

Section 1.8

4. If x is bigger than 1, then $2^x < 3^x$ and $17 < 7(3^x)$. Hence $2^x + 17 < 8(3^x)$, so $2^x + 17$ is $O(3^x)$.

14. (a) No. x^3/x^2 tends to infinity as x tends to infinity so x^3 is not $O(x^2)$.

(b) Yes, with $C=1$ and $k=0$.

(c) Yes, with $C=1$ and $k=0$.

(d) Yes, with $C=1$ and $k=1$.

(e) Yes. One way to see this is to take \ln : $\ln(x^3) = 3 \ln x$, and $\ln(3^x) = x \ln 3$. Since $\ln x$ is $O(x)$ and the exponential function is increasing, this implies that x^3 is $O(3^x)$.

(f) Yes, with $C=2$ and $k=0$.

22. (a) Not $\Omega(x)$, not $\theta(x)$.

(b) $\Omega(x)$ and $\theta(x)$.

(c) $\Omega(x)$, not $\theta(x)$.

(d) Not $\Omega(x)$, not $\theta(x)$.

(e) $\Omega(x)$ and $\theta(x)$.

(f) $\Omega(x)$ and $\theta(x)$.

38. If f is $\theta(g)$, then f is $O(g)$ and f is $\Omega(g)$. Likewise, if g is $\theta(h)$, then g is $O(h)$ and g is $\Omega(h)$. Hence there exists $C_1 > 0$ and $k_1 > 0$ so that if $x > k_1$, then $|f(x)| \leq C_1 |g(x)|$, and there exists $C_2 > 0$ and $k_2 > 0$ so that if $x > k_2$, then $|g(x)| \leq C_2 |h(x)|$. Hence if $x > \max(k_1, k_2)$, then $|f(x)| \leq C_1 C_2 |h(x)|$, so f is $O(h)$. A similar argument with lower bounds on f implies that f is $\Omega(h)$, so f is $\theta(h)$.

Section 2.1

26.

procedure *big terms*(a_1, \dots, a_n : real numbers)

total := 0

index := 1

nextBigTermIndex := 0

array *bigTerms*[n]

while (*index* < $n+1$)

begin

 if $a_{\text{index}} > \text{total}$ **then**

begin

nextBigTermIndex := *nextBigTermIndex* + 1

bigTerms[*nextBigTermIndex*] = a_{index}

end

total := *total* + a_{index}

index := *index* + 1

end

{The array *bigTerms*[] contains the terms bigger than the sum of all previous entries, and *nextBigTermIndex* contains the number of big terms. (A separate array could be used to store the positions, but this was not required.)}

Section 2.2

4. Starting with x , it takes one multiplication to get x^2 , then squaring again gives x^4 in two multiplications, x^8 in three, and in general, k multiplications to get x^m , where $m=2^k$. Hence this procedure is $O(k)$, whereas multiplying by x repeatedly takes 2^k-1 multiplications to get x^m , which is $O(2^k)$. Hence repeated squaring is more efficient.

14. (a) One way to prove that the algorithm is optimal is to use induction. If there is only one element in the sequence, then no comparisons are needed to find the maximum, and if there are two elements, then one comparison must be used, as in the given algorithm. For the induction step, we suppose that any algorithm to find the maximum element from a sequence of m elements requires at least $m-1$ comparisons among the elements. In any algorithm to find the maximum of $m+1$ elements, there are two elements, say a and b that appear in the final comparison. Since this is the final comparison, a must be the largest element in some subset A of the original elements and b must be the largest in a subset B with the union of A and B containing all $m+1$ elements. Hence $|A| + |B| \geq m+1$, and we may assume that $|A| \leq m$ and $|B| \leq m$ since otherwise the final comparison was irrelevant. By the induction hypothesis, the number of comparison needed to find a was at least $|A|-1$, and the number needed to find b was at least $|B|-1$. Hence the number of comparisons before the final comparison was at least $|A|+|B|-2 \geq m-1$, and adding in the final comparison, we have at least m comparisons needed to find the maximum of $m+1$ elements. By induction, any algorithm to find the maximum of a sequence of n integers requires at least $n-1$ comparisons, so the given algorithm is optimal with regard to the number of comparisons. See the grading guide for additional discussion.

(b) To search a randomly ordered list of integers for a given value, in the worst case (in which the given value is not in the list), each of the elements in the list must be compared to the given value, so the algorithm must use at least n comparisons to search a list of length n . Hence the linear search algorithm is optimal with respect to the number of comparisons.

20. Each time through the loop there is one comparison (not counting the index comparison) plus one addition. If the comparison is satisfied, then there is also the addition of the new element to the list of elements satisfying the condition. Hence in the worst case there are n nonindex comparisons, $n+1$ index comparisons, n additions, and n list additions, for a running time of $O(n)$.

Section 2.3

12. To solve this, we determine the number of 5's and the number of 2's in the prime factorization of $100!$. Since 2 times 5 is 10, the smaller of these two number of factors represents the number of times 10 divides $100!$ evenly, hence the number of 0's at the end. In the integers between 1 and 100, there are 20 numbers that are divisible by 5 and 4 that are divisible by 25. Hence there are 24 factors of 5. There are 50 numbers divisible by 2, so there are at least 50 factors of 2. Hence there must be 24 0's at the end of $100!$.

16. (a) The proper divisors of 6 are 1, 2, 3, which add to 6, hence 6 is perfect. The proper divisors of 28 are 1, 2, 4, 7, 14, which add to 28, hence 28 is perfect.

(b) Let $a = 2^{p-1}$ and $b = (2^p)-1$. If b is prime, then the divisors of ab are 1, 2, 4, 8, ..., 2^{p-1} plus each of these factors multiplied by b . Since we exclude ab from the list of proper divisors, we take the sum

$$(1+2+4+\dots+2^{p-1}) + b(1+2+4+\dots+2^{p-2}) = (2^p-1) + b(2^{p-1}-1).$$

Using $b = (2^p)-1$, this is equal to $2^{p-1}((2^p)-1) = ab$, so this is a perfect number.

40. (a) 58

(b) 60

(c) 52

(d) 3

48. The number must satisfy $1(0) + 2(2) + 3(0) + 4(1) + 5(5) + 6(7) + 7(Q) + 8(8) + 9(9) + 10(1) = 0 \pmod{11}$. This is $230 + 7Q = 0 \pmod{11}$. Subtracting 231 (which is congruent to 0 mod 11) from both sides, we have $-1 + 7Q = 0 \pmod{11}$, or $7Q = 1 \pmod{11}$. Hence we want $7Q$ equal to one of 1, 12, 23, 34, 45, 56, or 67. Since 56 is the only one of these divisible by 7, we have $Q = 8$. Checking, $230 + 7(8) = 286$, which is divisible by 11, as desired.