CS 5413 homework assignments

Problem 2-2 in the text

Problem 2-4 in the text

Problem 3-3 in the text

Problem 3-7 in the text

2-2 Correctness of bubblesort

Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order. The procedure BUBBLESORT sorts array $A[1:n]$.

```
BUBBLESORT(A, n)
1  for i = 1 to n − 1
2      for j = n downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

a. Let A' denote the array A after BUBBLESORT(A, n) is executed. To prove that BUBBLESORT is correct, you need to prove that it terminates and that

$$A'[1] \le A'[2] \le \cdots \le A'[n] \,. \tag{2.5}$$

In order to show that BUBBLESORT actually sorts, what else do you need to prove?

The next two parts prove inequality (2.5).

b. State precisely a loop invariant for the **for** loop in lines 2–4, and prove that this loop invariant holds. Your proof should use the structure of the loop-invariant proof presented in this chapter.

c. Using the termination condition of the loop invariant proved in part (b), state a loop invariant for the **for** loop in lines 1–4 that allows you to prove inequality (2.5). Your proof should use the structure of the loop-invariant proof presented in this chapter.

d. What is the worst-case running time of BUBBLESORT? How does it compare with the running time of INSERTION-SORT?

2-3 Correctness of Horner's rule

You are given the coefficents $a_0, a_1, a_2, \ldots, a_n$ of a polynomial

$$P(x) = \sum_{k=0}^{n} a_k x^k$$

a. In terms of Θ-notation, what is the running time of this procedure?

b. Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare with HORNER?

c. Consider the following loop invariant for the procedure HORNER:

> At the start of each iteration of the **for** loop of lines 2–3,
>
> $$p = \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k .$$
>
> Interpret a summation with no terms as equaling 0. Following the structure of the loop-invariant proof presented in this chapter, use this loop invariant to show that, at termination, $p = \sum_{k=0}^{n} A[k] \cdot x^k$.

2-4 *Inversions*

Let $A[1:n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A.

a. List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?

c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify merge sort.)

The procedure HORNER implements Horner's rule to evaluate $P(x)$, given the coefficients $a_0, a_1, a_2, \ldots, a_n$ in an array $A[0:n]$ and the value of x.

```
HORNER(A, n, x)
1  p = 0
2  for i = n downto 0
3      p = A[i] + x · p
4  return p
```

a. In terms of Θ-notation, what is the running time of this procedure?

b. Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare with HORNER?

c. Consider the following loop invariant for the procedure HORNER:

> At the start of each iteration of the **for** loop of lines 2–3,
>
> $$p = \sum_{k=0}^{n-(i+1)} A[k+i+1] \cdot x^k \,.$$
>
> Interpret a summation with no terms as equaling 0. Following the structure of the loop-invariant proof presented in this chapter, use this loop invariant to show that, at termination, $p = \sum_{k=0}^{n} A[k] \cdot x^k$.

2-4 *Inversions*

Let $A[1:n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A.

a. List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?

c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify merge sort.)

Problems

3-1 Asymptotic behavior of polynomials

Let

$$p(n) = \sum_{i=0}^{d} a_i n^i ,$$

where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notations to prove the following properties.

a. If $k \geq d$, then $p(n) = O(n^k)$.

b. If $k \leq d$, then $p(n) = \Omega(n^k)$.

c. If $k = d$, then $p(n) = \Theta(n^k)$.

d. If $k > d$, then $p(n) = o(n^k)$.

e. If $k < d$, then $p(n) = \omega(n^k)$.

3-2 *Relative asymptotic growths*

Indicate, for each pair of expressions (A, B) in the table below whether A is O, o, Ω, ω, or Θ of B. Assume that $k \geq 1$, $\epsilon > 0$, and $c > 1$ are constants. Write your answer in the form of the table with "yes" or "no" written in each box.

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					
f.	$\lg(n!)$	$\lg(n^n)$					

3-3 *Ordering by asymptotic growth rates*

a. Rank the following functions by order of growth. That is, find an arrangement $g_1, g_2, \ldots, g_{30}$ of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{29} = \Omega(g_{30})$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ belong to the same class if and only if $f(n) = \Theta(g(n))$.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(3/2)^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln\ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg\lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2\lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

b. Give an example of a single nonnegative function $f(n)$ such that for all functions $g_i(n)$ in part (a), $f(n)$ is neither $O(g_i(n))$ nor $\Omega(g_i(n))$.

3-4 *Asymptotic notation properties*

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min\{f(n), g(n)\})$.

c. $f(n) = O(g(n))$ implies $\lg f(n) = O(\lg g(n))$, where $\lg g(n) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

d. $f(n) = O(g(n))$ implies $2^{f(n)} = O\left(2^{g(n)}\right)$.

e. $f(n) = O\left((f(n))^2\right)$.

f. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$.

h. $f(n) + o(f(n)) = \Theta(f(n))$.

3-5 *Manipulating asymptotic notation*

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove the following identities:

a. $\Theta(\Theta(f(n))) = \Theta(f(n))$.

b. $\Theta(f(n)) + O(f(n)) = \Theta(f(n))$.

c. $\Theta(f(n)) + \Theta(g(n)) = \Theta(f(n) + g(n))$.

d. $\Theta(f(n)) \cdot \Theta(g(n)) = \Theta(f(n) \cdot g(n))$.

$$\widetilde{O}(g(n)) = \{f(n) : \text{there exist positive constants } c, k, \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n)\lg^k(n) \text{ for all } n \ge n_0\}.$$

e. Define $\widetilde{\Omega}$ and $\widetilde{\Theta}$ in a similar manner. Prove the corresponding analog to Theorem 3.1.

3-7 *Iterated functions*

We can apply the iteration operator * used in the $\lg^*$ function to any monotonically increasing function $f(n)$ over the reals. For a given constant $c \in \mathbb{R}$, we define the iterated function f_c^* by

$$f_c^*(n) = \min\{i \ge 0 : f^{(i)}(n) \le c\},$$

which need not be well defined in all cases. In other words, the quantity $f_c^*(n)$ is the minimum number of iterated applications of the function f required to reduce its argument down to c or less.

For each of the functions $f(n)$ and constants c in the table below, give as tight a bound as possible on $f_c^*(n)$. If there is no i such that $f^{(i)}(n) \le c$, write "undefined" as your answer.

	$f(n)$	c	$f_c^*(n)$
a.	$n-1$	0	
b.	$\lg n$	1	
c.	$n/2$	1	
d.	$n/2$	2	
e.	$\sqrt{n}$	2	
f.	$\sqrt{n}$	1	
g.	$n^{1/3}$	2	

Chapter notes

Knuth [259] traces the origin of the O-notation to a number-theory text by P. Bachmann in 1892. The o-notation was invented by E. Landau in 1909 for his discussion of the distribution of prime numbers. The Ω and Θ notations were advocated by Knuth [265] to correct the popular, but technically sloppy, practice in the literature of using O-notation for both upper and lower bounds. As noted earlier in this chapter, many people continue to use the O-notation where the Θ-notation is more technically precise. The soft-oh notation $\widetilde{O}$ in Problem 3-6 was introduced