CPS 130 Solutions

Guangwei Yuan

Fall 1999

Midterm 2 of Spring 99

Problem 1: (a)

adjacency list

\begin{verbatim}
COMPUTE-GS(G)
V[Gs] ← V[G]
for each u ∈ V[Gs]
   Adj[u] ← ∅
for each u ∈ V[G]
   for each v ∈ Adj[u]
      INSERT(Adj[u'], v), where u' ∈ V[Gs] and u' = u
      for each w ∈ Adj[v], w ≠ u
         INSERT(Adj[u'], w)

return Gs
\end{verbatim}

(Note here INSERT does not insert duplicated elements.)

adjacency matrix

\begin{verbatim}
COMPUTE-GS(G, A)
for i ← 0 to |V[G]| - 1
   for j ← 0 to |V[G]| - 1
      As[i][j] ← 0
for i ← 0 to |V[G]| - 1
   for j ← 0 to |V[G]| - 1
      for k ← 0 to |V[G]| - 1
         if A[i][k] = 1 and A[k][j] = 1 then
            As[i][j] ← 1
            break

return As
\end{verbatim}

(b) \(O(VE^2)\) for adjacency list, and \(O(V^3)\) for adjacency matrix.
(c) The adjacency list representation is good for sparse graphs. If the graph is dense, the adjacency matrix is preferred.

**Problem 2:** I think this problem is not stated clearly enough. Suppose we use pre-order, and start from a, left to right.

DFS: a, b, c, d, e, f, g, i, h, j  
BFS: a, b, c, d, f, i, g

**Problem 3:** (a)

post-order: b, i, k, l, j, f, g, h, c, d, e, a  
pre-order: a, b, c, f, i, j, k, l, g, h, d, e

(b) Please look at the algorithm described in the lecture.

**Problem 4:**

```
   a
  / \  /
 b   c / \
   \     \
    \    
     \   
      \ 
       d
```

**Problem 5:** There are no Spanning Trees problems in Midterm 2, and you can work this problem out without much difficulty. So I skip it here.

**Problem 6:** The basic idea is to traverse the graph $G$ by breadth first search, and color vertices in each level of the BFS tree white and black alternatively. If you reach a vertex and put one color on it, but it already has a different color, then you can say $G$ is not bipartite.