Minimum Spanning Trees

11-11-2003

Opening Discussion

- What is a graph? I’d like informal and formal descriptions of the structure. What are the two ways we talked about for representing the graphs in a program?
- Do you have any questions about the assignment?

Code for Searches

- Now let’s go look at some code for doing these searches. Doing them efficiently is slightly different depending on whether you have an adjacency list or an adjacency matrix representation.
**Topology Sorting**

- For a DAG, a depth first search can easily produce an ordering for all the vertices such that if there is an edge \((u,v)\) then \(u\) comes before \(v\) in the ordering.
- This ordering can be made by sorting the elements in reverse order by their finishing time. This is done quickly by simply inserting the nodes at the beginning of a linked list when finished.

**Strongly Connected Components**

- A strongly component of a graph is a maximal subset, \(C\), of \(V\) such that for every pair of nodes, \(u\) and \(v\), in \(C\) there is a path from \(u\) to \(v\) and a path from \(v\) to \(u\).
- To find these we do a DFS of \(G\), then compute \(G'\) where all the edges have been reversed. Now we do a DFS of \(G'\), but visit nodes in order of descending finishing time. The trees in the forest of that traversal are the strongly connected components.

**Minimum Spanning Tree**

- A spanning tree is a set of edges that connects all the vertices of a graph. If the graph has \(V\) vertices then the spanning tree has \(|V|-1\) edges.
- It is called a tree because there are no cycles in it.
- If the graph is weighted then we can find a spanning tree that minimizes the weights of the edges in it.
Procedure and Definitions

- The procedure for building the minimum spanning tree is to start with an empty set and repeatedly add "safe" edges into it. A "safe" edge is one that is an element of some minimum spanning tree.
- To help with this we define a cut as a partitioning of the graph into two sets, \( C=(S,V-S) \). An edge that connects a vertex in \( S \) to one in \( V-S \) crosses the cut.
- A cut respects a set of edges \( A \) if no edge in \( A \) crosses the cut.

A Helpful Theorem

- Given a graph \( G=(V,E) \) and a weight function \( \omega \) on \( E \), let \( A \) be a subset of \( E \) that is a subset of some minimum spanning tree for \( G \) and let \( (S,V-S) \) be a cut that respects \( A \). If the edge \((u,v)\) is a light edge crossing the cut, then \((u,v)\) is safe for \( A \).
- A light edge is one with a minimum weight that satisfies a requirement.

Kruskal’s Algorithm

- In this algorithm we always add the lowest weight edge in the graph that doesn’t create a cycle. As a result, while the algorithm executes, we go through a process of connecting a forest of trees to produce a single tree.
- To make this fast we first sort the elements of \( E \) by weight and simply walk that in a loop then use fast structures for disjoint sets (Ch. 21).
**Prim’s Algorithm**

- This algorithm starts with a particular node, and adds edges to a single tree, A, that is built. At each step the edge added is the smallest to connects the tree to a vertex not yet in the tree.
- To make it efficient we keep a min-priority queue of the "keys" of nodes where a key is the minimum edge connecting a node not in the tree to some node in the tree.

---

**Minute Essay**

- Which of these two algorithms is more intuitive to you? We haven't tried to do any of these things in a generic way (using templates or other polymorphism). Can you speculate on why? What does this say about graphs as a data structure?
- Remember that assignment #4 should be turned in today.