CS317: Algorithms — Homework Assignment #2

Due: See Canvas for Assignment Due Dates (20 points)

NOTE: NO LATE ASSIGNMENTS WILL BE ACCEPTED because we often review them in class on the due date.

Please upload the document containing your answers. They can be handwritten and scanned, but they must be clearly legible to receive a grade on the assignment. PDF is the best format for canvas. You DO NOT Need to include this cover sheet in your upload. It is formatted for the grader to use for me, as needed.

Chapter 2: Work the following problems. Point values are provided for each problem.

Problem #	Points	Grader's Notes
Sec 2.1, #7a	1	
Sec 2.1, #10	2	
Sec 2.2, #2	2	
Sec 2.3, #1a,b,c	3	
Sec 2.3, #6a,b,c,d	4	
Sec 2.4, #1 a-d	4	
Sec 2.4, #4 a-c	2	
Sec 2.4, #10	1	

Chapter 3

Problem #	Points	Grader's Notes
Sec 3.1, #3	1	

2.1)

7. Gaussian elimination, the classic algorithm for solving systems of n linear equations in n unknowns, requires about $\frac{1}{3}n^3$ multiplications, which is the algorithm's basic operation.

a. How much longer should you expect Gaussian elimination to work on a system of 1000 equations versus a system of 500 equations?

$\frac{1}{3}n^3$ multiplications

Number of equations	Number of operations	Quick math
1000	333,333,333.3	$\frac{333,333,333}{41,666,666}$ ≈ 8 times faster for 500 equations
500	41,666,666	

10. *Invention of chess*

a. According to a well-known legend, the game of chess was invented many centuries ago in northwestern India by a certain sage. When he took his invention to his king, the king liked the game so much that he offered the inventor any reward he wanted. The inventor 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 chessboard, two on the second, four on the third, eight on the fourth, and so on, until all 64 squares had been filled. If it took just 1 second to count each grain, how long would it take to count all the grain due to him?

b. How long would it take if instead of doubling the number of grains for each square of the chessboard, the inventor asked for adding two grains?

1	2	3	4	5	6
1	2	4	8	16	32

For every increase in n, the grains of rice double.

$\sum_{i=1}^{64} 2^{n-1}$

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$

2^{64} seconds... 1.84×10^{19} seconds
or 5.85×10^{10} decades
... A long time. Impossibly long.

b. How long would it take if instead of doubling the number of grains for each square of the chessboard, the inventor asked for adding two grains?

$n + n+2 + n+2 \ldots + n+2$ $\quad 1+3+5+7+9\ldots. \Rightarrow \sum_{i=1}^{64} (2n-1)$

On the 64th tile there would be $128-1$ grains of rice, or 127 seconds

---

2.2)

2. Use the informal definitions of O, Θ, and Ω to determine whether the following assertions are true or false.

$O \leq \quad \Omega \geq \quad \Theta =$

a. $n(n+1)/2 \in O(n^3)$ $\quad \frac{n^2+n}{2} \leq n^3$

True, $n^2 \leq n^3$

b. $n(n+1)/2 \in O(n^2)$ $\quad \frac{n^2+n}{2} \leq n^2$

True, $n^2 \leq n^2$

c. $n(n+1)/2 \in \Theta(n^3)$ $\quad \frac{n^2+n}{2} \leq n^3$

False, $n^2 \neq n^3$

d. $n(n+1)/2 \in \Omega(n)$ $\quad \frac{n^2+n}{2} \leq n^3$

False, $n^2 \not\geq n^3$

2.3)

1. Compute the following sums.

a. $1+3+5+7+\cdots+999$

b. $2+4+8+16+\cdots+1024$

c. $\sum_{i=3}^{n+1} 1$ **d.** $\sum_{i=3}^{n+1} i$

$2 \cdot \sum_{n=1}^{500} n \Rightarrow$ Sum formula: $\frac{1}{2} n(n+1)$

$\frac{1}{2}(500 \times 501) = 125\,250$

$2 \times 125250 = 250{,}500$

a) $1+3+5+7+\ldots+999:$

$999 = 2n-1$

$998 = 2n$

$n = 499$

$\sum_{i=1}^{500} (2n-1)$,

$\sum_{n=1}^{500} 1 = 500$

$\sum_{i=1}^{500} 2n - \sum_{i=1}^{500} 1 = 250{,}500 - 500 = 250{,}000$

b) $2+4+8+10+\dots+1024$

$\sum_{i=1}^{512} 2n = 2\cdot\sum_{i=1}^{512} n = 1/2 \cdot 512(513) = 131328 \times 2 = \boxed{262656}$

$1024 = 2n \Rightarrow n = 512$ operations

$\sum_{i=1}^{512} 2n = 262656 = 2+4+8+10+\dots+1024$

c) $\sum_{i=3}^{n+1} 1 = \sum_{i=1}^{n+1} 1 - \sum_{i=1}^{2} 1 \Rightarrow \sum_{i=1}^{n+1} 1 = 1\cdot(n+1)$

$\Rightarrow \sum_{i=1}^{2} 1 = 1\cdot 2 = 2$

$n+1-2 = n-1$

$\boxed{\sum_{i=3}^{n+1} = n-1}$

$A[1,2] = [2,1]$

6

ALGORITHM *Enigma*$(A[0..n-1, 0..n-1])$
//Input: A matrix $A[0..n-1, 0..n-1]$ of real numbers
for $i \leftarrow 0$ **to** $n-2$ **do**
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[i, j] \neq A[j, i]$
      return false
return true

a. What does this algorithm compute?

Checks whether numbers in an array are repeated or not. If none repeate, then it returns false. Checks for Array Symmetry

b. What is its basic operation?

if $A[i,j] \neq A[j,i]$

c. How many times is the basic operation executed?

$n^2/2$

d. What is the efficiency class of this algorithm?

Efficiency class: $\Theta(n^2)$

2.4

1. Solve the following recurrence relations.

a. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$

b. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$

c. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$

d. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)

e. $x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$ (solve for $n = 3^k$)

1)

a) **a.** $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$

$x(n-1) + 5$

$x(1) = 0$

$x(2) = x(1) + 5 = 0 + 5 = 5$

$x(3) = x(2) + 5 = 5 + 5 = 10$

$x(4) = x(3) + 5 = 10 + 5 = 15$

$5(n-1)$ for all $n \geq 1$

b) **b.** $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$

$x(1) = 4$

$x(n) = 3x(n-1)$ Sub. $n-1 = 3x(n-2)$

$x(2) = 3[3x(n-2)]$ Sub. $n-2 = 3x(n-3)$

$x(3) = 3^2[3x(n-3)]$ Sub $n-3 = 3x(n-4)$

$x(4) = 3^3[3x(n-4)]$

$x(n) = 3^i\, x(n-i) \Rightarrow 3^{n-1}\, x(n-(n-1)) = 3^{n-1}(x(1)) = 4 \times 3^{n-1}$

$x(n) = 3x(n-1) = \boxed{4 \times 3^{n-1}}$

c. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$

$c = 1$ $g(n) = n$ $x(1) = x(0) + 1 = 1$

$$x(n) = 1^{n-1} \cdot 1 + \sum_{i=2}^{n} (1^{n-i}) \cdot (i)$$

$$= 1 + \sum_{i=2}^{n} i \Rightarrow 1 + (2 + 3 + \ldots + n) = \boxed{\frac{n(n+1)}{2}}$$

d. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)

$x(2^{k-1}) + 2^k \Rightarrow [x(2^{k-2}) + 2^{k-1}] + 2^k = x(2^{k-1}) + 2^{k-1} + 2^k$

$= [x(2^{k-3}) + 2^{k-2}] + 2^{k-1} + 2^k = x(2^{k-3}) + 2^{k-2} + 2^{k-1} + 2^k$

$= x(2^{k-i}) + 2^{k-i+1} + \ldots + 2^k = x(2^{k-k}) + 2^{k-k+1} + \ldots + 2^k$

$= x(2^{k-k}) + 2^1 + 2^2 + \ldots + 2^k = 1 + 2^1 + 2^2 + \ldots + 2^k$

$= 2^{k+1} - 1 = 2 \cdot 2^k - 1 = \boxed{2n - 1}$

4. 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.

$Q(n) = Q(n-1) + 2n - 1$ For $n > 1$

$Q(2) = Q(1) + 4 - 1 = 4$

$Q(3) = Q(2) + 6 - 1 = 9$

$Q(4) = Q(3) + 8 - 1 = 16$

From this, it seems like $\boxed{Q(n) = n^2}$

b. Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.

else do one multiplication if $n=1$, no multiplications are performed

$Q(n) = \begin{cases} 0 & \text{if } n=1 \\ Q(n-1)+1 & \text{otherwise} \end{cases}$

$Q(n) = Q(n-1) + 1$

$Q(1) = 0$

$Q(2) = Q(1) + 1 = 1$

$Q(3) = Q(2) + 1 = 2$

$Q(4) = Q(3) + 1 = 3$

$\boxed{Q(n) = n - 1}$

c. Set up a recurrence relation for the number of additions/subtractions made by this algorithm and solve it.

else, do 1 addition and one subtraction

if $n=1$, no additions are performed: $Q(n) = \begin{cases} 0 & \text{if } n=1 \\ C(n-1)+2 & \text{otherwise} \end{cases}$

$Q(1) = 0$

$Q(2) = C(1) + 2 = 2$

$Q(3) = C(2) + 2 = 4$

$\boxed{2n - 2}$

10. Consider the following algorithm to check whether a graph defined by its adjacency matrix is complete.

ALGORITHM *GraphComplete*$(A[0..n-1, 0..n-1])$

```
//Input: Adjacency matrix A[0..n − 1, 0..n − 1]) of an undirected graph G
//Output: 1 (true) if G is complete and 0 (false) otherwise
if n = 1 return 1   //one-vertex graph is complete by definition
else
    if not GraphComplete(A[0..n − 2, 0..n − 2]) return 0
    else for j ← 0 to n − 2 do
            if A[n − 1, j] = 0 return 0
        return 1
```

What is the algorithm's efficiency class in the worst case?

The worst case would be if the graph was complete and had to go through every item. In this case, the time complexity would be $O(n^2)$ because it needs 2 checks for each time it goes through the algorithm.

3.1

3. For each of the algorithms in Problems 4, 5, and 6 of Exercises 2.3, tell whether or not the algorithm is based on the brute-force approach.

ALGORITHM *Mystery*(n)

```
//Input: A nonnegative integer n
S ← 0
for i ← 1 to n do
    S ← S + i ∗ i
return S
```

I would say this is brute force because it is very straight forward and is a running sum of the square of numbers from 1->n.
- While it is brute force though, this may be the fastest implementation.

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
```

This is also brute force. You are searching through all of the elements just to find the difference between the highest and lowest value.

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
```

I would say this is brute force b/c it checks every element of the matrix to check if it is symmetric.