CS3230 : Design and Analysis of Algorithms (Fall 2014)

Tutorial Set #1

[For discussion during Week 3]

S-Problem Due: Friday, 22-Aug, before noon.

OUT: 19-Aug-2014

Tutorials: Tue & Wed, 26-27 Aug 2014

S-Problem Due: Friday, 22-Aug, before Noon.

IMPORTANT: Read "Remarks about Homework" – also applies to tutorials.

Prepare your answers to all the D-Problems in every tutorial set.

When preparing to present your answers,

- Think of a CLEAR EXPLANATION
- Illustrate with a good worked example;
- Describe the main ideas,
- Can you sketch why the solution works;
- Give analysis of running time, if appropriate
- Can you think of other (perhaps simpler) solutions?

In CS3230, you learn to develop high-level abstractions when describing algorithms. Try not to speak in ML/AL (machine/assembly language) or "for (j=0; j<n; j++) do". Instead give names to your sets (of objects or things or data structures), talk about Depth-First Search, Binary Search, traverse the graph, sort the set, use a priority queue, etc. You are no longer in CS1010, CS1020, CS2010 or CS2020. Speak with greater sophistication, and at a higher level of abstraction.

Remember:

- You can **freely quote** standard algorithms and data structures covered in the lectures (and including from pre-requisites) modules, textbook. Explain **any modifications** you make to them, and how they may affect the running time.
- There is no need to copy/re-prove algorithms or theorems already cover already;
- Unless otherwise specified, you are expected to **prove (justify)** your results. All logarithms are base 2, unless otherwise stated.

Examples:

a. Use Quicksort to sort the array $X[1..n]$ in increasing order;
b. Organize the set S as a Max-Heap (array-based);
c. Run a post-order traversal of the tree T, and at each node, the processing of the node is …
d. Run Dijkstra's algorithm for single-source shortest path from vertex w on graph G=(V, E).
e. Do <some-std-alg Q>, but with the following modifications: blah, blah, blah….
f. By the Handshaking Lemma, $(d_1 + d_2 + d_2 + \ . . . + d_n) = 2e$
(OK, if you still don't know the Handshaking Lemma, Google it and learn it.)

Of course, it is your responsibility to ensure that the algorithm that you quote ACTUALLY solves your problem.

***Routine Practice Problems** -- do not turn these in -- but make sure you know how to do them.*

R1. **(a)** Show, by definition, that $f(n) = O(n)$, where $f(n) = 819n$.
(a') Show that $f(n) = O(n^2)$, $f(n) = O(n^{819})$.

(b) Show, by definition, that $g(n) = O(n^2)$, where $g(n) = 17n^2$.
(b') Show that $g(n) = O(n^3)$, $g(n) = O(n^{17})$.

(c) Show, by definition, that $h(n) = O(n^2)$, where $h(n) = 17n^2 + 819n$.
(c') Show that $h(n) = O(n^3)$, $h(n) = O(n^{836})$.

(d) Show, by definition, that $k(n) = O(n^2)$, where $k(n) = 17n^2 + 37n(\lg n)$
(d') Show that $g(n) = O(n^3)$, $g(n) = O(n^{54})$.

R2. **(a)** Show, by definition, that $f(n) = \Theta(n)$, where $f(n) = 819n$.
(a') Is $f(n) = \Theta(n^2)$? Is $f(n) = \Theta(n^{819})$?

(b) Show, by definition, that $g(n) = \Theta(n^2)$, where $g(n) = 17n^2$.
(b') Is $g(n) = \Theta(n)$? Is $g(n) = \Theta(n^3)$? Is $g(n) = \Theta(n^{17})$?

(c) Show, by definition, that $h(n) = \Theta(n^2)$, where $h(n) = 17n^2 + 819n$.
(c') Is $h(n) = \Theta(n)$? Is $h(n) = \Theta(n^3)$? Is $g(n) = \Theta(n^{836})$?

(d) Show, by definition, that $k(n) = \Theta(n^2)$, where $k(n) = 17n^2 + 37n(\lg n)$
(d') Is $k(n) = \Theta(n \lg n)$? Is $k(n) = \Theta(n^3)$? Is $k(n) = \Theta(n^{54})$?

R3. From **T1-R1** and **T1-R2**, can you see the big different between O-notation and Θ-notation?

S-Problems: (To do and submit by Friday, 22-Aug, before noon.)
Solve this S-problem and submit for grading.

S1: (Two Important Processes in CS)

[Repeated-halving]
Start with a number n. Repeatedly "*divide by two* (throw away the remainder)" until we reach 0. How *many steps* will you take? Let $h(n)$ be the number of steps.

[Repeated-doubling]
Start with the number 1. Repeatedly *multiply by two* until we get a number greater than or equal to n. How *many steps* will you take? Let $d(n)$ be the number of steps.

(a) [In a nice table, list the value of $h(n)$ $d(n)$ for n = 1-25, 31, 32, 33, 63, 64, 65, 100, 127, 128, 129, 100, 1023, 1024, 1025, 10^6, 10^9.]

(b) Write the repeated-halving and repeated-doubling processes in pseudo-code.

(c) Write down any relationship you see between the two processes? between $h(n)$ and $d(n)$?

(d) Give an *exact mathematical formulae* for $h(n)$ and $d(n)$.
[Hint: As a self-check, you can test the formula out yourself.]

D-Problems:

Solve these D-problems and prepare to discuss them in tutorial class. You will call upon one of you to present your solution *or your best attempt at a solution*. Your solution presentation does NOT need to be fully correct, given your best attempt. The TA will help clarify and correct any issues or errors.

D1. Discuss solution to HW0-S1. (Hint: Compare with problem T1-S1.)

D2. Discuss solution to HW0-S3.

D3. [Proving O, Ω, Θ by definition]

Let $f(n) = 14n^2 - 6n + 707$ $g(n) = 19n^2 - 707n(\lg n) + 8n - 30$.

(a) Prove the following by using the definitions of O, Ω, Θ. Namely, find the respective constants c, c_1, c_2, and the positive integer n_0.

(i) $f(n) = O(n^2)$ (ii) $f(n) = \Omega(n^2)$ (iii) $f(n) = \Theta(n^2)$

(iv) $g(n) = O(n^2)$ (v) $g(n) = \Omega(n^2)$ (vi) $g(n) = \Theta(n^2)$

(b) Prove the results in T1-D3(a) above by making use of some of the following short-cut methods: sum-rule or product-rule, by polynomial rule, L'Hopital's rule (limits), etc. (*Find the method that is easiest for you.*)

D4. [Analysis of Algorithm Ace.] Analyze the following algorithm (called Ace) and give a *good estimate* of its running time (in Θ notation).

```
Procedure Ace (N) :
    1. Set A[k]=1 for all k=2,3,...,N                   // Initialize A to all "primes"
    2. for (p := 2; p <= N; p++) do begin
    3.    if (A[p] = 1) then Print p;
    4.    for (j := 2p; j <= N; j := j + p) do A[j] :=0;  // mark j as "not prime"
    5. end
```

[**Note:** The running time is NOT $\Theta(n^2)$.]
[**Hint:** The loop in Step 4, ask how many times does it loop?]