



# Bubblesort

Sorting Algorithms

A card for you to write:

## Bubblesort Characteristics

- Long series of swaps up the array on each pass
- In the general case – slow –  $O(n^2)$
- In its best case – almost sorted – fast – close to  $O(n)$
- First sorting algorithm to be coded in 1955.

A card for you to write:

## Bubblesort Swapping Tradeoff

(-) In the general case, bubble sort is one of the slowest sorts. It's method of swapping one-by-one up the array takes forever.

(+) If the array is almost sorted, it is one of the only sorts that can stop as soon as it is sorted. The slow swapping method ALSO checks if the array is sorted and allows the code to stop early.

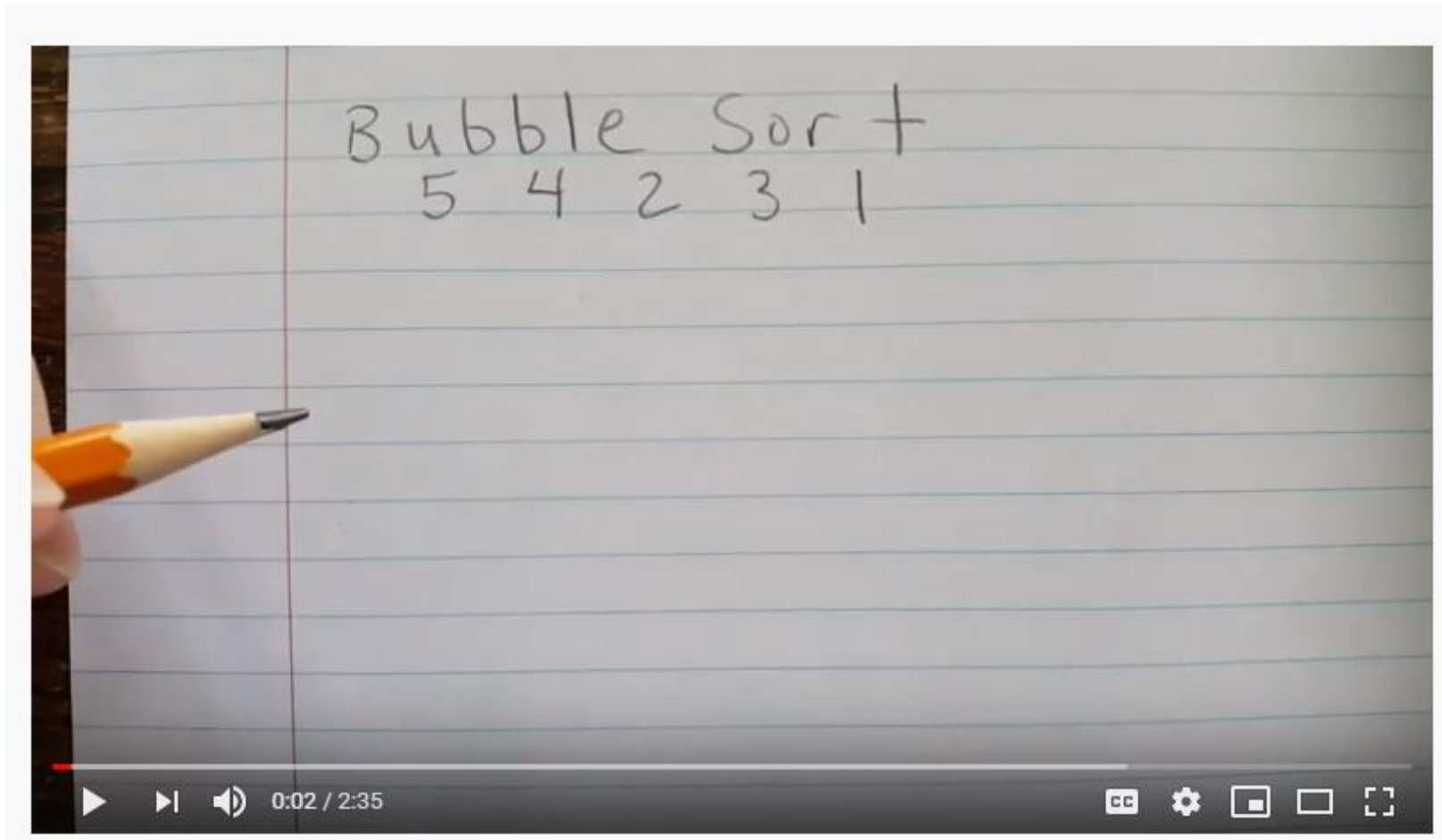
A card for you to write:

## Bubblesort Easy to Code Tradeoff

(+) Bubblesort is simple to understand and is simple to code. This is why it was the first sorting algorithm to be coded.

(-) It's simplicity means it isn't very clever or efficient. That means it is SLOW.

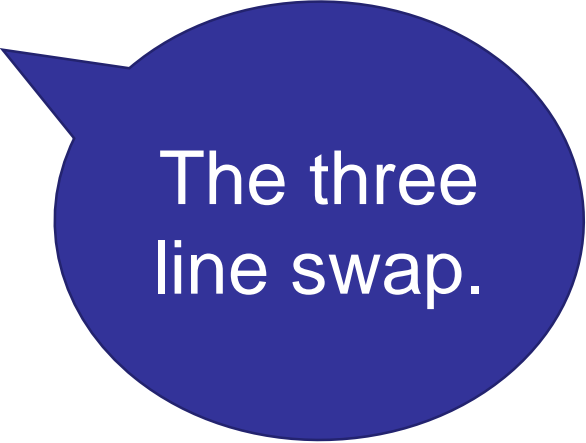
I posted a video to YouTube tracing Bubble Sort:



<https://www.youtube.com/watch?v=W2rIMC7HIW8>

# The Bubble Sort Code


```
int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
    for (int j = 0 ; j < n - 1 - i ; j++)
    {
        if (a [j + 1] < a [j])
        {
            int temp = a [j];
            a [j] = a [j + 1];
            a [j + 1] = temp;
        }
    }
}
```



The three  
line swap.

# The Bubble Sort Code

```
int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}
```



If the two  
neighbours  
are out of  
order




Swap  
them!

# The Bubble Sort Code

```
int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}
```



Swapping  
out of  
order  
pairs!



Go down  
the array  
once!




# The Bubble Sort Code

```
int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}
```



Swapping  
out of  
order  
pairs!



Go down  
the array  
once!





```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4



```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4
2	3	1	4

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4
2	3	1	4
2	1	3	4

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4
2	3	1	4
2	1	3	4

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4
2	3	1	4
2	1	3	4
1	2	3	4

```

int n = a.length;
for (int i = 0 ; i < n - 1 ; i++)
{
  for (int j = 0 ; j < n - 1 - i ; j++)
  {
    if (a [j + 1] < a [j])
    {
      int temp = a [j];
      a [j] = a [j + 1];
      a [j + 1] = temp;
    }
  }
}

```

[0]	[1]	[2]	[3]
4	3	2	1
3	4	2	1
3	2	4	1
3	2	1	4
2	3	1	4
2	1	3	4
1	2	3	4









[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	5	7	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8
2	1	3	5	7	8



[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8
2	1	3	5	7	8
1	2	3	5	7	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8
2	1	3	5	7	8
1	2	3	5	7	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8
2	1	3	5	7	8
1	2	3	5	7	8

[0]	[1]	[2]	[3]	[4]	[5]
8	2	3	1	7	5
2	8	3	1	7	5
2	3	8	1	7	5
2	3	1	8	7	5
2	3	1	7	8	5
2	3	1	7	5	8
2	1	3	7	5	8
2	1	3	5	7	8
1	2	3	5	7	8



In a **bubble sort**,  
the “heaviest”  
item sinks to  
the bottom of the  
list while the  
“lightest” floats up  
to the top

beef stock



chicken stock



laughing stock

