Program Design & Data Structures (Course 1DL201)
Uppsala University Autumn 2017/Spring 2018
Homework Assignment 2: Analysis of Algorithms

Prepared by Dave Clarke

Lab:	Friday, 8 December, 2017
Submission Deadline:	18:00, Friday, 15 December, 2017
Lesson:	Friday, 19 January, 2018
Resubmission Deadline:	18:00, Friday, 26 January, 2018

Goal

The goal of this assignment is to promote and assess your understanding of the key concepts of algorithm analysis: growth of functions, converting code to recurrences, and solving recurrences (by hand).

Please submit your answers to the following questions in a report in PDF format. We strongly prefer computer written solutions over handwritten ones. You can generate nice maths using LaTeX.

Please do not just provide equations as answers. Explain how you obtained your answers.

1 Real World Algorithm

Consider the following algorithm which is executed by people. The objective of the algorithm is to sort a group of people by age. The people all stand in a line side-by-side, facing the same direction, and alternate the following steps:

- people in even-numbered positions check with the person on their right to see who is younger. If the person on the right is younger, they swap.
- people in odd-numbered positions check with the person on their right to see who is younger. If the person on the right is younger, they swap.

(Essentially, people alternate between checking left and right, and the youngest person moves to the left.)

- Determine the number of steps **per person** required to sort n people.

- Determine the number of steps **in total** required to sort n people.
- Discuss briefly how this human computation model compares to the computations performed by a computer.

2 Growth of Functions

Recall the definition of $\Theta(g(n))$:

> $f(n) = \Theta(g(n))$ if and only if there exists $n_0 \geq 0$ and $c_1, c_2 > 0$ such that for all $n \geq n_0$ the following holds $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$.

Find the **precise** bound (a function $g(n)$ such that $t(n) = \Theta(g(n))$) of the function:

$$t(n) = 10 + 15n + 16n^2 \sin(n) + n^3.$$

Prove that your answer is correct by finding n_0, c_1, and c_2 satisfying the definition above.

3 Recurrences

Compute the first 10 values of the following recurrence (starting from $n = 0$):

$$\begin{aligned} f(0) &= 1 \\ f(n) &= 2f(n-1) + 2n + 12, \qquad n > 0 \end{aligned}$$

4 From Code to Recurrence

For the following programs, give a recurrence indicating the run-time based on the input. Indicate whether your recucurrence is defined in terms of the size of the input, the value of the input, or something else. You do **not** need to give the closed form of the recurrence.

1. The following function appends one list onto the end of another. Find $T_{\texttt{append}}$ that describes the runime cost of `append n m`.

```
append :: [a] -> [a] -> [a]
append []     ys = ys
append (x:xs) ys = x : append xs ys
```

2. Let `p` be an arbitrary (but fixed) integer in the following function. What is the recurrence $T_{\texttt{split}}(n)$ that describes the runtime cost of `split p xs`?

```
split :: Int -> [Int] -> ([Int], [Int])
split _ [] = ([], [])
split p (x:xs) =
  let
    (lows, highs) = split p xs
  in
    if x < p
      then (x:lows, highs)
      else (lows, x:highs)
```

3. The function `concat` takes a list of lists and appends them together to form one large list. What is the recurrence $T_{\texttt{concat}}(n)$ that describes the runtime cost of `concat xss`?

```
concat :: [[a]] -> [a]
concat [] = []
concat (x:xs) = append x (concat xs)
```

Hint: Think carefully about what n is and what it represents.

5 Solving Recurrences

For the following questions you do **not** need to prove your answer, e.g., using induction, you just need to apply the appropriate method to produce a reasonable guess for the closed form.

1. Use the substitution method to obtain a closed form for the following recurrence:

$$\begin{aligned} f(0) &= 5 \\ f(n) &= f(n-1) + n + 2, \qquad n > 0 \end{aligned}$$

Hint: $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$ — you do **not** need to prove this fact.

2. Use the expansion method to obtain a closed form for the following recurrence:

$$g(0) = \begin{cases} 2 & \text{if } n = 0 \\ 2g(n-1) + 3 & \text{if } n > 0 \end{cases}$$

Hint: keep track of 2s and 3s.

Grading

Your answers are graded on a U/K/4/5 scale for correctness of results and explicitness of reasoning. Each of the five questions will be graded separately.

If your answer for *any* of the questions shows little or no evidence of engagement in the question, you get a U grade for the *entire* homework assignment. Thus, you need to get a grade of K (or better) on *all* questions to pass the assignment.

If your answer shows some relevant engagement in the question, e.g., choice of a partially correct strategy or some partially correct reasoning with trial and error, you get (at least) a K for your answer.

If your answer is mostly correct and your reasoning is comprehensible, with very few major omissions or errors, you get a 4 for your answer.

If your answer is correct and your reasoning is explicit, with at most minor omissions or oversights, you get a 5 for your answer.

Lessons and Resubmission

A K grade on *any* question means that you are required to attend the **lesson** discussing the assignment and to subsequently resubmit those answers for which you received a K.

Your resubmitted answers will be re-graded. Grades of K will improve to 3 if you provide a mostly correct solution (cf. the criteria for grade 4 above). You cannot get a better grade than 3 for a question where you originally got a K.

Final Grade

You pass the assignment if (and only if) your grade is at least 3 on *all* questions after resubmission. In this case, your final grade for the assignment is the arithmetic mean of the five per-question grades.

Modalities

- The assignment will be conducted in groups of 2. Groups have been assigned via the Student Portal. *If you cannot find your partner by Wednesday, December 6, please contact Andreas Löscher* `<andreas.loscher@it.uu.se>` *to assign you a new partner—if possible. Note that this deadline is strict! You are not guaranteed a partner if you ask later than this.*
- Answers must be submitted via the Student Portal. Only one report per group shall be submitted. Ensure that **both** group members' names appear on the report.

We assume that by submitting a solution you are certifying that it is solely the work of your group, except where explicitly attributed otherwise. We reserve the right to use plagiarism detection tools and point out that they are extremely powerful. You have already been warned about the consequences of cheating and plagiarism.

Good luck!