CSE100 Algorithm Design and Analysis — Homework set #1

Spring semester 2012 — Miguel Á. Carreira-Perpiñán

Total possible marks: 100. Homeworks must be solved individually. This set covers chapters 1–4 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al.

Exercise 1: asymptotic behavior (20 points).

1. (10 points) Assume you have two computers, C_A and C_B capable of performing 10^6 and 10^8 operations per second, respectively. Both computers run a set of algorithms whose precise complexities $f(n)$ are given below. Determine the size n^* of the biggest input that can be processed in 1 second for each computer, as in the example.

$f(n)$	n^* for C_A	n^* for C_B
$\sqrt{n}$	10^{12}	10^{16}
$14n + 4$		
$2 + \lg(n^{10})$		
$-\sqrt{n}\log_3 n + n$		
$10n^{\lg 2} - 1$		
$n!$		
$(2n)^{\lceil n/5+1 \rceil}$		

   The precise complexity tells you how many operations are performed to solve an instance of size n. Assume each operation takes the same time and that the input sizes are natural numbers $1, 2, 3, \dots$

2. (10 points) Prove formally that $f(n) = an^3 + bn + c$ where $a > 0$ is $\Theta(n^3)$. *Hint*: find values for the constants c_1, c_2, n_0 in the definition of $\Theta(\cdot)$ and show it holds.

Exercise 2: running time (38 points). Consider the following function, where A is an array containing integers, u and v are integers with $u < v$, and the initial call is FCN-X$(A, u, v, 1, A.\mathit{length})$.

```
FCN-X(A, u, v, p, r)
if p > r
    return 0
if p == r
    if A[p] > v
        return 1
    elseif A[p] < u
        return −1
    else
        return 0
q = ⌊p + (r − p)/2⌋
return (FCN-X(A, u, v, p, q) + FCN-X(A, u, v, q + 1, r))
```

Answer the following questions concisely:

1. (5 points) What does the function do?

2. (10 points) Give a recurrence for its running time $T(n)$ using asymptotic notation and solve it. What can you say about its worst, average and best case?

3. (10 points) Write the function using an incremental (as opposed to divide-and-conquer) approach.
4. (8 points) Give its running time $T(n)$ in asymptotic notation.
5. (5 points) How much did $T(n)$ improve using the incremental approach? Can you think of a way to reduce the running time order?

Exercise 3: recurrent equations (42 points). (This is based on textbook problems 4.1 and 4.3.) Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$.

1. (6 points) $T(n) = T(2n/3) + \sqrt{n}$
2. (6 points) $T(n) = 4\,T(\sqrt{n}) + 1$
3. (6 points) $T(n) = 27\,T(n/3) + n^3$
4. (6 points) $T(n) = 13\,T(n/2) + n^3 + 3n\sqrt{n} + 1$
5. (6 points) $T(n) = 7\,T(n/3) + n^2$
6. (6 points) $T(n) = 2\,T(n/8) + n^{1/3}$
7. (6 points) $T(n) = T(n-1) + n$

Justify your answers. If you use the master theorem, specify which case and show that its hypotheses are satisfied. If you use recursion trees to find a good guess, verify the guess with the substitution method.

Bonus exercise: recurrent equations (10 points). Consider the recurrence

$$T(n) = aT(n/b) + f(n).$$

Case 3 of the master theorem requires that $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$ and that $f(n)$ satisfies the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n. Prove that the regularity condition is always satisfied if $f(n) = n^k$. (This means we don't need to check it when f is a polynomial.)

Bonus exercise: recurrent equations (30 points). Consider the following particular type of recurrence:

$$T(n) = \begin{cases} 1, & n = 1 \\ aT(n/b) + n^c, & n > 1 \end{cases}$$

where $a \geq 1$ and $b > 1$ are integers, and $k = log_b n$ is integer (that is, we can only pick sizes n of the form $n = b^k$ where $k \geq 0$ is an integer). Prove the master theorem for this recurrence, that is, prove that

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{if } c < \log_b a \\ \Theta(n^c \lg n), & \text{if } c = \log_b a \\ \Theta(n^c), & \text{if } c > \log_b a. \end{cases}$$

Hint: draw the recursion tree, indicating the tree levels, subproblem sizes and subproblem costs, and then sum all the costs.