ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ(ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННЫХ СИСТЕМ

ОТЧЕТ ПО ТЕМЕ:

**«БЫСТРАЯ СОРТИРОВКА»**

Выполнила: Федорчук Виктория, гр. 11-002

Казань 2021г.

1. *Название алгоритма, краткая историческая справка*

**Быстрая сортировка**, **сортировка Хоара** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *quicksort*), часто называемая ***qsort***  —[алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), разработанный английским информатиком [Тони Хоаром](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D0%B0%D1%80,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7_%D0%AD%D0%BD%D1%82%D0%BE%D0%BD%D0%B8_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4) во время его работы в [МГУ](https://ru.wikipedia.org/wiki/%D0%9C%D0%93%D0%A3) в [1960 году](https://ru.wikipedia.org/wiki/1960_%D0%B3%D0%BE%D0%B4).

1. *Основной принцип устройства. Особенности*

QuickSort является существенно улучшенным вариантом алгоритма сортировки с помощью прямого обмена (его варианты известны как «[Пузырьковая сортировка](https://ru.wikipedia.org/wiki/%D0%9F%D1%83%D0%B7%D1%8B%D1%80%D1%8C%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0)» и «[Шейкерная сортировка](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D0%B9%D0%BA%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0" \o "Шейкерная сортировка)»), известного в том числе своей низкой эффективностью.

Принципиальное отличие состоит в том, что в первую очередь производятся перестановки на наибольшем возможном расстоянии и после каждого прохода элементы делятся на две независимые группы. (Таким образом улучшение самого неэффективного прямого метода сортировки дало в результате один из наиболее эффективных улучшенных методов.)

Общая идея алгоритма состоит в следующем:

* Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
* Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».
* Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

На практике массив обычно делят не на три, а на две части: например, «меньшие опорного» и «равные и большие»; такой подход в общем случае эффективнее, так как упрощает алгоритм разделения.

1. *Оценка временной сложности*

Операция разделения массива на две части относительно опорного элемента занимает время O(log2 n) {\displaystyle O(\log \_{2}n)}. Поскольку все операции разделения, проделываемые на одной глубине рекурсии, обрабатывают разные части исходного массива, размер которого постоянен, суммарно на каждом уровне рекурсии потребуется также {\displaystyle O(n)}O(n) операций. Следовательно, общая сложность алгоритма определяется лишь количеством разделений, то есть глубиной рекурсии. Глубина рекурсии, в свою очередь, зависит от сочетания входных данных и способа определения опорного элемента.

1. *Выводы, плюсы и минусы алгоритма, его применимость*

**Достоинства:**

* + Один из самых быстродействующих (на практике) из алгоритмов внутренней сортировки общего назначения.
  + Алгоритм очень короткий: запомнив основные моменты, его легко написать «из головы», невелика константа при {\displaystyle n\log n}n log n.
  + Требует лишь {\displaystyle O(1)}O(1) дополнительной памяти для своей работы. (Не улучшенный рекурсивный алгоритм в худшем случае {\displaystyle O(n)}O(n) памяти)
  + Хорошо сочетается с механизмами [кэширования](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88-%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) и [виртуальной памяти](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C).
  + Допускает естественное распараллеливание (сортировка выделенных подмассивов в параллельно выполняющихся подпроцессах).
  + Допускает эффективную модификацию для сортировки по нескольким ключам (в частности — [алгоритм Седжвика](https://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A1%D0%B5%D0%B4%D0%B6%D0%B2%D0%B8%D0%BA%D0%B0&action=edit&redlink=1) для сортировки строк): благодаря тому, что в процессе разделения автоматически выделяется отрезок элементов, равных опорному, этот отрезок можно сразу же сортировать по следующему ключу.
  + Работает на [связных списках](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA) и других структурах с последовательным доступом, допускающих эффективный проход как от начала к концу, так и от конца к началу.

**Недостатки:**

* Сильно деградирует по скорости (до {\displaystyle O(n^{2})}O(n^2)) в худшем или близком к нему случае, что может случиться при неудачных входных данных.
* Прямая реализация в виде функции с двумя рекурсивными вызовами может привести к ошибке [переполнения стека](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D1%82%D0%B5%D0%BA%D0%B0), так как в худшем случае ей может потребоваться сделать {\displaystyle O(n)}O(n) вложенных рекурсивных вызовов.
* [Неустойчив](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D0%BE%D0%B9%D1%87%D0%B8%D0%B2%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0).

1. *Список литературы*

* <https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0>
* <https://habr.com/ru/company/otus/blog/524948/>

1. *Код программы*

#include <stdio.h>  
#include <stdlib.h>  
  
using namespace std;  
  
void quickSort(int \*numbers, int left, int right) {  
 int pivot; // разрешающий элемент  
 int l\_hold = left; //левая граница  
 int r\_hold = right; // правая граница  
 pivot = numbers[left];  
 while (left < right) // пока границы не сомкнутся  
 {  
 while ((numbers[right] >= pivot) && (left < right))  
 right--; // сдвигаем правую границу пока элемент [right] больше [pivot]  
 if (left != right) // если границы не сомкнулись  
 {  
 numbers[left] = numbers[right]; // перемещаем элемент [right] на место разрешающего  
 left++; // сдвигаем левую границу вправо  
 }  
 while ((numbers[left] <= pivot) && (left < right))  
 left++; // сдвигаем левую границу пока элемент [left] меньше [pivot]  
 if (left != right) // если границы не сомкнулись  
 {  
 numbers[right] = numbers[left]; // перемещаем элемент [left] на место [right]  
 right--; // сдвигаем правую границу вправо  
 }  
 }  
 numbers[left] = pivot; // ставим разрешающий элемент на место  
 pivot = left;  
 left = l\_hold;  
 right = r\_hold;  
 if (left < pivot) // Рекурсивно вызываем сортировку для левой и правой части массива  
 quickSort(numbers, left, pivot - 1);  
 if (right > pivot)  
 quickSort(numbers, pivot + 1, right);  
}  
  
int main() {  
 int n;  
 int a[n];  
 // Заполнение массива случайными числами  
 for (int i = 0; i < n; i++)  
 a[i] = rand() % 201 - 100;  
 // Вывод элементов массива до сортировки  
 for (int i = 0; i < n; i++)  
 printf("%4d ", a[i]);  
 printf("\n");  
 quickSort(a, 0, n - 1); // вызов функции сортировки  
 // Вывод элементов массива после сортировки  
 for (int i = 0; i < n; i++)  
 printf("%4d ", a[i]);  
 printf("\n");  
 getchar();  
 return 0;  
}

1. *Входные данные*

<https://github.com/Viktoria223/JAVA/tree/main/Tests2>

1. *График, время*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| входные данные | 1 | 2 | 3 | 4 | 5 | среднее значение |
| 100 | 0,002 | 0,001 | 0 | 0,001 | 0,002 | 0,0012 |
| 300 | 0,002 | 0 | 0 | 0,001 | 0,002 | 0,001 |
| 500 | 0,003 | 0,001 | 0,001 | 0 | 0,003 | 0,0016 |
| 700 | 0,002 | 0,001 | 0,001 | 0,001 | 0,002 | 0,0014 |
| 900 | 0,002 | 0,001 | 0,002 | 0,001 | 0,002 | 0,0016 |
| 1100 | 0,002 | 0,002 | 0,001 | 0,002 | 0,002 | 0,0018 |
| 1300 | 0,002 | 0,001 | 0,001 | 0,001 | 0,002 | 0,0014 |
| 1500 | 0,002 | 0,001 | 0,001 | 0,001 | 0,002 | 0,0014 |
| 1700 | 0,003 | 0,002 | 0,002 | 0,002 | 0,002 | 0,0022 |
| 1900 | 0,002 | 0,002 | 0,002 | 0,002 | 0,002 | 0,002 |
| 2100 | 0,002 | 0,002 | 0,002 | 0,002 | 0,002 | 0,002 |
| 2300 | 0,003 | 0,002 | 0,003 | 0,002 | 0,003 | 0,0026 |
| 2500 | 0,003 | 0,002 | 0,002 | 0,002 | 0,002 | 0,0022 |
| 2700 | 0,003 | 0,002 | 0,003 | 0,002 | 0,003 | 0,0026 |
| 2900 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 |
| 3100 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 |
| 3300 | 0,003 | 0,002 | 0,003 | 0,002 | 0,003 | 0,0026 |
| 3500 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 | 0,003 |
| 3700 | 0,003 | 0,003 | 0,004 | 0,003 | 0,003 | 0,0032 |
| 3900 | 0,003 | 0,003 | 0,004 | 0,003 | 0,003 | 0,0032 |
| 4100 | 0,003 | 0,003 | 0,004 | 0,003 | 0,003 | 0,0032 |
| 4300 | 0,003 | 0,003 | 0,005 | 0,003 | 0,003 | 0,0034 |
| 4500 | 0,003 | 0,003 | 0,004 | 0,003 | 0,003 | 0,0032 |
| 4700 | 0,003 | 0,004 | 0,004 | 0,004 | 0,003 | 0,0036 |
| 4900 | 0,004 | 0,003 | 0,005 | 0,003 | 0,004 | 0,0038 |
| 5100 | 0,004 | 0,004 | 0,005 | 0,004 | 0,004 | 0,0042 |
| 5300 | 0,004 | 0,004 | 0,005 | 0,004 | 0,004 | 0,0042 |
| 5500 | 0,004 | 0,004 | 0,005 | 0,004 | 0,004 | 0,0042 |
| 5700 | 0,003 | 0,004 | 0,005 | 0,004 | 0,003 | 0,0038 |
| 5900 | 0,004 | 0,004 | 0,005 | 0,004 | 0,004 | 0,0042 |
| 6100 | 0,004 | 0,004 | 0,005 | 0,004 | 0,004 | 0,0042 |
| 6300 | 0,004 | 0,005 | 0,006 | 0,005 | 0,004 | 0,0048 |
| 6500 | 0,004 | 0,004 | 0,006 | 0,004 | 0,004 | 0,0044 |
| 6700 | 0,004 | 0,004 | 0,006 | 0,004 | 0,004 | 0,0044 |
| 6900 | 0,004 | 0,005 | 0,005 | 0,005 | 0,004 | 0,0046 |
| 7100 | 0,005 | 0,005 | 0,005 | 0,005 | 0,005 | 0,005 |
| 7300 | 0,004 | 0,005 | 0,006 | 0,005 | 0,004 | 0,0048 |
| 7500 | 0,004 | 0,005 | 0,006 | 0,005 | 0,004 | 0,0048 |
| 7700 | 0,004 | 0,005 | 0,007 | 0,005 | 0,004 | 0,005 |
| 7900 | 0,005 | 0,005 | 0,006 | 0,005 | 0,005 | 0,0052 |
| 8100 | 0,005 | 0,006 | 0,006 | 0,006 | 0,005 | 0,0056 |
| 8300 | 0,005 | 0,006 | 0,006 | 0,006 | 0,005 | 0,0056 |
| 8500 | 0,005 | 0,005 | 0,006 | 0,005 | 0,005 | 0,0052 |
| 8700 | 0,005 | 0,006 | 0,006 | 0,006 | 0,005 | 0,0056 |
| 8900 | 0,005 | 0,006 | 0,007 | 0,006 | 0,005 | 0,0058 |
| 9100 | 0,004 | 0,006 | 0,007 | 0,006 | 0,004 | 0,0054 |
| 9300 | 0,005 | 0,005 | 0,008 | 0,005 | 0,005 | 0,0056 |
| 9500 | 0,006 | 0,006 | 0,007 | 0,006 | 0,006 | 0,0062 |
| 9700 | 0,006 | 0,007 | 0,007 | 0,006 | 0,007 | 0,0066 |
| 9900 | 0,006 | 0,007 | 0,008 | 0,006 | 0,008 | 0,007 |
| 10000 | 0,006 | 0,007 | 0,008 | 0,007 | 0,008 | 0,0072 |