

✓

§

§

(hi

(hi

(hi 0:00

[

([https://log2base2.com/?utm\\_src=textcourse&utm\\_target=ltext](https://log2base2.com/?utm_src=textcourse&utm_target=ltext))  
Greedy Approach

# Bubble Sort Algorithm

In bubble sort, each pair of adjacent elements are compared and the elements are swapped if they are not follow the ordering rule.

## Example

Let's take an array of 5 elements.

```
int arr[5] = {50, 25, 5, 20, 10}
```

## Step 1

## Algorithms

	Adjacent Pairs	Compare	swap	After Swap
Sc	50 25 5 20 10	$50 > 25$	yes	25 50 5 20 10
St	25 50 5 20 10	$50 > 5$	yes	25 5 50 20 10
(http	25 5 50 20 10	$50 > 20$	yes	25 5 20 50 10
(http	25 5 20 50 10	$50 > 10$	yes	25 5 20 10 50
D:				sorted
G:	<a href="https://www.log2base2.com">www.log2base2.com</a>			

We can notice that one element has been sorted after the above process.

In general, to sort **N** element using bubble sort, we need to do the same process **N-1** times.

From next iteration onwards, we can skip the sorted elements. i.e. 50

## Step 2

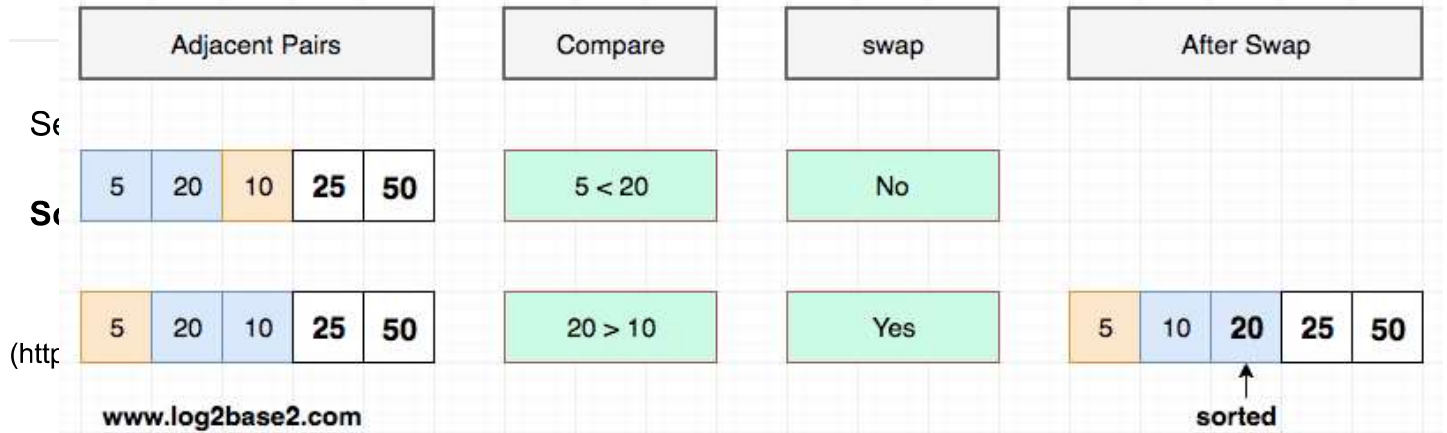
	Adjacent Pairs	Compare	swap	After Swap
	25 5 20 10 50	$25 > 5$	yes	5 25 20 10 50
	5 25 20 10 50	$25 > 20$	yes	5 20 25 10 50
	5 20 25 10 50	$25 > 10$	yes	5 20 10 25 50
	<a href="https://www.log2base2.com">www.log2base2.com</a>			

sorted

(<https://www.log2base2.com/>) Courses

### Step 3

#### Algorithms

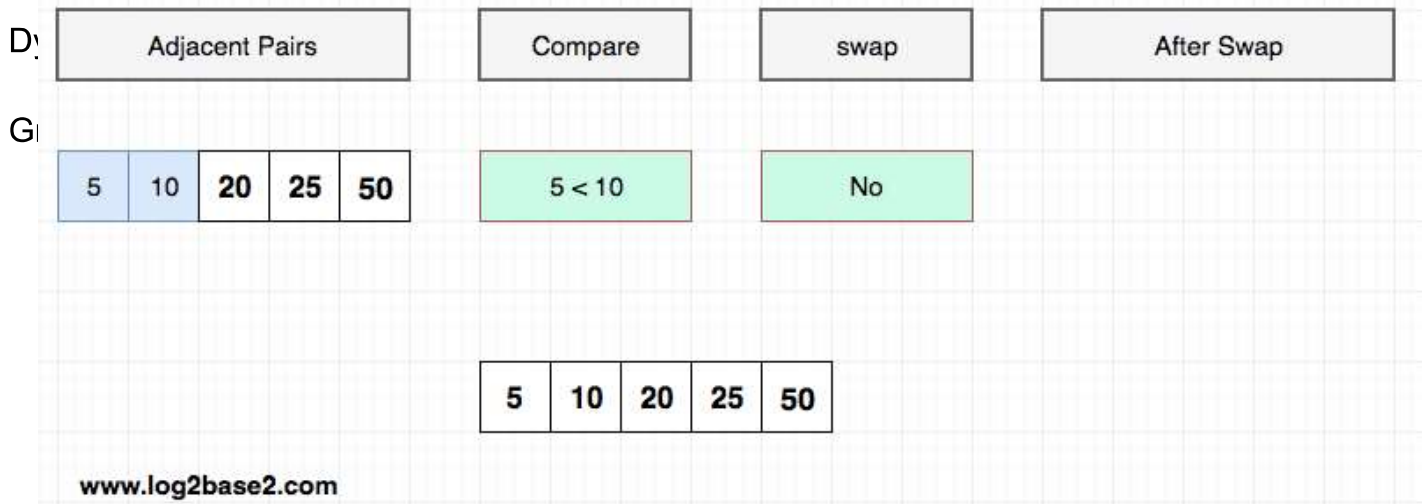


(<https://www.log2base2.com/algorithms/sorting/bubble-sort-algorithm-in-c.html>)

### Step 4

Quick sort

(<https://www.log2base2.com/algorithms/sorting/quick-sort.html>)



## Algorithms

### Bubble Sort Program

Searching

```

/*
 * Program : Bubble sort
 * Language : C
 */

```

Sorting

```

/*
Selection Sort
#include<stdio.h>

```

(https://www.log2base2.com/algorithms/sorting/selection-sort.html)

```

Bubble Sort
void swap(int *x, int *y)

```

(https://www.log2base2.com/algorithms/sorting/bubble-sort-algorithm-in-c.html)

```

int temp = *x;
QuickSort *x = *y;

```

(https://www.log2base2.com/algorithms/sorting/quick-sort.html)

Dynamic Programming

```

void bubbleSort(int arr[])

```

Greedy Approach

```

int i,j;
for(i = 0; i < size-1; i++)
{
    for(j = 0; j < size-1-i; j++)
    {
        if(arr[j] > arr[j+1])
            swap(&arr[j],&arr[j+1]);
    }
}

```

```

int main()
{
    int arr[size] = {50, 25, 5, 20, 10}, i;

    bubbleSort(arr);

    printf("After the bubble sort\n");
    for(i = 0; i < size; i++)
        printf("%d ",arr[i]);

    return 0;
}

```

Run it (https://www.log2base2.com/algorithms/sorting/try-it-bubble-sort-algorithm-in-c-1.html) Courses

## Algorithms

Searching

## Sorting

Selection Sort

(https://www.log2base2.com/algorithms/sorting/selection-sort.html)

Bubble Sort

(https://www.log2base2.com/algorithms/sorting/bubble-sort-algorithm-in-c.html)

## Binary Dollar Question

Quicksort

(https://www.log2base2.com/algorithms/sorting/quick-sort.html)

In the above program, we have used **j < size-1-i** condition in the inner for loop.

Dynamic Programming

for(j = 0; j < size-1-i; j++)

Greedy Approach

What change it will bring? Can you guess?

## Optimizing the Bubble Sorting Algorithm

We can optimize the bubble sort algorithm further.

Let's take a sorted array.

### Example

```
int arr[5] = {5, 10, 15}
```

Bubble sort operation will be like below.

## Algorithms

	Adjacent Pairs	Compare	swap	After Swap
Sc				
St	5 10 15 20	$5 < 10$	No	
(http	5 10 15 20	$10 < 15$	No	
(http	5 10 15 20	$15 < 20$	No	

(<https://www.log2base2.com/algorithms/sorting/quick-sort.html>)

## Dynamic Programming

## Greedy Approach

## Observation

We can notice that the sorted array doesn't need any swap operation.

So, if no swap operation takes place, we can ensure that the array is sorted and we can break the process.

It will improve the efficiency of the bubble sort algorithm.

## Example

## Algorithms

/\*

\* Program : Improved Bubble Sort

\* Language : C

Searching

#include<stdio.h>

Sorting #define size 5

Selection Sort  
**void swap(int \*x, int \*y)**

{

(<https://www.log2base2.com/algorithms/sorting/selection-sort.html>)  
**int temp = \*x;**

Bubble Sort  
**\*y = \*x;**

(<https://www.log2base2.com/algorithms/sorting/bubble-sort-algorithm-in-c.html>)  
**\*y = temp;**

}

Quicksort

(<https://www.log2base2.com/algorithms/sorting/quick-sort.html>)  
**void bubbleSort(int arr[])**  
{

Dynamic Programming;

Greedy Approach  
**for(i = 0; i < size-1; i++)**

{

**flag = 0;**

**//if any swap operation takes place, set flag as 1**

**for(j = 0; j < size-1-i; j++)**

{

**if(arr[j] > arr[j+1])**

{

**swap(&arr[j], &arr[j+1]);**

**flag = 1;**

}

}

**//if flag remains 0, break the process**

**if(flag == 0)**

**break;**

}

}

**int main()**

{

**int arr[size] = {50, 25, 5, 20, 10}, i;**

**bubbleSort(arr);**

**printf("After the bubble sort\n");**

**for(i = 0; i < size; i++)**

**printf("%d ", arr[i]);**

**Algorithms**  
return 0;  
}

Searching  
Run it (<https://www.log2base2.com/algorithms/sorting/try-it-bubble-sort-algorithm-in-c.html>)

**Sorting**

Selection Sort

(<https://www.log2base2.com/algorithms/sorting/selection-sort.html>)

PAGE -- (<https://www.log2base2.com/algorithms/sorting/selection-sort.html>)  
(<https://www.log2base2.com/algorithms/sorting/bubble-sort-algorithm-in-c.html>)

Quicksort PAGE ++ (<https://www.log2base2.com/algorithms/sorting/quick-sort.html>)  
(<https://www.log2base2.com/algorithms/sorting/quick-sort.html>)

Dynamic Programming

Greedy Approach

YouTube (<https://www.youtube.com/c/log2base2>) / Facebook (<https://www.facebook.com/log2base2>) /  
Twitter (<https://twitter.com/log2base2>) / Instagram (<https://www.instagram.com/log2base2/>)