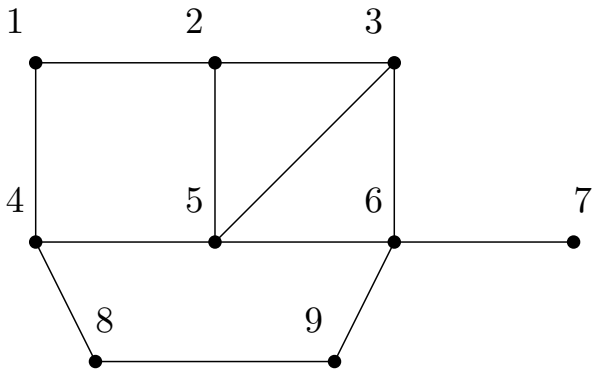
Universität Karlsruhe
Fakultät für Informatik
ITI Wagner

Sechste Übung

Problem 2 (a):



Problem 3:

```
void menger()
{
  edge_map<bool> visited(gr, false);              // simulate orienting of edge
  node_map<edge> lastOutEdge(gr, nil);            // last outgoing edge visited
  int nmbPaths(0);

  edge loopStartEdge = startEdge;
  lastOutEdge[startNode] = startEdge;
  do
  {
    stack<edge> dfsEdges;
    dfsEdges.push(loopStartEdge);

    while (!dfsEdges.empty())
    {
      edge currDfsEdge = dfsEdges.pop();
      visited[currDfsEdge] = true;                // orient edge
      visited[gr.reversal(currDfsEdge)] = true;   // block reverse edge
      gw.set_color(currDfsEdge, nmbPaths+2);      // color edge

      node nextDfsNode = target(currDfsEdge);
      if (nextDfsNode == targetNode)
      {
        nmbPaths++;
        dfsEdges.clear();
      }
      else
      {
        edge nextDfsEdge;
        if (lastOutEdge[nextDfsNode] == nil)      // no outgoing edge yet
          nextDfsEdge = getRightmostAdjEdge(currDfsEdge);
        else nextDfsEdge
               = getRightmostAdjEdge(gr.reversal(lastOutEdge[nextDfsNode]));
        lastOutEdge[nextDfsNode] = nextDfsEdge;

        if (nextDfsEdge != nil &&                 // there exists outg. edge
            nextDfsEdge != currDfsEdge &&         // different from current one
            !visited[nextDfsEdge])                // not blocked yet
        {
          dfsEdges.push(currDfsEdge);
          dfsEdges.push(nextDfsEdge);
        }
        else gw.set_color(currDfsEdge, 1);        // uncolor edge
      }
    }

    loopStartEdge = getRightmostAdjEdge(gr.reversal(loopStartEdge));
  } while (loopStartEdge != startEdge);
  cout << "#paths: " << nmbPaths << endl;
}
```