Algorithms in the Real World (15-853), Fall 04
Assignment #3

Please answer all of the questions, except question 1 (see note). Due Nov. 17.

Problem 1: (20pts)

This problem is hard (particularly part 2), so as mentioned in class you don't need to do it. If you do make progress, it will count as extra credit.
Consider the incremental 2d Convex Hull algorithm described in class. Gary showed using backwards analysis that the algorithm runs in expected $O(n \log n)$ time.
In many practical situations only a small number of vertices end up on the hull, and the algorithm might run much faster.

1. Show that if only h points up on the hull, the incremental algorithm runs in expected $O(n \log h)$ time, or describe a counterexample.
2. Lets say that for a random sample of the points of size m (the total number of points is n), the expected size of the hull is $O(\sqrt{m})$. What is the expected running time of the algorithm in this case?

Problem 2: (20pts)

1. Show that in a binary tree there is (1/4)-(3/4) edge separator of size 1. Here we are assuming edges and vertices are unweighted and we want to balance the vertices. Generalize this theorem to trees where each node has degree at most d.
2. Show that in any tree, the bisection width (*i.e.*, the number of nodes that must be removed in order to partition the tree into two equal-sized (to within 1) pieces is at most $O(\log n)$.

Problem 3: (20pt)

Consider applying divide-and-conquer to graphs and lets say that merging the two recursive solutions takes $f(s)$ time, where s is the number of edges separating the two graphs. For each of the following $f(s)$, and assuming you are given an edge separator tree for which all separators for the subgraphs of size n are 1/3-2/3 balanced and bounded by $kn^{1/2}$, what is the running time of such an approach.

1. s
2. $s \log s$
3. s^2
4. s^4

Problem 4: (10pt)
Prove that given a class of graphs satisfying an $O(n^{(d-1)/d})$, $d \geq 1$, edge-separator theorem, all members must have bounded degree.

Problem 5: (30pt)
In class, and in the Karypis and Kumar reading, we covered a multilevel edge-separator algorithm. In this problem you need to generalize this technique to work for vertex separators. There are two ways to do this.

A. The first way is to generate an edge separator, and then find a vertex separator based on it. The Karypis and Kumar reading suggest using a minimum vertex cover. Briefly (a paragraph or so) explain how you might do this efficiently (i.e. polynomial time). You can look up the sources mentioned in the paper, but you will learn more by trying it on your own. Can you find an example where your method will fail (i.e. there is a good vertex separator but the method does not work)?

B. The second way is to try to build vertex separators directly into the algorithm. Along these lines, please answer the following questoins:

1. Argue why coarsening using a maximal matching is or is not still appropriate.
2. Describe what we should keep track of when coarsening (contracting) the graph (e.g. on the edge separator version each edge kept a weight representing the number of original edges between two multivertices).
3. Describe how we project the solution of the coarsened version back onto the original graph (the recursive solution must return a vertex separator).
4. Describe a variant of Kernighan-Lin or (preferably) the Fiduccia-Mattheyses heuristic for vertex separators. Be explicit about what the gain metric is.

Note that there is not necessarily a right and wrong answer for this problem.