Program Design & Data Structures (Course 1DL201)
Uppsala University Autumn 2014/Spring 2015
Homework Assignment 2: Analysis of Algorithms

Prepared by Tjark Weber

Lab: Friday, 5 December, 2014
Submission Deadline: 18:00, Friday, 12 December, 2014

Lesson: TBA (in 2015)
Resubmission Deadline: TBA (in 2015)

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, solving recurrences (by hand), and the master theorem.

Please submit your answers to the following questions in a report in PDF format.

1 Growth of Functions

Consider the sequence $F(n)$ of Fibonacci numbers, given by the following recurrence:

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

1. Prove that $F(n) = O(2^n)$. (Hint: use complete induction.)
2. Prove that $F(n) = \Omega(2^{n/2})$. (Hint: use complete induction.)

2 From Code to Recurrences

1. Consider the following implementation of list concatenation:

```
[]     ++ ys = ys
(x:xs) ++ ys = x : (xs++ys)
```

Let `ys` be an arbitrary, but fixed list. What is the recurrence that describes the run-time cost $T_{\texttt{++}}(n)$ for `xs ++ ys`? Here, $n \geq 0$ is the length of `xs`. (You do *not* need to give the closed form of the recurrence.)

2. Consider the following implementation of a function `fib` to compute the Fibonacci numbers:

```
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)
```

Let $n \geq 0$. What is the recurrence that describes the run-time cost $T_{\texttt{fib}}(n)$ for `fib n`? (You do *not* need to give the closed form of the recurrence.)

3. Consider the following function `power` that computes powers of integers:

```
power b 0 = 1
power b n | even n = power b (n `div` 2) * power b (n `div` 2)
          | otherwise = b * power b (n `div` 2) * power b (n `div` 2)
```

Assume that `even n` and `n `div` 2` can be computed in $\Theta(1)$.

Let b be an arbitrary, but fixed integer. Let $n \geq 0$. What is the recurrence that describes the run-time cost $T_{\texttt{power}}(n)$ for `power b n`? (You do *not* need to give the closed form of the recurrence.)

3 Solving Recurrences

1. Use the expansion method to obtain a closed form for the following recurrence. (You do *not* need to prove your answer, e.g., by induction.)

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

2. Use the substitution method to obtain a closed form for the following recurrence. (You do *not* need to prove your answer, e.g., by induction.)

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

(Hint: $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$. You do *not* need to prove this fact.)

4 The Master Theorem

Either use the master theorem to obtain a tight asymptotic bound for the following recurrence, or explain why the master theorem cannot be used here.

$$T(n) := 3 \cdot T(n/2) + n^2 \cdot \log(\log n)$$

5 Example: Quicksort

Recall the quicksort algorithm, a sorting algorithm that was developed by Tony Hoare in 1960. In Haskell, the algorithm can be implemented as follows:

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

quicksort [] = []
quicksort (x:xs) =
  let
    (lows, highs) = partition x xs
  in
    quicksort lows ++ x : quicksort highs
```

1. Let p be an arbitrary, but fixed integer. What is the recurrence that describes the run-time cost $T_{\texttt{partition}}(n)$ for `partition p xs`? Here, $n \geq 0$ is the length of `xs`.

2. Give a closed form of $T_{\texttt{partition}}(n)$. Briefly explain how you obtained your answer. (However, you do *not* need to prove your answer, e.g., by induction.)

3. What is the recurrence that describes the run-time cost $T_{\texttt{quicksort}}(n)$ for `quicksort xs`? Here, $n \geq 0$ is the length of `xs`. (You do *not* need to give the closed form of the recurrence.)

   Consider two different cases:

   (a) Assume that the lists `lows` and `highs` that are obtained from (recursive) calls to `partition` always have equal length.

   (b) Assume that the list `highs` that is obtained from (recursive) calls to `partition` is always empty (i.e., all list elements are smaller than the pivot).

Answer the question seperately for each case.

(Hint: To simplify the recurrence, remember your analysis of $T_{\texttt{++}}(n)$ from Exercise 2.1. What is the closed form of $T_{\texttt{++}}(n)$?)

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 until Tuesday, December 2, please contact Andreas Löscher* `<andreas.loscher@it.uu.se>` *to assign you a new partner—if possible.*
- 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!