Published on 27.07.2017
http://www.ollejnik.com

ARRAY PACKAGE SORTING ALGORITHMS — vladsort series or can we do it faster than qsort?

Yes, we can. This is the answer to the question posed in the title of this article.
And perhaps one of the array sorting algorithms proposed here (vladsort_3) can be the LARGEST and the FASTEST of them all.

SOURCE CODES OF TWO ARRAY SORTING ALGORITHMS are published IN THIS ARTICLE—vladsort_1 and vladsort_2.
THE THIRD ALGORITHM—vladsort_3—IS PUBLISHED ON THE OTHER PAGE (http://ollejnik.com/files//array_package_sorting_algorithms-2.pdf).

All three proposed algorithms are a hybrid of counting sort and selection sort.
One of the sorting algorithms proposed here (vladsort_3)—in many cases, on random number arrays—is 5—30% faster than quick sorting (qsort).
But sorting time of vladsort_3 depends very much on the value (number) of variable "t". At some values of "t" —sorting time can be slower than qsort and much so.

BUT AT CERTAIN VALUES OF NUMBER "t" - vladsort_3—SORTS FASTER THAN qsort.

"t" is the value of a "piece", a package, a segment of numbers in an array that are sorted in a certain range. Its value can be 1 or more, an integer.
For example, in the algorithm vladsort_3, with t = 1, 1,024 numbers will be sorted at once for one pass on the array. For t = 2, 2,048 numbers. For t = 10, 10,240; for t = 100—10,2400(!) numbers.

What defines "t"?
The value depends on several parameters:
1. Array size
2. The value between the minimum and maximum number
3. The "density" of numbers in the array.
4. Some other parameters.
So, on different arrays there is a certain optimal value of the number "t", at which vladsort_3 runs faster than qsort. And this optimal number, so to speak, the "golden mean", I got experimentally.
How to get the value of "t" by computation, knowing the above parameters of the array—I do not know yet.

But first things first.

****

ATTENTION! Algorithms are published "as is". Therefore, the author does not bear responsibility for their possibly incorrect operation and the consequences associated with it.
Algorithms are NOT written by a PROFESSIONAL, they may contain extra variables and unnecessary calculations, as well as, possibly, unrecognized bugs.

All the algorithms and fragments of the algorithms presented in this article are written in C ++.

So far, the level of my knowledge does not allow me to write a program more exquisitely, therefore, I apologize for the possibly clumsy code.

BACKGROUND OF THE CREATION OF THE ALGORITHM.

One day, about six months ago, I noticed one detail while comparing different sorting algorithms. Many algorithms turned out to sort small and very small arrays, with almost the same speed. But the medium-sized, large and very large arrays are sorted in different ways and with a big difference in the sorting time. Sorting algorithms have both linear and quadratic sorting time.
For example, I measured the sorting speed of two algorithms —qsort (quick sorting) and bubble sorting. **ALL measurements were carried out on the same powerful computer.**
And the difference in sorting time on an array of random numbers turned out to be huge.

(Numbers are obtained with the rand () function, unless otherwise specified).

A question may arise, HOW WAS THE TIME OF SORTING MEASURED? Very simple: using a for loop or a label, do a loop of 1000 (or 100, 10,000 or other value) iterations. The algorithm body, together with the rand () function, is put into a loop. Time is measured using an ordinary second hand. Then we divide time by the number of iterations.
This way, we get the AVERAGE TIME OF 1,000 SORTINGS, and this time is almost always the same.
(There is still time for placing numbers in the array, but under the same conditions, the sorting time measured by this method depends only on the speed of the algorithm).

But I think almost all algorithms sort an array OF TWO numbers with almost the same speed.
And then I thought, "And what if we divide a large array into small partially sorted "pieces" and then quickly sort them?" This idea is not new, there are many such algorithms.

BUT, THE POINT IS, FIRST TO DO ROUGH SORTING OF THE ARRAY, AND ONLY THEN—PRECISION SORTING.
To see what became of it, read on.

SELECTION SORT IT THE MOTHER OF ALL GIVEN ALGORITHMS

Here are TWO slower algorithms and ONE (located on the other page), the latest and the quickest, faster than qsort (only for a certain value of "t", which for each array may be different). And I called them "vladsort" (there is a "timsort" after all).

The "parent" of all three algorithms is widely known selection sort.

Here is the snippet of the selection sort code (one of the versions):

```
for (mas_vel = 100000; mas_vel > 0; mas_vel--) {
                macs = massiv[0];
                for (c = 1; c < mas_vel; c++) if (macs < massiv[c]) macs = massiv[c];
                for (c = 0; c<mas_vel; c++) if (macs == massiv[c]) b = c;
                massiv[b] = massiv[mas_vel - 1];
                massiv[mas_vel - 1] = macs;
        }
```

As can be seen from the snippet, we first find the largest number. Then we find the number of the cell in which the largest number is found. And then we move the largest number to the end of the array, and the number that was in the last cell of the array is moved to the cell where the largest number was. And this sequence of operations is repeated until the array is completely sorted.

Selection sort is about 4 times faster than bubble sort.
Here is the sorting time by bubble, selection and quick sort
(Obtained experimentally):
array 100,000 .... 15 ...... 3.5 ...... 0.020 ....sec
array 200,000 ....60 .... 15 .... 0.040 ....sec

As is known, the sorting time is linear in qsort, and in bubble and in the selection the sort, time is quadratic.
What does this mean in practice? If an array doubles in size, the sorting time doubles for linear, and quadruples for quadratic. Also, qsort is considered the quickest sorting up to date, from all existing ones. Bubble sort is one of the slowest (some are even slower).

The idea behind the very first sorting algorithm I wrote (vladsort_1) is as follows:
Sorting consists of TWO STEPS.

******* SORT STAGE 1 *****.**
For every two passes of the algorithm, we immediately find a certain number (let's call it a package the size of which depends on the value of "t") of the smallest and largest numbers and move them to the beginning and the end of the algorithm. The next pass of the algorithm starts from the end of the first "package" and to the beginning of the second "package". Here, the pre-sorting ends—in the middle of the array. And the array is obtained from the "pieces" is a set of numbers already sorted in a certain range.

******* SORT STAGE 2 *****.**
By selection sort (the fragment is shown above, is "configured" specifically for this task), we sort the "pieces" in the array already sorted in a certain range.

Here is the algorithm for sorting arrays—vladsort_1 running on the same principle:

```
// Array Sorting Algorithm vladsort_1.
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
	setlocale(LC_ALL, "Rus");
	int masiv[1000];
	int a, aa, b, c, d, h, s, v, vv, x, y, z, i, xx, t;

	xx = z = 1000;
	t = 10;

	// Put a set of random numbers in an array.

	for (c = 0; c < z; c++) masiv[c] = rand();

	// Find the minimum number.

	a = masiv[0];
	for (b = 1; b < z; b++) if (a > masiv[b]) a = masiv[b];

	// Find the maximum number.

	c = masiv[0];
	for (b = 1; b < z; b++) if (c < masiv[b]) c = masiv[b];

	// Calculate the number of necessary iterations.

	h = c - a;
	h = h + 2;
	for (; ;) {
		d = h % t;
		if (d == 0) {
			d = h % 2;
			if (d == 0) break;
		}
		h++;
	}
	b = h / 10;
	b = b / 2;
	h = h + a;

	// Start sorting the array (stage 1).

	aa = a;
	c = h;
	s = z;
	z--;
	y = 0;
	for (d = 0; d < b; d++) {
		a = a + t;
		h = h - t;
		v = y;
		for (; v <= z; v++) {
			if (masiv[v] < a) {
				for (; masiv[y] < a;) {
					if (y == v) {
						y++;
						goto num;
					}
					y++;
				}
				x = masiv[y];
				masiv[y] = masiv[v];
```

```
                masiv[v] = x;
                y++;
                continue;
            num:
                continue;
            }
            if (masiv[v] >= h) {
                for (; z > v; z--) {
                    if ((z - 1) == v) break;
                    if (masiv[z] < h) break;
                }
                x = masiv[v];
                masiv[v] = masiv[z];
                masiv[z] = x;
                z--;
                if (masiv[v] < a) v--;
            }
        }
    }

    // Start sorting the array (stage 2).

    x = s - 1;
    b = b * 2;
    z = 0;
    for (d = 0; d <= b; d++) {
        v = vv = z;
        aa = aa + t;
        for (; v < s; v++) {
            if (masiv[v] < aa) {
                if (v == x) goto mit;
                if (masiv[v + 1] >= aa) {
                mit:
                    if (vv == v) {
                        z = ++v;
                        break;
                    }
                    z = v = ++v;
                    for (; v > vv; v--) {
                        h = masiv[vv];
                        for (c = (vv + 1); c < v; c++) if (h < masiv[c]) h = masiv[c];
                        for (c = vv; c < v; c++) if (h == masiv[c]) y = c;
                        if (masiv[y] == masiv[v - 1]) continue;
                        masiv[y] = masiv[v - 1];
                        masiv[v - 1] = h;
                    }
                    break;
                }
                continue;
            }
            if (masiv[v] >= aa) break;
        }
    }
    return 0;
}
```

This algorithm for t = 10 is several times slower than quick sorting.

And what if "pieces" are found and moved to the beginning and to the end of the array, not one but 10 or 20, I thought.
Then I wrote the following algorithm, which in 2 passes finds and moves at once 20 "packages" of numbers. For t = 10, the algorithm pre-sort 400 numbers at once over the two passes. 200 from the "head" of the array and 200 from the "tail" of the array.

And such an algorithm was written, it is an algorithm for sorting arrays — vladsort_2.

This algorithm has about 1380 lines.
When testing this algorithm (t = 15), it turned out that on an array of 50 000 numbers, it is ahead of the quick sorting by mere 2%. And this algorithm proved to be much quicker than the first algorithm.
By sorting speed, on some arrays of random numbers, it came close to quick sorting, and on an array of 50,000 numbers was even slightly faster.

vladsort_2 array sorting algorithm source code is given at the end of this article.

And then...

And then vladsort_3 array sorting algorithm, which sorts at the first stage, 512 “pieces” at once—from the beginning and from the end of the array. This for t = 1. And this algorithm is faster than quick sorting (for certain values of “t”) by up to 30% on arrays of random numbers.
And at t = 10, it sorts immediately 10,240 (!) numbers; at t = 20, 20,480 numbers; and at t = 100—do the math yourself.
The algorithm can sort from 1,024 to hundreds of thousands and millions of numbers (and more), in just 2 passes.

The algorithms for sorting arrays vladsort_2 and vladsort_3 are almost the same in structure and in action by the executable code. I even used vladsort_2 as a template, when writing the algorithm code for vladsort_3.

Yes, array sorting algorithm vladsort_3 is huge!!

************************* vladsort_3 is:
about 35,000(!) lines of source code (maybe less, I wrote as quickly as I could).
more than 4,096 local variables (512 times eight).
the code compilation takes over 9 minutes.
the "weight" of the source is over 900 KB (it can be slimmed down though).
all 35,000 lines fit into one function.
the compiler hangs up on the Creating code line if an array is moved to the global area and the variables are local.
(still I found no solution to this issue, we must look for options, maybe try other compilers.
I used the Visual Studio Community 2015 built-in compiler).

****

So, the compiler behaves strangely. It somehow doesn't throw errors!! For example, if in one place of code instead of variable h142, it will be just 142, the code will be created, as it was h142! But there will be no sorted numbers in the already sorted array. Thus, the variable "142" will be "perceived" by the compiler as h142!!
Probably, the stack of the parser is overflowing and it simply goes into stupor and stops analysing...
I made three mistakes when writing the code (one of them is mentioned above), for which I scoured the code for a few days until I found and fixed them.

Well, now the COMPARISON of qsort and vladsort_3 sorting times
(obtained experimentally):

for a number-only array, sorted in the reverse order:
(t = 20, the number 20 is optimal for such arrays)
.... vladsort_3 ....
200,000 ....0.01 sec .... (qsort - 0.014 sec.) .40% quicker than qsort
10,000 ....0.000375 sec.
5,000 ....0.000175 sec.
1,000 ....0.0000375 sec.
~~~~~~

for t = 4.
 array ......qsort .... vladsort_3 ....
 200, 000 ... 0.040 sec ....0.0365 sec, ... 10% quicker than qsort.
 100,000 ... 0.020 sec......... 0.0175 sec, ...14% quicker than qsort.
.. 50,000 ... 0.0097 sec ....0.00888 sec, ... 9% quicker than qsort.
 . 40,000 ... 0.00783 sec ....0.0072 sec, ... 9% quicker than qsort.
 . 30,000 ... 0.00575 sec ....0.00543 sec, ... 6% quicker than qsort.
 . 20,000 ... 0.00375 sec ....0.0037 sec, ... 1% quicker than qsort.
 . 10,000 ... 0.00174 sec ....0.00195 sec, ... 12% SLOWER than qsort.
 ... 5,000 ... 0.00086 sec ....0.001 sec, ... 16% SLOWER than qsort.
 ... 1,000 ... 0.000147 sec ...0.000235 sec ... 60% SLOWER than qsort.
 ...... 100 ... 0.0000114 sec. 0.0000475 sec. 317% SLOWER than qsort.
~~~~~~

As you can see from the measurements, vladsort_3 on arrays of 20,000 or more elements is equal or ahead of the quick sorting in terms of array ordering time.
On arrays of less than 20,000 elements, vladsort_3 is slower.
But this is for t = 4.
If we take a different value for "t", the sorting speed on smaller arrays will exceed that of qsort.

For example:
t = 80 ..... array 10,000 - 0.00145 sec, 20% quicker than qsort
t = 80 ..... array 20,000 - 0.0033 sec, 14% quicker than qsort
t = 80 ..... array 30,000 - 0.0055 sec, 4.5% quicker than qsort

For example:
t = 4, ...array 1,000 - 0.000235 sec. ....60% slower than qsort.
t = 40, ...array 1,000 - 0.000175 sec ....19% slower than qsort.
t = 300, ..array 1,000 - 0.00012 sec. ....23% QUICKER than qsort.

AREAS OF APPLICATION OF ARRAY SORTING ALGORITHM vladsort_3.

The proposed algorithm potentially can replace quick sorting. On arrays of a few hundred numbers to very large ones.
After all, on all the arrays with which I experimented, vladsort_3 is quicker than qsort (but maybe slower on smaller arrays having only one really big number).
But this is in principle ...
Due to the large size and the strong dependence of the sorting speed on the given value of the variable "t", and also not a big gain in time, compared to qsort — in practice, I do not think that it can replace qsort.
But in certain niche cases, it is possible.
For example, if you want to sort input arrays with similar sets of numbers, when the optimal value of "t" can be specified more or less accurately and in advance.

There is another option—to conduct a PRELIMINARY ANALYSIS OF THE ARRAY before sorting.
BUT...
1. The analysis of the array will take some time. The more detailed analysis takes more time.
2. Which parameters should the analysis of the array take into account?
Additional time will pass for all this, which possibly will offset this petty gain in sorting time.
(I'm thinking over this one, and I do not know what to do about it just yet).

****
****

And now, here is vladsort_2 source code
(source code):

```
// vladsort_2 array sorting algorithm.
/*
This algorithm sorts arrays of numbers in two stages.
At the first stage, a "rough" sort is performed.
In this algorithm, 20 packages of numbers in a certain range are sorted at once, from the beginning and from the end of an array.
The length of the "piece" depends on the given number of variable □"t".

In the second stage, we sort the "pieces" of numbers by selection sorting, which is best suited for this task.
*/

#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
        setlocale(LC_ALL, "Rus");
        int masiv[50000];
        int a, b, c, d, h, s, v, vv, x, y, z, bb, an, t, i;

        int a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20;
        int h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, h13, h14, h15, h16, h17, h18, h19, h20;
        int aa, a2a, a3a, a4a, a5a, a6a, a7a, a8a, a9a, a10a, a11a, a12a, a13a, a14a, a15a, a16a, a17a, a18a, a19a, a20a;
        int hh, h2h, h3h, h4h, h5h, h6h, h7h, h8h, h9h, h10h, h11h, h12h, h13h, h14h, h15h, h16h, h17h, h18h, h19h, h20h;
        int ah, a2h, a3h, a4h, a5h, a6h, a7h, a8h, a9h, a10h, a11h, a12h, a13h, a14h, a15h, a16h, a17h, a18h, a19h, a20h;
        int ha, h2a, h3a, h4a, h5a, h6a, h7a, h8a, h9a, h10a, h11a, h12a, h13a, h14a, h15a, h16a, h17a, h18a, h19a, h20a;
        int as, as2, as3, as4, as5, as6, as7, as8, as9, as10, as11, as12, as13, as14, as15, as16, as17, as18, as19, as20;
        int az, az2, az3, az4, az5, az6, az7, az8, az9, az10, az11, az12, az13, az14, az15, az16, az17, az18, az19, az20;

        z = 50000;
        t = 15;
```

Put a set of random numbers in an array.

```
for (c = 0; c < z; c++) masiv[c] = rand();

// Find the minimum number.

a = masiv[0];
for (b = 1; b < z; b++) if (a > masiv[b]) a = masiv[b];

// Find the maximum number.

c = masiv[0];
for (b = 1; b < z; b++) if (c < masiv[b]) c = masiv[b];

// Calculate the number of necessary iterations.

an = a;
b = c - a;
b = b + 2;
for (; ;) {
        d = b % t;
        if (d == 0) {
                d = b % 2;
                if (d == 0) break;
        }
        b++;
}
h = b;
b = b / t;
bb = b;
b = b / 2;
h = h + a;

//  Вычисляем сколько надо ДВАДЦАТОК.

for (; ;) {
        d = b % 20;
        if (d == 0) break;
        b++;
}
d = b / 20;
s = z;

//  Start sorting the array (stage 1).

//   a20 - это < 10 (или <38).
//   a - это < 200 (или <228).

a20 = a;
y = --z;
v = b = 0;
for (c = 0; c < d; c++) {
        a19 = a20 + t;
        a18 = a19 + t;
        a17 = a18 + t;
        a16 = a17 + t;
```

```
		a15 = a16 + t;
		a14 = a15 + t;
		a13 = a14 + t;
		a12 = a13 + t;
		a11 = a12 + t;
		a10 = a11 + t;
		a9 = a10 + t;
		a8 = a9 + t;
		a7 = a8 + t;
		a6 = a7 + t;
		a5 = a6 + t;
		a4 = a5 + t;
		a3 = a4 + t;
		a2 = a3 + t;
		a = a2 + t;
		x = a + t;

		h = h - t;
		h2 = h - t;
		h3 = h2 - t;
		h4 = h3 - t;
		h5 = h4 - t;
		h6 = h5 - t;
		h7 = h6 - t;
		h8 = h7 - t;
		h9 = h8 - t;
		h10 = h9 - t;
		h11 = h10 - t;
		h12 = h11 - t;
		h13 = h12 - t;
		h14 = h13 - t;
		h15 = h14 - t;
		h16 = h15 - t;
		h17 = h16 - t;
		h18 = h17 - t;
		h19 = h18 - t;
		h20 = h19 - t;

		aa = a2a = a3a = a4a = a5a = a6a = a7a = a8a = a9a = a10a = a11a = a12a = a13a = a14a = a15a = a16a = a17a = 
a18a = a19a = a20a = 0;
		hh = h2h = h3h = h4h = h5h = h6h = h7h = h8h = h9h = h10h = h11h = h12h = h13h = h14h = h15h = h16h = h17h = 
h18h = h19h = h20h = 0;

		for (; b <= z; b++) {
			if (masiv[b] < x) {
				if (masiv[b] >= a10) {
					if (masiv[b] >= a5) {
						if (masiv[b] >= a3) {
							if (masiv[b] >= a) {
								aa++;
								continue;
							}
							if (masiv[b] >= a2)   a2a++;
							else   a3a++;
							continue;
						}
						else {
							if (masiv[b] >= a4)   a4a++;
							else   a5a++;
							continue;
						}
					}
					else {
						if (masiv[b] >= a8) {
							if (masiv[b] >= a6) {
								a6a++;
```

```
                    continue;
                }
                if (masiv[b] >= a7)   a7a++;
                else   a8a++;
                continue;
            }
            else {
                if (masiv[b] >= a9)   a9a++;
                else   a10a++;
                continue;
            }
        }
    }
    else {
        if (masiv[b] >= a15) {
            if (masiv[b] >= a13) {
                if (masiv[b] >= a11) {
                    a11a++;
                    continue;
                }
                if (masiv[b] >= a12) a12a++;
                else   a13a++;
                continue;
            }
            else {
                if (masiv[b] >= a14)   a14a++;
                else   a15a++;
                continue;
            }
        }
        else {
            if (masiv[b] >= a18) {
                if (masiv[b] >= a16) {
                    a16a++;
                    continue;
                }
                if (masiv[b] >= a17)   a17a++;
                else   a18a++;
                continue;
            }
            else {
                if (masiv[b] >= a19)   a19a++;
                else   a20a++;
                continue;
            }
        }
    }
}
if (masiv[b] >= h20) {
    if (masiv[b] >= h10) {
        if (masiv[b] >= h5) {
            if (masiv[b] >= h3) {
                if (masiv[b] >= h) {
                    hh++;
                    continue;
                }
                if (masiv[b] >= h2)   h2h++;
                else   h3h++;
                continue;
            }
            else {
                if (masiv[b] >= h4)   h4h++;
                else   h5h++;
                continue;
            }
        }
```

```
            else {
                if (masiv[b] >= h8) {
                    if (masiv[b] >= h6) {
                        h6h++;
                        continue;
                    }
                    if (masiv[b] >= h7)   h7h++;
                    else   h8h++;
                    continue;
                }
                else {
                    if (masiv[b] >= h9)   h9h++;
                    else   h10h++;
                    continue;
                }
            }
        }
        else {
            if (masiv[b] >= h15) {
                if (masiv[b] >= h13) {
                    if (masiv[b] >= h11) {
                        h11h++;
                        continue;
                    }
                    if (masiv[b] >= h12)   h12h++;
                    else   h13h++;
                    continue;
                }
                else {
                    if (masiv[b] >= h14)   h14h++;
                    else   h15h++;
                    continue;
                }
            }
            else {
                if (masiv[b] >= h18) {
                    if (masiv[b] >= h16) {
                        h16h++;
                        continue;
                    }
                    if (masiv[b] >= h17)   h17h++;
                    else   h18h++;
                    continue;
                }
                else {
                    if (masiv[b] >= h19)   h19h++;
                    else   h20h++;
                    continue;
                }
            }
        }
    }
}

as = v;
as20 = a20a = a20a + v;
a20h = a20a - 1;
as19 = a19a = a19a + a20a;
a19h = a19a - 1;
as18 = a18a = a18a + a19a;
a18h = a18a - 1;
as17 = a17a = a17a + a18a;
a17h = a17a - 1;
as16 = a16a = a16a + a17a;
a16h = a16a - 1;
as15 = a15a = a15a + a16a;
```

```
a15h = a15a - 1;
as14 = a14a = a14a + a15a;
a14h = a14a - 1;
as13 = a13a = a13a + a14a;
a13h = a13a - 1;
as12 = a12a = a12a + a13a;
a12h = a12a - 1;
as11 = a11a = a11a + a12a;
a11h = a11a - 1;
as10 = a10a = a10a + a11a;
a10h = a10a - 1;
as9 = a9a = a9a + a10a;
a9h = a9a - 1;
as8 = a8a = a8a + a9a;
a8h = a8a - 1;
as7 = a7a = a7a + a8a;
a7h = a7a - 1;
as6 = a6a = a6a + a7a;
a6h = a6a - 1;
as5 = a5a = a5a + a6a;
a5h = a5a - 1;
as4 = a4a = a4a + a5a;
a4h = a4a - 1;
as3 = a3a = a3a + a4a;
a3h = a3a - 1;
as2 = a2a = a2a + a3a;
a2h = a2a - 1;
aa = aa + a2a;
ah = aa - 1;

az20 = z;
az = hh = z - hh;
ha = hh + 1;
az2 = h2h = hh - h2h;
h2a = h2h + 1;
az3 = h3h = h2h - h3h;
h3a = h3h + 1;
az4 = h4h = h3h - h4h;
h4a = h4h + 1;
az5 = h5h = h4h - h5h;
h5a = h5h + 1;
az6 = h6h = h5h - h6h;
h6a = h6h + 1;
az7 = h7h = h6h - h7h;
h7a = h7h + 1;
az8 = h8h = h7h - h8h;
h8a = h8h + 1;
az9 = h9h = h8h - h9h;
h9a = h9h + 1;
az10 = h10h = h9h - h10h;
h10a = h10h + 1;
az11 = h11h = h10h - h11h;
h11a = h11h + 1;
az12 = h12h = h11h - h12h;
h12a = h12h + 1;
az13 = h13h = h12h - h13h;
h13a = h13h + 1;
az14 = h14h = h13h - h14h;
h14a = h14h + 1;
az15 = h15h = h14h - h15h;
h15a = h15h + 1;
az16 = h16h = h15h - h16h;
h16a = h16h + 1;
az17 = h17h = h16h - h17h;
h17a = h17h + 1;
az18 = h18h = h17h - h18h;
```

```
h18a = h18h + 1;
az19 = h19h = h18h - h19h;
h19a = h19h + 1;
h20h = h19h - h20h;
h20a = h20h + 1;

b = v;

for (; b <= z; ) {
        if (masiv[b] < x) {
                if (masiv[b] >= a10) {
                        if (masiv[b] >= a5) {
                                if (masiv[b] >= a3) {
                                        if (masiv[b] >= a) {
                                                if (b <= a2a) {
                                                        if (b >= as2) {
                                                                if (b < a2a) {
                                                                        b = a2a;
                                                                        continue;
                                                                }
                                                                if (b == ah) {
                                                                        b++;
                                                                        continue;
                                                                }
                                                                if (a2a < ah)   a2a++;
                                                                b = a2a;
                                                                continue;
                                                        }
                                                }
                                                vv = masiv[b];
                                                masiv[b] = masiv[a2a];
                                                masiv[a2a] = vv;
                                                if (b < a2a) {
                                                        if (a2a < ah) a2a++;
                                                        continue;
                                                }
                                                if (a2a < ah)   a2a++;
                                                b++;
                                                continue;
                                        }
                                        if (masiv[b] >= a2) {
                                                if (b <= a3a) {
                                                        if (b >= as3) {
                                                                if (b < a3a) {
                                                                        b = a3a;
                                                                        continue;
                                                                }
                                                                if (b == a2h) {
                                                                        b++;
                                                                        continue;
                                                                }
                                                                if (a3a < a2h)   a3a++;
                                                                b = a3a;
                                                                continue;
                                                        }
                                                }
                                                vv = masiv[b];
                                                masiv[b] = masiv[a3a];
                                                masiv[a3a] = vv;
                                                if (b < a3a) {
                                                        if (a3a < a2h) a3a++;
                                                        continue;
                                                }
                                                if (a3a < a2h)   a3a++;
                                                b++;
                                                continue;
                                        }
                                        else {
                                                if (b <= a4a) {
                                                        if (b >= as4) {
                                                                if (b < a4a) {
                                                                        b = a4a;
                                                                        continue;
                                                                }
                                                                if (b == a3h) {
                                                                        b++;
                                                                        continue;
                                                                }
                                                                if (a4a < a3h)   a4a++;
                                                                b = a4a;
                                                                continue;
                                                        }
                                                }
                                                vv = masiv[b];
                                                masiv[b] = masiv[a4a];
                                                masiv[a4a] = vv;
                                                if (b < a4a) {
                                                        if (a4a < a3h) a4a++;
                                                        continue;
                                                }
                                                if (a4a < a3h)   a4a++;
                                                b++;
                                                continue;
                                        }
                                }
                                else {
                                        if (masiv[b] >= a4) {
                                                if (b <= a5a) {
                                                        if (b >= as5) {
                                                                if (b < a5a) {
                                                                        b = a5a;
                                                                        continue;
                                                                }
                                                                if (b == a4h) {
                                                                        b++;
                                                                        continue;
                                                                }
                                                                if (a5a < a4h)   a5a++;
                                                                b = a5a;
                                                                continue;
                                                        }
                                                }
                                                vv = masiv[b];
                                                masiv[b] = masiv[a5a];
                                                masiv[a5a] = vv;
                                                if (b < a5a) {
                                                        if (a5a < a4h) a5a++;
                                                        continue;
                                                }
                                                if (a5a < a4h)   a5a++;
                                                b++;
                                                continue;
                                        }
                                        else {
                                                if (b <= a6a) {
                                                        if (b >= as6) {
                                                                if (b < a6a) {
                                                                        b = a6a;
                                                                        continue;
                                                                }
                                                                if (b == a5h) {
                                                                        b++;
                                                continue;
                                            }
                                            if (a6a < a5h)   a6a++;
                                            b = a6a;
                                            continue;
                                        }
                                    }
                                    vv = masiv[b];
                                    masiv[b] = masiv[a6a];
                                    masiv[a6a] = vv;
                                    if (b < a6a) {
                                        if (a6a < a5h) a6a++;
                                        continue;
                                    }
                                    if (a6a < a5h)   a6a++;
                                    b++;
                                    continue;
                                }
                            }
}
else {
    if (masiv[b] >= a8) {
        if (masiv[b] >= a6) {
            if (b <= a7a) {
                if (b >= as7) {
                    if (b < a7a) {
                        b = a7a;
                        continue;
                    }
                    if (b == a6h) {
                        b++;
                        continue;
                    }
                    if (a7a < a6h)   a7a++;
                    b = a7a;
                    continue;
                }
            }
            vv = masiv[b];
            masiv[b] = masiv[a7a];
            masiv[a7a] = vv;
            if (b < a7a) {
                if (a7a < a6h) a7a++;
                continue;
            }
            if (a7a < a6h)   a7a++;
            b++;
            continue;
        }
        if (masiv[b] >= a7) {
            if (b <= a8a) {
                if (b >= as8) {
                    if (b < a8a) {
                        b = a8a;
                        continue;
                    }
                    if (b == a7h) {
                        b++;
                        continue;
                    }
                    if (a8a < a7h)   a8a++;
                    b = a8a;
                    continue;
                }
            }
            vv = masiv[b];
```

```
                masiv[b] = masiv[a8a];
                masiv[a8a] = vv;
                if (b < a8a) {
                        if (a8a < a7h) a8a++;
                        continue;
                }
                if (a8a < a7h)   a8a++;
                b++;
                continue;
        }
        else {
                if (b <= a9a) {
                        if (b >= as9) {
                                if (b < a9a) {
                                        b = a9a;
                                        continue;
                                }
                                if (b == a8h) {
                                        b++;
                                        continue;
                                }
                                if (a9a < a8h)   a9a++;
                                b = a9a;
                                continue;
                        }
                }
                vv = masiv[b];
                masiv[b] = masiv[a9a];
                masiv[a9a] = vv;
                if (b < a9a) {
                        if (a9a < a8h) a9a++;
                        continue;
                }
                if (a9a < a8h)   a9a++;
                b++;
                continue;
        }
}
else {
        if (masiv[b] >= a9) {
                if (b <= a10a) {
                        if (b >= as10) {
                                if (b < a10a) {
                                        b = a10a;
                                        continue;
                                }
                                if (b == a9h) {
                                        b++;
                                        continue;
                                }
                                if (a10a < a9h)   a10a++;
                                b = a10a;
                                continue;
                        }
                }
                vv = masiv[b];
                masiv[b] = masiv[a10a];
                masiv[a10a] = vv;
                if (b < a10a) {
                        if (a10a < a9h) a10a++;
                        continue;
                }
                if (a10a < a9h)   a10a++;
                b++;
                continue;
        }
```

```
            else {
                if (b <= a11a) {
                    if (b >= as11) {
                        if (b < a11a) {
                            b = a11a;
                            continue;
                        }
                        if (b == a10h) {
                            b++;
                            continue;
                        }
                        if (a11a < a10h)   a11a++;
                        b = a11a;
                        continue;
                    }
                }
                vv = masiv[b];
                masiv[b] = masiv[a11a];
                masiv[a11a] = vv;
                if (b < a11a) {
                    if (a11a < a10h) a11a++;
                    continue;
                }
                if (a11a < a10h)   a11a++;
                b++;
                continue;
            }
        }
    }
}
else {
    if (masiv[b] >= a15) {
        if (masiv[b] >= a13) {
            if (masiv[b] >= a11) {
                if (b <= a12a) {
                    if (b >= as12) {
                        if (b < a12a) {
                            b = a12a;
                            continue;
                        }
                        if (b == a11h) {
                            b++;
                            continue;
                        }
                        if (a12a < a11h)   a12a++;
                        b = a12a;
                        continue;
                    }
                }
                vv = masiv[b];
                masiv[b] = masiv[a12a];
                masiv[a12a] = vv;
                if (b < a12a) {
                    if (a12a < a11h) a12a++;
                    continue;
                }
                if (a12a < a11h)   a12a++;
                b++;
                continue;
            }
            if (masiv[b] >= a12) {
                if (b <= a13a) {
                    if (b >= as13) {
                        if (b < a13a) {
                            b = a13a;
                            continue;
```

```
                                                }
                                                if (b == a12h) {
                                                        b++;
                                                        continue;
                                                }
                                                if (a13a < a12h)   a13a++;
                                                b = a13a;
                                                continue;
                                        }
                                }
                                vv = masiv[b];
                                masiv[b] = masiv[a13a];
                                masiv[a13a] = vv;
                                if (b < a13a) {
                                        if (a13a < a12h) a13a++;
                                        continue;
                                }
                                if (a13a < a12h)   a13a++;
                                b++;
                                continue;
                        }
                        else {
                                if (b <= a14a) {
                                        if (b >= as14) {
                                                if (b < a14a) {
                                                        b = a14a;
                                                        continue;
                                                }
                                                if (b == a13h) {
                                                        b++;
                                                        continue;
                                                }
                                                if (a14a < a13h)   a14a++;
                                                b = a14a;
                                                continue;
                                        }
                                }
                                vv = masiv[b];
                                masiv[b] = masiv[a14a];
                                masiv[a14a] = vv;
                                if (b < a14a) {
                                        if (a14a < a13h) a14a++;
                                        continue;
                                }
                                if (a14a < a13h)   a14a++;
                                b++;
                                continue;
                        }
                }
                else {
                        if (masiv[b] >= a14) {
                                if (b <= a15a) {
                                        if (b >= as15) {
                                                if (b < a15a) {
                                                        b = a15a;
                                                        continue;
                                                }
                                                if (b == a14h) {
                                                        b++;
                                                        continue;
                                                }
                                                if (a15a < a14h)   a15a++;
                                                b = a15a;
                                                continue;
                                        }
                                }
```

```
            vv = masiv[b];
            masiv[b] = masiv[a15a];
            masiv[a15a] = vv;
            if (b < a15a) {
                if (a15a < a14h) a15a++;
                continue;
            }
            if (a15a < a14h)   a15a++;
            b++;
            continue;
        }
        else {
            if (b <= a16a) {
                if (b >= as16) {
                    if (b < a16a) {
                        b = a16a;
                        continue;
                    }
                    if (b == a15h) {
                        b++;
                        continue;
                    }
                    if (a16a < a15h)   a16a++;
                    b = a16a;
                    continue;
                }
            }
            vv = masiv[b];
            masiv[b] = masiv[a16a];
            masiv[a16a] = vv;
            if (b < a16a) {
                if (a16a < a15h) a16a++;
                continue;
            }
            if (a16a < a15h)   a16a++;
            b++;
            continue;
        }
    }
}
else {
    if (masiv[b] >= a18) {
        if (masiv[b] >= a16) {
            if (b <= a17a) {
                if (b >= as17) {
                    if (b < a17a) {
                        b = a17a;
                        continue;
                    }
                    if (b == a16h) {
                        b++;
                        continue;
                    }
                    if (a17a < a16h)   a17a++;
                    b = a17a;
                    continue;
                }
            }
            vv = masiv[b];
            masiv[b] = masiv[a17a];
            masiv[a17a] = vv;
            if (b < a17a) {
                if (a17a < a16h) a17a++;
                continue;
            }
            if (a17a < a16h)   a17a++;
```

```
                b++;
                continue;
        }
        if (masiv[b] >= a17) {
                if (b <= a18a) {
                        if (b >= as18) {
                                if (b < a18a) {
                                        b = a18a;
                                        continue;
                                }
                                if (b == a17h) {
                                        b++;
                                        continue;
                                }
                                if (a18a < a17h)   a18a++;
                                b = a18a;
                                continue;
                        }
                }
                vv = masiv[b];
                masiv[b] = masiv[a18a];
                masiv[a18a] = vv;
                if (b < a18a) {
                        if (a18a < a17h) a18a++;
                        continue;
                }
                if (a18a < a17h)   a18a++;
                b++;
                continue;
        }
        else {
                if (b <= a19a) {
                        if (b >= as19) {
                                if (b < a19a) {
                                        b = a19a;
                                        continue;
                                }
                                if (b == a18h) {
                                        b++;
                                        continue;
                                }
                                if (a19a < a18h)   a19a++;
                                b = a19a;
                                continue;
                        }
                }
                vv = masiv[b];
                masiv[b] = masiv[a19a];
                masiv[a19a] = vv;
                if (b < a19a) {
                        if (a19a < a18h) a19a++;
                        continue;
                }
                if (a19a < a18h)   a19a++;
                b++;
                continue;
        }
}
else {
        if (masiv[b] >= a19) {
                if (b <= a20a) {
                        if (b >= as20) {
                                if (b < a20a) {
                                        b = a20a;
                                        continue;
                                }
```

```
                            if (b == a19h) {
                                b++;
                                continue;
                            }
                            if (a20a < a19h)   a20a++;
                            b = a20a;
                            continue;
                        }
                    }
                    vv = masiv[b];
                    masiv[b] = masiv[a20a];
                    masiv[a20a] = vv;
                    if (b < a20a) {
                        if (a20a < a19h) a20a++;
                        continue;
                    }
                    if (a20a < a19h)   a20a++;
                    b++;
                    continue;
                }
                else {
                    if (b <= v) {
                        if (b >= as) {
                            if (b < v) {
                                b = v;
                                continue;
                            }
                            if (b == a20h) {
                                b++;
                                continue;
                            }
                            if (v < a20h)   v++;
                            b = v;
                            continue;
                        }
                    }
                    vv = masiv[b];
                    masiv[b] = masiv[v];
                    masiv[v] = vv;
                    if (b < v) {
                        if (v < a20h) v++;
                        continue;
                    }
                    if (v < a20h)   v++;
                    b++;
                    continue;
                }
            }
        }
    }
}
if (masiv[b] >= h20) {
    if (masiv[b] >= h10) {
        if (masiv[b] >= h5) {
            if (masiv[b] >= h3) {
                if (masiv[b] >= h) {
                    if (b >= z) {
                        if (b <= az20) break;

                    }
                    vv = masiv[b];
                    masiv[b] = masiv[z];
                    masiv[z] = vv;
                    if (b < z) {
                        if (z > ha) z--;
                        continue;
        }
        if (z > ha)   z--;
        b++;
        continue;
    }
    if (masiv[b] >= h2) {
        if (b >= hh) {
            if (b <= az) {
                b = az;
                b++;
                continue;
            }
        }
        vv = masiv[b];
        masiv[b] = masiv[hh];
        masiv[hh] = vv;
        if (b < hh) {
            if (hh > h2a) hh--;
            continue;
        }
        if (hh > h2a)   hh--;
        b++;
        continue;
    }
    else {
        if (b >= h2h) {
            if (b <= az2) {
                b = az2;
                b++;
                continue;
            }
        }
        vv = masiv[b];
        masiv[b] = masiv[h2h];
        masiv[h2h] = vv;
        if (b < h2h) {
            if (h2h > h3a) h2h--;
            continue;
        }
        if (h2h > h3a)   h2h--;
        b++;
        continue;
    }
}
else {
    if (masiv[b] >= h4) {
        if (b >= h3h) {
            if (b <= az3) {
                b = az3;
                b++;
                continue;
            }
        }
        vv = masiv[b];
        masiv[b] = masiv[h3h];
        masiv[h3h] = vv;
        if (b < h3h) {
            if (h3h > h4a) h3h--;
            continue;
        }
        if (h3h > h4a)   h3h--;
        b++;
        continue;
    }
    else {
        if (b >= h4h) {
                if (b <= az4) {
                    b = az4;
                    b++;
                    continue;
                }
            }
            vv = masiv[b];
            masiv[b] = masiv[h4h];
            masiv[h4h] = vv;
            if (b < h4h) {
                if (h4h > h5a) h4h--;
                continue;
            }
            if (h4h > h5a)   h4h--;
            b++;
            continue;
        }
    }
}
else {
    if (masiv[b] >= h8) {
        if (masiv[b] >= h6) {
            if (b >= h5h) {
                if (b <= az5) {
                    b = az5;
                    b++;
                    continue;
                }
            }
            vv = masiv[b];
            masiv[b] = masiv[h5h];
            masiv[h5h] = vv;
            if (b < h5h) {
                if (h5h > h6a) h5h--;
                continue;
            }
            if (h5h > h6a)   h5h--;
            b++;
            continue;
        }
        if (masiv[b] >= h7) {
            if (b >= h6h) {
                if (b <= az6) {
                    b = az6;
                    b++;
                    continue;
                }
            }
            vv = masiv[b];
            masiv[b] = masiv[h6h];
            masiv[h6h] = vv;
            if (b < h6h) {
                if (h6h > h7a) h6h--;
                continue;
            }
            if (h6h > h7a)   h6h--;
            b++;
            continue;
        }
        else {
            if (b >= h7h) {
                if (b <= az7) {
                    b = az7;
                    b++;
                    continue;
                }
                }
                vv = masiv[b];
                masiv[b] = masiv[h7h];
                masiv[h7h] = vv;
                if (b < h7h) {
                    if (h7h > h8a) h7h--;
                    continue;
                }
                if (h7h > h8a)   h7h--;
                b++;
                continue;
            }
        }
        else {
            if (masiv[b] >= h9) {
                if (b >= h8h) {
                    if (b <= az8) {
                        b = az8;
                        b++;
                        continue;
                    }
                }
                vv = masiv[b];
                masiv[b] = masiv[h8h];
                masiv[h8h] = vv;
                if (b < h8h) {
                    if (h8h > h9a) h8h--;
                    continue;
                }
                if (h8h > h9a)   h8h--;
                b++;
                continue;
            }
            else {
                if (b >= h9h) {
                    if (b <= az9) {
                        b = az9;
                        b++;
                        continue;
                    }
                }
                vv = masiv[b];
                masiv[b] = masiv[h9h];
                masiv[h9h] = vv;
                if (b < h9h) {
                    if (h9h > h10a) h9h--;
                    continue;
                }
                if (h9h > h10a)   h9h--;
                b++;
                continue;
            }
        }
    }
}
else {
    if (masiv[b] >= h15) {
        if (masiv[b] >= h13) {
            if (masiv[b] >= h11) {
                if (b >= h10h) {
                    if (b <= az10) {
                        b = az10;
                        b++;
                        continue;
                    }
                }
        vv = masiv[b];
        masiv[b] = masiv[h10h];
        masiv[h10h] = vv;
        if (b < h10h) {
            if (h10h > h11a) h10h--;
            continue;
        }
        if (h10h > h11a)   h10h--;
        b++;
        continue;
    }
    if (masiv[b] >= h12) {
        if (b >= h11h) {
            if (b <= az11) {
                b = az11;
                b++;
                continue;
            }
        }
        vv = masiv[b];
        masiv[b] = masiv[h11h];
        masiv[h11h] = vv;
        if (b < h11h) {
            if (h11h > h12a) h11h--;
            continue;
        }
        if (h11h > h12a)   h11h--;
        b++;
        continue;
    }
    else {
        if (b >= h12h) {
            if (b <= az12) {
                b = az12;
                b++;
                continue;
            }
        }
        vv = masiv[b];
        masiv[b] = masiv[h12h];
        masiv[h12h] = vv;
        if (b < h12h) {
            if (h12h > h13a) h12h--;
            continue;
        }
        if (h12h > h13a)   h12h--;
        b++;
        continue;
    }
}
else {
    if (masiv[b] >= h14) {
        if (b >= h13h) {
            if (b <= az13) {
                b = az13;
                b++;
                continue;
            }
        }
        vv = masiv[b];
        masiv[b] = masiv[h13h];
        masiv[h13h] = vv;
        if (b < h13h) {
            if (h13h > h14a) h13h--;
            continue;
        }
```

```
                        if (h13h > h14a)   h13h--;
                        b++;
                        continue;
                    }
                    else {
                        if (b >= h14h) {
                            if (b <= az14) {
                                b = az14;
                                b++;
                                continue;
                            }
                        }
                        vv = masiv[b];
                        masiv[b] = masiv[h14h];
                        masiv[h14h] = vv;
                        if (b < h14h) {
                            if (h14h > h15a) h14h--;
                            continue;
                        }
                        if (h14h > h15a)   h14h--;
                        b++;
                        continue;
                    }
                }
            }
            else {
                if (masiv[b] >= h18) {
                    if (masiv[b] >= h16) {
                        if (b >= h15h) {
                            if (b <= az15) {
                                b = az15;
                                b++;
                                continue;
                            }
                        }
                        vv = masiv[b];
                        masiv[b] = masiv[h15h];
                        masiv[h15h] = vv;
                        if (b < h15h) {
                            if (h15h > h16a) h15h--;
                            continue;
                        }
                        if (h15h > h16a)   h15h--;
                        b++;
                        continue;
                    }
                    if (masiv[b] >= h17) {
                        if (b >= h16h) {
                            if (b <= az16) {
                                b = az16;
                                b++;
                                continue;
                            }
                        }
                        vv = masiv[b];
                        masiv[b] = masiv[h16h];
                        masiv[h16h] = vv;
                        if (b < h16h) {
                            if (h16h > h17a) h16h--;
                            continue;
                        }
                        if (h16h > h17a)   h16h--;
                        b++;
                        continue;
                    }
                    else {
```

```
						if (b >= h17h) {
							if (b <= az17) {
								b = az17;
								b++;
								continue;
							}
						}
						vv = masiv[b];
						masiv[b] = masiv[h17h];
						masiv[h17h] = vv;
						if (b < h17h) {
							if (h17h > h18a) h17h--;
							continue;
						}
						if (h17h > h18a)   h17h--;
						b++;
						continue;
					}
				}
				else {
					if (masiv[b] >= h19) {
						if (b >= h18h) {
							if (b <= az18) {
								b = az18;
								b++;
								continue;
							}
						}
						vv = masiv[b];
						masiv[b] = masiv[h18h];
						masiv[h18h] = vv;
						if (b < h18h) {
							if (h18h > h19a) h18h--;
							continue;
						}
						if (h18h > h19a)   h18h--;
						b++;
						continue;
					}
					else {
						if (b >= h19h) {
							if (b <= az19) {
								b = az19;
								b++;
								continue;
							}
						}
						vv = masiv[b];
						masiv[b] = masiv[h19h];
						masiv[h19h] = vv;
						if (b < h19h) {
							if (h19h > h20a) h19h--;
							continue;
						}
						if (h19h > h20a)   h19h--;
						b++;
						continue;
					}
				}
			}
		}
	}
	b++;
}

b = v = aa;
```

```
                z = h20h;
                a20 = x;
                h = h20;
        }

        // Start sorting the array (stage 2).

        aa = an;
        x = y;
        b = bb;
        z = 0;
        for (d = 0; d <= b; d++) {
                v = vv = z;
                aa = aa + t;
                for (; v < s; v++) {
                        if (masiv[v] < aa) {
                                if (v == x) goto mit;
                                if (masiv[v + 1] >= aa) {
                                mit:
                                        if (vv == v) {
                                                z = ++v;
                                                break;
                                        }
                                        z = v = ++v;
                                        for (; v > vv; v--) {
                                                h = masiv[vv];
                                                for (c = (vv + 1); c < v; c++) if (h < masiv[c]) h = masiv[c];
                                                for (c = vv; c < v; c++) if (h == masiv[c]) y = c;
                                                if (masiv[y] == masiv[v - 1]) continue;
                                                masiv[y] = masiv[v - 1];
                                                masiv[v - 1] = h;
                                        }
                                        break;
                                }
                                continue;
                        }
                        if (masiv[v] >= aa) break;
                }
        }

// для предотвращения закрытия окна, я применяю
// вот этот "кусочек":  -  cin >> i;

        cout << "Массив отсортирован.\n";
        cout << "Чтобы вывести массив, введите число и нажмите Enter\n";
        cin >> i;
        for (a = 0; a < s; a++) cout << masiv[a] << ' ';
        cin >> i;
        cout << i;
        return 0;
}
```
The end of the algorithm.