СПб НИУ ИТМО

Кафедра ИПМ

Комбинаторные алгоритмы

Отчет по лабораторной работе № 1

"Комбинаторные алгоритмы (сортировка)"

Варианты сортировок:

* методом выбора
* методом простых вставок
* быстрая сортировка

Работу выполнил

Студент 1 курса КТУ

Группы № 1125

Журавлев Виталий

Санкт-Петербург

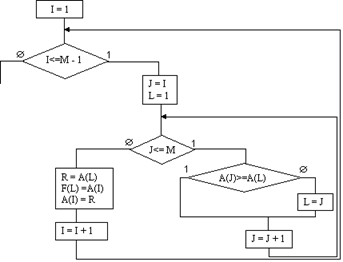
2013

***Описание алгоритмов:***

1. **Сортировка выбором**

При сортировке массива a[1], a[2], ..., a[m] методом простого выбора среди всех элементов находится элемент с наименьшим значением a[i], и a[1] и a[i] обмениваются значениями. Затем этот процесс повторяется для получаемых подмассивов a[2], a[3], ..., a[n], ... a[j], a[j+1], ..., a[m] до тех пор, пока мы не дойдем до подмассива a[m], содержащего к этому моменту наибольшее значение.

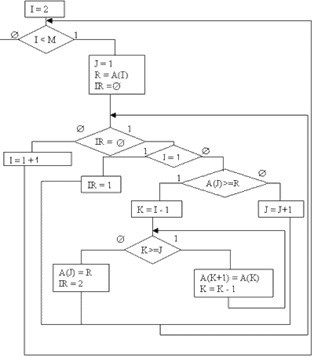
Для метода сортировки простым выбором требуемое число сравнений – n\*(n-1)/2. Порядок требуемого числа пересылок (включая те, которые требуются для выбора минимального элемента) в худшем случае составляет O(n2). Однако порядок среднего числа пересылок есть O(n\*ln n), что в ряде случаев делает этот метод предпочтительным.



1. **Метод простых вставок.**

Пусть в заданной последовательности а1, А2, ..., АN первые I-1 элементов уже отсортированы. Найдем в этой части последовательности место для I-го элемента. Для этого будем сравнивать его по порядку со всеми элементами, стоящими левее, до тех пор пока не окажется, что некоторый АK<=аI. Затем сдвинем часть последовательности АK+1, АK+2, ..., АI-1 на один элемент вправо и освободимтаким образом место АK+1для элемента АI , куда его и поставим. Считая, что первые три элементауже упорядочены, ищем место для четвертого по вышеизложенному правилу. Знак**<=**(а не <) показывает, когда нужно прекратить сравнения, т. е. движение влево по последовательности. При этом достигается устойчивость сортировки вставками.

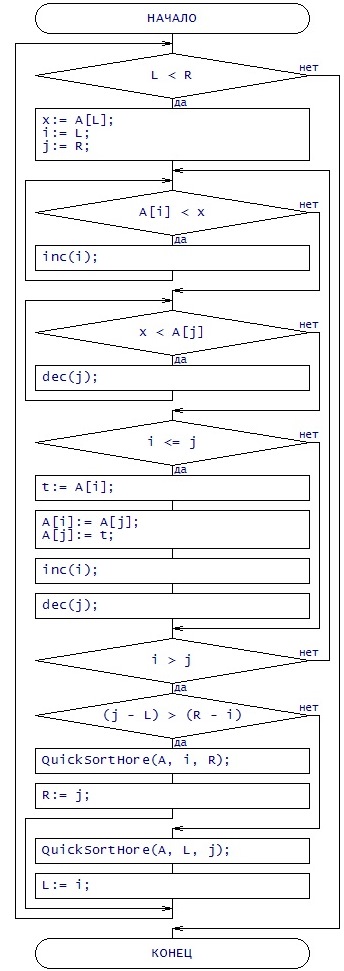
Время выполнения алгоритма зависит от входных данных: чем большее множество нужно отсортировать, тем большее время выполняется сортировка. На время выполнения влияет и исходная упорядоченность массива. Так, лучшим случаем является отсортированный массив, а худшим — массив, отсортированный в обратном порядке. Временная сложность алгоритма при худшем варианте входных данных — O(n²).



1. **Сортировка методом быстрой сортировки** **(Quicksort)**

Основная идея алгоритма состоит в том, что случайным образом выбирается некоторый элемент массива x, после чего массив просматривается слева, пока не встретится элемент a[i] такой, что a[i] > x, а затем массив просматривается справа, пока не встретится элемент a[j] такой, что a[j] < x. Эти два элемента меняются местами, и процесс просмотра, сравнения и обмена продолжается, пока мы не дойдем до элемента x. В результате массив окажется разбитым на две части - левую, в которой значения ключей будут меньше x, и правую со значениями ключей, большими x. Далее процесс рекурсивно продолжается для левой и правой частей массива до тех пор, пока каждая часть не будет содержать один элемент.

Алгоритм недаром называется быстрой сортировкой, поскольку для него оценкой числа сравнений и обменов является O(n\*log n). На самом деле, в большинстве утилит, выполняющих сортировку массивов, используется именно этот алгоритм.



***Текст программы:***

class Program

{

static private Stopwatch stopwatch;

static void Choice(double[] SortedArray)

{

for (int i = 0; i < SortedArray.Length - 1; i++)

{

int min = i;

for (int j = i + 1; j < SortedArray.Length; j++)

{

if (SortedArray[j] < SortedArray[min])

min = j;

}

if (min != i)

{

double t = SortedArray[i];

SortedArray[i] = SortedArray[min];

SortedArray[min] = t;

}

}

}

static void SimpleInserts(double[] SortedArray)

{

int a, b;

double Key;

for (a = 1; a < SortedArray.Length; ++a)

{

Key = SortedArray[a];

b = a - 1;

while ((Key < SortedArray[b]) && (b > 0))

{

SortedArray[b + 1] = SortedArray[b];

b = b - 1;

}

SortedArray[b + 1] = Key;

}

}

static void QuickSort(double[] SortedArray, long first, long last)

{

double p = SortedArray[(last - first) / 2 + first];

double Temp;

long i = first, j = last;

while (i <= j)

{

while (SortedArray[i] < p && i <= last) ++i;

while (SortedArray[j] > p && j >= first) --j;

if (i <= j)

{

Temp = SortedArray[i];

SortedArray[i] = SortedArray[j];

SortedArray[j] = Temp;

++i; --j;

}

}

if (j > first) QuickSort(SortedArray, first, j);

if (i < last) QuickSort(SortedArray, i, last);

}

static bool LoadAll()

{

Console.WriteLine("Choose number of menu and press [Enter]\n");

Console.WriteLine("\tSortings:\n");

Console.WriteLine("(1) The method of choice");

Console.WriteLine("(2) The method of simple inserts");

Console.WriteLine("(3) Quick sorting\n");

string ans = Console.ReadLine();

string[] filepaths = Directory.GetFiles(@"E:\Sort", "\*", SearchOption.TopDirectoryOnly);

foreach (string s in filepaths)

{

string[] StringArray = File.ReadAllText(s).Split(new[] { " " }, StringSplitOptions.RemoveEmptyEntries);

double[] SortedArray = new double[StringArray.Length];

switch (ans)

{

case "1": File.AppendAllText(@"E:/data.csv", System.IO.Path.GetFileName(s) + ";" + GetTiksCh(SortedArray) + "\n"); break;

case "2": File.AppendAllText(@"E:/data.csv", System.IO.Path.GetFileName(s) + ";" + GetTiksIns(SortedArray) + "\n"); break;

case "3": File.AppendAllText(@"E:/data.csv", System.IO.Path.GetFileName(s) + ";" + GetTiksQuick(SortedArray) + "\n"); break;

default: return false;

}

}

return true;

}

public static double GetTiksCh(double[] arr)

{

stopwatch.Restart();

Choice(arr);

stopwatch.Stop();

return stopwatch.Elapsed.TotalMilliseconds;

}

public static double GetTiksIns(double[] arr)

{

stopwatch.Restart();

SimpleInserts(arr);

stopwatch.Stop();

return stopwatch.Elapsed.TotalMilliseconds;

}

public static double GetTiksQuick(double[] arr)

{

stopwatch.Restart();

QuickSort(arr, 0, arr.Length - 1);

stopwatch.Stop();

return stopwatch.Elapsed.TotalMilliseconds;

}

static void Main(string[] args)

{

stopwatch = new Stopwatch();

bool LoadOK = false;

while (!LoadOK)

{

LoadOK = LoadAll();

if (!LoadOK) Console.WriteLine("Invalid Number! Choose again!");

}

if (LoadOK) Console.WriteLine("Array sucsessfully sorted!");

Console.Read();

}

}

***Результаты замеров***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Метод выбора** | | | | **Метод простых вставок** | | | | **QuickSort** | | | |
| Вар. | Элем. | % | Время, | Вар. | Элем. | % | Время, | Вар. | Элем. | % | Время, |
| разуп | мс | разуп | мс | разуп | мс |
| 0 | 64 | 0 | 0,0174 | 0 | 64 | 0 | 0,0016 | 0 | 64 | 0 | 0,0911 |
| 1 | 64 | 0 | 0,0165 | 1 | 64 | 0 | 0,0004 | 1 | 64 | 0 | 0,0267 |
| 2 | 64 | 0 | 0,0165 | 2 | 64 | 0 | 0,0008 | 2 | 64 | 0 | 0,0081 |
| 0 | 64 | 25 | 0,016 | 0 | 64 | 25 | 0,002 | 0 | 64 | 25 | 0,0105 |
| 1 | 64 | 25 | 0,016 | 1 | 64 | 25 | 0,002 | 1 | 64 | 25 | 0,0089 |
| 2 | 64 | 25 | 0,0165 | 2 | 64 | 25 | 0,0024 | 2 | 64 | 25 | 0,0101 |
| 0 | 64 | 50 | 0,016 | 0 | 64 | 50 | 0,0032 | 0 | 64 | 50 | 0,0121 |
| 1 | 64 | 50 | 0,0165 | 1 | 64 | 50 | 0,0032 | 1 | 64 | 50 | 0,0101 |
| 2 | 64 | 50 | 0,016 | 2 | 64 | 50 | 0,0028 | 2 | 64 | 50 | 0,0097 |
| 0 | 64 | 75 | 0,016 | 0 | 64 | 75 | 0,0036 | 0 | 64 | 75 | 0,0101 |
| 1 | 64 | 75 | 0,016 | 1 | 64 | 75 | 0,0032 | 1 | 64 | 75 | 0,0109 |
| 2 | 64 | 75 | 0,016 | 2 | 64 | 75 | 0,0032 | 2 | 64 | 75 | 0,0105 |
| 0 | 64 | 100 | 0,016 | 0 | 64 | 100 | 0,0036 | 0 | 64 | 100 | 0,0113 |
| 1 | 64 | 100 | 0,0165 | 1 | 64 | 100 | 0,004 | 1 | 64 | 100 | 0,0113 |
| 2 | 64 | 100 | 0,0165 | 2 | 64 | 100 | 0,004 | 2 | 64 | 100 | 0,0121 |
| 0 | 128 | 0 | 0,062 | 0 | 128 | 0 | 0,0008 | 0 | 128 | 0 | 0,0125 |
| 1 | 128 | 0 | 0,062 | 1 | 128 | 0 | 0,0008 | 1 | 128 | 0 | 0,0129 |
| 2 | 128 | 0 | 0,0615 | 2 | 128 | 0 | 0,0008 | 2 | 128 | 0 | 0,0145 |
| 0 | 128 | 25 | 0,062 | 0 | 128 | 25 | 0,006 | 0 | 128 | 25 | 0,0158 |
| 1 | 128 | 25 | 0,062 | 1 | 128 | 25 | 0,0068 | 1 | 128 | 25 | 0,0178 |
| 2 | 128 | 25 | 0,0615 | 2 | 128 | 25 | 0,0076 | 2 | 128 | 25 | 0,0186 |
| 0 | 128 | 50 | 0,062 | 0 | 128 | 50 | 0,0089 | 0 | 128 | 50 | 0,0198 |
| 1 | 128 | 50 | 0,0615 | 1 | 128 | 50 | 0,0117 | 1 | 128 | 50 | 0,0198 |
| 2 | 128 | 50 | 0,062 | 2 | 128 | 50 | 0,0113 | 2 | 128 | 50 | 0,0206 |
| 0 | 128 | 75 | 0,062 | 0 | 128 | 75 | 0,0117 | 0 | 128 | 75 | 0,0218 |
| 1 | 128 | 75 | 0,0615 | 1 | 128 | 75 | 0,0117 | 1 | 128 | 75 | 0,021 |
| 2 | 128 | 75 | 0,062 | 2 | 128 | 75 | 0,0149 | 2 | 128 | 75 | 0,0206 |
| 0 | 128 | 100 | 0,0615 | 0 | 128 | 100 | 0,0125 | 0 | 128 | 100 | 0,0214 |
| 1 | 128 | 100 | 0,062 | 1 | 128 | 100 | 0,0121 | 1 | 128 | 100 | 0,0226 |
| 2 | 128 | 100 | 0,062 | 2 | 128 | 100 | 0,0133 | 2 | 128 | 100 | 0,0214 |
| 0 | 256 | 0 | 0,2409 | 0 | 256 | 0 | 0,002 | 0 | 256 | 0 | 0,0243 |
| 1 | 256 | 0 | 0,2409 | 1 | 256 | 0 | 0,0016 | 1 | 256 | 0 | 0,0239 |
| 2 | 256 | 0 | 0,2414 | 2 | 256 | 0 | 0,0012 | 2 | 256 | 0 | 0,0243 |
| 0 | 256 | 25 | 0,2414 | 0 | 256 | 25 | 0,0283 | 0 | 256 | 25 | 0,034 |
| 1 | 256 | 25 | 0,2414 | 1 | 256 | 25 | 0,0235 | 1 | 256 | 25 | 0,036 |
| 2 | 256 | 25 | 0,2601 | 2 | 256 | 25 | 0,0271 | 2 | 256 | 25 | 0,036 |
| 0 | 256 | 50 | 0,2409 | 0 | 256 | 50 | 0,0457 | 0 | 256 | 50 | 0,0421 |
| 1 | 256 | 50 | 0,2414 | 1 | 256 | 50 | 0,0891 | 1 | 256 | 50 | 0,0413 |
| 2 | 256 | 50 | 0,2414 | 2 | 256 | 50 | 0,0368 | 2 | 256 | 50 | 0,0413 |
| 0 | 256 | 75 | 0,2414 | 0 | 256 | 75 | 0,0465 | 0 | 256 | 75 | 0,0441 |
| 1 | 256 | 75 | 0,2409 | 1 | 256 | 75 | 0,0433 | 1 | 256 | 75 | 0,0441 |
| 2 | 256 | 75 | 0,2414 | 2 | 256 | 75 | 0,0421 | 2 | 256 | 75 | 0,0429 |
| 0 | 256 | 100 | 0,2409 | 0 | 256 | 100 | 0,0465 | 0 | 256 | 100 | 0,0437 |
| 1 | 256 | 100 | 0,2414 | 1 | 256 | 100 | 0,0498 | 1 | 256 | 100 | 0,0425 |
| 2 | 256 | 100 | 0,2409 | 2 | 256 | 100 | 0,0518 | 2 | 256 | 100 | 0,0441 |
| 0 | 512 | 0 | 0,9535 | 0 | 512 | 0 | 0,0036 | 0 | 512 | 0 | 0,0494 |
| 1 | 512 | 0 | 0,9535 | 1 | 512 | 0 | 0,0024 | 1 | 512 | 0 | 0,0494 |
| 2 | 512 | 0 | 0,954 | 2 | 512 | 0 | 0,0028 | 2 | 512 | 0 | 0,0486 |
| 0 | 512 | 25 | 0,9549 | 0 | 512 | 25 | 0,128 | 0 | 512 | 25 | 0,0757 |
| 1 | 512 | 25 | 0,954 | 1 | 512 | 25 | 0,0903 | 1 | 512 | 25 | 0,0737 |
| 2 | 512 | 25 | 0,954 | 2 | 512 | 25 | 0,1021 | 2 | 512 | 25 | 0,0769 |
| 0 | 512 | 50 | 0,9535 | 0 | 512 | 50 | 0,2208 | 0 | 512 | 50 | 0,0887 |
| 1 | 512 | 50 | 0,9535 | 1 | 512 | 50 | 0,1556 | 1 | 512 | 50 | 0,085 |
| 2 | 512 | 50 | 0,9799 | 2 | 512 | 50 | 0,1584 | 2 | 512 | 50 | 0,0854 |
| 0 | 512 | 75 | 0,954 | 0 | 512 | 75 | 0,1657 | 0 | 512 | 75 | 0,0923 |
| 1 | 512 | 75 | 0,9535 | 1 | 512 | 75 | 0,1345 | 1 | 512 | 75 | 0,0899 |
| 2 | 512 | 75 | 0,954 | 2 | 512 | 75 | 0,132 | 2 | 512 | 75 | 0,0891 |
| 0 | 512 | 100 | 0,9535 | 0 | 512 | 100 | 0,141 | 0 | 512 | 100 | 0,0927 |
| 1 | 512 | 100 | 0,954 | 1 | 512 | 100 | 0,1426 | 1 | 512 | 100 | 0,0907 |
| 2 | 512 | 100 | 0,954 | 2 | 512 | 100 | 0,1479 | 2 | 512 | 100 | 0,0931 |
| 0 | 1024 | 0 | 3,7987 | 0 | 1024 | 0 | 0,004 | 0 | 1024 | 0 | 0,1004 |
| 1 | 1024 | 0 | 3,7965 | 1 | 1024 | 0 | 0,0036 | 1 | 1024 | 0 | 0,1008 |
| 2 | 1024 | 0 | 3,7974 | 2 | 1024 | 0 | 0,0036 | 2 | 1024 | 0 | 0,1004 |
| 0 | 1024 | 25 | 3,7965 | 0 | 1024 | 25 | 0,2929 | 0 | 1024 | 25 | 0,16 |
| 1 | 1024 | 25 | 3,796 | 1 | 1024 | 25 | 0,2641 | 1 | 1024 | 25 | 0,1523 |
| 2 | 1024 | 25 | 3,8518 | 2 | 1024 | 25 | 0,2966 | 2 | 1024 | 25 | 0,1624 |
| 0 | 1024 | 50 | 3,2726 | 0 | 1024 | 50 | 0,5089 | 0 | 1024 | 50 | 0,1823 |
| 1 | 1024 | 50 | 3,2686 | 1 | 1024 | 50 | 0,4526 | 1 | 1024 | 50 | 0,1835 |
| 2 | 1024 | 50 | 3,3136 | 2 | 1024 | 50 | 0,4558 | 2 | 1024 | 50 | 0,1815 |
| 0 | 1024 | 75 | 3,269 | 0 | 1024 | 75 | 0,5364 | 0 | 1024 | 75 | 0,194 |
| 1 | 1024 | 75 | 3,2726 | 1 | 1024 | 75 | 0,5348 | 1 | 1024 | 75 | 0,1924 |
| 2 | 1024 | 75 | 3,2699 | 2 | 1024 | 75 | 0,5462 | 2 | 1024 | 75 | 0,1924 |
| 0 | 1024 | 100 | 3,2351 | 0 | 1024 | 100 | 0,5624 | 0 | 1024 | 100 | 0,1977 |
| 1 | 1024 | 100 | 3,0566 | 1 | 1024 | 100 | 0,562 | 1 | 1024 | 100 | 0,2013 |
| 2 | 1024 | 100 | 3,0513 | 2 | 1024 | 100 | 0,6134 | 2 | 1024 | 100 | 0,1932 |
| 0 | 2048 | 0 | 12,3083 | 0 | 2048 | 0 | 0,0068 | 0 | 2048 | 0 | 0,2034 |
| 1 | 2048 | 0 | 12,5956 | 1 | 2048 | 0 | 0,0068 | 1 | 2048 | 0 | 0,2034 |
| 2 | 2048 | 0 | 12,1026 | 2 | 2048 | 0 | 0,0068 | 2 | 2048 | 0 | 0,2058 |
| 0 | 2048 | 25 | 12,3667 | 0 | 2048 | 25 | 1,1507 | 0 | 2048 | 25 | 0,3237 |
| 1 | 2048 | 25 | 12,107 | 1 | 2048 | 25 | 1,105 | 1 | 2048 | 25 | 0,3261 |
| 2 | 2048 | 25 | 12,2958 | 2 | 2048 | 25 | 1,173 | 2 | 2048 | 25 | 0,3261 |
| 0 | 2048 | 50 | 12,0869 | 0 | 2048 | 50 | 1,8781 | 0 | 2048 | 50 | 0,3845 |
| 1 | 2048 | 50 | 12,3078 | 1 | 2048 | 50 | 1,8072 | 1 | 2048 | 50 | 0,3857 |
| 2 | 2048 | 50 | 12,7255 | 2 | 2048 | 50 | 1,7136 | 2 | 2048 | 50 | 0,3821 |
| 0 | 2048 | 75 | 12,3471 | 0 | 2048 | 75 | 2,0767 | 0 | 2048 | 75 | 0,4007 |
| 1 | 2048 | 75 | 12,6135 | 1 | 2048 | 75 | 2,054 | 1 | 2048 | 75 | 0,3991 |
| 2 | 2048 | 75 | 12,124 | 2 | 2048 | 75 | 2,0965 | 2 | 2048 | 75 | 0,3967 |
| 0 | 2048 | 100 | 12,7942 | 0 | 2048 | 100 | 2,2736 | 0 | 2048 | 100 | 0,4254 |
| 1 | 2048 | 100 | 13,0231 | 1 | 2048 | 100 | 2,2841 | 1 | 2048 | 100 | 0,4116 |
| 2 | 2048 | 100 | 13,136 | 2 | 2048 | 100 | 2,2938 | 2 | 2048 | 100 | 0,4238 |
| 0 | 4096 | 0 | 58,4991 | 0 | 4096 | 0 | 0,0133 | 0 | 4096 | 0 | 0,4514 |
| 1 | 4096 | 0 | 50,0632 | 1 | 4096 | 0 | 0,0137 | 1 | 4096 | 0 | 0,4449 |
| 2 | 4096 | 0 | 50,2055 | 2 | 4096 | 0 | 0,0133 | 2 | 4096 | 0 | 0,4299 |
| 0 | 4096 | 25 | 50,9097 | 0 | 4096 | 25 | 4,3357 | 0 | 4096 | 25 | 0,7285 |
| 1 | 4096 | 25 | 50,8115 | 1 | 4096 | 25 | 4,8629 | 1 | 4096 | 25 | 0,7022 |
| 2 | 4096 | 25 | 50,2011 | 2 | 4096 | 25 | 4,6696 | 2 | 4096 | 25 | 0,7423 |
| 0 | 4096 | 50 | 54,3331 | 0 | 4096 | 50 | 7,0879 | 0 | 4096 | 50 | 1,3169 |
| 1 | 4096 | 50 | 81,412 | 1 | 4096 | 50 | 6,9667 | 1 | 4096 | 50 | 0,861 |
| 2 | 4096 | 50 | 49,9592 | 2 | 4096 | 50 | 6,9761 | 2 | 4096 | 50 | 0,812 |
| 0 | 4096 | 75 | 53,0052 | 0 | 4096 | 75 | 8,697 | 0 | 4096 | 75 | 0,8598 |
| 1 | 4096 | 75 | 59,5161 | 1 | 4096 | 75 | 8,2727 | 1 | 4096 | 75 | 0,8764 |
| 2 | 4096 | 75 | 56,7682 | 2 | 4096 | 75 | 8,2622 | 2 | 4096 | 75 | 0,8468 |
| 0 | 4096 | 100 | 54,77 | 0 | 4096 | 100 | 8,7221 | 0 | 4096 | 100 | 0,872 |
| 1 | 4096 | 100 | 89,0367 | 1 | 4096 | 100 | 8,9178 | 1 | 4096 | 100 | 0,8651 |
| 2 | 4096 | 100 | 50,8418 | 2 | 4096 | 100 | 8,9693 | 2 | 4096 | 100 | 0,8659 |
| 0 | 8192 | 0 | 223,1011 | 0 | 8192 | 0 | 0,0271 | 0 | 8192 | 0 | 0,8922 |
| 1 | 8192 | 0 | 276,3651 | 1 | 8192 | 0 | 0,0271 | 1 | 8192 | 0 | 0,8926 |
| 2 | 8192 | 0 | 836,4118 | 2 | 8192 | 0 | 0,0267 | 2 | 8192 | 0 | 0,893 |
| 0 | 8192 | 25 | 790,3628 | 0 | 8192 | 25 | 17,3454 | 0 | 8192 | 25 | 1,526 |
| 1 | 8192 | 25 | 719,2069 | 1 | 8192 | 25 | 17,7097 | 1 | 8192 | 25 | 1,4506 |
| 2 | 8192 | 25 | 222,2707 | 2 | 8192 | 25 | 19,0051 | 2 | 8192 | 25 | 1,4729 |
| 0 | 8192 | 50 | 217,5536 | 0 | 8192 | 50 | 28,1135 | 0 | 8192 | 50 | 1,6949 |
| 1 | 8192 | 50 | 200,9981 | 1 | 8192 | 50 | 28,6638 | 1 | 8192 | 50 | 1,7744 |
| 2 | 8192 | 50 | 224,969 | 2 | 8192 | 50 | 27,8728 | 2 | 8192 | 50 | 1,7018 |
| 0 | 8192 | 75 | 252,6476 | 0 | 8192 | 75 | 33,0141 | 0 | 8192 | 75 | 1,7764 |
| 1 | 8192 | 75 | 197,7375 | 1 | 8192 | 75 | 33,6077 | 1 | 8192 | 75 | 1,8546 |
| 2 | 8192 | 75 | 216,0444 | 2 | 8192 | 75 | 33,3934 | 2 | 8192 | 75 | 1,8149 |
| 0 | 8192 | 100 | 889,9775 | 0 | 8192 | 100 | 35,5459 | 0 | 8192 | 100 | 1,8477 |
| 1 | 8192 | 100 | 548,1301 | 1 | 8192 | 100 | 35,9405 | 1 | 8192 | 100 | 1,8412 |
| 2 | 8192 | 100 | 229,2515 | 2 | 8192 | 100 | 35,5965 | 2 | 8192 | 100 | 1,8664 |
| 0 | 16384 | 0 | 881,9382 | 0 | 16384 | 0 | 0,0534 | 0 | 16384 | 0 | 1,8939 |
| 1 | 16384 | 0 | 1858,157 | 1 | 16384 | 0 | 0,053 | 1 | 16384 | 0 | 1,8826 |
| 2 | 16384 | 0 | 1472,653 | 2 | 16384 | 0 | 0,0538 | 2 | 16384 | 0 | 1,9575 |
| 0 | 16384 | 25 | 791,8421 | 0 | 16384 | 25 | 74,3132 | 0 | 16384 | 25 | 3,1723 |
| 1 | 16384 | 25 | 776,8877 | 1 | 16384 | 25 | 73,8091 | 1 | 16384 | 25 | 3,0577 |
| 2 | 16384 | 25 | 906,1787 | 2 | 16384 | 25 | 73,9716 | 2 | 16384 | 25 | 3,1736 |
| 0 | 16384 | 50 | 969,9942 | 0 | 16384 | 50 | 111,871 | 0 | 16384 | 50 | 3,6922 |
| 1 | 16384 | 50 | 987,7434 | 1 | 16384 | 50 | 110,446 | 1 | 16384 | 50 | 3,7486 |
| 2 | 16384 | 50 | 970,7252 | 2 | 16384 | 50 | 109,536 | 2 | 16384 | 50 | 3,6323 |
| 0 | 16384 | 75 | 989,1682 | 0 | 16384 | 75 | 130,765 | 0 | 16384 | 75 | 3,7883 |
| 1 | 16384 | 75 | 976,8157 | 1 | 16384 | 75 | 133,181 | 1 | 16384 | 75 | 3,849 |
| 2 | 16384 | 75 | 996,6889 | 2 | 16384 | 75 | 131,706 | 2 | 16384 | 75 | 3,8608 |
| 0 | 16384 | 100 | 970,4503 | 0 | 16384 | 100 | 141,129 | 0 | 16384 | 100 | 3,9666 |
| 1 | 16384 | 100 | 1002,387 | 1 | 16384 | 100 | 145,79 | 1 | 16384 | 100 | 3,9625 |
| 2 | 16384 | 100 | 968,9469 | 2 | 16384 | 100 | 143,467 | 2 | 16384 | 100 | 3,8819 |
| 0 | 32768 | 0 | 3895,903 | 0 | 32768 | 0 | 0,1061 | 0 | 32768 | 0 | 4,0018 |
| 1 | 32768 | 0 | 3889,116 | 1 | 32768 | 0 | 0,1357 | 1 | 32768 | 0 | 3,9868 |
| 2 | 32768 | 0 | 3904,227 | 2 | 32768 | 0 | 0,115 | 2 | 32768 | 0 | 4,4435 |
| 0 | 32768 | 25 | 3888,058 | 0 | 32768 | 25 | 294,204 | 0 | 32768 | 25 | 6,6965 |
| 1 | 32768 | 25 | 3891,86 | 1 | 32768 | 25 | 298,352 | 1 | 32768 | 25 | 6,784 |
| 2 | 32768 | 25 | 3919,927 | 2 | 32768 | 25 | 297,152 | 2 | 32768 | 25 | 6,7358 |
| 0 | 32768 | 50 | 3888,813 | 0 | 32768 | 50 | 441,577 | 0 | 32768 | 50 | 7,8302 |
| 1 | 32768 | 50 | 3895,617 | 1 | 32768 | 50 | 442,845 | 1 | 32768 | 50 | 7,8298 |
| 2 | 32768 | 50 | 3912,042 | 2 | 32768 | 50 | 441,216 | 2 | 32768 | 50 | 7,6637 |
| 0 | 32768 | 75 | 3934,468 | 0 | 32768 | 75 | 539,106 | 0 | 32768 | 75 | 8,2018 |
| 1 | 32768 | 75 | 3898,396 | 1 | 32768 | 75 | 529,635 | 1 | 32768 | 75 | 8,1751 |
| 2 | 32768 | 75 | 3903,854 | 2 | 32768 | 75 | 528,277 | 2 | 32768 | 75 | 8,3023 |
| 0 | 32768 | 100 | 3906,226 | 0 | 32768 | 100 | 567,604 | 0 | 32768 | 100 | 8,2748 |
| 1 | 32768 | 100 | 3888,761 | 1 | 32768 | 100 | 568,463 | 1 | 32768 | 100 | 8,3461 |
| 2 | 32768 | 100 | 3896,016 | 2 | 32768 | 100 | 565,848 | 2 | 32768 | 100 | 8,3392 |
| 0 | 65536 | 0 | 15542,35 | 0 | 65536 | 0 | 0,2248 | 0 | 65536 | 0 | 8,3218 |
| 1 | 65536 | 0 | 15607,52 | 1 | 65536 | 0 | 0,2248 | 1 | 65536 | 0 | 8,2663 |
| 2 | 65536 | 0 | 15586,2 | 2 | 65536 | 0 | 0,2261 | 2 | 65536 | 0 | 8,3692 |
| 0 | 65536 | 25 | 15587,64 | 0 | 65536 | 25 | 1160,9 | 0 | 65536 | 25 | 13,799 |
| 1 | 65536 | 25 | 15602,08 | 1 | 65536 | 25 | 1161,04 | 1 | 65536 | 25 | 13,496 |
| 2 | 65536 | 25 | 15590,17 | 2 | 65536 | 25 | 1162,71 | 2 | 65536 | 25 | 13,955 |
| 0 | 65536 | 50 | 15607,88 | 0 | 65536 | 50 | 1775,88 | 0 | 65536 | 50 | 16,317 |
| 1 | 65536 | 50 | 15620,52 | 1 | 65536 | 50 | 1793,68 | 1 | 65536 | 50 | 16,189 |
| 2 | 65536 | 50 | 15556,55 | 2 | 65536 | 50 | 1792,3 | 2 | 65536 | 50 | 15,846 |
| 0 | 65536 | 75 | 15581,08 | 0 | 65536 | 75 | 2126,29 | 0 | 65536 | 75 | 16,707 |
| 1 | 65536 | 75 | 15906,76 | 1 | 65536 | 75 | 2118,68 | 1 | 65536 | 75 | 16,792 |
| 2 | 65536 | 75 | 15587,59 | 2 | 65536 | 75 | 2102,82 | 2 | 65536 | 75 | 17,223 |
| 0 | 65536 | 100 | 15618,29 | 0 | 65536 | 100 | 2272,04 | 0 | 65536 | 100 | 16,92 |
| 1 | 65536 | 100 | 15541,09 | 1 | 65536 | 100 | 2272,72 | 1 | 65536 | 100 | 17,285 |
| 2 | 65536 | 100 | 15656,41 | 2 | 65536 | 100 | 2281,04 | 2 | 65536 | 100 | 17,41 |

**Сводные таблицы**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сортировка методом выбора | | | | | | |
|  | **0** | **25** | **50** | **75** | **100** | **Общий итог** |
| 64 | 0,0168 | 0,0162 | 0,0162 | 0,0160 | 0,0163 | 0,0163 |
| 128 | 0,0618 | 0,0618 | 0,0618 | 0,0618 | 0,0618 | 0,0618 |
| 256 | 0,2411 | 0,2476 | 0,2412 | 0,2412 | 0,2411 | 0,2424 |
| 512 | 0,9537 | 0,9543 | 0,9623 | 0,9538 | 0,9538 | 0,9556 |
| 1024 | 3,7975 | 3,8148 | 3,2849 | 3,2705 | 3,1143 | 3,4564 |
| 2048 | 12,3355 | 12,2565 | 12,3734 | 12,3615 | 12,9844 | 12,4623 |
| 4096 | 52,9226 | 50,6408 | 61,9014 | 56,4298 | 64,8828 | 57,3555 |
| 8192 | 445,2927 | 577,2801 | 214,5069 | 222,1432 | 555,7864 | 403,0018 |
| 16384 | 1404,2495 | 824,9695 | 976,1543 | 987,5576 | 980,5947 | 1034,7051 |
| 32768 | 3896,4154 | 3899,9485 | 3898,8243 | 3912,2392 | 3897,0010 | 3900,8857 |
| 65536 | 15578,6923 | 15593,2949 | 15594,9841 | 15691,8095 | 15605,2631 | 15612,8088 |
| **Общий итог** | **1944,9981** | **1905,7714** | **1887,5737** | **1898,8258** | **1920,0818** | **1911,4502** |
| Сортировка методом простых вставок | | | | | | |
|  | **0** | **25** | **50** | **75** | **100** | **Общий итог** |
| 64 | 0,0009 | 0,0021 | 0,0031 | 0,0033 | 0,0039 | 0,0027 |
| 128 | 0,0008 | 0,0068 | 0,0106 | 0,0128 | 0,0126 | 0,0087 |
| 256 | 0,0016 | 0,0263 | 0,0572 | 0,0440 | 0,0494 | 0,0357 |
| 512 | 0,0029 | 0,1068 | 0,1783 | 0,1441 | 0,1438 | 0,1152 |
| 1024 | 0,0037 | 0,2845 | 0,4724 | 0,5391 | 0,5793 | 0,3758 |
| 2048 | 0,0068 | 1,1429 | 1,7996 | 2,0757 | 2,2838 | 1,4618 |
| 4096 | 0,0134 | 4,6227 | 7,0102 | 8,4106 | 8,8697 | 5,7854 |
| 8192 | 0,0270 | 18,0201 | 28,2167 | 33,3384 | 35,6943 | 23,0593 |
| 16384 | 0,0534 | 74,0313 | 110,6177 | 131,8840 | 143,4620 | 92,0097 |
| 32768 | 0,1189 | 296,5693 | 441,8793 | 532,3393 | 567,3050 | 367,6424 |
| 65536 | 0,2252 | 1161,5500 | 1787,2867 | 2115,9300 | 2275,2667 | 1468,0517 |
| **Общий итог** | **0,0413** | **141,4875** | **216,1393** | **256,7929** | **275,7882** | **178,0498** |
| Быстрая сортировка | | | | | | |
|  | **0** | **25** | **50** | **75** | **100** | **Общий итог** |
| 64 | 0,0420 | 0,0098 | 0,0106 | 0,0105 | 0,0116 | 0,0169 |
| 128 | 0,0133 | 0,0174 | 0,0201 | 0,0211 | 0,0218 | 0,0187 |
| 256 | 0,0242 | 0,0353 | 0,0416 | 0,0437 | 0,0434 | 0,0376 |
| 512 | 0,0491 | 0,0754 | 0,0864 | 0,0904 | 0,0922 | 0,0787 |
| 1024 | 0,1005 | 0,1582 | 0,1824 | 0,1929 | 0,1974 | 0,1663 |
| 2048 | 0,2042 | 0,3253 | 0,3841 | 0,3988 | 0,4203 | 0,3465 |
| 4096 | 0,4421 | 0,7243 | 0,9966 | 0,8610 | 0,8677 | 0,7783 |
| 8192 | 0,8926 | 1,4832 | 1,7237 | 1,8153 | 1,8518 | 1,5533 |
| 16384 | 1,9113 | 3,1345 | 3,6910 | 3,8327 | 3,9370 | 3,3013 |
| 32768 | 4,1440 | 6,7388 | 7,7746 | 8,2264 | 8,3200 | 7,0408 |
| 65536 | 8,3191 | 13,7500 | 16,1173 | 16,9073 | 17,2050 | 14,4598 |
| **Общий итог** | **1,4675** | **2,4048** | **2,8208** | **2,9455** | **2,9971** | **2,5271** |

***Графики***

График 1. а) Сортировка методом выбора

График 1. б) Сортировка методом выбора, аппроксимация

График 2. а) Сортировка методом простых вставок

График 2. б) Сортировка методом простых вставок, аппроксимация

График 3. а) Быстрая сортировка

График 3. б) Быстрая сортировка, аппроксимация

График 4. а) Сравнение сортировок при упорядоченности 0%

График 4. б) Сравнение сортировок при упорядоченности 50%

График 4. а) Сравнение сортировок при упорядоченности 100%

***Вывод***

В процессе выполнения данной лабораторной работы были выявлены следующие особенности исследуемых алгоритмов сортировки:

1. **Метод выбора**

Данный способ прост в исполнении и понимании, но не предпочтителен в случае большого числа элементов. Алгоритм не использует дополнительной памяти: все операции происходят "на месте". Является неустойчивой сортировкой, если входная последовательность почти упорядочена, то сравнений все равно будет столько же. Существует также двунаправленный вариант сортировки методом выбора, в котором на каждом проходе отыскиваются и устанавливаются на свои места и минимальное, и максимальное значения.

1. **Метод простых вставок**

Достаточно легкий для программирования и небольшой алгоритм, но как показал опыт исполнения плохо подходящий для реальной работы с большими массивами, т.к. совершенно не отличается быстродействием. Отлично работает при выявлении уже отсортированных массивов любой длинны и, как оказалось, является самым быстродейственным из трех алгоритмов для массивов включающим до 128 элементов, но с продолжением увеличения количества элементов начинает серьезно проигрывать. Для данного метода характерна пропорциональность неотсортированности исходного массива и времени его сортировки.

1. **Быстрая сортировка**

При выборе опорного элемента из данного диапазона случайным образом худший случай становится очень маловероятным и ожидаемое время выполнения алгоритма сортировки — O(n log n).Данный метод отлично подходит для реализации сортировки массивов среднего и большого размера. При степени неотсортированности от 50 до 100 наблюдается, нечувствительность к количествам инверсий в массиве.

**Таким образом:**

Наиболее универсальным методом, является метод быстрой сортировки («QuickSort»), он показывает стабильно высокие результаты на любых размерах массивов. На втором месте находится метод простых вставок. Его использование может быть обосновано большее простым алгоритмом с точки зрения программиста. На последнем же месте по скорости находится метод выбором.

При использовании небольших массивов данных нет большой разницы по скорости между методами сортировки, поэтому целесообразнее применять метод простых вставок.

При использовании средних и больших массивов, к тому же с большим процентом инверсии лучше всего использовать метод быстрой сортировки.

При необходимости экономить память лучше всего использовать метод выбора.