Тема лекции:

Алгоритмы сортировки данных

Сортировка – процесс упорядочивания данных по заданному правилу.

Обычно с упорядоченными элементами проще работать, чем с произвольно расположенными: легче найти необходимые элементы, исключить или добавить новые.

В общем, методы сортировок разделяют на два типа: сортировку массивов и сортировку файлов.

На практике массивы хранятся в оперативной памяти устройства, а файлы в более «медленных» внешних устройствах хранения но более вместительных.

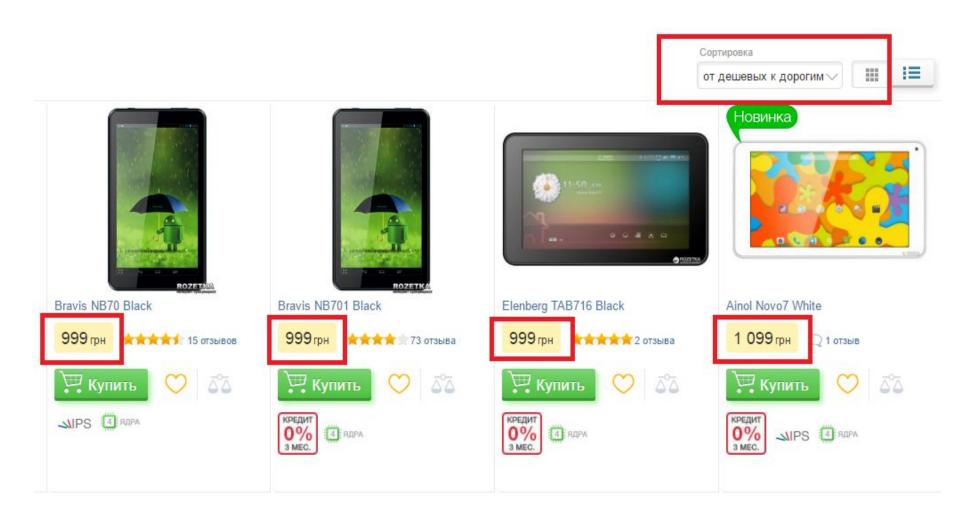


- К методам сортировки применяются два основных требования:
- 1) Математическое алгоритм должен быть корректен математически;
- 2)Бизнес-требование алгоритм должен удовлетворять требованиям конкретной бизнес-задачи.

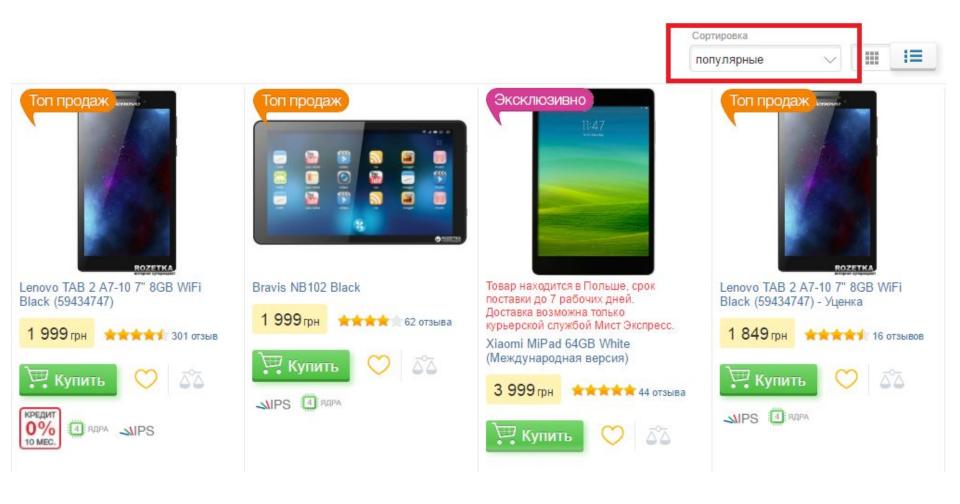
Математическая корректность может не удовлетворять бизнес-требование, но бизнес-требование должно быть математически корректным.



Пример математической корректности алгоритма



Пример бизнес-требования к алгоритму



Сортировка массивов

Главное требование к разработке алгоритмов сортировки массивов — экономное использование доступной оперативной памяти.

Хорошая мера эффективности – число необходимых сравнений и перестановок элементов.

Хорошие методы сортировки требуют порядка n*log(n) сравнений, простые же — n^2 .

Характеристики простых методов:

- 1)Хороши для понимания основных принципов сортировок. 2)Легки для понимания.
- 3)Обычно более быстры для малых *п* но нельзя использовать для больших **п**.

Пузырьковая сортировка O(n²)

Дано: массив значений { A[1], A[2], ..., A[n] } n – число элементов

Псевдокод:

```
Для і от 1 до n-1 выполнять
Для k от i+1 до n выполнять
Если A[k] < A[i] то
tmp:=A[k];
A[k]:=A[i];
A[i]:=tmp;
```

Пузырьковая сортировка O(n²)

6 5 3 1 8 7 2 4

Сортировка вставками O(n²)

```
Дано: массив значений { A[1], A[2], ..., A[n] } 
n – число элементов
```

Псевдокод:

```
Для і от 1 до n выполнять key = A[i]; k=i-1; Пока (k>0) и (A[j]>key) выполнять A[k+1]:=A[k]; k:=k-1; A[k+1]:=key;
```

Сортировка вставками O(n²)

6 5 3 1 8 7 2 4

Сортировка слиянием O(n log₂(n))

- 1. Сортируемый массив разбивается на две части примерно одинакового размера;
- 2. Каждая из получившихся частей сортируется отдельно, например этим же самым алгоритмом;
- 3. Два упорядоченных массива половинного размера соединяются в один.

Алгоритм объединения двух упорядоченных массивов A и B (K1 – длина A, K2 – длина B):

i = 1; k = 1;
Пока (i<=K1) и (k<=K2) выполнять
если A[i] < B[k] то поместить в выходной массив A[i] и i=i+1
иначе поместить в выходной массив B[k] и k=k+1
Поместить в выходной массив оставшиеся элементы массива
А с i-го по K1 и массива В с k-го по K2.

Сортировка слиянием $O(n log_2(n))$

6 5 3 1 8 7 2 4

- 1) Выбираем в массиве некоторый элемент, который будем называть опорным элементом. Например, средний по положению.
- 2) Операция разделения массива: реорганизуем массив таким образом, чтобы все элементы, меньшие или равные опорному элементу, оказались слева от него, а все элементы, большие опорного справа от него. Обычный алгоритм операции:
 - 1. Два индекса I и г, приравниваются к минимальному и максимальному индексу разделяемого массива соответственно.
 - 2. Вычисляется индекс опорного элемента m. В нашем случае m=(I+r)/2.
 - 3. Индекс I последовательно увеличивается до m до тех пор, пока I-й элемент не превысит опорный.
 - 4. Индекс r последовательно уменьшается до m до тех пор, пока r-й элемент не окажется меньше либо равен опорному.
 - 5. Если r = I найдена середина массива операция разделения закончена, оба индекса указывают на опорный элемент.
 - 6. Если I < r найденную пару элементов нужно обменять местами и продолжить операцию разделения с тех значений I и r, которые были достигнуты.
- 3) Рекурсивно упорядочиваем подмассивы, лежащие слева и справа от опорного элемента, пока размеры подмассивов не будут равны 1

Дано множество

```
{9,6,3,4,10,8,2,7}
```

Берем **9** в качестве базового элемента. Сравниваем **9** с противоположно стоящим элементом, в данном случае это **7**. **7** меньше, чем **9**, следовательно элементы меняются местами.

```
{7,6,3,4,10,8,2,9}
```

Далее начинаем последовательно сравнивать элементы с **9**, и менять их местами в зависимости от сравнения.

```
{7,6,3,4,10,8,2,9}
{7,6,3,4,10,8,2,9}
{7,6,3,4,10,8,2,9}
{7,6,3,4,9,8,2,10} - 9 и 10 меняем местами.
{7,6,3,4,8,9,2,10} - 9 и 8 меняем местами.
{7,6,3,4,8,2,9,10} - 2 и 9 меняем местами.
```

После такого перебрасывания элементов весь массив разбивается на два подмножества, разделенных элементом 9.

```
{7,6,3,4,8,2} и {10}
```

Далее по уже отработанному алгоритму сортируются эти подмножества. Подмножество из одного элемента естественно можно не сортировать. Выбираем в первом подмножестве базовый элемент 7.

```
{7,6,3,4,8,2}

{2,6,3,4,8,7} - меняем местами 2 и 7.

{2,6,3,4,8,7}

{2,6,3,4,8,7}

{2,6,3,4,8,7}

{2,6,3,4,8,7}

{2,6,3,4,7,8} - меняем местами 7 и 8
```

Получили снова два подмножества.

Далее алгоритм продолжается.

