COUNT ($i = N - 1$):	0	0	0	0	2	0	2	0	0	0	0	0	0	0	13	12
COUNT ($i = N - 2$):	0	0	0	0	3	0	3	0	0	0	0	0	0	11	13	12
COUNT ($i = N - 3$):	0	0	0	0	4	0	4	0	1	0	0	0	9	11	13	12
COUNT ($i = N - 4$):	0	0	1	0	5	0	5	0	2	0	0	7	9	11	13	12
COUNT ($i = N - 5$):	1	0	2	0	6	1	6	1	3	1	2	7	9	11	13	12
.....
COUNT ($i = 2$):	6	1	8	0	15	3	14	4	10	5	2	7	9	11	13	12

Подсчет распределения



- Подсчет распределения применим, когда имеется много равных ключей и все они попадают в интервал $u \leq k_j \leq v$, ширина которого $(v - u)$ невелика.
- Предположим, что все ключи лежат между 1 и 100. При первом просмотре следует подсчитать, сколько имеется ключей, равных 1, 2, ..., 100, а при втором – расположить записи в соответствующих местах области вывода.

Подсчет распределения



- **Подсчет распределения.** Алгоритм сортирует записи r_1, r_2, \dots, r_n , используя вспомогательный массив $\text{count}[u] \dots \text{count}[v]$ в предположении, что все ключи – целые числа в диапазоне $u \leq k_j \leq v$, $1 \leq j \leq N$. На последнем шаге все записи в требуемом порядке переносятся в область вывода s_1, \dots, s_N .
- **В1.** [Сбросить счетчики count] Инициализировать count нулями.
- **В2.** [Цикл по i] Выполнить шаг В3 при $1 \leq j \leq N$; затем перейти к шагу В4.
- **В3.** [Увеличить $\text{count}[k_j]$] Увеличить значение $\text{count}[k_j]$ на 1.
- **В4.** [Накопление] (К этому моменту значение $\text{count}[i]$ есть число ключей, равных i) Присвоить $\text{count}[i] = \text{count}[i] + \text{count}[i - 1]$ для $i = u + 1, u + 2, \dots, v$.
- **В5.** [Цикл по j] (К этому моменту значение $\text{count}[i]$ есть число ключей, меньших или равных i , в частности $\text{count}[v] = i$) Выполнить шаг В6 при $j = N, N - 1, \dots, 1$ и завершить выполнение процедуры.
- **В6.** [Вывод r_j] Присвоить $i = \text{count}[k_j]$, $s_i = r_j$, $\text{count}[k_j] = i - 1$

Подсчет распределения



- Сам на доске!

Сортировка простыми вставками



- Способ предполагает, что перед рассмотрением записи r_j предыдущие записи r_1, \dots, r_{j-1} уже размещены так, что $k_1 \leq k_2 \leq \dots \leq k_{j-1}$. Будем сравнивать по очереди k_j с $k_{j-1}, k_{j-2} \dots$ до тех пор, пока не обнаружим, что запись r_j следует вставить между r_i и r_{i-1} . Тогда подвинем записи r_{i+1}, \dots, r_{j-1} на одну позицию и поместим новую запись в позицию $i + 1$. Поскольку r_i как бы «погружается» на положенный ей уровень, этот способ называют *просеиванием* или *погружением*.

Метод простых вставок



- **Сортировка методом простых вставок.** Записи r_1, \dots, r_N перерасмещаются на одном и том же месте. После завершения сортировки их ключи будут упорядочены: $k_1 \leq \dots \leq k_N$.
- **С1** [Цикл по j] Выполнить шаги С2 до С5 при $j = 2, 3, \dots, N$ и после этого завершить выполнение процедуры.
- **С2** [Установка i, k, r] Присвоить $i = j - 1, k = k_j, r = r_j$. (На последующих шагах будет предпринята попытка вставить запись r в нужное место, сравнивая k с k_i при убывающих значениях i .)
- **С3** [Сравнение $k : k_i$] Если $k \geq k_i$, перейти к шагу С5 (нашли искомое место для записи r).
- **С4** [Перемещение r_i , уменьшение i] Присвоить $r_{i+1} = r_i, i = i - 1$. Если $i > 0$, вернуться к шагу С3. (Если $i = 0$, k – наименьший из рассмотренных до сих пор ключей, а значит, запись r должна занять первую позицию).
- **С5** [Перенос r на место r_{i+1}] Присвоить $r = r_{i+1}$.

Метод простых вставок



ПРИМЕР СОРТИРОВКИ МЕТОДОМ ПРОСТЫХ ВСТАВОК

[^] 503 : 087
087 503 ; [^] 512
[^] 087 503 512 : 061
061 087 503 512 ; [^] 908
061 087 [^] 503 512 908 : 170
061 087 170 503 512 [^] 908 : 897
.
061 087 154 170 275 426 503 509 512 612 653 677 [^] 765 897 908 : 703
061 087 154 170 275 426 503 509 512 612 653 677 703 765 897 908
