CSI 33 LECTURE NOTES (Ojakian)

Topic 3: Algorithm Analysis

OUTLINE
(References: Ch 1.3)

1. Algorithm Efficiency: empirical and abstract
2. Binary Search

---

1. Empirical Timing of Programs

PROBLEM 1. *Run the Python and C++ programs in the two files: PythonLoops.py and CPPLoops.cpp. Do the following:*

(a) Within each language compare the runs of the various loops.

(b) Between Python and C++ compare the runs of similar loops.

(c) Can you explain the differences? How do the timed results meet expectations and not expectations?

2. Theoretical Algorithm Analysis - Big O and Big Theta

(a) Different names that roughly mean the same thing:

i. Algorithm Analysis
ii. Complexity Analysis
iii. Theta-Analysis
iv. Asymptotic Analysis - captures idea: care about run time as input size tends to infinity.

(b) Which Cost Model?

i. *Uniform Cost Model*: All basic operations (like arithmetic) take the same constant amount of time (*We will use this one unless otherwise specified!*)
ii. *Logarithmic Cost Model*: Basic operations take time proportional to their length.

(c) Definition (for functions) of Big O, Big Omega, and Big Theta

i.

PROBLEM 2. *Let* $f(n) = 10n + 2000$ *and let* $g(n) = n$.
Show that: $f(n)$ *is* $O(g(n))$.

PROBLEM 3. *Show that* $5n^2 + 100n$ *is* $\Theta(n^2)$.

ii. See table from Power Point Chap 1, p.36.

PROBLEM 4. *Find crossing points and try graphing ...*

(d) Application to programs.

i. Find a function describing the number of steps the program takes (in terms of the input size)
ii. Determine the Theta Category of that function

PROBLEM 5. *Do a theta analysis of the Python and C++ programs from Loop files.*

PROBLEM 6. *From chapter 1 (p.36) do exercise 8(b).*

3. Linear Search of List

PROBLEM 7. *Time linear searches on a Python list versus C++ array*

PROBLEM 8. *Do the Big Theta analysis for linear search.*

4. Binary Search

(a) Recall recursion

PROBLEM 9. *Write a recursive function for factorial.*

PROBLEM 10. *Consider writing a recursive function that takes a list as input and returns True if and only if the list is a palandrome (example:* $[2, 3, 9, 3, 2]$ *). Do it two ways in Python:*

i. First, using slicing, sending a new list into the recursion call

ii. Second, keeping the list unchanged, and maintaining the useful indices.

(b) Recall Binary Search

i. List must be sorted

ii. Follow the usual binary search algorithm, returning the index of the item or -1 if it is not found.

iii. We use the convention for middle element of: Rounding down.

iv.

PROBLEM 11. *Consider the list:* $[2, 3, 8, 9, 15, 17, 50, 60, 62, 100]$
Carry out the binary search algorithm by hand, first for 50, then for 4.

PROBLEM 12. *Study the recursive Python function for binary search.*

PROBLEM 13. *Compare run times of linear search versus binary search.*

PROBLEM 14. *Carry out an asymptotic analysis of binary search.*