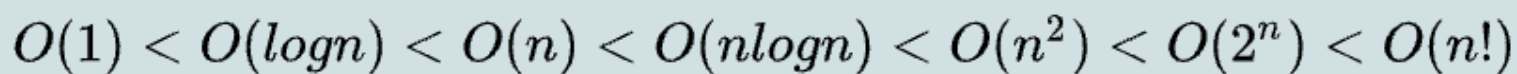


Algorithm Complexity II

Sorting Algorithms

ivanovitch.silva@ufrn.br
@ivanovitchm





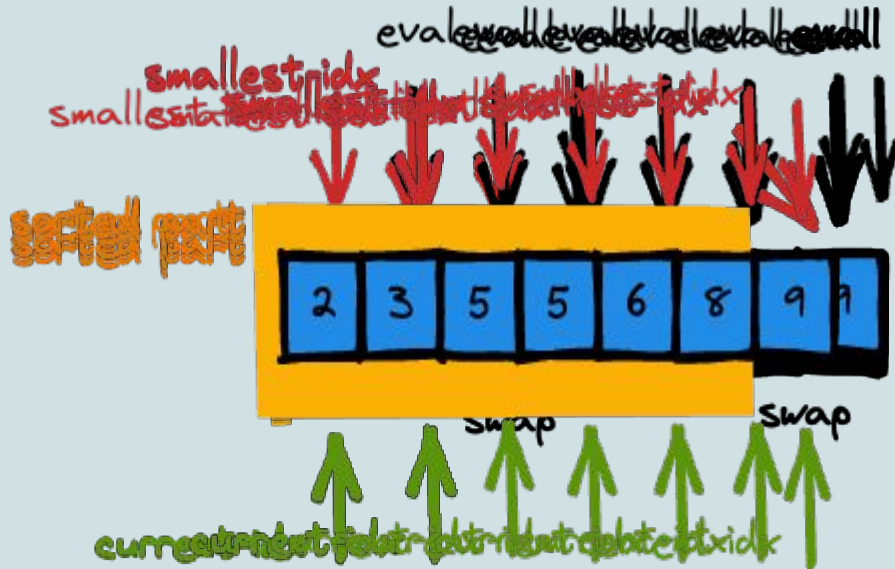
Sorting Algorithm	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	Worst Case
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$
Counting sort	$O(n+k)$	$O(n+k)$	$O(n+k)$	$O(n+k)$
Radix sort	$O(d(n+b))$	$O(d(n+b))$	$O(d(n+b))$	$O(n+b)$
Bucket sort	$O(n+k)$	$O(n+k)$	$O(n^2)$	$O(n+k)$
Shell sort	$O(n \log n)$	$O(n \log n)^2$	$O(n \log n)^2$	$O(1)$

Sorting Algorithm	Comparison Based	Stable	Recursive	In-place	Adaptive	Online
Insertion sort	✓	✓	✗	✓	✓	✓
Bubble sort	✓	✓	✗	✓	✓	✗
Selection sort	✓	✗	✗	✓	✗	✗
Merge sort	✓	✓	✓	✗	✗	✗
Heapsort	✓	✗	✗	✓	✗	✗
Quicksort	✓	✗	✓	✓	✗	✗
Counting sort	✗	✓	✗	✗	✗	✗
Radix sort	✗	✓	✗	✗	✗	✗
Bucket sort	✗	✓	✗	✗	✗	✗
Shell sort	✓	✗	✗	✓	✓	✗

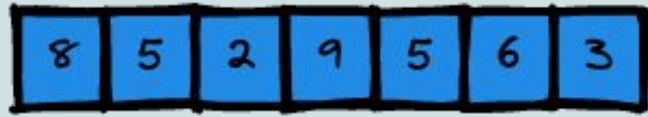
Selection Sort



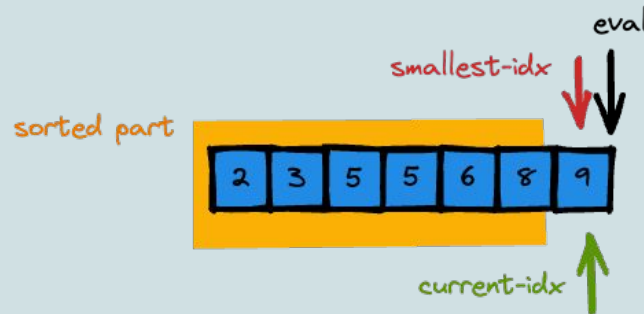
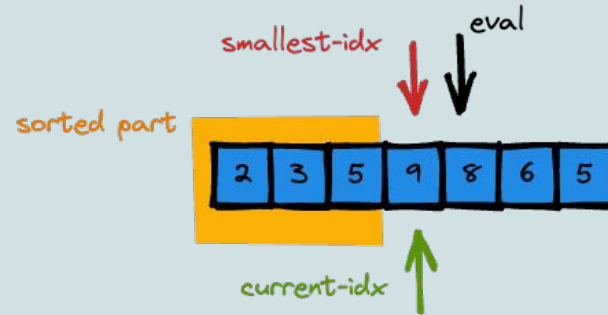
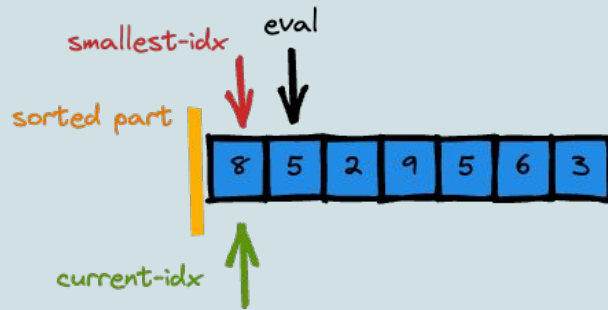
sorted part
unsorted part
swap



Selection Sort



sorted part
unsorted part
swap

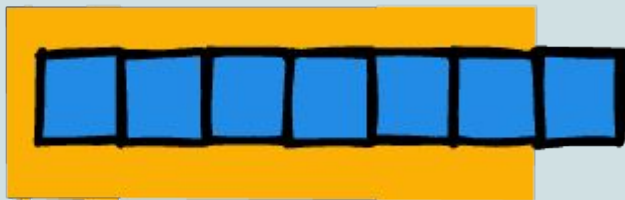


Selection Sort

Calc of complexity



sorted part



$$T(n) = N$$

$$T(n) = N + (N - 1)$$

$$T(n) = N + (N - 1) + \dots + 1$$

$$S_n = \frac{N(N+1)}{2}$$

$$O(N^2)$$

```
1: def selectionSort(array):
2:     currentIdx = 0
3:     while currentIdx < len(array) - 1:
4:         smallestIdx = currentIdx
5:         for i in range(currentIdx + 1, len(array)):
6:             if array[smallestIdx] > array[i]:
7:                 smallestIdx = i
8:         swap(currentIdx, smallestIdx, array)
9:         currentIdx += 1
10:    return array
11:
12: def swap(i, j, array):
13:     array[i], array[j] = array[j], array[i]
```