CSE 101 Class Notes Today: Graphs, Recursion vs. Iteration

September 29, 2006

Breadth- and Depth-First Search

Problem: Given a directed graph $G = (V, E)$, find the set of nodes reachable from some node x.

In doing this, we want to avoid visiting a node more than once, so we need to keep track of these visited nodes. Initially, `Visited[x]` is `false` for all x. We also need to track the set F of "frontier" nodes, which we have seen (as neighbors of a processed node) but have not yet processed.

Algorithm

```
1      Frontier <- { x }
2      Visited[x] = T
3      forall y != x
4          Visited[y] = F
5      while Frontier not empty
6          y <- Choose(Frontier)
7          for z in Adj(y)
8              if !Visited[z]
9                  Insert(Frontier, z)
10                 Visited[z] = T
11         Remove(Frontier, y)
```

What data structure should we use? It must support (1) inserting an element, (2) choosing an arbitrary element, and (3) removing that element. Both stacks (push, top, pop) and queues (enqueue, head, dequeue (or "decapitate"?)) support these operations in constant time.

Running Time

After a node is added to F, it is marked `Visited`. No visited node is added to F. Since a `Visited` mark is never removed, this implies that each node is added to F at most once. Therefore the loop at line 7 is done at most once per node y.

This loop takes $O(\deg(y))$, since it is executed once for each edge from y. The sum of all nodes' out-degrees in a directed graph is $|E|$. Therefore the total running time for lines 7-10 is $O(\sum_{v \in V} \deg(v)) = O(|E|)$.

The remaining lines are executed $|V|$ times, and take constant time. Therefore the total running time is $O(|V| + |E|)$.

Note that this running time is the same whether we use a stack or a queue. However, which we use will determine whether we will execute a depth-first (stack) or breadth-first (queue) search. Using a stack, however, we can gather additional information by remembering the order in which we insert and remove items. In particular, no node is removed from the stack until all its children have been removed. This order of removal (children before parents) is referred to as depth-first search.