Algorithms in the Real World
Assignment #3 Due Date: 29 Oct 03

Do either one of problems 5 and 6.

Problem 1: (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 2: (20pt)

1. Prove that the bisection width (i.e. the number of edges that must be removed to separate a graph into two equal-sized parts, within 1) of the complete graph with n vertices is $(n/2)^2$.

2. Prove that the bisection width of the n-node hypercube is $n/2$. This should be proved from below and above. (Hint: show that the complete graph can be embedded in the hypercube so that vertices map to vertices, edges in the complete graph map to paths in the hypercube, and each hypercube ends up supporting the same number of paths.)

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: (20pt) 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 directly (do not use a postprocessing stage). In particular:

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.

Problem 5: (20pts)
Suppose that rather than associating a single bit with a graph node in a Tornado code, we associate a symbol drawn from a finite field.

1. Using a systematic Reed-Solomon code capable of correcting one symbol erasure instead of a simple 1-bit parity function, show how the Tornado code construction described in class can be generalized to correct for erasures of symbols rather than bits.
2. Now using a systematic Reed-Solomon code capable of correcting one symbol error (rather than just an erasure), show how to construct a Tornado code capable of correcting errors rather than just erasures. Your code should add at total of $O(m)$ parity symbols to a message of m symbols, and should be able to recover a message, with high probability, even if every bit is incorrect with some constant probability $p > 0$ (p may be very small, but still a constant).

   You may find it helpful to make multiple copies of some of the structures in the graph described in class.

   Another tip is that the (α, δ) "unshared neighbor" property proved for bipartite expander graphs in which there are n nodes on the left and $n/2$ nodes on the right can be generalized to bipartite expander graphs in which there are n nodes on the left and n/c nodes on the right, for any constant $c > 0$. (In particular, you may be interested in values of $c > 2$.) Changing the value of c will impact the relationship between α and δ (for fixed δ, larger c will imply smaller α), but as long as c is a constant, α and δ will also be constants.
3. Describe how to decode your code.

Problem 6: (20pts)
Recall that nested dissection is an algorithm that determines the order in which variables are to eliminated when using Gaussian elimination to solve a linear system $Ax = b$. The goal of nested

dissection is to produce an ordering that creates only a small amount of "fill". Fill occurs when a zero entry in a matrix row becomes non-zero. It is helpful to view nested dissection as a graph algorithm, where the graph G is defined by viewing the matrix A as an adjacency matrix. (There is an edge between two nodes in G if the corresponding entry in A is non-zero.) Gaussian elimination removes nodes from the graph one-by-one, and each time a node is removed, a clique is formed on its neighbors. Any clique edges that were not already present in the graph constitute fill.

In this problem we will prove an upper bound on the fill generated by nested dissection for *planar graphs* that have special "cycle" separator functions. A class of planar graphs $\mathscr{C}$ is said to have an $f(N)$-cycle-separator-theorem if, for any N-node graph G that belongs to the class, there is a simple cycle C of length $O(f(N))$ whose removal partitions the graph into two subgraphs G_1 and G_2, where $N_1 \geq N/3$, $N_2 \geq N/3$ (where N_1 and N_2 are the number of nodes in G_1 and G_2, respectively), where G_1 lies inside the cycle, G_2 lies outside the cycle (the designation of inside and outside doesn't matter), and both G_1 and G_2 either belong to the class $\mathscr{C}$ or contain only $O(1)$ nodes.

1. Let $\mathscr{C}$ be the class of $m \times n$ two-dimensional meshes, where $m/2 \leq n \leq m$. (Each such graph has $N = mn$ nodes.) Prove that $\mathscr{C}$ has a $\sqrt{N}$-cycle-separator-theorem.

Now suppose that we apply the following nested dissection algorithm to a graph with a cycle separator theorem.

- Remove the cycle separator C (nodes and edges) from G. Place the nodes in C last in the elimination order, in arbitrary order.
- Recursively order G_1. Place these nodes in the order before the nodes of C.
- Recursively order G_2. Place these nodes in the order before the nodes of G_1.

The recursion terminates if G is a graph consisting of a constant number of nodes that does not belong to the class of graphs $\mathscr{C}$, and hence cannot be further broken up.

The execution of this algorithm can be viewed as a decomposition of the graph. The natural representation of this decomposition is a binary tree constructed recursively, where the root of the tree is a node representing the cycle C, the left subtree is the tree for G_2, and the right subtree is the tree for G_1. Each node u of the graph G is represented by a single node $t(u)$ in the decomposition tree.

2. (Easy.) Prove that if there is an edge between nodes u and v in G, then either $t(u) = t(v)$, or $t(u)$ is an ancestor of $t(v)$, or $t(v)$ is an ancestor of $t(u)$.
3. (Easy corollary.) Prove that there can be fill between nodes u and v of G only if either $t(u) = t(v)$, or $t(u)$ is an ancestor of $t(v)$, or $t(v)$ is an ancestor of $t(u)$.
4. Suppose that u is a node of G represented by $t(u)$. Prove that there is fill between u and nodes v represented by only $O(d)$ tree nodes, where d is the depth of the tree. In this step you will need to make use of the fact that all of the separators are simple cycles.
5. Prove that for the class of $m \times n$ meshes with the $\sqrt{n}$-cycle-separator-theorem, the total fill generated by the nested dissection algorithm is $O(N \log N)$.