Cheatsheet - Comparison and Non-Comparison Sorting Algorithms

Fabio Lama – fabio.lama@pm.me

1. About

This cheatsheet provides an overview of some common sorting algorithms.

2. Comparison Sort Overview

Name	Worst case complexity	Best case complexity
Bubble	$\Theta(N^2)$	$\Theta(N)$
Insertion	$\Theta(N^2)$	$\Theta(N)$
Selection	$\Theta(N^2)$	$\Theta(N^2)$
Quicksort	$\Theta(N^2)$	$\Theta(N \times \log N)$
Mergesort	$\Theta(N \times \log N)$	$\Theta(N \times \log N)$

Because comparison sorts must compare pairs of elements, **they cannot** run faster than $N \times \log N$.

3. Bubble Sort

3.1. Pseudocode

```
function BubbleSort($A$, $N$)
    swapped $= true$
    while (swapped) do
        swapped $= false$
        for $0 \le i < N - 1$ do
            if $(A[i] > A[i+1])$ then
                swap($A[i]$, $A[i+1]$)
                swapped $= true$
            end if
        end for
        $N = N - 1$
    end while
    return $A$
end function
```

3.2. Time Complexity

The **best case** for bubble sort is:

$$T(N) = C_0 \times N + C_1$$

Additionally:

- $T(N)$ is $O(N)$, $O(N^2)$ and $O(N^3)$, etc.
- $T(N)$ is $\Omega(N)$, $\Omega(\log N)$ and $\Omega(1)$, etc.
- $T(N)$ is $\Theta(N)$

The **worst case** for bubble sort is:

$$T(N) = C_0 \times N^2 + C_1 \times N + C_2.$$

Additionally:

- $T(N)$ is $O(N^2)$ and $O(N^3)$, etc.
- $T(N)$ is $\Omega(N^2)$, $\Omega(\log N)$ and $\Omega(1)$, etc.
- $T(N)$ is $\Theta(N^2)$

4. Insertion Sort

4.1. Pseudocode

```
function $\text{InsertionSort}(A, N)$
    for $1 \le j \le N - 1$ do
        $\texttt{ins} = A[j]$
        $i = j - 1$
        while $(i \ge 0$ and $\texttt{ins} < A[i])$ do
            $A[i + 1] = A[i]$
            $i = i - 1$
        end while
        $A[i + 1] = \texttt{ins}$
    end for
end function
```

5. Selection Sort

5.1. Pseudocode

```
function $\text{SelectionSort}(A, N)$
    for $0 \le i < N - 1$ do
        $\texttt{min} = \text{pos\_min}(A, i, N - 1)$
        $\texttt{swap}(A[i], A[\texttt{min}])$
    end for
end function
```

The function $\text{pos\_min}(A, a, b)$ returns the position of the minimum value between positions a and b (both inclusive) in array A.

6. Quicksort

6.1. Pseudocode

```
function $\text{Quicksort}(A, \texttt{low}, \texttt{high})$
    if $\texttt{low} < \texttt{high}$ then
        $p = \text{partition}(A, \texttt{low}, \texttt{high})$
        $\text{Quicksort}(A, \texttt{low}, p - 1)$
        $\text{Quicksort}(A, p + 1, \texttt{high})$
    end if
end function
```

TODO: This needs further explanation.

The function $\text{partition}(A, \texttt{low}, \texttt{high})$ selects a number (the pivot), moves all numbers lower than the pivot to the left part of the array and moves the privot to its final position.

7. Mergesort

7.1. Pseudocode

```
function $\text{MergeSort}(A, \text{int } l, \text{int } h)$
    if $(l < h)$
        $\texttt{mid} = l + \text{floor}((h - 1)/2)$
        $\text{MergeSort}(A, l, \texttt{mid})$
        $\text{MergeSort}(A, \texttt{mid} + 1, h)$
        $\text{Merge}(A, l, \texttt{mid}, h)$]
    end if
end function
```

The function Merge creates two arrays of both halves (left and right) and then merges them to produce a single, sorted array.

Last updated 2024-05-21 20:46:28 UTC