

# Homework 4

22C:44 Algorithms  
Due Tuesday, March 12, 2002

- Sort the following functions so that they are ordered by growth rate (slowest growing first):  $3\sqrt{n}$ ,  $100n^2 + \log n$ ,  $400n - n^3 + n^5$ ,  $(\log n)^2$ ,  $\log \log n$ ,  $2^{n+1}$ ,  $3^n$ ,  $2^{(2n)}$ ,  $n^3$ ,  $n \log n^2$ ,  $n!$ ,  $1000 \log n$ ,  $n + \cos(n)$
- Show carefully, using the definition of Big-Oh, that  $(n + 4)^2 = O(n^2)$ . To do this, you need to choose particular values for constants  $c$  and  $n_0$  of the definition of Big-Oh.
- You are given programs A, B, C, and D.  $80n^2$  is an upper bound on the worst-case running time of program A. B is  $\Theta(n \log n)$  in the worst case, and  $\Theta(n)$  in the best and average cases. The running time of program C is characterized exactly by  $T(n) = 2n$ . The worst-case running time of D is  $\Omega(1)$ .

NOTE: Make sure you include short explanations with each of your answers below.

- Is it possible that program A will run faster than program B on *all* possible inputs?
  - Is it possible that program B will run faster than program A on *all* possible inputs?
  - Is it possible that program B will run faster than program C on *all* possible inputs?
  - Is it possible that program C will run faster than program B on *all* possible inputs?
  - Is it possible that program D runs slower than each of A, B, and C, on *all* possible inputs?
- Suppose functions  $f$  and  $g$  are defined as follows

$$f(n) = \begin{cases} n^3 & \text{if } n \bmod 3 = 0 \\ n & \text{otherwise} \end{cases}$$

$$g(n) = \begin{cases} n^2 & \text{if } n \bmod 2 = 0 \\ 1 & \text{otherwise} \end{cases}$$

Recall that  $n \bmod c = 0$  means that  $n$  is a multiple of  $c$ .

Give tight upper ( $O$ ) and lower ( $\Omega$ ) bounds for  $T(n)$  in each of the following cases.

- $T(n) = f(n) + g(n)$
- $T(n) = f(n)/g(n)$
- $T(n) = f(g(n))$

HINT: It might be helpful to eliminate the  $f$ 's and  $g$ 's from  $T(n)$ . That is, rewrite  $T(n)$  in terms of  $n$  only. For example, if we have  $T(n) = f(n) + g(n)$  then we can rewrite  $T(n)$  as

$$T(n) = \begin{cases} n^3 + 1 & \text{if } n \bmod 3 = 0, \text{ but } n \bmod 2 \neq 0 \\ n^3 + n^2 & \text{if } n \bmod 6 = 0 \\ n^2 + n & \text{if } n \bmod 2 = 0, \text{ but } n \bmod 3 \neq 0 \\ n + 1 & \text{otherwise} \end{cases}$$

5. Give a tight bound on the running time of each the following (i.e. find  $f(n)$  such that the program is  $\Theta(f(n))$ ). Assume  $n \geq 1$ .

a.

```

for (i = 5; i < n*n; i++) {
    j = 1;
    while (j < n) {
        print(i*j + i*i - 3);
        A[i,j] = j-i;
        j = j + 2;
    }
}

```

b.

```

for (i = 1; i < n/2; i++)
    B[i] = i*i*i;
i = 1;
while (i < n) {
    for (j = 1; j < n; j++)
        for (k = 1; k < n; k++)
            A[j,k] := i * j;
    i = i * 2;
}
for (i = 1; i < n; i++)
    for (j = 1; j < n; j++)
        A[i] += A[j + i] * i + j;

```

6. Determine  $f(n)$  such that if

$$T(1) = 1$$

$$T(n) = 4T(n/2) + n \quad \text{for } n \geq 2$$

then  $T(n)$  is  $\Theta(f(n))$ . Use the “recursion tree” or “iteration” method for solving recurrences.