Как правильно делать сортировку в Java

<https://www.toptal.com/developers/sorting-algorithms> - ссылка на страницу, где представлено визуальное определение скорости работы разных алгоритмов сортировки в зависимости от начального состояния отсортированнлсти списка.

Зачем дальше везде добавляется метод toString() я не знаю!!!!

**Использование метода sort()**

Если нужно отсортировать коллекцию (**List**), то применяйте метод **Collections.sort**().

// Collections.sort(…)

List<ObjectName> list = **new** ArrayList<ObjectName>();

**Collections.sort**(list,

**new** Comparator<ObjectName>() {

**public** **int** compare(ObjectName o1, ObjectName o2) {

**return** o1.toString().compareTo(o2.toString());

}

});

Если требуется отсортировать **массив**, используйте метод **Arrays.sort**().

// Arrays.sort(…)

ObjectName[] arr = **new** ObjectName[10];

Arrays.sort(arr,

**new** Comparator<ObjectName>() {

**public** **int** compare(ObjectName o1, ObjectName o2) {

**return** o1.toString().compareTo(o2.toString());

}

});

Метод **sort**() очень удобен, когда коллекция или массив, уже заполнены значениями.

**Применение самосортирующихся структур данных**

Если нужно отсортировать множество (**Set**), используйте структуру **TreeSet** для сортировки.

// TreeSet

Set<ObjectName> sortedSet = **new** **TreeSet**<ObjectName>(

**new** Comparator<ObjectName>() {

**public** **int** compare(ObjectName o1, ObjectName o2) {

**return** o1.toString().compareTo(o2.toString());

}

});

sortedSet.addAll(unsortedSet);

Если вам требуется отсортировать словарь (**Map**), используйте структуру **TreeMap** для сортировки. TreeMap сортируется по ключу (key).

// TreeMap – использующий String ключи и компаратор (Comparator) CASE\_INSENSITIVE\_ORDER, упорядочивающий строки (String) методом compareToIgnoreCase (т.е. игнорируя регистр)

Map<String, Integer> sortedMap = **new** **TreeMap**<String, Integer> (String. CASE\_INSENSITIVE\_ORDER);

sortedMap.putAll(unsortedMap);

//TreeMap – общий случай, компаратор указывается вручную

Map<ObjectName, String> sortedMap = **new** TreeMap<ObjectName, String>(

**new** Comparator<ObjectName>() {

**public** **int** compare(ObjectName o1, ObjectName o2) {

**return** o1.toString().compareTo(o2.toString());

}

});

sortedMap.putAll(unsortedMap);

Вышеописанный подход очень полезен в тех случаях, если вам нужно проводить большое количество операций поиска элементов в коллекции. Самосортирующиеся структуры данных имеют эффективность O (log(n)), что лучше, чем O(n). Это означает, что при удвоении количества данных в коллекции время поиска не удваивается, а увеличивается на постоянную величину (*прим. перев.*)

**Плохой подход к задаче сортировки**

До сих пор можно встретить примеры, когда программисты самостоятельно описывают алгоритмы сортировки. Рассмотрим код сортировки, представленный ниже (сортировка double-массива по возрастанию (*прим. перев.*)). Этот код не только не эффективен, но и не читабелен. И таких примеров много.

**double** t;

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = i + 1; j < N; j++)

**if** (r[j] < r[i]) {

t = r[i];

r[i] = r[j];

r[j] = t;

}

**ДЛЯ ТОГО, ЧТО БЫ ОСТАВИТЬ КОММЕНТАРИЙ ВЫ ДОЛЖНЫ**[**АВТОРИЗОВАТЬСЯ**](https://javarush.ru/login)

[**Александр Моцар**](https://javarush.ru/users/274253)28 уровень, Минск

31 октября 2018, 21:33

последняя ручная сортировка - сортировка выбором, это действительно самый медленный вариант который есть. НО стоит отметить что сортировки методом Arrays.sort и Collections.sort реализованы методом "Quick sort" и в самом неблагоприятном случае время для такого вида сортировки ничем не отличается от приведенной выше самописной сортировки. Хотя бесспорно Arrays.sort() лучше использовать чем то, что предлагают в конце.

К чему вообще стал писать этот коммент: две недели назад был на двух собеседованиях и на каждом из них, что было достаточно удивительно, меня попросили написать код сортировки ПРИЧЕМ В РУЧНУЮ. Ответ типа сортируем методом sort() не принимался. На втором собеседовании интервьюера вполне устроил код самой простой пузырьковой сортировки, приведенный выше вариант тоже вполне бы сгодился. А вот на первом, такой простой вариант уже не прокатил) Также совершенно недавно знакомый делал тестовое задание на java junior в котором одной из задач было объединить два не отсортированных массива чисел в один отсортированный) Естественно предполагалось что человек не просто два раза вызовет метод sort(), а как минимум сделает это но в двух разных потоках. В общем к моему удивлению, такие вопросы на интервью любят. Поэтому лучше кроме стандартных методов потратить полчаса времени и реализовать для себя пару вариантов сортировки вручную, желательно с применением многопоточности. Например пузырьковую ну и сортировку слиянием, алгоритмы достаточно простые. Чтоб на собеседовании не вспоминать лихорадочно "как жеж там оно это пишется, ведь просто ж все)", а взять и написать за минуту, да еще и О-нотацию указать для каждого варианта.