Сортировка подсчётом

Материал из Википедии — свободной энциклопедии

Сортировка подсчётом[1] (англ. *counting sort*[2]; сортировка посредством подсчёта[3] англ. *sorting by counting*[4]) — алгоритм сортировки, в котором используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов. Применение сортировки подсчётом целесообразно лишь тогда, когда сортируемые числа имеют (или их можно отобразить в) диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством, например, миллион натуральных чисел меньших 1000.

Предположим, что входной массив состоит из n целых чисел в диапазоне от 0 до $k-1$, где $k \in \mathbb{N}$. Далее алгоритм будет обобщён для произвольного целочисленного диапазона. Существует несколько модификаций сортировки подсчётом, ниже рассмотрены три линейных и одна квадратичная, которая использует другой подход, но имеет то же название.

Содержание

Простой алгоритм

Это простейший вариант алгоритма. Создать вспомогательный массив `C[0..k - 1]`, состоящий из нулей, затем последовательно прочитать элементы входного массива `A`, для каждого `A[i]` увеличить `C[A[i]]` на единицу. Теперь достаточно пройти по массиву `C`, для каждого $j \in \{0, \ldots, k-1\}$ в массив `A` последовательно записать число `j` `C[j]` раз.

```
SimpleCountingSort:
    for i = 0 to k - 1
        C[i] = 0;
    for i = 0 to n - 1
        C[A[i]] = C[A[i]] + 1;
    b = 0;
    for j = 0 to k - 1
        for i = 0 to C[j] - 1
            A[b] = j;
            b = b + 1;
```

Алгоритм со списком

Этот вариант (англ. *pigeonhole sorting, count sort*) используется, когда на вход подается массив структур данных, который следует отсортировать по ключам (`key`). Нужно создать вспомогательный массив `C[0..k - 1]`, каждый `C[i]` в дальнейшем будет содержать список элементов из входного массива. Затем последовательно прочитать элементы входного массива `A`, каждый `A[i]` добавить в список `C[A[i].key]`. В заключении пройти по массиву `C`, для каждого $j \in \{0, \ldots, k-1\}$ в массив `A` последовательно записывать элементы списка `C[j]`. Алгоритм устойчив.

```
ListCountingSort
    for i = 0 to k - 1
        C[i] = NULL;
    for i = 0 to n - 1
        C[A[i].key].add(A[i]);
    b = 0;
    for j = 0 to k - 1
        p = C[j];
        while p != NULL
            A[b] = p.data;
            p = p.next();
            b = b + 1;
```

Устойчивый алгоритм

В этом варианте помимо входного массива `A` потребуется два вспомогательных массива — `C[0..k - 1]` для счётчика и `B[0..n - 1]` для отсортированного массива. Сначала следует заполнить массив `C` нулями, и для каждого `A[i]` увеличить `C[A[i]]` на 1. Далее подсчитывается количество элементов меньших или равных $k-1$. Для этого каждый `C[j]`, начиная с `C[1]`, увеличивают на `C[j - 1]`. Таким образом в последней ячейке будет находиться количество элементов от 0 до $k-1$ существующих во входном массиве. На последнем шаге алгоритма читается входной массив с конца, значение `C[A[i]]` уменьшается на 1 и в каждый `B[C[A[i]]]` записывается `A[i]`. Алгоритм устойчив.

```
StableCountingSort
    for i = 0 to k - 1
        C[i] = 0;
    for i = 0 to n - 1
        C[A[i]] = C[A[i]] + 1;
    for j = 1 to k - 1
        C[j] = C[j] + C[j - 1];
    for i = n - 1 to 0
        C[A[i]] = C[A[i]] - 1;
        B[C[A[i]]] = A[i];
```

Обобщение на произвольный целочисленный диапазон

Возникает несколько вопросов. Что делать, если диапазон значений (min и max) заранее не известен? Что делать, если минимальное значение больше нуля или в сортируемых данных присутствуют отрицательные числа? Первый вопрос можно решить линейным поиском min и max, что не повлияет на асимптотику алгоритма. Второй вопрос несколько сложнее. Если min больше нуля, то следует при работе с массивом `C` из `A[i]` вычитать min, а при обратной записи прибавлять. При наличии отрицательных чисел нужно при работе с массивом `C` к `A[i]` прибавлять |min|, а при обратной записи вычитать.

Анализ

В первых двух алгоритмах первые два цикла работают за $\Theta(k)$ и $\Theta(n)$, соответственно; двойной цикл за $\Theta(n+k)$. В третьем алгоритме циклы занимают $\Theta(k)$, $\Theta(n)$, $\Theta(k)$ и $\Theta(n)$, соответственно. Итого все три алгоритма имеют линейную временную трудоёмкость $\Theta(n+k)$. Используемая память в первых двух алгоритмах равна $\Theta(k)$, а в третьем

$\Theta(n+k)$.

Квадратичный алгоритм сортировки подсчётом

Также сортировкой подсчётом называют немного другой алгоритм. В нём используются входной массив `A` и вспомогательный массив `B` для отсортированного множества. В алгоритме следует для каждого элемента входного массива `A[i]` подсчитать количество элементов меньших него c_1 и количество элементов, равных ему, но стоящих ранее c_2 ($c = c_1 + c_2$). `B[c]` присвоить `A[i]`. Алгоритм устойчив.

```
SquareCountingSort
    for i = 0 to n - 1
        c = 0;
        for j = 0 to i - 1
            if A[j] <= A[i]
                c = c + 1;
        for j = i + 1 to n - 1
            if A[j] < A[i]
                c = c + 1;
        B[c] = A[i];
```

Анализ

Очевидно, временная оценка алгоритма равна $\Theta(n^2)$, память $\Theta(n)$.

Примеры реализации

C++

Простой алгоритм.

```
 1 void counting_sort(int* vec, unsigned int len, int min, int max)
 2 {
 3     assert(min <= max);
 4     assert(vec != NULL);
 5
 6     int * cnt = new int[max-min+1];
 7
 8     for (int i = min; i <= max; ++i)
 9         cnt[i - min] = 0;
10
11     for (int i = 0; i < len; ++i)
12         ++cnt[vec[i] - min];
13
14     for (int i = min; i <= max; ++i)
15         for(int j = cnt[i - min]; j--;)
16             *vec++ = i;
17
18     delete [] cnt;
19 }
```

Компонентный Паскаль

Простой алгоритм.

```
PROCEDURE CountingSort (VAR a: ARRAY OF INTEGER; min, max: INTEGER);
  VAR
    i, j, c: INTEGER;
    b: POINTER TO ARRAY OF INTEGER;
BEGIN
  ASSERT(min <= max);
  NEW(b, max - min + 1);
  FOR i := 0 TO LEN(a) - 1 DO INC(b[a[i] - min]) END;
```

```
  i := 0;
  FOR j := min TO max DO
    c := b[j - min];
    WHILE c > 0 DO
      a[i] := j; INC(i); DEC(c)
    END
  END
END CountingSort;
```

Реализация на PascalABC.Net

```
1 const
2   n = 5;
3   m = 12; // Максимальное значение всех элементов в a.
4
5 var
6   a: array [0..n] of integer;
7   c: array [0..m] of integer; // Вспомогательный массив.
8   i, j: integer; // Переменные, играющие роль индексов.
9   k:integer;
10 begin
11   for i := 0 to n do
12     a[i] := Random(m); // Заполнение массива.
13   for i := 0 to m do
14     c[i] := 0;  //Обнуление вспомогательного массива
15   for i := 0 to n do
16     c[a[i]] := c[a[i]] + 1;
17   j := 0; // Обнуление j. Хороший стиль программирования обнулять все переменные изначально, поскольку не
все компиляторы это делают автоматически.
18   for i := 0 to m do
19     for  k := 1 to c[i] do
20     begin
21         a[j] := i;
22         Inc(j);
23     end;
24   Writeln(a);
25 end.
```

См. также

- Алгоритм сортировки
- О-большое
- Временная сложность алгоритма

Примечания

1. *Кормен.* Сортировка подсчетом // Алгоритмы: Вводный курс. — Вильямс, 2014. — С. 71. — 208 с. — ISBN 978-5-8459-1868-0.
2. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. & Stein, Clifford (2001), "8.2 Counting Sort", *Introduction to Algorithms* (2nd ed.), MIT Press and McGraw-Hill, с. 168–170, ISBN 0-262-03293-7.
3. *Кнут.* Сортировка посредством подсчёта // Искусство программирования. — Т. 3. — С. 77.
4. Knuth, D. E. (1998), "Section 5.2, Sorting by counting", *The Art of Computer Programming, Volume 3: Sorting and Searching* (2nd ed.), Addison-Wesley, с. 75-80, ISBN 0-201-89685-0.

Литература

- *Левитин А. В.* Глава 7. Пространственно-временной компромисс: Сортировка подсчетом // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 331–339. — 576 с. — ISBN 978-5-8459-0987-9
- *Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клифорд.* **Глава 8. Сортировка за линейное время** // Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е издание. — М.: «Вильямс», 2005. — С. 224 - 226. — ISBN 5-8459-0857-4.

Ссылки

- Визуализатор1 (http://rain.ifmo.ru/cat/view.php/vis/sorts/linear-2005) — Java-аплет.
- Визуализатор2 (http://rain.ifmo.ru/cat/view.php/vis/sorts/linear-2001) — Java-аплет.

Источник — https://ru.wikipedia.org/w/index.php?title=Сортировка_подсчётом&oldid=97560246