CS 290H: Sparse matrix algorithms // Homework 2

Assigned October 17, 2004

Due by class time Thursday, October 28

1. **[10 points]** Prove that every tree with n vertices has some single vertex whose removal leaves no connected component with more than $2n/3$ vertices. (That is, trees have 1-separators.)

2. **[40 points]** Let A be any $n \times n$ symmetric matrix, let $G = G(A)$ be its graph, let $G^+ = G^+(A)$ be its filled graph, and let $T = T(A)$ be its elimination tree. You may use the result of problem (1) for this problem.

2(a) **[10 points]** Let j be a vertex of G, G^+, and T, so $1 \leq j \leq n$. Let D_j be the set of vertices that are descendants of j in T. (Remember that j is its own descendant by definition.) Let E_j be the set of vertices that are adjacent in G to vertices in D_j, but are not themselves in D_j. Finally, let $C_j = E_j \cup \{j\}$. Prove that the subgraph of G^+ induced by C_j is a clique, that is, every pair of vertices in C_j is joined by an edge of G^+. (This is the same as exercise 6.2 on page 129 of GLN, with different notation.)

2(b) **[10 points]** Let j and D_j be as above. Let E_j^+ be the set of vertices that are adjacent in G^+ to vertices in D_j, but are not themselves in D_j. Finally, let $C_j^+ = E_j^+ \cup \{j\}$. Prove that $C_j^+ = C_j$. (This is exercise 6.3 on page 129 of GLN.)

2(c) **[10 points]** Prove that there is a clique of G^+ whose removal leaves no connected component with more than $2n/3$ vertices.

2(d) **[10 points]** Let A be any $n \times n$ symmetric positive definite matrix. Suppose there exists a permutation P such that the Cholesky factor of PAP^T has t nonzeros. Prove that $G(A)$ has a separator (a set of vertices whose removal leaves no connected component with more than $2n/3$ vertices) with at most $2\sqrt{t}$ vertices.

[Optional, extra credit] Let A be any $n \times n$ symmetric positive definite matrix with at most d nonzeros per row or column. Prove that there exists a nested dissection order (i.e. a choice of separators for the graph of A and, recursively, for the hierarchy of subgraphs that appear in nested dissection) for which the number of nonzeros in the Cholesky factor of A is within a factor of $d \log^2 n$ of the smallest possible number of nonzeros for any elimination order.

3. **[50 points]** Write a Matlab mexFunction `X = lsolve (L,B)` that solves a sequence of sparse triangular systems with sparse right-hand sides.

Input L is an $n \times n$ square Matlab sparse matrix that you can assume is lower triangular. Input B is an $n \times m$ Matlab sparse matrix. The output X should be an $n \times m$ Matlab sparse matrix that is identical to the result of `X = L \ B`.

Unlike homework 1, for this homework you can use Matlab to help get the indices of each column of X into sorted order. Matlab's matrix transpose operator works correctly even if its input has out-of-order row indices, and it produces a matrix with sorted row indices. It works by doing a lexicographic sort of the nonzeros by rows and columns, which takes time $O(\text{nnz}(X) + n + m)$. So, you can wrap your routine in a Matlab m-file that finishes with `X = (X')'`.

There are several possible approaches to this problem. For full credit, you should implement the following two approaches and compare them to each other and to Matlab. The first approach alone is worth half credit.

Approach 1: Loop over the columns of B, creating a column of X for each one. Compute the current column of X in a dense n-vector that starts out equal to the column of B. Loop over the entries of the dense vector; whenever you encounter a nonzero entry, compute the final value of that element of X (by dividing by a diagonal element of L) and update the dense vector by subtracting a suitable multiple of a suitable column of L. At the end of the dense vector, compress that column into the data structure for X.

Note that Approach 1 takes at least order n time per column, so its run time is at least $\Omega(nm)$.

Approach 2: Instead of a simple dense n-vector, use a SPA to compute each column of X. As in the efficient matrix-multiplication code, use $O(n)$ time to initialize the SPA once at the very beginning, but don't use $O(n)$ time for each column.

The trick here is to figure out, when you start each column of X, which columns of L will be needed as updates—that is, which elements of the current column of X will be nonzero. You can do this by performing a depth-first search in the directed graph $G(L^T)$, beginning from the vertices representing the nonzeros in the current column of B. The vertices you reach in the search are exactly those that will represent the nonzeros in the current column of X. During the search you can produce a list of those vertices in topological order. (That is, in an order that is consistent with the partial order defined by the graph $G(L^T)$.) Then you can go through the topologically ordered list, updating the SPA with the corresponding columns of L. (Convince yourself that the updates must occur in topological order for the triangular solve to work correctly. They don't need to occur in increasing numerical order, though, so you don't have to sort the nonzeros into numerical order.)

To test your code, generate triangular matrices L and right-hand sides B by

```
A = grid5(k);
A = sprand(A);
A = A(:, colamd(A));
[L,U,P] = lu(A);
n = k*k;
B = sprand(n, n/10, 3/n);
```

The routine `grid5`, from the meshpart toolbox referenced on the course web site, produces the $n \times n$ matrix of the 2-D model problem on the $k \times k$ grid, $n = k^2$. Try various n from 100 to as large as you can get. (The limit will be how much L fills in during the LU factorization; you ought to be able to get n up to 10,000 or so anyway.) Use Matlab to plot the running times of your two codes and Matlab's `L \ B`.

Turn in all your code, and also the plot with your run times and Matlab's, and also a Matlab transcript of the session that creates the plot and verifies that your output matrices agree with Matlab's.

Again, I'll give a little extra credit for beating Matlab on large matrices, and for the fastest code submitted.