Solutions to Homework 9

22C:044 Algorithms, Fall 2000

1 First we use BFS algorithm to compute the distances from an arbitrary
source node \( s \) to all nodes. (If the graph is not connected this would be
repeated with different source nodes until all vertices have been found.
In the pseudocode below I assume that the graph is connected.) Then
we verify that each edge connects two vertices whose distances have
different parities, that is, one end point has even distance to \( s \) and the
other end point has odd distance to \( s \).

Bipartite(G)
1. BFS(G,s) where \( s \) is an arbitrarily chosen vertex
2. for all \( u \in V \) do
3. for all \( v \in \text{Adj}[u] \) do
4. if \( d[u] + d[v] \) is even then return FALSE
5. return TRUE

2(a) A straightforward algorithm based on the definitions uses BFS to com-
pute the eccentricities of the vertices. All vertices are tried one after
the other as the source \( s \) of the search (loop on lines 4–9). Each time,
the maximum value of \( d[v] \) gives the eccentricity of the source. We find
the minimum eccentricity among all sources \( s \), and print out all nodes
that have that eccentricity. The eccentricities of all vertices are stored
in array Ecc[..]:

Centers(G)
1. Allocate array Ecc
2. for every \( s \in V \) set Ecc[s] \( \leftarrow \) 0
3. \( \min \leftarrow \infty \)
4. for every \( s \in V \) do
5. begin
6. BFS(G,s)
7. for every \( v \in V \) do if \( d[v] > \text{Ecc}[s] \) then Ecc[s] \( \leftarrow \) d[v]
8. if Ecc[s] < \( \min \) then \( \min \leftarrow \text{Ecc}[s] \)
9. end
10. for every \( v \in V \) do if Ecc[v] = \( \min \) then print \( v \)
2(b) We first select an arbitrary node $s$ as the source and execute a BFS search to record the distances of all nodes from $s$. Let $v$ be a node that has the longest distance to $s$. It is fairly obvious that $v$ is the endpoint of a maximum length simple path in the tree. Therefore, we execute a second BFS search using $v$ as the source node to record the distances of all nodes from $v$, and to construct shortest paths from $v$ to all nodes. Let $u$ have a maximum distance. The center(s) is (are) the middle point(s) of a shortest path connecting $u$ and $v$:

TreeCenters($G$)
1. Choose arbitrary $s \in V$
2. BFS($G$, $s$)
3. find $v$ with maximum $d[v]$ value
4. BFS($G$, $v$)
5. find $u$ with maximum $d[u]$ value
6. set $x \leftarrow u$
7. for \( \lfloor d[u]/2 \rfloor \) times do $x \leftarrow \pi[x]$
8. if $d[u]$ is even then return $x$
9. else return $x$ and $\pi[x]$

3 The program code is given on the internet site. The output of the program (after 8 hours on a 650 MHz Pentium!) is as follows:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.382</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.397</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.425</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.363</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.376</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.396</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.413</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.382</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.455</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.483</td>
<td>1.000</td>
<td></td>
</tr>
</tbody>
</table>

Clearly increasing the probability $p$ for fixed $n$ increases the number of edges in the graph, and therefore the probability that the graph is connected increases. Also if $p$ is fixed then increasing $n$ introduces more edges because the average number of edges per node is $pn$. This also increases the probability that the graph is connected. According to this experiment it seems that probability $p = \ln(n)/n$ is the threshold between connected and unconnected graphs.