

Википедия

Сортировка выбором

Материал из Википедии — свободной энциклопедии

Сортировка выбором (*Selection sort*) — алгоритм сортировки. Может быть как устойчивый, так и неустойчивый. На массиве из *n* элементов имеет время выполнения в худшем, среднем и лучшем случае $\Theta(n^2)$, предполагая что сравнения делаются за постоянное время.

Содержание

Алгоритм без дополнительного выделения памяти

Реализация на VBA

Литература

См. также

Ссылки

Алгоритм без дополнительного выделения памяти

Шаги алгоритма:

1. находим номер минимального значения в текущем списке

2. производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)

3. теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

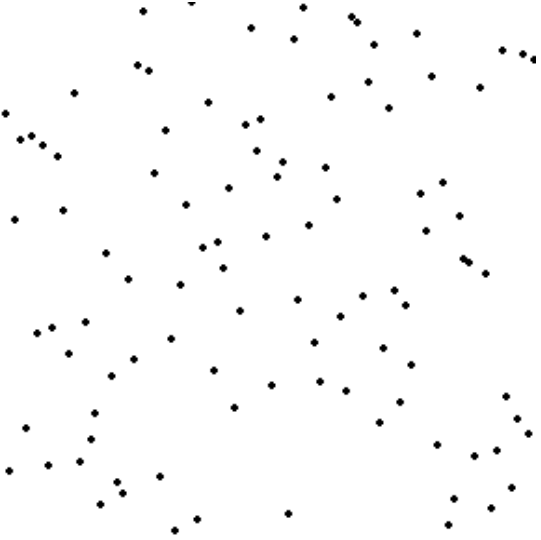
Для реализации устойчивости алгоритма необходимо в пункте 2 минимальный элемент непосредственно вставлять в первую неотсортированную позицию, не меняя порядок остальных элементов.

Пример неустойчивой реализации:

C++

```
1 #include <cstdlib>
2 #include <utility>
3 using namespace std;
4
5 template<typename T>
6 void selection_sort(T array[], size_t size) {
7     for (size_t idx_i = 0; idx_i < size - 1; idx_i++) {
8         size_t min_idx = idx_i;
9         for (size_t idx_j = idx_i + 1; idx_j < size; idx_j++) {
10             if (array[idx_j] < array[min_idx]) {
11                 min_idx = idx_j;
```

Сортировка выбором



Действие алгоритма на примере сортировки случайных точек.

Предназначение	Алгоритм сортировки
Структура данных	Массив
Худшее время	$O(n^2)$
Лучшее время	$O(n^2)$
Среднее время	$O(n^2)$
Затраты памяти	$O(n)$ всего, $O(1)$ дополнительно

```

12     }
13 }
14
15     if (min_idx != idx_i) {
16         swap(array[idx_i], array[min_idx]);
17         min_idx = idx_i;
18     }
19 }
20 }

```

C#

```

1 public static IList<int> Selection(IList<int> list)
2 {
3     for (int i = 0; i < list.Count-1; i++)
4     {
5         int min = i;
6         for (int j = i + 1; j < list.Count; j++)
7         {
8             if (list[j] < list[min])
9             {
10                 min = j;
11             }
12         }
13         int dummy = list[i];
14         list[i] = list[min];
15         list[min] = dummy;
16     }
17     return list;
18 }

```

PL/SQL

```

1 type sort_choice_list is table of integer index by binary_integer;
2
3 function SORT_CHOICE return sort_choice_list
4 is
5     list sort_choice_list;
6     l_min pls_integer;
7     l_dummy pls_integer;
8 begin
9
10     for n in 1..100 loop
11         list(n):=dbms_random.random; --инициализация массива случайными числами
12     end loop;
13
14     for i in list.first..list.last loop
15         l_min:=i;
16         for j in (i + 1)..list.last loop
17             if (list(j) < list(l_min)) then
18                 l_min := j;
19             end if;
20         end loop;
21         l_dummy:=list(i);
22         list(i):=list(l_min);
23         list(l_min) := l_dummy;
24     end loop;
25
26     return list;
27
28 end SORT_CHOICE;

```

Java

```

1 public static void sort(int[] arr) {
2     for (int min = 0; min < arr.length - 1; min++) {
3         int least = min;
4         for (int j = min + 1; j < arr.length; j++) {
5             if (arr[j] < arr[least]) {
6                 least = j;
7             }
8         }
9         int tmp = arr[min];

```

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

Сортировка
выбором

```

10         arr[min] = arr[least];
11         arr[least] = tmp;
12     }
13 }

```

Ruby

```

1 def selection_sort(array)
2   for min in 0..array.count-2
3     least = min
4     for j in (min + 1)..array.count-1
5       if array[j] < array[least]
6         least = j
7       end
8     end
9     temp = array[min]
10    array[min] = array[least]
11    array[least] = temp
12  end
13 end

```

Покажем, почему данная реализация является неустойчивой.

Рассмотрим следующий массив из элементов, каждый из которых имеет два поля. Сортировка идет по первому полю.

Массив до сортировки:

{ (2, a), (2, b), (1, a) }

Уже после первой итерации внешнего цикла будем иметь отсортированную последовательность:

{ (1, a), (2, b), (2, a) }

Теперь заметим, что взаимное расположение элементов (2, a) и (2, b) изменилось. Таким образом, рассматриваемая реализация является неустойчивой.

Так как после каждого прохода по внутреннему циклу делается только один обмен, то общее число обменов равно $N-1$, что в $N/2$ раз меньше, чем в сортировке пузырьком.

Число проходов по внутреннему циклу равно $N-1$ даже в случае сортировки частично или полностью отсортированного массива.

Наихудший случай:

Число сравнений в теле цикла равно $(N-1)*N/2$.

Число сравнений в заголовках циклов $(N-1)*N/2$.

Число сравнений перед операцией обмена $N-1$.

Суммарное число сравнений N^2-1 .

Число обменов $N-1$.

Наилучший случай:

Время сортировки 10000 коротких целых чисел на одном и том же программно-аппаратном комплексе сортировкой выбором составило ≈ 40 сек., а ещё более улучшенной сортировкой пузырьком ≈ 30 сек.

Пирамидальная сортировка сильно улучшает базовый алгоритм, используя структуру данных «куча» для ускорения нахождения и удаления минимального элемента.

Существует также двунаправленный вариант сортировки методом выбора, в котором на каждом проходе отыскиваются и устанавливаются на свои места и минимальное, и максимальное значения.

Реализация на VBA

```
Sub VSort()
    For i = 1 To 10 Step 1
        For j = i + 1 To 10 Step 1
            If ActiveSheet.Cells(i, 1) > ActiveSheet.Cells(j, 1) Then
                ActiveSheet.Cells(i, 1) = ActiveSheet.Cells(i, 1) + ActiveSheet.Cells(j, 1)
                ActiveSheet.Cells(j, 1) = ActiveSheet.Cells(i, 1) - ActiveSheet.Cells(j, 1)
                ActiveSheet.Cells(i, 1) = ActiveSheet.Cells(i, 1) - ActiveSheet.Cells(j, 1)
            End If
        Next j
    Next i
End Sub
```

```
' С использованием переменной c.
Sub VSort()
    For i = 1 To 10 Step 1
        For j = i + 1 To 10 Step 1
            If ActiveSheet.Cells(i, 1) > ActiveSheet.Cells(j, 1) Then
                c = ActiveSheet.Cells(i, 1)
                ActiveSheet.Cells(i, 1) = ActiveSheet.Cells(j, 1)
                ActiveSheet.Cells(j, 1) = c
            End If
        Next j
    Next i
End Sub
```

Здесь сортируются ячейки напрямую. `ActiveSheet.Cells(i, 1)` - обращение к ячейке с (1, i). Обратите внимание, что координаты ячеек в VBA задаются как (y, x) - номер колонки и номер ряда, в котором расположена ячейка. Также помните, что ячейки индексируются с 1.

Литература

- Левитин А. В.* Глава 3. Метод грубой силы: Сортировка выбором // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 143—144. — 576 с. — ISBN 978-5-8459-0987-9
- Роберт Седжвик. Часть III. Глава 6. Элементарные методы сортировки: 6.2 Сортировка выбором* // Алгоритмы на C++ = Algorithms in C++. — М.: «Вильямс», 2011. — С. 246-247. — ISBN 978-5-8459-1650-1.
- Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К.* Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд. — М.: Вильямс, 2005. — 1296 с. — ISBN 5-8459-0857-4.

См. также

- Список алгоритмов сортировки
- Сортировка пузырьком
- Сортировка вставками

Ссылки

- Статья "Сортировка выбором" на сайте [algotist.manual.ru](http://algotist.manual.ru/sort/select_sort.php) (http://algotist.manual.ru/sort/select_sort.php). Проверено 25 сентября 2012. Архивировано (<http://www.webcitation.org/6BSxd6Dqs>) 17 октября 2012 года.
- Динамическая визуализация 7 алгоритмов сортировки с открытым исходным кодом (<https://airtucha.github.io/SortVis/>)
-

Получено от "https://ru.wikipedia.org/w/index.php?title=Сортировка_выбором&oldid=96975671"

Эта страница в последний раз была отредактирована 21 декабря 2018 в 16:15.

Текст доступен по лицензии [Creative Commons Attribution-ShareAlike](#); в отдельных случаях могут действовать дополнительные условия.

