Data Structures and Algorithms [19ECSC201]

Chapter 1: Fundamentals of Algorithms and Problem Solving

Exercise - Handout

Problem 1.

For each of the following algorithms,

- Computing the sum of n numbers
- Computing n!
- Finding the largest element in a list of n numbers
- Euclid's algorithm

Indicate:

a. A natural size metric for its inputs
b. Basic Operation
c. Will Basic Operation count be different for inputs of the same size?

Problem 2.

Consider a variation of sequential search that scans a list to return the number of occurrences of a given search key in the list. Will its efficiency differ from the efficiency of classic sequential search?

Problem 3.

Solve: 1+3+5+… +999

Problem 4.

There are 22 gloves in a drawer: 5 pairs of red gloves, 4 pairs of yellow, and 2 pairs of green. You select the gloves in the dark and can check them only after a selection has been made. What is the smallest number of gloves you need to select to have at least one matching pair in the best case? In the worst case?

Problem 5.

Suggest how any sorting algorithm can be augmented in a way to make the best-case count of its key comparisons equal to just n−1. Do you think it would be a worthwhile addition to any sorting algorithm?

Problem 6.

For each of the following functions, indicate by how much the function value will change if its argument is increased fourfold.

a. $\log_2 n$
b. $\sqrt{n}$
c. n

d. n^2
e. n^3
f. 2^n

Problem 7.
Indicate whether the first function of each of the following pairs has a smaller, same, or larger order of growth (to within a constant multiple) than the second function:
a. $n(n + 1)$and $2000n^2$
b. $100n^2$ and $0.01n^3$
c. $\log_2 n$ and $\ln n$
d. $\log_2^2 n$ and $\log_2 n^2$
e. 2^{n-1} and 2^n
f. $(n-1)!$ and $n!$

Problem 8.
According to a well-known legend, the game of chess was invented many centuries ago in north-western India by a sage named Shashi. When he took his invention to his king, the king liked the game so much that he offered the inventor any reward he wanted. Shashi asked for some grain to be obtained as follows: just a single grain of wheat was to be placed on the first square of the chess board, two on the second, four on the third, eight on the fourth, and so on, until all 64 squares had been filled.

What would the ultimate result of this algorithm have been?

Problem 9.
Order the following functions according to their orders of growth (from the lowest to the highest):
$(n-2)!$ $5\log(n+100)^{10}$ 2^{2n} $0.001n^4 + 3n^3$ $\ln^2 n$ $(n)^{1/3}$ 3^n

Problem 10.
```
ALGORITHM MaxElement(A[0… n-1])
// Determines the value of largest element in a given array
// Input: An array A[0… n-1] of real numbers
// Output: The value of the largest element in A
maxval ← A[0]
for i ← 1 to n-1 do
   if A[i] > maxval
        maxval ← A[i]
return maxval
```

a. What is the Basic Operation?

b. How many times is it executed? Perform efficiency analysis
c. Write your conclusions on orders of growth

Problem 11.

```
ALGORITHM Mystery(n)
// Input: A non-negative integer
S ← 0
for i ← 1 to n do
   S ← S + i * i
return S
```

a. What does this algorithm compute?
b. What is its basic operation?
c. How many times is the BO executed?
d. Suggest an improvement or a better algorithm altogether and indicate its efficiency class. If you cannot do it, try to prove that, in fact, it cannot be done.

Problem 12.

```
ALGORITHM matrixMultiplication(A[0… n-1, 0… n-1], B[0… n-1, 0… n-1])
// Multiplies two n-by-n matrices
// Input: Two n-by-n matrices A and B
// Output: matrix C = AB
for i ← 0 to n-1 do
   for j ← 0 to n-1 do
      C[i, j] ← 0.0
      for k ← 0 to n-1 do
          C[i, j] ← C[i, j] + A[i, k] * B[k, j]
return C
```

a. What is the basic operation?
b. Find the orders of growth

Problem 13.

Improve the implementation of the matrix multiplication algorithm (Refer problem 12) by reducing the number of additions made by the algorithm. What effect will this change have on the algorithm's efficiency?

Problem 14.

Solve:

$x(n) = x(n-1) + 5$ for $n > 1$

$x(1) = 0$

Problem 15.
For the algorithm given below, perform efficiency analysis:

```
ALGORITHM Binary (n)
// Count the number of digits in a binary representation of a given positive decimal
// integer
// Input: n, a positive decimal integer
// Output: Number of digits in a binary representation
if n = 1
   return 1
return Binary(n/2) + 1
```

Problem 16.
Find the orders of growth for:

```
ALGORITHM Factorial (n)
// Computes factorial of a given number
// Input: positive integer n
// Output: factorial of a supplied n
if n = 0
   return 1
return n*Factorial(n-1)
```

Problem 17.
Consider the following algorithm:

```
ALGORITHM Secret(A[0..n−1])
//Input: An array A[0..n−1] of n real numbers
minval ← A[0]
maxval ← A[0]
for i ← 1 to n−1 do
    if A[i] < minval
        minval ← A[i]
    if A[i] > maxval
        maxval ← A[i]
return maxval – minval
```

a. What does this algorithm compute?
b. What is its basic operation?
c. How many times is the basic operation executed?
d. What is the efficiency class of this algorithm?
e. Suggest an improvement or a better algorithm altogether and indicate its efficiency

Problem 18.

Consider the following algorithm:

```
ALGORITHM Enigma(A[0..n−1,0..n−1])
//Input: A matrix A[0..n−1,0..n−1] of real numbers
for i ←0 to n−2 do
    for j ←i +1 to n−1 do
        if A[i, j] != A[j, i]
            return false
return true
```

a. What does this algorithm compute?

b. What is its basic operation?

c. How many times is the basic operation executed?

d. What is the efficiency class of this algorithm?

e. Suggest an improvement or a better algorithm altogether and indicate its efficiency

Problem 19.

For the algorithm given below, identify and analyse for worst case efficiency.

```
ALGORITHM UniqueElements(A[0… n-1])
// Determines whether all the elements in a given array are distinct
// Input: An array A[0...n-1]
// Output: Returns true if all the elements in A are distinct and false otherwise.
for i ← 0 to n-2 do
    for j ← i+1 to n-1 do
        if A[i] = A[j] return false
return true
```

Problem 20.

Solve:

$x(n) = 3x(n-1)$ for $n > 1$ and $x(1) = 4$

Problem 21.

Consider the following recursive algorithm for computing the sum of the first n cubes: $S(n)=1^3 + 2^3 + \dots + n^3$.

```
ALGORITHM S(n)
//Input: A positive integer n
//Output: The sum of the first n cubes
if n =1
    return 1
else
    return S(n−1) + n∗n∗n
```

a. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.
b. How does this algorithm compare with the straightforward non-recursive algorithm for computing this function?

Problem 22.
Consider the following recursive algorithm.

```
ALGORITHM Q(n)
//Input: A positive integer n
if n =1
   return 1
else
   return Q(n−1) + 2∗n−1
```

a. Set up a recurrence relation for this function's values and solve it to determine what this algorithm computes.
b. Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.
c. Set up a recurrence relation for the number of additions/subtractions made by this algorithm and solve it.

Problem 23.
a. Design a recursive algorithm for computing 2^n for any nonnegative integer n that is based on the formula: $2^n = 2^{n-1} + 2^{n-1}$.
b. Set up a recurrence relation for the number of additions made by the algorithm and solve it.
c. Draw a tree of recursive calls for this algorithm and count the number of calls made by the algorithm.
d. Is it a good algorithm for solving this problem?

~*~*~*~*~*~*~*~*~