

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 $\Theta(n)$ extra memory your solution 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. $(7-6+2) \bmod 7$. 39, 9, -, -, -, -, 32
enqueue (32)	39, 9, 32, -, -, -, 32
size()	“Return 4” $(7-6+3) \bmod 7$. 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. $(7-5+1) \bmod 7$
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?

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 $T(f(x)) = O(n) + O(\log n)$
 $O(T(n)) = O(n)$.
2. prove with $f(n)$ is $O(g(n))$.

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

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

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

$$100n^3 \geq 100$$

$$\left. \begin{array}{l} 3n^3 \geq 3n^2 \\ 2n^3 \geq 2n \\ 100n^3 \geq 100 \end{array} \right\} f(n) \leq 7n^3 + 3n^3 + 2n^3 + 100n^3 = 112n^3 \text{ for all } n \geq 1.$$

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

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

$$\text{when } C=1 \Rightarrow n^2+1 \leq C(n^2+2n+1)$$

$$\therefore n^2+1 \leq n^2+2n+1 \text{ so } C=1 \text{ and } n_0 \geq 0. O(f(n)) = (n^2+1)/(n+1) \text{ is } O(n)$$

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

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

$$n! = n(n-1)!$$

assume:

$$n(n-1)! \leq Cn^n$$

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

$$= \int_1^n \ln x dx$$

$$= n \ln n - (n+1)$$

$$\rightarrow \ln(n!) \leq \ln C + n \ln n$$

$$\rightarrow \ln(n!) \leq \ln C + n \ln n$$

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

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

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

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

The function $f(n)$ is $O(g(n))$ if and only when $C \geq e^7$ 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 satisfy two constant c and n_0 is greater or equal to 1. such that

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

Since:

$$\begin{aligned} \because a &= b^{\log_b a} \\ \log_b a &= \log \\ \log_2 n &\leq c \log_{10} n \\ \Rightarrow \frac{1}{\log_n 2} &\leq \frac{c}{\log_n 10} \quad \leftarrow \text{we assume } 2^b = 10 \\ &\quad b = \log_2 10 \\ \Rightarrow \frac{1}{\log_n 2} &\leq \frac{c}{\log_n 2^b} \quad \leftarrow \log_n 10 = \log_n 2^{3.3219} \\ \Rightarrow \log_n^2 \frac{1}{\log_n 2} &\leq \frac{c}{b} \cdot \frac{1}{\log_n 2} \cdot \log_n^2 \end{aligned}$$

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

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

The statement $O(f(n))$ is $O(g(n))$ is false.

if and only if when c is

$$c \geq 3.321928 = \log_2 10$$

But if and only if when

$$\log_n^2 \geq 0 \quad \forall n_0 \geq 1$$

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

proving with method of negation \rightarrow prove such that $f(n) = n^3$ is $O(100n^2)$.

\Rightarrow we assume there is a function $g(n) = 100n^2$ that satisfy two

constant $c \geq 1$ $n \geq n_0$ such that

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

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

$$\Rightarrow n \geq c 100$$

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

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

\therefore thus the statement violate the condition. for

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

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

we assume: There exist a function $g(n)$ such that $C_1 g(n) \leq f(n) \leq C_2 g(n)$. for $g(n) = 2^n$.
such that exist a constant C_i and n_0 that is $C \geq 0$ and $n_0 \geq 1$
since $C_1 2^n \leq 2^{n+1} \leq C_2 2^n$ $\log b \Rightarrow \log_2 b$.

$$\ln C_1 \log 2 \leq (n+1) \log 2 \leq \log C_2 + n \log 2$$

$$\Rightarrow \log C_1 + n \leq (n+1) \leq \log C_2 + n$$

$$\Rightarrow \log C_1 \leq 1 \Rightarrow 1 \leq \log C_2$$

$$\Rightarrow C_1 \leq 2 \Rightarrow 2 \leq C_2$$

$$\Rightarrow C_1 = 1 \Rightarrow C_2 = 2 \quad \square$$

in conclusion. There exist two constant C_1 and C_2 that satisfy $C_1 g(n) \leq f(n) \leq C_2 g(n)$
for $C_1 = 1$ and $C_2 = 2$. for arbitrary $n \geq 1$.
 $g(n) \leq f(n) \leq 2g(n)$.

$$\text{so } \lceil f(n) \rceil = 2g(n) \quad \Theta(f(n)) = \Theta(2^n)$$

$$\lfloor f(n) \rfloor = g(n)$$

3. (a). find the largest subarray \rightarrow Pseudocode.

algorith X. (Array size of N A)

1. $F = 0$. $MS = 0$; $i = 0$. $j = 0$ $S = 0$ $E = 0$.

2. for $k \leftarrow 1$ to N . (b) \rightarrow line 2. n . from line 3 to 15 has

3. if $(A[k-1] < A[k])$ total runtime of 12.

4. if $(F \neq 1)$. So the total runtime is.

5. $S = k-1$. $F = 1$; $n \cdot 12 = 12n$.

6. if $(k == N-1)$ And the running time complexity of.

7. $E = N-1$; this function X is $O(f(n) = 12n) = O(n)$.

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

4. (a) findME ([] a size n, [] range r size = m) . (b)

1. low = 0. high = n-1

2. while (low <= high) {

3. mid = (low + high) / 2

4. if (a[mid] > r[mid])

5. high = mid - 1

6. else if (a[mid] < r[mid])

7. low = mid + 1

8. else

9. return r[mid]

Since the range the function run in with $\log_2 n$.

Since the array divide in half every time.

5. On the document.

6. \therefore assume $f(n)$ is neither $O(n^2)$ nor $\Omega(n^2)$.

H₁ There is not existing a value C_1 that satisfy $f(n) \leq C_1 O(n^2)$ for $C_1 \geq 1$

H₂ There is not existing a value C_2 that satisfy $C_2 n^2 \leq f(n)$ for $C_2 \geq 1$

Assumption: $f(n) = n^3 (\sin(n))$ is a example for this question

\rightarrow prove $n^3 (\sin(n)) \leq C n^2$ for all $C \geq n^3 (\sin(n))$ $n_0 > 0$

\Rightarrow if $C = 2$ $2^3 (\sin(2)) \rightarrow 2 \cdot 2^2$

\Rightarrow thus we conclude $O(f(n)) \neq O(n^2)$

\Rightarrow prove $n^3 (\sin(n)) \geq C n^2 \forall C < n^3 (\sin(n))$ $n_0 > 0$

\Rightarrow if $C = 4$ $C n^2 > n^3 (\sin(n))$

\Rightarrow thus we conclude $\Omega(f(n)) \neq \Omega(n^2) \therefore f(n)$ is neither $O(n^2)$ nor Ω

7. 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 = j

Analysis n=0 Table of substitution

① As n=1 $T(1) = 2$ $1 \times (1+1)$

n=2 $T(2) = 20$ $2^2 (2^2+1)$

n=3 $T(3) = 90$ $3^2 \times (3^2+1)$

n=4 $T(4) = 272$ $4^2 \times (4^2+1)$

n=5 $T(5) = 650$ $5^2 \times (5^2+1)$

For analyzing the 1st for loop run

total $n \cdot n$ times = n^2

And the 2nd for loop runs

in the size $n+1$

$$n = n \quad T(n) = \frac{n^2 \times (n^2 + 1)}{2} \\ = (n^4 + n^2) / 2$$

$$O(T(n)) = O(n^4)$$

Thus the function of running time \therefore n

$$\& \text{ Ex is } \frac{n^2(n^2+1)}{2}$$

Q8. given variable. Stack inbox, Stack outbox.

(a) enqueue(e) {
 inbox.push(e) }

dequeue() {
 if outbox is empty.
 push all of the element that pop
 from inbox.

Return outbox.pop(). }

(b) enqueue(e). cost $O(1)$.

dequeue() cost $O(n)$.
for stack pop n times to
get to the first element.