## Design and Analysis of Algorithms

## CS575 Spring 2020

## Answers to Theory Assignment 1

1. [20 points] Fill in all the missing values. For column A you need to compute the sums. For column B (the last two rows) you need to guess a function that does not contradict any of the yes/no answers already in the next three columns. Fill in each empty entry in the last three columns with either a yes/no answer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function | Function | O | Ω | Θ |
| A | B | A = O(B) | A = Ω(B) | A = Θ(B) |
| *n*4 | *n*3 lg *n* | no | yes | no |
| *n* | *n*2 | yes | no | no |
| (*n*+1)! | *n*! | no | yes | no |
| *lg n* | *nk* where *k* > 0 | yes | no | no |
|  | *n*2 | yes | yes | yes |
| *=* | *n*4 (or any polynomial) | no | yes | no |

1. [20 points] Prove the following using the original definitions of O, , , o,
   1. 3n3 + 50n2 + 4n - 9 O(n3)

Answer: For n 3n3 + 50n2 + 4n – 93n3 + 50n3 + 4n3 =57 n3. Thus, 3n3 + 50n2 + 4n - 9 O(n3)

* 1. 1000n3 (n2)

Answer: For n ≥ 1, 1000 n3 ≥ n2. Thus, 1000n3 (n2)

* 1. 10n3 +7n2 (n2)

To prove this, we need to show that 10n3 +7n2 ≥ c n2 for any c ≥ 0. If we divide both sides of the inequality by n2, we get 10n + 7 ≥ c , i.e., n ≥ (c-7)/10. Thus, pick any N≥ max((c-7)/10, 1) that conclude the proof. (For any c, you can pick an N value that satisﬁes the inequality.)

* 1. 78n3 o(n4)

To prove this, we need to show that 78n3≤ c n4 for any c ≥ 0. If we divide both sides of the inequality by n3, we get 78 ≤ cn, i.e. 78/c ≤ n Thus, pick any N ≥ 78/c to conclude the proof. (For any c, you can pick an N value that satisﬁes the inequality.)

* 1. 
     1. First, *n*2 + 3*n* – 10 ≤ *n*2 + 3*n* ≤ 4*n*2 for all *n* ≥ 1, that is *n*2 + 3*n* – 10 ≤ *cn*2 for constants *c* = 4 and *n*0 = 1 and *n* ≥ *n*0. Therefore, *n*2 + 3*n* – 10 ∈ *O*(*n*2)
     2. Next, *n*2 + 3*n* – 10 = *n*2 + (3*n* – 10) ≥ *n*2 when 3*n* – 10 ≥ 0, which holds when *n* ≥ 4. That is *n*2 + 3*n* – 10 ≥ *cn*2 for constants *c* = 1 and *n*0 = 4 and *n* ≥ *n*0. Therefore, *n*2 + 3*n* – 10 ∈ Ω(*n*2).

In summary, for c1 = 1, *c*2 = 4 and *n*0 = 4, we have *c*1*n*2 ≤ *n*2 + 3*n* – 10 ≤ *c*2*n*2 when *n* ≥ *n*0. Therefore, *n*2 + 3*n* – 10 ∈ Θ(*n*2) according to the definition of Θ.

1. [15 points] Prove the following using limits.
2. *n*1/*n* Θ(1) [Hint: you can use x=elnx ]

Answer: *n*1/*n*=. =0.

Or you can use the following:

Let *x* = *n*1/*n*. Then lg *x* = (1/*n*)lg *n* = (lg *n*) / *n*. We know This means that when *n* becomes infinity, lg *x* = 0, which in turn implies that *x* = 1 when *n* becomes infinity, 🡺

1. (nk)

= =

=

=

=



Answer: ) lim[lg3n/n0.5] = lim[(3lg2n/(nln2)) / (0.5/n0.5)] = C1\*lim[lg2n/n0.5] = … = C2lim[lg n/n0.5]= … = C3lim[1/n0.5] = 0, and from Theorem 1.3 this implies that lg3n ∈ o(n0.5).

1. [10 points] Order the functions below by increasing growth rates (no justification required):

*nn*, *n*, *n*ln *n*,, , ln *n,* 10, *n*1/*n*, *n*!, *lg(n10)*, 2*n*

Let *gi*(*n*) be the *i*th function from the left after the ordering (the leftmost function has the slowest growth rate). In the order, *gi*(*n*) should satisfy *gi*(*n*) ∈ *O*(*gi*+1(*n*)). If two or more functions are equivalent (in terms of Θ), put them in [ ] separated by comma (e.g., [*n*2, 5*n*2]).

**Answer**: The order is: [10, *n*1/*n*], [ln *n*, *lg(n10)*] ,], [*n*, (= *n*lg 2 = *n*)], *n*ln *n*, 2*n*, *n*!, *nn*

=

Both 10 and *n*1/*n* are in Θ(1). It can be shown that Let *x* = *n*1/*n*. Then lg *x* = (1/*n*)lg *n* = (lg *n*) / *n*. We know This means that when *n* becomes infinity, lg *x* = 0, which in turn implies that *x* = 1 when *n* becomes infinity, 🡺

That n! ∈ *O*(*nn*) can be shown using Stirling’s Approximation. From this approximation, we know n! ≤ . Thus . Therefore, n! ∈ *O*(*nn*).

The other relationships are either easy to see or have been discussed in class.

1. [20 points] Let *f*(*n*) and *g*(*n*) be asymptotically positive functions. For each of the following conjectures, either prove it is true or provide a counter example to show it is not true.
   1. .
   2. .
   3. .
   4. implies where lg(g(n))1 and 1 for sufficiently large n.

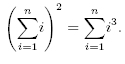
**Answer**: a) is true. Proof: It is easy to see that max(*f*(*n*), *g*(*n*)) ≤ (*f*(*n*) + *g*(*n*)) ≤ 2\*max(*f*(*n*), *g*(*n*)) for any *n*. Based on the definition of Θ,  is true.

b) is not true. A counter example is as follows. Let *f*(*n*) = 2*n* and *g*(*n*) = *n*. Then *f*(*n*) ∈ *O*(g(*n*)) is true. However, 2*f*(*n*) = 22*n* = 4*n* grows faster than 2*g*(*n*) = 2*n*. So 2*f*(*n*) ∈ *O*(2*g*(*n*) ) is not true.

c) is true. Proof: If *f*(*n*) ∈ *O*(*g*(*n*)), then there exist positive constants c and *n*0, such that *f*(*n*) ≤ *cg*(*n*)) for all *n* ≥ *n*0. This means that there is a positive constant *c*1 = 1/*c* and constant *n*0 which make *g*(*n*) ≥ (1/*c*) *f*(*n*) = *c*1 *f*(*n*) for *n* ≥ *n*0, that is, *g*(*n*) ∈ Ω(*f*(*n*)) is true.

d) is true. Proof: If *f*(*n*) ∈ *O*(*g*(*n*)), then there exist positive constants c and *n*0, such that *f*(*n*) ≤ *cg*(*n*)) for all *n* ≥ *n*0. lg(*f*(*n*)) ≤ lg(*c) + lg(g*(*n*)) ≤ lg(*c)lg(g(n)) + lg(g*(*n*)) ≤ (lg(*c) +1) lg(g*(*n*))

1. [10 points] Prove that for all integers n>0,

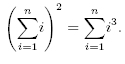


by mathematical induction. Divide your proof into the three required parts: Induction Base, Induction Hypothesis, and Induction Steps.

**Answer**:

**Induction base**: For n=1, (1)2=13.

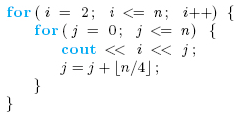
**Induction hypothesis**: Assume for arbitrary positive integer n, that



**Induction step: We need to show**

To that end,

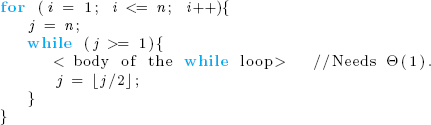
1. [15 points] Consider the following algorithm:



1. What is the output when n=4?
2. What is the time complexity T(n). You may assume that n is divisible by 4.

Answer:

1. n = 4: 2 0 2 1 2 2 2 3 2 4 3 0 3 1 3 2 3 3 3 4 4 0 4 1 4 2 4 3 4 4
2. T(n) = (n-1)(5) ∈ O(n) because the inner loop will always have 5 loops.
3. [10 points] What is the time complexity T(n) of the nested loops below? For simplicity, you may assume that n is a power of 2. That is, n = 2k for some positive integer k. Give some justification for your answer.



Answer:

T(n) = n\*log n = O(n\*log n)