CSI 2110 Computer Science Fall 2016 University of Ottawa

Assignment #1 (35 points, weight 5%)
Due: Wednesday September 28, 9:30PM

The assignment is to be uploaded on blackboard electronically (you may type or write by hand legibly and scan it). Only a single PDF file is accepted.

Late submission is accepted from 1 min late up to 24 hs for 30% off (i.e. your mark is multiplied by 0.7).

1. (2 marks) Given an *n*-element array A, Algorithm X executes an $O(n)$-time computation for each even number in A and an $O(\log n)$-time computation for each odd number in A.

   (a) What is the best-case running time of Algorithm X?

   (b) What is the worst-case running time of Algorithm X?

2. (9 marks) Use the definition of "$f(n)$ is $O(g(n))$" to prove the following statements.

   (a) $f(n)=7n^3+3n^2-2n+100$ is $O(n^3)$.

   (b) $f(n) = (n^2 + 1)/(n + 1)$ is $O(n)$.

   (c) $f(n)=n!$ is $O(n^n)$.

   (d) $f(n) = \log_2 n$ is $O(\log_{10} n)$.

   (e) $f(n) = n^3$ **is not** $O(100n^2)$.[=

   (f) $f(n) = 2^{n+1}$ is $\Theta(2^n)$

3. Given an array, A, of n integers, give an $O(n)$-time algorithm that finds the longest subarray of A such that all the numbers in that subarray are in sorted order. Your algorithm outputs two integers: the initial and final indices of the longest subarray.

   (a) (4 marks) Give the algorithm pseudocode.

(b) (1 mark) Justify your big-Oh (1 mark).

A solution that uses extra memory that is in *O(1)* is worth 100%; if you use *Θ(n)* extra memory your solutions is worth 80%.
Example: If n = 10 and A= [8,6,7,10,−2,4,5,6,2,5] then the algorithm outputs 4 and 7, since A[4..7] = [−2, 4, 5, 6] is the longest sorted subarray.

4. Suppose you are given a sorted array, A, of n distinct integers in the range from 1 to n+1, so there is exactly one integer in this range missing from A. Give an O(log n)-time algorithm for finding the integer in this range that is not in A. Hint: the algorithm resembles binary search

(a) (4 marks) Give the algorithm pseudocode.

(b) (1 mark) Justify your big-Oh (1 mark).

5. (4 marks) Fill a table showing a series of following queue operations and their effects on an initially empty queue *Q* of integer objects. Here *Q* is implemented with an Array of size 7.

Operation	**Output Q**
enqueue (4)	4, -, -, -, -, -, -
dequeue ()	<4> -, -, -, -, -, -, -
dequeue ()	<error message> -, -, -, -, -, -, -
enqueue (44)	-, 44, -, -, -, -, -
enqueue (7)	-, 44, 7, -, -, -, -
enqueue (6)	-, 44, 7, 6, -, -, -
dequeue ()	<44> -, -, 7, 6, -, -, -
isEmpty()	Return False -, -, 7, 6, -, -, -
enqueue (3)	-, -, 7, 6, 3, -, -
enqueue(5)	-, -, 7, 6, 3, 5, -
dequeue ()	Return 7 -, -, -, 6, 3, 5, -

dequeue ()	Return 6, -, -, -, -, 3, 5, -
dequeue ()	Return 3, -, -, -, -, -, 5, -
dequeue ()	Return 5, -, -, -, -, -, -, -
enqueue(32)	-, -, -, -, -, -, 32
enqueue(39)	39, -, -, -, -, -, 32
enqueue(9)	39, 9, -, -, -, -, 32
size()	Return 3. (7-6+2) mod 7. 39, 9, -, -, -, -, 32
enqueue (32)	39, 9, 32, -, -, -, 32
size()	"Return 4" (7-6+3) mod 7. 39, 9, 32, -, -, -, 32
dequeue ()	"Return 32" 39, 9, 32, -, -, -, -
enqueue (6)	39, 9, 32, 6, -, -, -
enqueue (5)	39, 9, 32, 6, 5, -, -
Dequeue ()	Return 39 -, 9, 32, 6, 5, -, -
front()	Return 9
size()	Return 4. (7-5+1)mod 7
enqueue (9)	-, 9, 32, 6, 5, 9, -

6. (2 marks) Give an example of a positive function $f(n)$ such that $f(n)$ is neither $O(n^2)$ nor $\Omega(n^2)$. Explain both assertions.

7. (3 marks) Give a big-Oh characterization, in terms of *n*, of the running time of the following method. Show your analysis!

```
public void Ex(int n)
int a = 1;
for (int i = 0 ; i < n*n ; i++)
  for (int j = 0; j <= i; j++)
      if( a <= j)
        a = i;
}
```

8. Give a big-Oh characterization (in terms of the number *n* of elements stored in the queue) of the running time of the following methods. Show your analysis!

   (a) (4 marks) Describe how to implement the queue ADT using two stacks. That is: write pseudocode algorithms which implement the *enqueue()* and *dequeue()* methods of the queue using the methods of the stack.

   (b) (1 mark) What are the running times of your *dequeue()* and *enqueue()* algorithms?

CSI 2110.

Assignment 1.

STId 8136867

assume algorithm $X \Rightarrow$ running time of $f(x)$

1. (a). The best case running time. of $f(x)$. is constant. For. array has no element in it. It denotes, as. $\Omega(f(n)) = C$. or $O(1)$.

(b) The worse case running time. of $f(x)$ is $O(n)$. For algorithm. runs at $\begin{cases} O(n) \text{ odd} \\ O(\log n) \text{ ev} \end{cases}$

we can conclude the worse running case obtain when array has both. odd and even. number. Thus. the $T(f(x)) = O(n) + O(\log$

$O(T(n)) = O(n)$.

2. prove. with $f(n)$ is $O(g(n))$.

(a) to prove $O(f(n)) = 7n^3 + 3n^2 - 2n + 100$ is $O(n^3)$.

We assume exist such a function $g(x) = Cn^3$ that satisfy a constant C and n_0 such that $f(n) = 7n^3 + 3n^2 - 2n + 100 \leq Cn^3$. for all $n \geq n_0$ and $C > 0$.

Since. Hypothesis $7n^3 + 3n^2 - 2n + 100 \leq Cn^3$ any $C \geq 1$ satisfied

$3n^3 \geq 3n^2$

$2n^3 \geq 2n$.

$100n^3 \geq 100$

$f(n) \leq 7n^3 + 3n^3 + 2n^3 + 100n^3 = 112n^3$ for all $n \geq 1$.

Since $C = 112$ and $n_0 \geq 1$. $\Rightarrow f(n) = O(n^3)$.

(b). To prove $O(f(n)) = (n^2+1)/(n+1)$ is $O(n)$.

We assume exist such a function $g(x) = C(n+1)$ that satisfy a constant c and n_0 such that: Hypothesis. $f(n) = (n^2+1)/(n+1) \leq Cn$. for all $n \geq n_0$. $C > 0$.

Since. $\frac{n^2+1}{n+1} \leq C \cdot (n+1) \Rightarrow n^2+1 \leq C(n+1)^2$

when $C = 1$ $\Rightarrow n^2+1 \leq C(n^2+2n+1)$.

$\because n^2+1 \leq n^2+2n+1$ so. $C = 1$ and $n_0 \geq 0$. $O(f(n)) = (n^2+1)/(n+1)$ is $O(g$

$O(g(n)) = O(n^2+2n+1) = O(n^2)$

The. statement is true, there exist a value C that $f(n) \leq O(g($

(c) to prove $O(f(n)) = n!$ is $O(n^n)$.

we assume exist such a function $g(x) = Cn^n$ that satisfy two constant C and n_0. Such that: $f(n) = n! \leq Cn^n$ for all $n \geq n_0$. and $n \geq 0$

Since.

$\because n! = n(n-1)!$ assume: $n(n-1)! \leq Cn^n$

Since $\ln(n!) = \sum_{j=1}^{n} \ln j$

$= \int_1^n \ln x \, dx \rightarrow \ln(n!) \leq \ln C + n\ln(n)$.

$= n\ln n - (n+1) \rightarrow \ln(n!) \leq \ln C + n\ln(n)$.

$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{if } n > 0 \end{cases}$

$\rightarrow n\ln n - (n+1) \leq \ln C + n\ln(n)$.

$\rightarrow -(n+1) \leq \ln C$.

$\rightarrow e^{-(n+1)} \leq C$. for. $n \geq 0$.

The function $f(n)$. is $O(g(n))$ if and only when $C \geq e^{-1}$ for arbitrary. $n_0 \geq \ln C - 1$.

(d) Prove $f(n) = \log_2 n$ is $O(\log_{10} n)$.

we assume there is a function $g(n) = c \cdot \log_{10} n$ that satify.
two constant c and n_0 is greater or equal to 1. such that.

$$O(f(n)) \leq O(g(n))$$

Since:

$$\log_2 n \leq c \log_{10} n$$

$\because a = b^{\log_b a}$

$\log_b a = \log$

$$\Rightarrow \frac{1}{\log_n 2} \leq \frac{c}{\log_n 10} \leftarrow \text{we assume } 2^b = 10 \quad \therefore b = \log_2 10$$

$$\Rightarrow \frac{1}{\log_n 2} \leq \frac{c}{\log_n 2^b} \quad \because \log_n 10 = \log_n 2^{3.3219}$$

$$\Rightarrow \log_n 2 \frac{1}{\log_n 2} \leq \frac{c}{b} \cdot \frac{1}{\log_n 2} \cdot \log_n 2$$

$$1 \leq \frac{c}{b}$$

$$b = \log_2 10 = 3.321928 \leq c.$$

The statement $O(f(n))$ is $O(g(n))$ is valate.
if and only if when c is.
$c \geq 3.321928 = \log_2 10$.
But if and only if when.
$\log_n 2 \geq 0 \ \forall n_0 \geq 1$

(e). Prove $f(n) = n^3$ is not $O(100n^2)$.

proving with method of negation → prove such that $f(n) = n^3$ is $O(100n^2)$.
⇒ we assume there is a function $g(n) = 100n^2$ that satisfy two constant $c \geq 1$ $n \geq n_0$ such that

$$f(n) \geq c\, 100n^2.$$

$$\Rightarrow n^3 \geq c\, 100n^2$$

$$\Rightarrow n \geq c100.$$

$$\therefore c = \frac{n}{100}.$$

$\forall n_0 > n_0$. $O(100n^2) \geq O(f(n))$.

∴ thus the statement violate the condition. for

$$\therefore O(f(n)) \geq O(100n^2).$$

(f). $f(n) = 2^{n+1}$ prove is $\Theta(2^n)$.

we assume: there exist a function. $c_1 g(n) \le f(n) \le c_2 g(n)$. for $g(n) = 2^n$.

Such that exist a constant c_i and n_0 that is $C \ge 0$ and. $n_0 \ge 1$

since. $c_1 2^n \le 2^{n+1} \le c_2 2^n$ $\qquad$ $\log b \Rightarrow \log_2 b$.

$\ln c_1 + n\log 2 \le (n+1)\log 2 \le \log c_2 + n\log 2$

$\Rightarrow \underbrace{\log c_1 + n \le (n+1)}\underbrace{\le \log c_2 + n}$.

$\Rightarrow \log c_1 \le 1 \qquad \Rightarrow 1 \le \log c_2$

$\Rightarrow c_1 \le 2. \qquad \Rightarrow 2 \le c_2.$

$\Rightarrow c_1 = 1. \qquad \Rightarrow c_2 = 2.$

inconclusion. There exist two constant c_1 and that satisfy $c_1 g(n) \le f(n) \le c_2 g(n)$ for $c_1 = 1$ and $c_2 = 2$. for arbitrary $n_0 \ge 1$

$g(n) \le f(n) \le 2g(n)$

so. $\left.\begin{array}{l}\lceil f(n) \rceil = 2g(n) \\ \lfloor f(n) \rfloor = g(n)\end{array}\right\} \Theta(f(n)) = \Theta(2^n)$

3. (a). find the largest subarray → Pseudocode.

algorth X (Array size of N A)

```
1.  F = 0.  MS = 0 ;   i = 0   j = 0   s = 0   E = 0
2.  for k ← 1 to N.
3.      if( A[k-1] < a[k] )
4.          if( F != 1 ).
5.              s = k-1.  F = 1
6.          if ( k == N - 1 )
7.              e = N - 1.
8.      else
9.          if (F == 1)
10.             end = k - 1.
11.             F = 0.
12.     if (Ms < e - s )
13.         Ms = e - s.
14.         i = s
15.         j = e.
16  return. i and j
```

(b) → line 2 n. frome line 3 to 15 has total runtime of 12.

So the total runtime is.

$n \cdot 12 = 12n$

And the running time complexity of this function X is $O(f(n) = 12n) = O(n)$.