Algortihms and datastructures

Kristoffer Klokker

2022

Contents

1 Algorithm analyse

An algorithm must stop for all input and give the correct output.
An algorithms quality is determined from:

- Speed
- Memory used
- Complexity of implmentation
- Other use cases like stability

The measure speed and memory the worst case of the algorithm is always used due to its simplicity in calculations and gurantee.
The measurement is often used as a function of input size using the big O notation, which says for a given input with n size it will take n^2 time to run for example.
To analyze an algorithm a model is used most often the ram model.

1.1 Ram-model

The ram model is a very simple interpretation of a computer.
The model consist of a CPU, memory and basic operation (add, sub, mult, shift, move, jump)
The time of the algorithm is then measured in amount of basic operations done.
The memory is determined as the amount of memory cell used.

2 Sorting algorithms

Sorting algorithms are used to sort an array of items in an ascending order.

2.1 Insertionsort

One of the most simple sorting algorithms.
Works by going through the list from index 1 and moves moves every entry before the element 1 up until the element is to the right of an element smaller than the element.
This will therefore have a run time of $O(n^2)$ due to the scenario where the

array is in decending order where it will moves $n, n-1, n-2, .., 1$ elements which is $\sum_{j=1}^{n} j = \frac{(n+1)n}{2} \leq \frac{2n^2}{2} = n^2$

```
1: Insertion-Sort(A, n)
2: for j = 1 don
3:     key = A[j]
4:     i = j − 1
5:     while i ≥ 0 and A[i] > key do
6:         a[i + 1] = A[i]
7:         i = i − 1
8:     end while
9:     A[i + 1] = key
10: end for
```

2.2 Selectionsort

Selectionsort is done by taking all elements and then searching for the smallest element and inserting it into the list.
This will result in the same run time $O(n^2)$ due to the same reasoning as insertionsort.

```
1: Selection-Sort(A, n)
2: Array output = newArray(n)
3: i = 0
4: while i < n do
5:     output[i] = A[i]
6:     A[i].remove()
7: end while
```