Name: David Xie

ID: 40065595

Date: September 25, 2020

COMP352 Assignment 1

Written Questions

Question 1

```
Algorithm reorder(A, n)
    Input: array A of n integers
    Output: reordered array

    temp ← 0
    check ← 0
    for i ← 1 to n do
        if A[check] % 2 = 0 then
            check ← check + 1
        else
            for j ← check to n-2 do
                temp ← A[j]
                A[j] ← A[j+1]
                A[j+1] ← temp
    return A
```

a) Time complexity: $O(n^2)$

b) Space complexity: $O(n)$

Question 2

```
Algorithm products(A, n)
    Input: array A of n integers
    Output: largest and smallest products of the array

    max1 ← 0
    min1 ← 0
    for i ← 1 to n-1 do
        if A[i] > A[max1] then
            max1 ← i
        if A[i] < A[min1] then
            min1 ← i
```

```
max2 ← 0
min2 ← 0
if max1 = max2 then
    max2 ← 1
if min1 = min2 then
    min2 ← 1
for i ← 0 to n-1 do
    if A[max1]-A[i] < A[max1]-A[max2] and max1 ≠ i then
        max2 ← i
    if A[i]-A[min1] < A[min2]-A[min1] and min1 ≠ i then
        min2 ← i

print the indexes and values of the two largest integers and its product
print the indexes and values of the two smallest integers and its product
```

a) The motives behind this design is to effectively find the indexes of the two largest and two smallest integers of an array of integers, using as little nested for loops as possible. Nested for loops would increase time complexity, which is why they were preferably avoided. The first for loop finds the index of the largest and the smallest integers in the array. The if statements following instantiation of the indexes of the second largest and smallest values are to ensure that they were not the same as the first indexes. The second for loop finds the second largest and smallest integers by comparing the differences between the values of integers. Finally, the design prints the indexes, values, and the product.

b) The time complexity is O(n). All primitive operations such as value assignments are O(1). However, the operations within the for loops are O(n). Big-O notation takes the worst case scenario, in this case, being O(n).

Question 3

a) f(n) is O(g(n)) if there is a c > 0 and a $n_0 \geq 1$ such that $f(n) \leq c \times g(n)$, for $n \geq n_0$

Here: $f(n) = 8000000n^2 \log(n) + n^3$, and $g(n) = n^3 \log(n)$

Want to show $f(n) \leq c \times g(n)$: $8000000n^2 \log(n) + n^3 \leq c \times n^3 \log(n)$

$$8000000n^2 \log(n) \leq 8000000n^3 \log(n) \;\; for\; n \geq 1$$
$$n^3 \leq n^3 \log(n) \;\; for\; n \geq 2$$

Conclusion: $8000000n^2\log(n) + n^3 \leq 8000001n^3\log(n)$ where $c = 8000001$ and $n_0 = 2$, therefore this proves the statement that f(n) IS O(g(n)).

b) f(n) is Θ(g(n)) if f(n) is f(n) is O(g(n)) and f(n) is $\Omega\big(g(n)\big)$; There are constants $c' > 0$ and $c'' > 0$, and an integer constant $n_0 \geq 1$ such that $c'g(n) \leq f(n) \leq c''g(n)$, for $n \geq n_0$

Here: $f(n) = 10^6n^2 + 3n^7 + 5n^3$, and $g(n) = n^3$

Want to show: $c'n^3 \leq 10^6n^2 + 3n^7 + 5n^3 \leq c''n^3$

However: $10^6n^2 + 3n^7 + 5n^3 \nleq c''n^3$ taking any $c'' > 0$ and any $n_0 \geq 1$, for all $n \geq n_0$ because n^7 grows faster than n^3 so eventually, the statement we want to prove will be false

Conclusion: f(n) is NOT $\Theta(g(n))$

c) f(n) is Θ(g(n)) if f(n) is f(n) is O(g(n)) and f(n) is $\Omega\big(g(n)\big)$; There are constants $c' > 0$ and $c'' > 0$, and an integer constant $n_0 \geq 1$ such that $c'g(n) \leq f(n) \leq c''g(n)$, for $n \geq n_0$

Here: $f(n) = 0.1n^3 + 0.0000005n^6$ and $g(n) = n^3$

Want to show: $c'n^3 \leq 0.1n^3 + 0.0000005n^6 \leq c''n^3$

However: $0.1n^3 + 0.0000005n^6 \nleq c''n^3$ taking any $c'' > 0$ and any $n_0 \geq 1$, for all $n \geq n_0$, because n^6 grows at a faster rate than n^3 so eventually, at an $n \geq n_0$, the statement we want to prove will be false

Conclusion: f(n) is NOT $\Theta(g(n))$

d) f(n) is $\Omega(g(n))$ if g(n) is O(f(n)); There is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq cg(n)$, for $n \geq n_0$

Here: $f(n) = n^4 + 0.0000001n^3$ and $g(n) = n^3$

Want to show: $n^4 + 0.0000001n^3 \geq cn^3$

Consider: $n^4 + 0.0000001n^3 \geq n^3$, for $c = 1$ and $n \geq 1$; Statement is true since n^4 grows at a faster rate than n^3

Conclusion: f(n) IS $\Omega(g(n))$

e) f(n) is Θ(g(n)) if f(n) is f(n) is O(g(n)) and f(n) is $\Omega\big(g(n)\big)$; There are constants $c' > 0$ and $c'' > 0$, and an integer constant $n_0 \geq 1$ such that $c'g(n) \leq f(n) \leq c''g(n)$, for $n \geq n_0$

<u>Here</u>: $f(n) = n!$ and $g(n) = 2^n$

Want to show: $c' \times 2^n \leq n! \leq c'' \times 2^n$

However: $n! \nleq c'' \times 2^n$ because n! grows at a faster rate than 2^n, therefore, eventually, at a certain $n \geq n_0$, the statement we want to prove will be false.

<u>Conclusion</u>: f(n) is NOT $\Theta(g(n))$

f) f(n) is $\Omega(g(n))$ if g(n) is O(f(n)); There is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq cg(n)$, for $n \geq n_0$

<u>Here</u>: $f(n) = n^n$ and $g(n) = n!$

Want to show: $n^n \geq cn!$

Consider: $n^n \geq n!$ where $c = 1$ and $n \geq 1$; Statement is true since n^n grows at a faster rate than n!

<u>Conclusion</u>: f(n) IS $\Omega(g(n))$