Sorting Algorithms

Paolo Bettelini

Contents

1 Bubble Sort

Given a list of numbers a.

We check each pair of adjacent numbers in the list (a_i, a_{i+1}).
If $a_i > a_{i+1}$, we swap a_i and a_{i+1}.
We repeat this process until we check every tuple without performing the swap operation.

	Best-case	**Average-case**	**Worst-case**
comparison	$O(n)$	$O(n^2)$	$O(n^2)$
swap	$O(1)$	$O(n^2)$	$O(n^2)$

Algorithm 1 Bubble Sort

```
swapped ← false
do
    swapped ← false
    for i ← 0 to length(a) − 1 do
        if a_i > a_{i+1} then
            swapped ← true
            swap a_i and a_{i+1}
while swapped
```

2 Selection Sort

Given a list of numbers a.

We find the minimum value in the list starting from an offset of 0.
We swap the minimum value and the value at the offset.
We increment the offset by 1 and repeat this process while the offset is less than the length of the list.

	Best-case	**Average-case**	**Worst-case**
comparison	$O(n^2)$	$O(n^2)$	$O(n^2)$
swap	$O(1)$	$O(n)$	$O(n)$

Algorithm 2 Selection Sort

```
for i ← 0 to length(a) − 1 do
    min ← i
    for j ← i + 1 to length(a) do
        if a_j < a_min then
            min ← j
    swap a_i and a_min
```

3 Gnome Sort

Given a list of numbers a.

We start at the beginning of the list. Until we reach the end of the list, we check each tuple.
If the tuple is sorted, we increment our position by 1. If the tuple is not sorted, we swap the adjacent numbers and decrement out position by 1.
If we are moving left and our position is 0, we go right instead.

Algorithm 3 Gnome Sort

```
i ← 0
while i < length(a) − 1 do
    if a_i > a_{i+1} then
        swap a_i and a_{i+1}
        if i = 0 then
            i ← i + 1
        else
            i ← i − 1
    else
        i ← i + 1
```