

Faculty of Science

**Course**: CSCI 3070U: Design and Analysis of Algorithms

**Lab:** #1

**Topic:** Running times, induction, asymptotic notation

1. Complete Problem 1-1 on p.15 of your textbook. You can use the table below, rather than mark up your textbook:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 second | 1 minute | 1 hour |
| lg n | 21000 | 260,000 | 23,600,000 |
|  | 1,000,000 | 3,600,000,000 | 12,960,000,000,000 |
| n | 1000 | 60,000 | 3,600,000 |
| n log n | 400 | 10,000 | 700,000 |
| n2 | 31 | 245 | 1900 |
| n3 | 10 | 39 | 153 |
| 2n | 10 | 16 | 25 |
| n! | 6 | 8 | 10 |

1. Using the pseudocode definition of reverse() below, which takes a list and returns the same list in reverse order, prove using structural induction that length(x) = length(reverse(x)).

REVERSE(List)

1. **if** (List == []) **then**

2. **return** []

3. Rest = List[2..]

4. **return** append(REVERSE(Rest), List[1])

Where:

* List[2..] returns a list with the same elements as List, but with the first element removed
* append(x, y) takes a list (x) and an element (y), and appends y to the end of x

Base: n = 0

Let L = []

REVERSE(L) = []

Intuitively, length(REVERSE(L)) = length(L)

Assume: For n=k, let Lk be a list of length k

Assume that length(REVERSE(Lk)) = length(Lk) is true

Prove: For n=k+1, let Lk+1 be a list such that the last k elements are equivalent to Lk, and an element exists at the start of the list, x0.

length(Lk+1) (by the above definition of Lk+1)

= length(Lk) + 1

length(REVERSE(Lk+1))

= length(append(Lk, x0)) (by the definition of REVERSE and [2..])

= length(Lk) + 1 (by the claim made about append())

1. For the above function, REVERSE, write T(n) as a function of n (the size of List) using a number of constant values for each operation being performed.

Where:

* List[2..] will take some constant (e.g. c3) multiplied by n operations
* Append(x, y) will take some constant (e.g. c4) multiplied by n operations

T(n) = c1 + c2 + c3n + c4n + c5T(n-1)