

Corrected algorithm to compute prime paths

Patrick Lam

February 2, 2009

The algorithm that I presented in class in Lecture 8 was slightly incorrect; I missed a corner case that the textbook alluded to. Thanks to Justin Tisi for pointing out the error. The problem is that path p might be non-extendable while still having outgoing edges, since p might only have edges that make non-simple paths. The corrected algorithm:

Input: Directed graph G

Output: List of prime paths in G , `primePaths`

```
nonextendablePaths ← ∅;
primePaths ← ∅;
worklist ← all paths of length 0, i.e. nodes;
while worklist ≠ ∅ do
  p ← worklist.removeFirst();
  pi ← initial node of p;
  pf ← final node of p;
  wasExtended ← false;
  if pf has no outgoing edges then
    nonextendablePaths += p;
  else
    foreach p'f such that (pf, p'f) is an edge in G do
      if p'f does not appear in p then
        worklist += p++p'f;
        wasExtended ← true;
      else
        if p'f = pi then
          nonextendablePaths += p++p'f;
          wasExtended ← true;
        end
      end
    end
  end
  if not wasExtended then
    nonextendablePaths += p;
  end
end
end
primePaths ← ∅;
foreach p ∈ nonextendablePaths do
  // (p could only be a suffix of a non-extendable path; I'd use a tree:)
  if p is not a proper subpath of any simple path then
    primePaths += p
  end
end
end
```

Bonus: What is the best (asymptotic) bound you can find for the number of prime paths? Is your bound tight? Next problem: finding test paths to tour all prime paths (to achieve prime path coverage). Usually you need far fewer test paths than prime paths. Book doesn't present an algorithm, but suggests extending prime paths, starting from the longest prime paths.