Cesar Camarena
T00223324

Chapter II HW

2.1

1. For each of the following algorithms, indicate (i) a natural size metric for its inputs, (ii) its basic operation, and (iii) whether the basic operation count can be different for inputs of the same size:
    a. Computing the sum of n numbers
    b. Computing n!
    c. Finding the largest element in a list of n numbers
    d. Euclid's algorithm
    e. Sieve of Eratosthenes
    f. Pen-and-pencil algorithm for multiplying two n-digit decimal integers

	i	ii	iii
a	n	addition	no
b	the amount of n's in the problem.	multiplication	no
c	n	comparison of two numbers	no
d	Whichever n is largest or smallest	Modulus	Yes
e	The amount of n's in the problem	Taking out a number from the list	No
f	N	Multiplication	no

2. a. Consider the definition-based algorithm for adding two n x n matrices. What is its basic operation? How many times is it performed as a function of the matrix order n? As a function of the total number of elements in the input matrices?

    Sum of the two corresponding elements of the matrices given. Happens n^2 times. n^2=N/2

    b. Answer the same questions for the definition-based algorithm for matrix multiplication.

    Multiplication. n^2 elements in the matrix gets multiplied by n elements of a vector.
n^3 = (N/2)^3/2

Cesar Camarena
T00223324

9. For each of the following pairs of functions, indicate whether the first function of each of the following pairs has a lower, same, or higher order of growth (to within a constant multiple) than the second function.

a. n(n + 1) and 2000n^2

same

b. 100n^2 and 0.01n^3

lower

c. log2n and ln(n)

same

d. log2^2n and log2n^2

higher

e. 2^(n-1) and 2^n

same

f. (n-1)! And n!

same

2.2

2. Use the informal definitions of O, θ, Ω to indicate the time efficiency class of sequential search

a. in the worst case

n

b. in the best case

1

c. in the average case

(1 – (p/2)n + (p/2) where p is between 0 and 1

Cesar Camarena
T00223324

5. List the following functions according to their order of growth from the lowest to the highest:

(n-2)!, 51g(n + 100)^10, 2^2n, 0.001n^4 + 3n^3 + 1, ln^2n, sqrt3(n), 3^n

51g(n + 100)^10, ln^2n, sqrt3(n), 0.00n^4 + 3n^3 + 1, 3^n, 2^2n, (n – 2)!

9. We mentioned in this section that one can check whether all elements of an array are distinct by a two-part algorithm based on the array's presorting.

a. If the presorting is done by an algorithm with a time efficiency of θ(nlogn), what will be a time-efficiency class of an entire algorithm?

θ(nlogn)

b. If the sorting algorithm used for presorting needs an extra array of size n, what will be the space-efficiency class of the entire algorithm?

θ(n)

2.3

1. a. 250, 000

b. 2, 046

c. n – 1

d. (n^2 + 3n – 4)/2

e. ((n^2 – 1)n)/4

f. (3^(n + 2) – 9)/2

g. (n^2(n + 1)^2)/4

h. n/(n + 1)

2. a. n^5

b. nlogn

c. n2^n

d. n^3

Cesar Camarena
T00223324

4. a. n^2

b. Multiplication

c. n

d. 2^b

e. Use (n(n + 1)(2n + 1))/6 to get θ(1)

2.4

1. a. 5(n – 1)

b. 4(3^(n – 1))

c. (n(n + 1))/2

d. 2n - 1

e. 1 + log3(n)

3. a. 2(n – 1)

b. S ←1

for i ←2 to n do

S ←S + i^3

return S

2.5

2. 144 paris

3. F(n + 1) for n ≥ 1

2.6

1. add the line

If j ≥ count ←count + 1

right after the while statement's end

4. nlogn algorithm