CPSC 327, 27 February 2013 — First Test, Take-home Part

This is a test. You should not work on the test or discuss it with other people in the class. You can use your textbook and notes, but you should not use other books or resources, including the Internet. You can ask the Professor for hints or clarification. There will be time to do that in class on Friday so that everyone can hear the answers. The test is due in class next Monday, March 4.

1. (*6 points*) Apply the Master Theorem (p. 490 in the textbook) to find the order of growth of $T(n)$ for each of the following recurrence relations. (You are being asked to find the function $f(n)$ such that $T(n) \in \Theta(f(n))$.)

a) $T(1) = 3$
$T(n) = 5 * T(n/3) + n^2$, for $n = 3^k$ where $k > 0$

b) $T(1) = 2$
$T(n) = 3 * T(n/2) + n$, for $n = 2^k$ where $k > 0$

2. (*6 points*) In class, we applied the Master Theorem to the following recurrence relation:

$T(1) = 1$
$T(n) = T(n/2) + n$, for $n = 2^k$ where $k > 0$

The result was that $T(n) \in \Theta(n)$. It looks like the recursive function call contributes nothing at all to the growth rate! To see what is going on, apply forward or backward substitution to the recurrence relation to solve it directly. You should solve it *only* for $T(2^k)$, that is, only when n is a power of two. Then explain in words why $\Theta(n)$ makes sense.

3. (*9 points*) A *coloring* of a graph assigns a color to each vertex, with the property that there there is no edge that connects two vertices of the same color. Given a set of k colors, it might or might not be possible to color a given graph with that set of colors.

Outline a brute-force, exhaustive-search algorithm to search for a coloring of a graph, using **three** colors, and estimate the run time efficiency of your algorithm. (Note: In the homework problem on bipartite graphs, you encountered the idea of coloring a graph using two colors. In that case, there was a simple, efficient algorithm. However, for the case of three or more colors, there is no such simple algorithm.)

4. (*9 points*) A graph can be represented either as an adjacency matrix or as adjacency lists. For a given graph, the two representations will use different amounts of memory. For very sparse graphs, adjacency lists will use less memory. For graphs that are nearly complete, an adjacency matrix will use less memory. But for a particular graph, the answer will depend on the number of edges in that graph as well as on the details of how the matrix and lists are implemented.

Investigate the following question, and write a report on your results: Choose some particular implementations in Java. Given a graph with n vertices and e edges, which representation will use less memory? The answer depends pretty much on how big e is compared to n^2, where n^2 is approximately equal to the maximum possible number of edges. Your answer will be along the lines of: The adjacency matrix takes less memory if e is at least such-and-such a percentage of n^2; otherwise, adjacency lists use less memory. Your assignment is to produce some estimate of the cutoff percentage.

For your computation, you can assume: A *boolean* value in Java occupies one byte of memory. A Java object uses 8 bytes, in addition to the memory for the member variables that it contains. An *int* uses 4 bytes. And every pointer also uses 4 bytes.

5. (*10 points*) What are the *space* efficiencies for depth-first search and breadth-first search of a graph? That is, how much *extra* space is needed to implement the search? The answer depends on the structure of the particular graph to which the search algorithm is applied. And the answer can be different for breadth-first and depth-first search.

Write an essay discussing this problem. Say as much as you can about the space efficiency of depth-first and breadth-first search. Try to get some idea of the worst case space efficiency for each algorithm. Include some specific graphs as examples, and look at the space requirements for each type of search. Try to select examples that illustrate your results. You should also try to say something about what you expect the average case behavior to be, without trying to find any exact results.