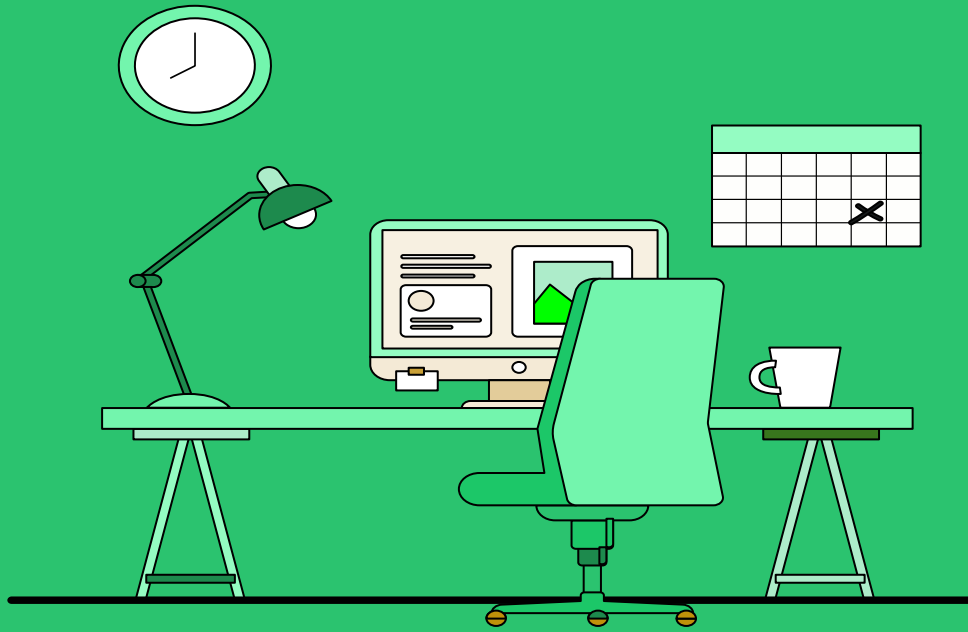
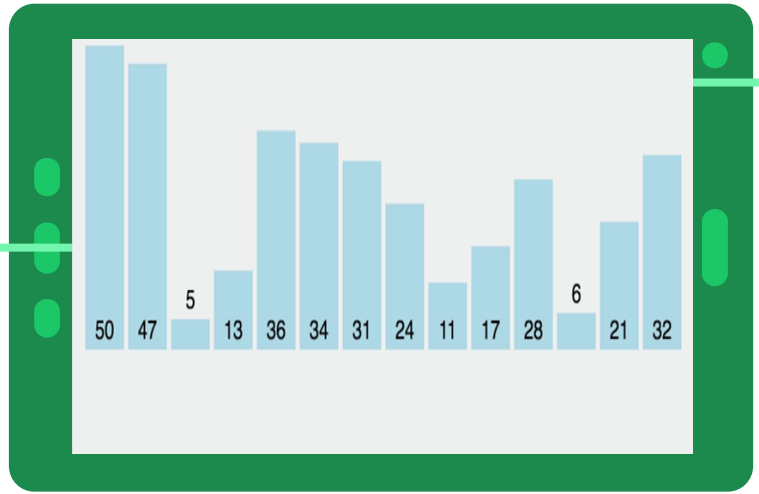
АЛГОРИТМИ СОРТУВАННЯ

C++



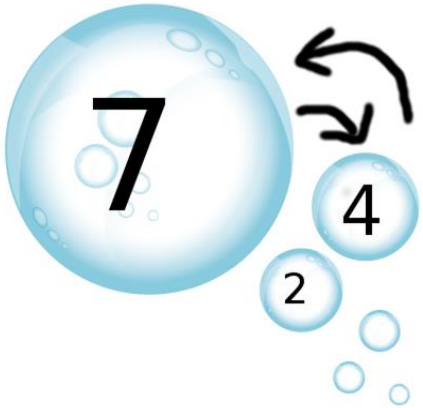
АЛГОРИТМИ СОРТУВАННЯ

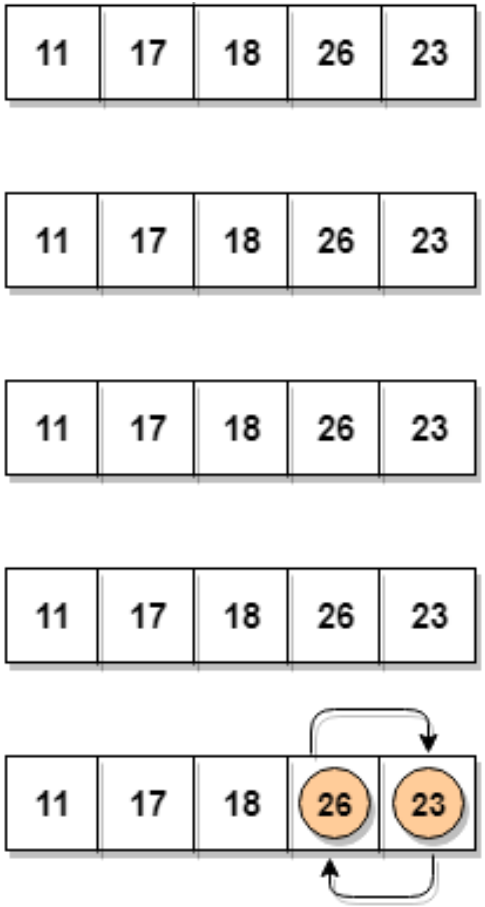
ДАНІ



ВПОРЯДКОВАНІ ДАНІ

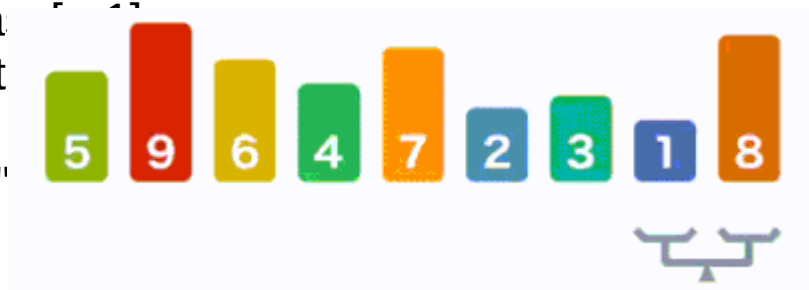
Бульбашка


7
4
2

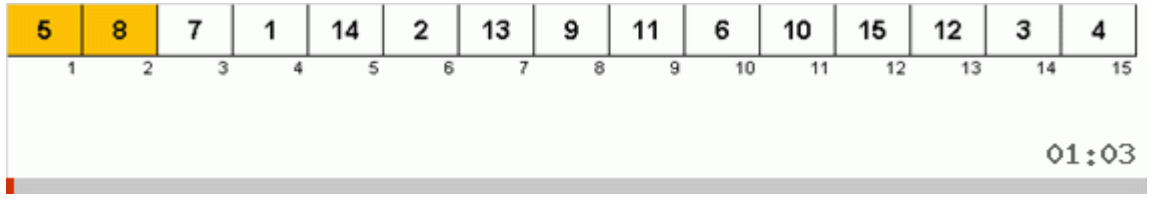

11 17 18 26 23
11 < 17
(No Swapping)
11 17 18 26 23
flag = 0
(flag remains 0)
17 < 18
(No Swapping)
11 17 18 26 23
flag = 0
18 < 26
(No Swapping)
11 17 18 26 23
flag = 0
26 < 23
(Swap then)
11 17 18 26 23
flag = 1

Bubble sort

```
#include <iostream>
using namespace std;
int main()
{  int n;
        cin >> n;
        int mass[n];
        for(int i = 0; i < n; ++i cin >> mass[i];

        for(int i = 1; i < n; ++i)
        for(int r = i; r < n-i; r++)
                if(mass[r] < mass[r+1])
                        {
                                int temp = mass[r];
                                mass[r] = ma
                                mass[r+1] = t
                        }
        for(int i = 0; i < n; ++i) cout << mass[i] << "
        return 0;
}
```

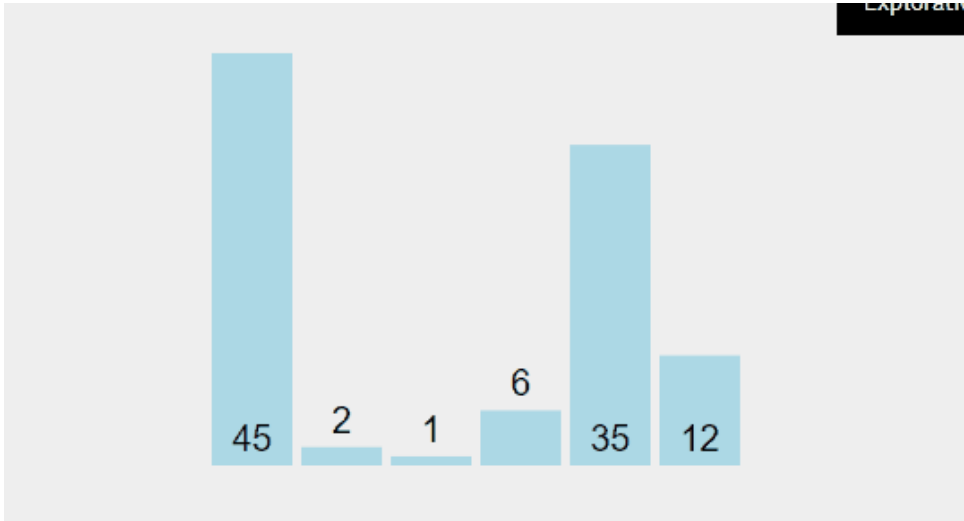


Gnome sort

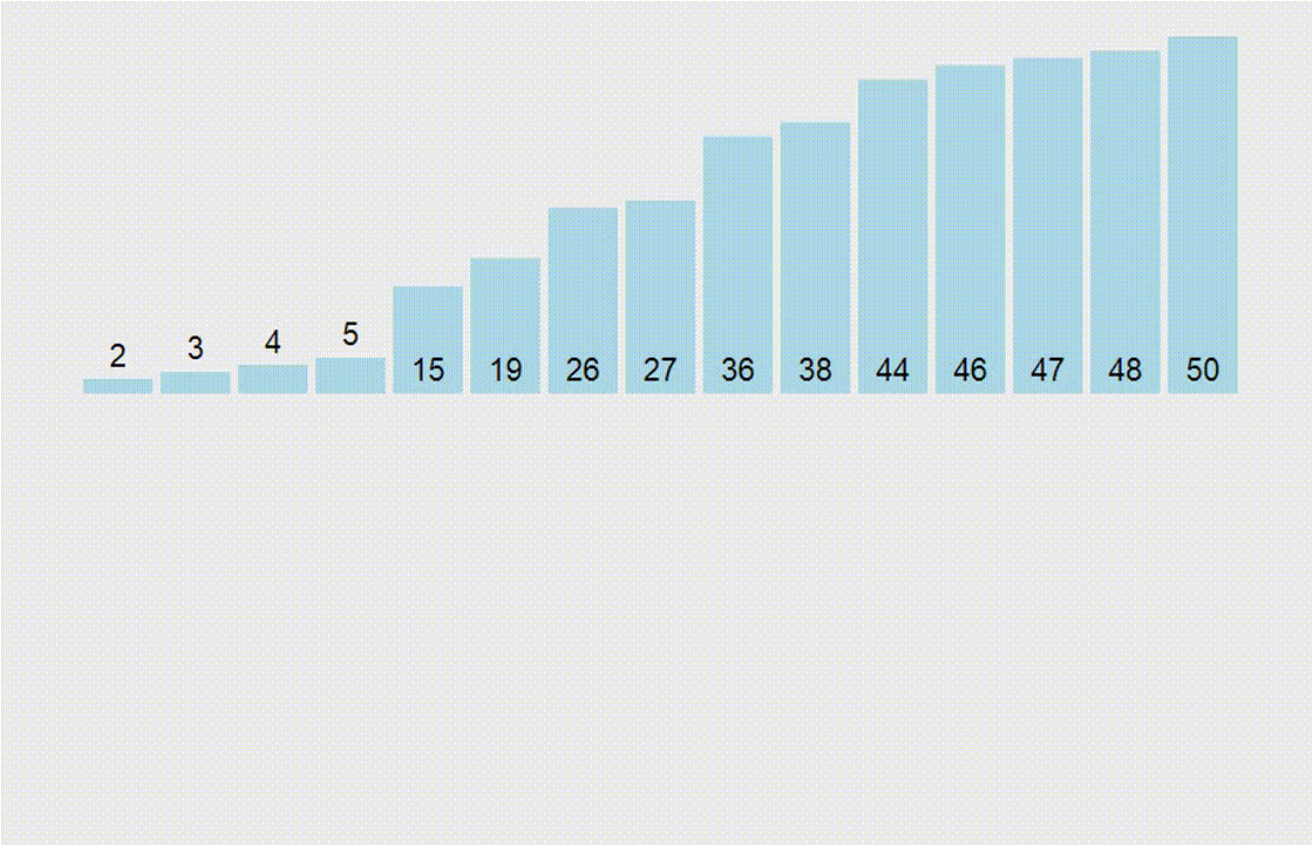


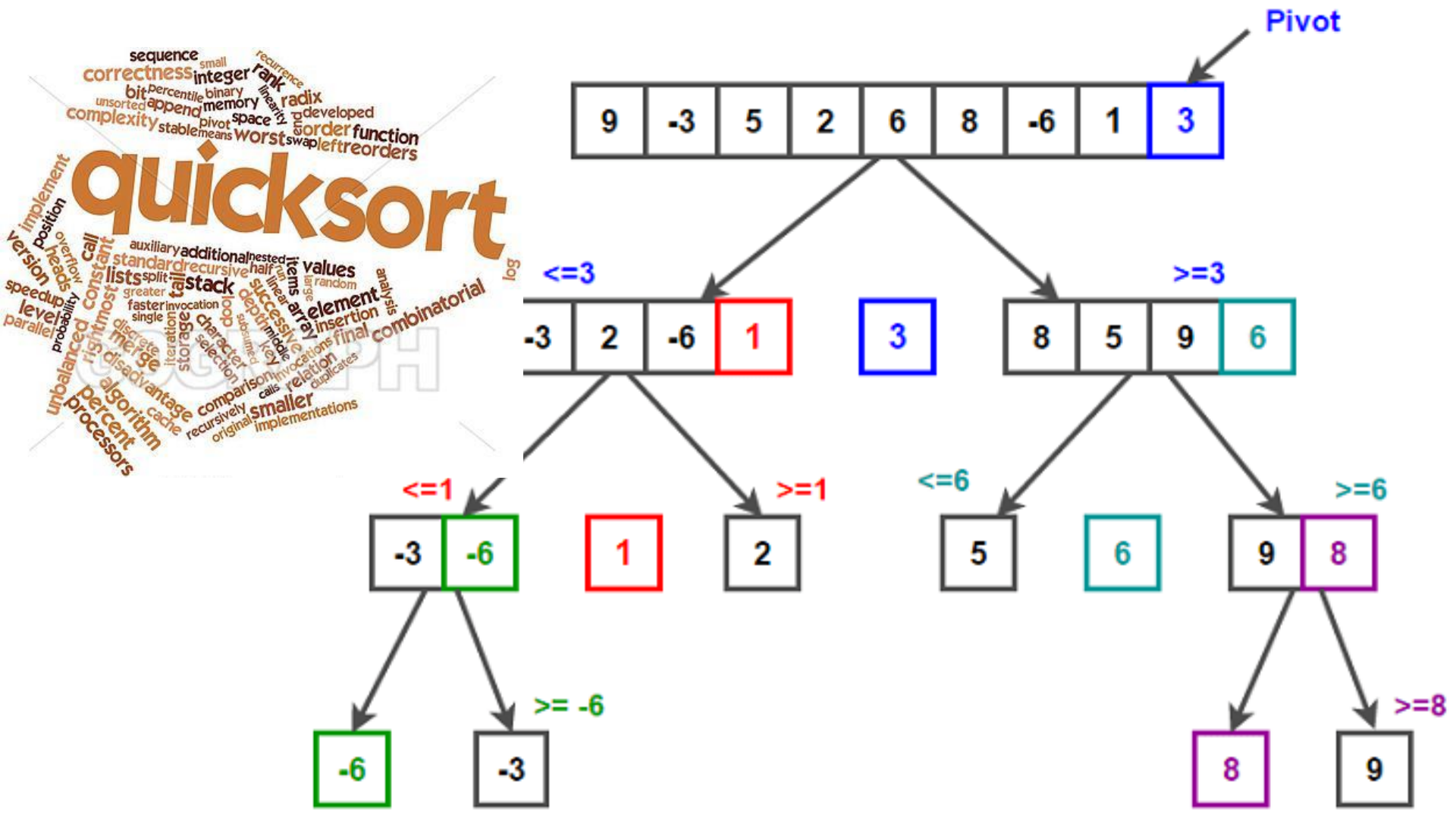
Selection sort

```
    for (int i =0 ; i < 10; i++)
 {
     for (int j = i; j < 10; j++)     {
         if (mas[j] > mas[i])     {
              tmp = mas[i];
              mas[i] = mas[j];
              mas[j] = tmp;
                   }
;}
```

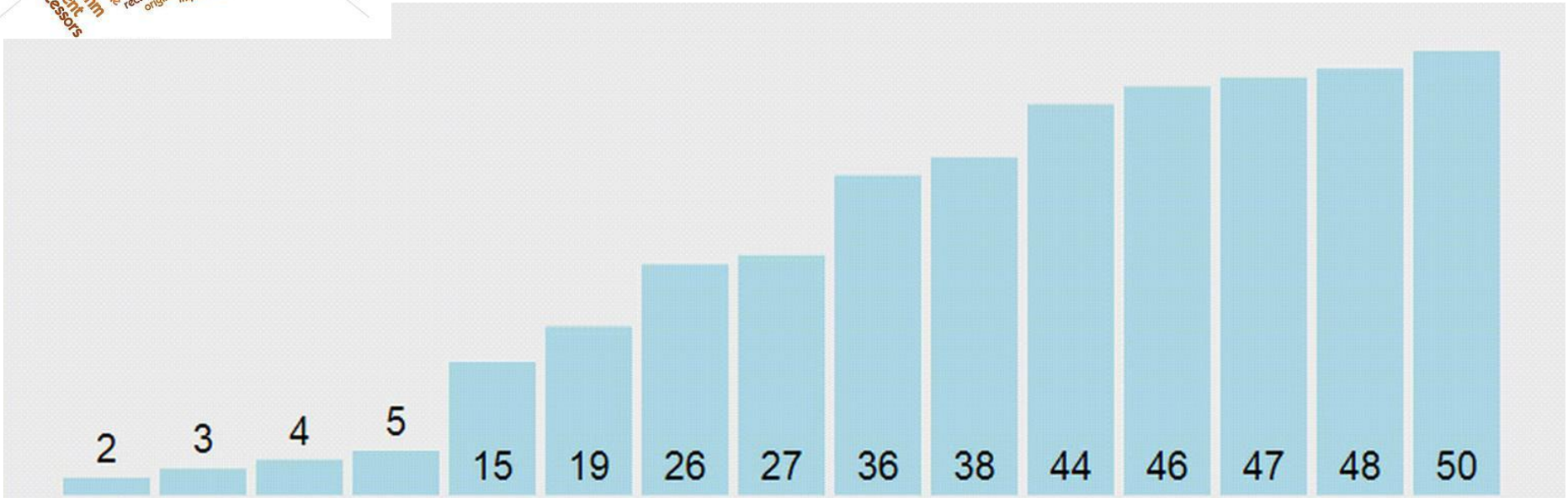


Insertion sort

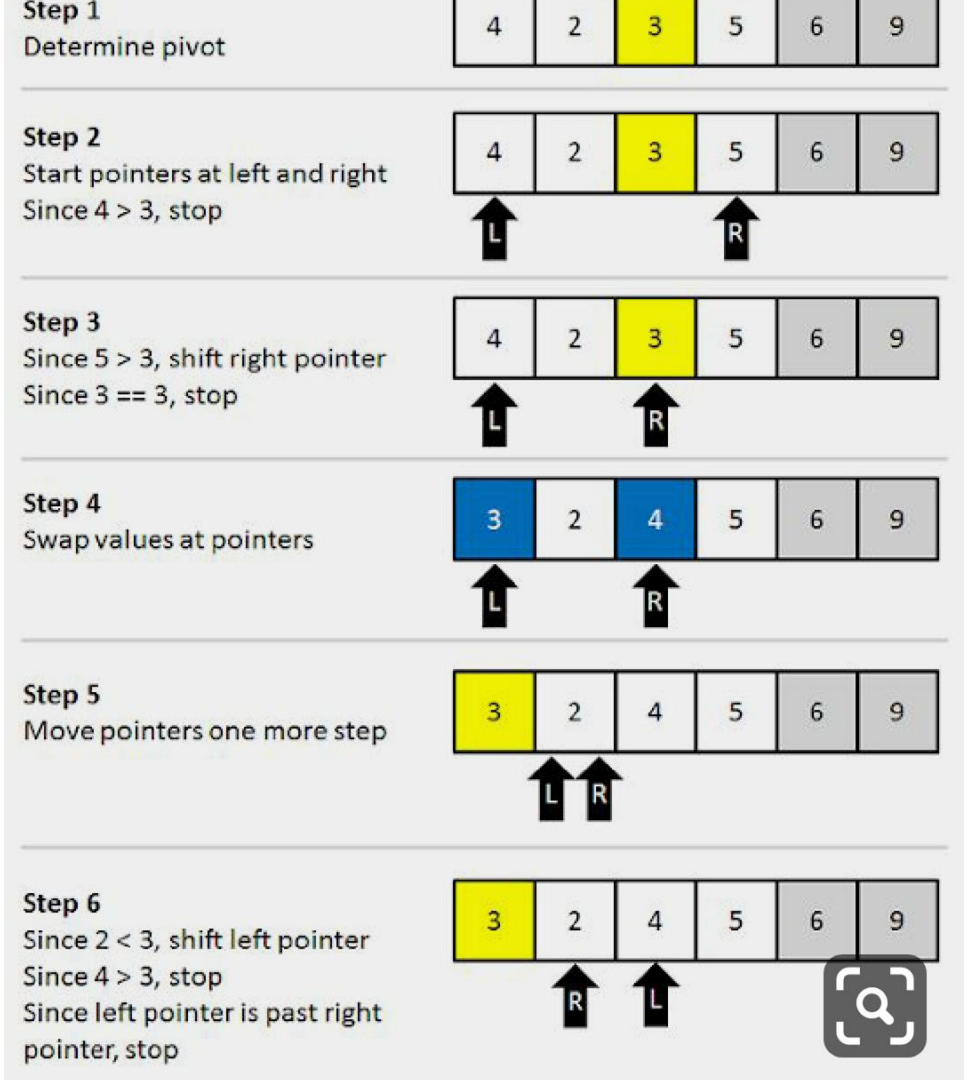



quicksort
sequence
correctness
integer
rank
radix
bit
append
memory
complexity
pivot
space
worst
order
function
left
reorders
developed
stable
means
swap
implement
position
version
heads
call
constant
lists
stack
auxiliary
additional
nested
standard
recursive
half
items
values
element
array
insertion
final
combinatorial
merge
disadvantage
algorithm
percent
processors
unbalanced
comparison
smaller
implementations
Pivot
9 -3 5 2 6 8 -6 1 3
<=3
>=3
-3 2 -6 1
3
8 5 9 6
<=1
>=1
<=6
>=6
-3 -6
1
2
5
6
9 8
>= -6
>=8
-6
-3
8
9


quicksort
sequence
small
correctness
integer
rank
radix
bit
percentile
binary
memory
append
unsorted
complexity
stable
means
pivot
space
worst
swap
order
function
developed
leftreorders
implement
position
overflow
heads
version
call
constant
standard
recursive
auxiliary
additional
nested
half
items
values
lists
split
stack
greater
faster
invocation
single
loop
element
insertion
final
combinatorial
analysis
random
speedup
level
parallel
probability
unbalanced
rightmost
merge
disadvantage
storage
iteration
character
selection
comparison
calls
recursively
original
smaller
implementations
relation
duplicates
cache
algorithm
Percent
processors
log

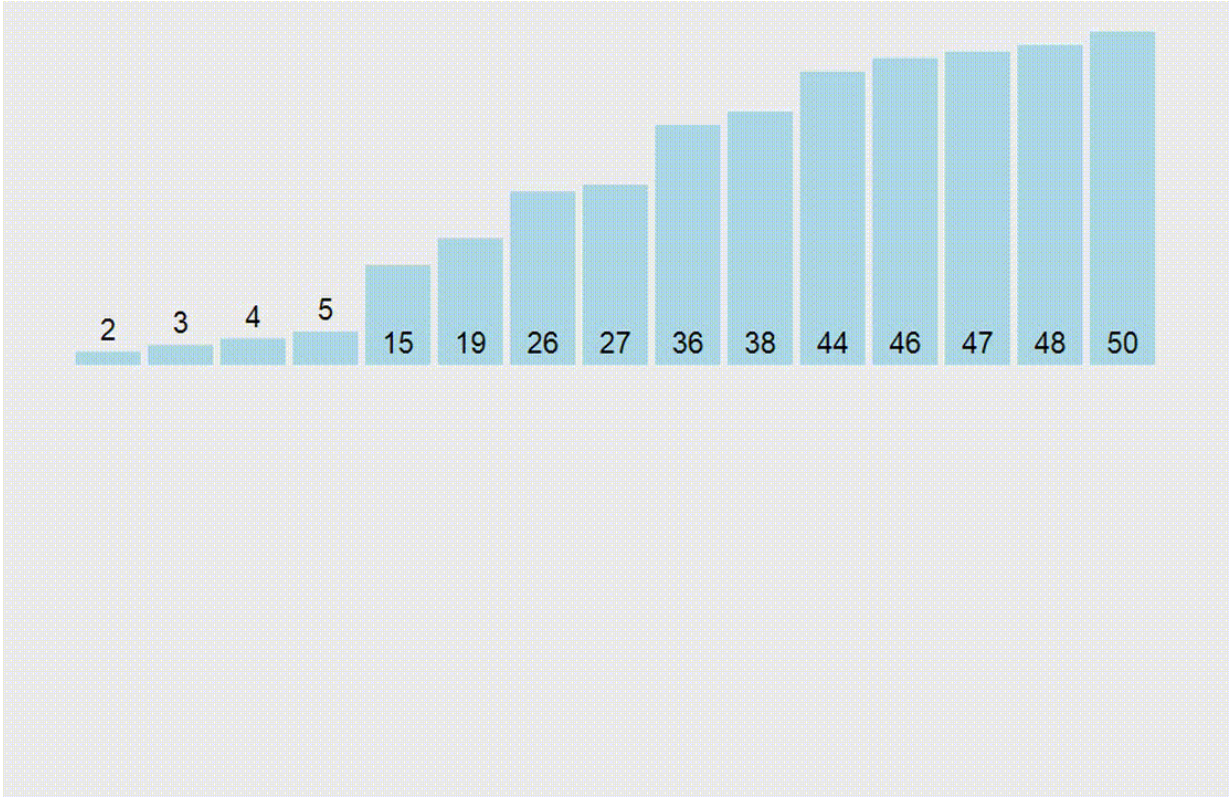

2
3
4
5
15
19
26
27
36
38
44
46
47
48
50

```
void quickSort(int arr[], int left, int right) {

      int i = left, j = right;
      int tmp;
     int pivot = arr[(left + right) / 2];

    while (i <= j) {
            while (arr[i] < pivot)
                  i++;
            while (arr[j] > pivot)
                  j--;
            if (i <= j) {
                  tmp = arr[i];
                  arr[i] = arr[j];
                  arr[j] = tmp;
                  i++;   j--;
            } };
      if (left < j)
            quickSort(arr, left, j);
      if (i < right)
            quickSort(arr, i, right);
```

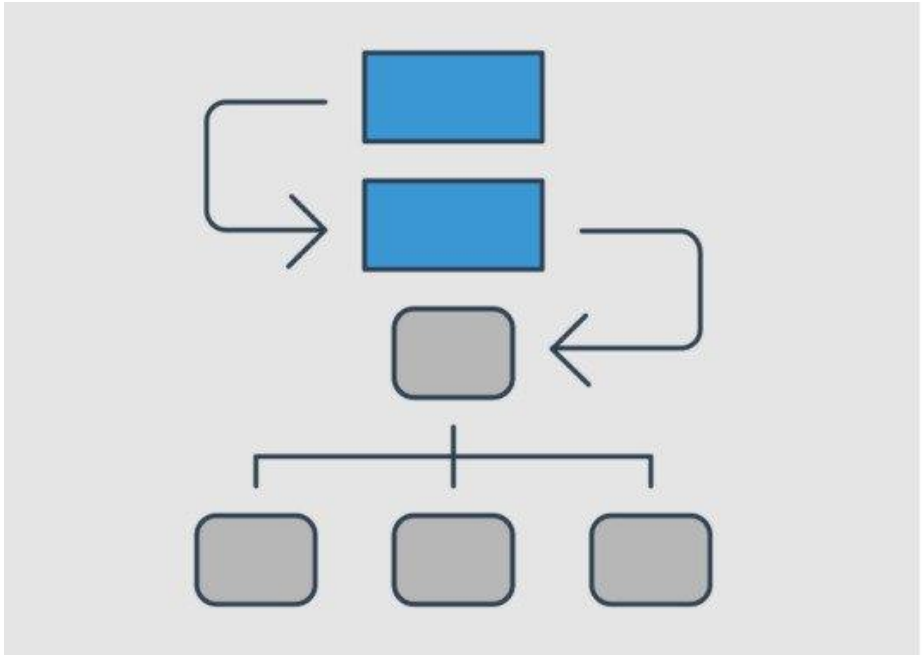


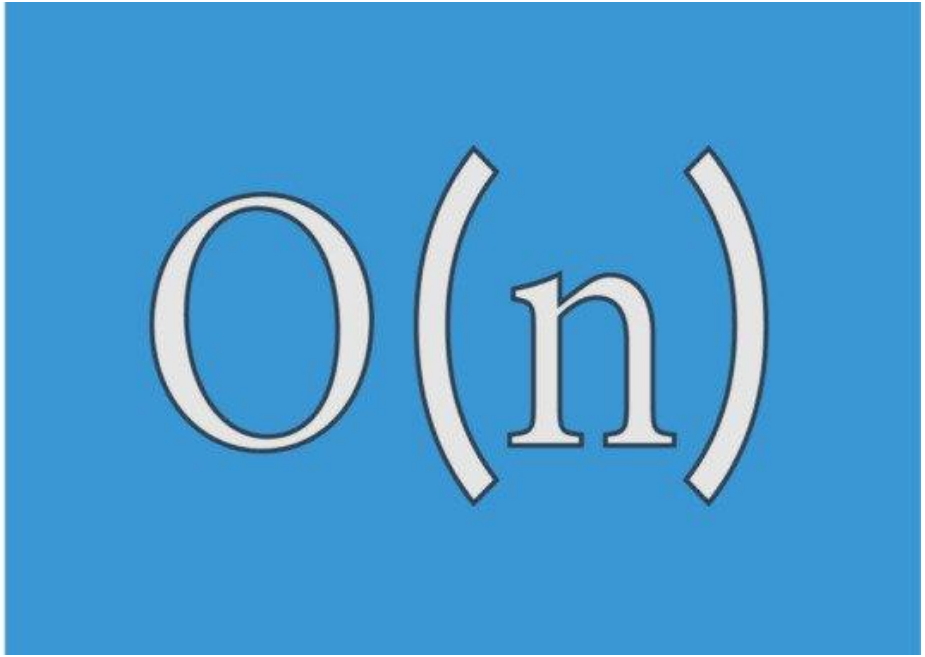
Radix sort

Merge sort



Оцінка ефективності алгоритмвв



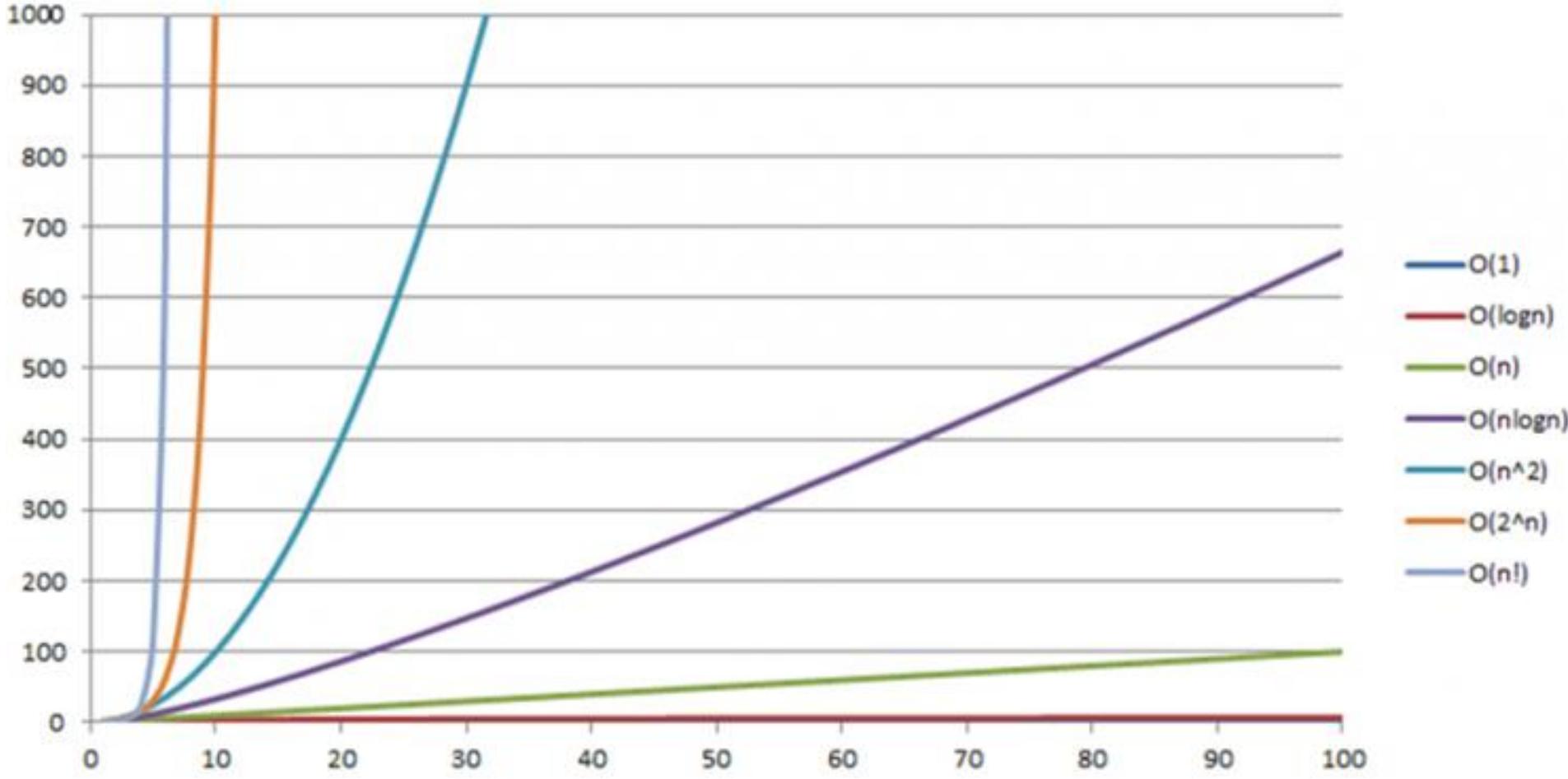


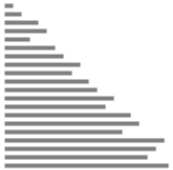
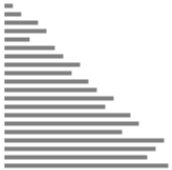
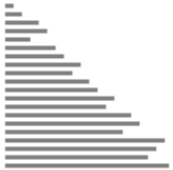
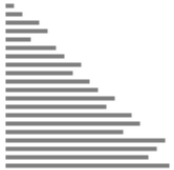
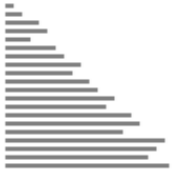
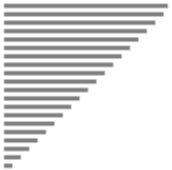
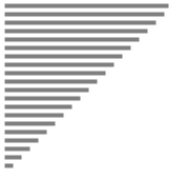
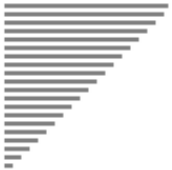
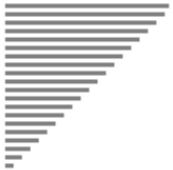
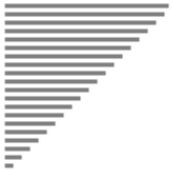
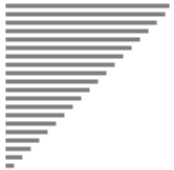
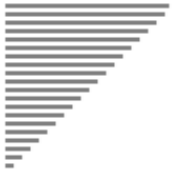
размер / сложность	10	20	30	40	50	60
n	0,00001 сек.	0,00002 сек.	0,00003 сек.	0,00004 сек.	0,00005 сек.	0,00005 сек.
n^2	0,0001 сек.	0,0004 сек.	0,0009 сек.	0,0016 сек.	0,0025 сек.	0,0036 сек.
n^3	0,001 сек.	0,008 сек.	0,027 сек.	0,064 сек.	0,125 сек.	0,216 сек.
n^5	0,1 сек.	3,2 сек.	24,3 сек.	1,7 минут	5,2 минут	13 минут
2^n	0,0001 сек.	1 сек.	17,9 минут	12,7 дней	35,7 веков	366 веков
3^n	0,059 сек.	58 минут	6,5 лет	3855 веков	2×10^8 веков	$1{,}3 \times 10^{13}$ веков

```
#include <iostream>
#include <ctime>

using namespace std;

int main()
{
    unsigned int start_time = clock();
    int i, j, k, count  = 0, n;
    cout << "n = ";
    cin >> n;
    for (i = 0; i < n; ++i)
        for (j = 0; j < n; ++j)
            for (k = 0; k < n; ++k)
                ++count;
    unsigned int end_time = clock();
    cout << count << endl;
    unsigned int search_time = end_time - start_time;
    cout << endl << "Время выполнения программы: " << search_time << " mc" << endl;

    return 0;
}
```

n=10
n=100
n=1000

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)


1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
O(1)
O(logn)
O(n)
O(nlogn)
O(n^2)
O(2^n)
O(n!)

	Insertion	Selection	Bubble	Shell	Merge	Heap	Quick	Quick3
Random								
Nearly Sorted								
Reversed								
Few Unique