CMPT 280

Timing analysis of Recursive Algorithms

Mark G. Eramian

University of Saskatchewan

Based on notes by G. Cheston and J. P. Tremblay

Timing Analysis for Recursive Algorithms

- If we define

  $$T_p^R(j) = \text{the time for the } j\text{-th recurive call of method } p$$

- Then the total time used by method p is:

  $$T_p = \sum_{j=1}^{k} T_p^R(j)$$

  where k is the number of recursive calls.

- If T_p^R has the same order for all calls, it can be factored out of the sum:

  $$T_p = \sum_{j=1}^{k} T_p^R(j) = T_p^R \cdot \sum_{j=1}^{k} 1 = T_p^R \cdot k$$

Timing Analysis for Recursive Algorithms

In other words:

- If $T_p^R(j)$ is constant and does not vary with j then the analysis is fairly easy.
- In such case, $T_p^R(j)$ can be written independently of j as just T_p^R (which is expressed in big-O notation).
- Then if k is the number of recursive calls we just multiply k by the cost of the recursive call and we have our answer:

$$k \cdot T_p^R$$

- If $T_p^R(j)$ varies with j we need to use the summation notation as on previous slide and compute the answer similarly to the way we did for dependent quadratic loops.

Practice

- Compute the worst-case time complexity of the traversal methods we added to `LinkedSimpleTree280<I>` in Lecture 08.
  - Exercise 1: Print the nodes with a pre-order traversal.
  - Exercise 2: In-order traversal.
  - Exercise 3: Post-order traversal.
  - Exercise 4: Count the number of nodes (with post-order).
  - Exercise 5: Count the height of the tree (with post-order).
  - Exercise 6: Print the nodes in level-order.
- All except one of them are recursive...

Timing Analysis for Recursive Algorithms

Keys to timing analysis of recursive algorithms:

- What is $T_p^R(j)$ for algorithm p?
- Does it vary with j?
- How many calls are done?