Umeå University
Institution för Datavetenskap

Datavetenskapens byggstenar 7.5 p
DV160HT15

OU4 Analysis of Complexity

Submitted 2015-12-25
Author: Lorenz Gerber (dv15lgr@cs.umu.se)
Instructor: Lena Kallin Westin / Johan Eliasson

Contents

1 Introduction

The aim with this laboration was to apply experimental and asymptotic complexity analysis of algorithms.

What is complexity analysis. Experimental, asymptotic. What is big O notation, what does it mean.

[**?**, pp. 117 – 132].

2 Material and Methods

2.1 Experimental Complexity Analysis

Describe experiment, describe the rules.

2.2 Asymptotic Complexity Analysis

Describe what was given and the rules to analyse.

Program 1 This is the shit

```
Algorithm bubbleSort(numElements, list[])
input:  numElements, the number of elements in the list
        list, a list of numbers to be sorted
output: the sorted list

1:  done <- false
2:  n <- 0
3:   while (n < numElements) and (done = false)
4:      done <- true
5:      for m <- (numElements -1) downto n
6:          if list[m] < list[m - 1] then
7:              tmp <- list[m]
8:              list[m] <- list[m - 1]
9:              list[m - 1] <- tmp
10:             done <- false
11:    n <- n + 1
12: return list
```

3 Results

3.1 Experimental Complexity Analysis

Show formulas, C, n0

3.2 Asymptotic Complexity Analysis

Worst Case

```
1:  1 * [done <- false] +
```

```
2:  1 * [n <- 0] +
3:  (numElements + 1) * ([n < numElements] + [done = false]) +
4:  numElements * [done <- true] +
5:  ((numElements(numElements+1)) / 2) * (1 * [m <-] + 1 * [numElements - 1]
6:  1 * [> n] + $\\
7:  1 * [list[m]] + 1* [m-1] + 1 * [list[]] + 1 * [<] +
8:  1 * [m-1] + 1 * [list[]] + 1 * [<-] +
9:  1 * [tmp] + 1 * [m-1] + 1 * [list[] <-] +
10: 1 * [done <- false] ) +
11: 1 * [n + 1] + 1 * [n <-] +
12: 1 * return

set numElements = x

1 + 1 + (x + 1) * 2 + x + (x(x+1) / 2) * ( 1 + 1 + 1 + 1 + 1 + 1
+ 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1) + 1 + 1 + 1
2 + 2x + 2 + x + (1 / 2x^2} + {1 / 2x) * 14 + 3
3x + 6 + 7x^2 + 7x
7x^2 + 10x + 7
```

Best case

```
1: 1 * [done <- false] +
2: 1 * [n <- 0] +
3: 2 * ([n < numElements] + [done = false]) +
4: 1 * [done <- true] +
5: (numElements - 1) * (1 * [m <-] + 1 * [numElements - 1] + 1 * [> n])

11: 1 * [n+1] + 1 * [<-]
12: 1 * [return]

set numElements = x

1 + 1 + 2 * 2 + 1 + (x-1) * (1 + 1 + 1) + 1 + 1 + 1
10 + 3x - 3
3x + 7
```

show plot

4 Discussion

4.1 Experimental Complexity Analysis

4.2 Asymptotic Complexity Analysis