Homework Set #10 Solution

TA: Minseo Park

This assignment covers textbook material in Chapters 22-24.
Note: Partial credit for wrong answers is only given if work is shown.

For this assignment, use the BFS procedure on p. 532 of our textbook and the DFS procedure on p. 541 instead of the pseudocode in the class handout.

1. (25 points) For the undirected, unweighted graph G1 in Figure 1:
   a) (3 points) Show an adjacency list representation of G1. Use lexicographic ordering.

   \[
   \begin{align*}
   &A \mid CG \\
   &B \mid CDF \\
   &C \mid ABF \\
   &D \mid BG \\
   &E \mid F \\
   &F \mid BCE \\
   &G \mid AD \\
   \end{align*}
   \]

   Figure 1: G1

   b) (2 points) Is an adjacency list representation better for G1 than an adjacency matrix? Justify your answer.

   \[
   \text{Answer:} \\
   \text{Yes, adjacency matrix needs much more space (}49:6(V^2)\text{) than space for adjacency list (}23:9(V+2E)\text{). This graph is sparse.}
   \]

c) (5 points) Draw the Breadth-First Search tree consisting of tree edges that result from a Breadth-First Search of G1 with node A as the source.

   \[
   \begin{align*}
   &A: 0 \\
   &C: 1 \\
   &B: 2 \\
   &G: 1 \\
   &F: 2 \\
   &D: 2 \\
   &E: 3 \\
   \end{align*}
   \]
d) (5 points) For each node reachable from A, show the shortest path in G1 from A to that node. Give the length (i.e. number of edges) of each such shortest path.

Answer)
A to B: A-C-B, length 2
A to C: A-C, length 1
A to D: A-G-D, length 2
A to E: A-C-F-E, length 3
A to F: A-C-F, length 2
A to G: A-G, length 1

e) (5 points) Draw the Depth-First Search spanning forest of trees that results from a Depth-First Search of G1. Classify and label each edge as either a tree edge, back edge, forward edge, or cross edge.

Answer)

f) (5 points) What is the longest length simple cycle in G1? Give an example of a simple cycle in G1 of that length.

Answer) The longest length simple cycle: 5 (A-C-B-D-G-A)
2. (25 points) For the directed, unweighted graph G2 in Figure 2:

a) (3 points) Show an adjacency matrix representation of G2. Use lexicographic ordering.

\[ \begin{array}{ccccccc}
  & A & B & C & D & E & F & G \\
  A & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
  B & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
  C & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
  D & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  E & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
  F & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
  G & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array} \]

b) (2 points) Is an adjacency list representation better for G2 than an adjacency matrix? Justify your answer.

\[ \text{Answer:} \quad \text{Yes, adjacency matrix needs much more (space } 49:6(V^2) \text{) than space for adjacency list } 17:6(V+E). \text{ This graph is also sparse.} \]

c) (5 points) Draw the Breadth-First Search tree consisting of tree edges that result from a Breadth-First Search of G2 with node C as the source.

\[ \text{Answer:} \]

\[ \begin{array}{c}
  C \\
  A \quad B \quad F \\
  G \quad D \quad E \\
\end{array} \]

d) (5 points) For each node reachable from C, show the shortest path in G2 from C to that node. Give the length of each such shortest path.

\[ \text{Answer:} \]

- C to A: C→ A, length 1
- C to B: C→ B, length 1
- C to D: C→ B→ D, length 2
- C to E: C→ F→ E, length 2
e) (5 points) Draw the Depth-First Search spanning forest of trees that results from a Depth-First Search of G2. Classify and label each edge as either a tree edge, back edge, forward edge, or cross edge. [This is assuming that it starts with vertex C. Other starting points are also possible.]

**Answer**

![Diagram of DFS spanning forest]

```
Adjacency list
A | CG
B | DF
C | ABF
D | -
E | F
F | E
G | D
```

f) (5 points) What is the longest length simple cycle in G2? Give an example of a simple cycle in G2 of that length.

**Answer**
The longest length: 2 (A-C-A) or (E-F-E)
Algorithm consists of 2 phases. First, compute $G^T$ in the usual way, so that $G^T$ is $G$ with its edges reversed. Then do a depth-first search on $G^T$, but in the main loop of DFS, consider the vertices in order of increasing values of $L(v)$. Each edge is either T(tree), C(cross), B(back), F(forward). If vertex $u$ is in the depth-first tree of tree edges with root $v$, then $\text{min}(u)=v$. Second phase labels back and cross edges as tree edges (make $G^T$ consist of only B and C edges), then performs $\text{DFS}$. Any vertices appearing in these trees that are not part of 1st groups of trees have their min as root of B/C edge tree. TVFare irrelevant as they point down a tree. The 2 phases together should include all vertices.

**Pseudocode**

Reachability($G$)

Compute $G^T$

$\text{DFS\_MIN}(G^T)$

$G^T \leftarrow$ subgraph of $G^T$ consisting of only edges labeled B or C

$\text{DFS\_MIN}(G^T)$

$\text{DFS\_MIN}(G)$

1. for each vertex $u \in V[G]$ \hspace{1cm} $> \text{initialization}$
2. \hspace{0.5cm} do color[$u$] $\leftarrow$ WHITE
3. \hspace{1cm} $\Pi [u] \leftarrow$ NIL
4. \hspace{1cm} $\text{min}[u] \leftarrow \infty$
5. \hspace{0.5cm} Sort $G$ by increasing order of $L[v]$’s value
6. $\text{for each vertex } u \in V[G]$
7. \hspace{0.5cm} do if color[$u$] $\leftarrow$ WHITE
8. \hspace{1cm} then $\text{DFS\_VISIT}(u)$

$\text{DFS\_MIN\_VISIT}(u)$

11. color[$u$] $\leftarrow$ GREY \hspace{1cm} $> \text{White vertex } u \text{ has just been discovered.}$
12. for each vertex $v \in R[u]$ \hspace{1cm} $> \text{explore edge } (u, v), R[u]: \text{the vertices that are reachable}$
13. \hspace{0.5cm} do $\Pi [v] \leftarrow u$
14. \hspace{1cm} if $L[\Pi [v]] < \text{min}[v]$ \hspace{1cm} $> \text{compare prefixed vertices } v(\Pi [v]) \text{ is reachable from } u$
15. \hspace{1cm} then $\text{min}[v] \leftarrow L[\Pi [v]]$ with $\text{min}[v]$.
16. \hspace{1cm} if color[$v$] $\leftarrow$ WHITE
Correctness Justification

To show correctness, first note that if \( u \) is in the depth-first tree rooted at \( v \) in \( G^T \), then there is a path \( v \rightarrow u \) in \( G^T \), and so there is a path \( u \rightarrow v \) in \( G \). Thus, the minimum vertex label of all vertices reachable from \( u \) is at most \( L(v) \), or in order words, \( L(v) \geq \min\{L(w) : w \in R(u)\} \). Now suppose that \( L(v) > \min\{L(w) : w \in R(u)\} \), so that there is a vertex \( w \in R(u) \) such that \( L(w) < L(v) \). At the time \( d[v] \) that we started the depth-first search from \( v \), we would have already discovered \( w \), so that \( d[w] < d[v] \). By the parenthesis theorem, either the intervals \( [d[v], f[v]] \) and \( [d[w], f[w]] \) are disjoint and neither \( v \) nor \( w \) is a descendant of the other, or we have the ordering \( d[w] < d[v] < f[v] < f[w] \) and \( v \) is a descendant of \( w \). The latter case cannot occur, since \( v \) is a root in the depth-first forest (which means that \( v \) cannot be a descendant of any other vertex). In the former case, since \( d[w] < d[v] \), we must have \( d[w] < f[w] < d[v] < f[v] \). In this case, since \( u \) is reachable from \( w \) in \( G^T \), we would have discovered \( u \) by the time \( f[w] \), so that \( d[w] < f[w] \). Since we discovered \( u \) during a search that started at \( v \), we have \( d[u] < d[v] \). Thus, \( d[v] < d[u] < f[w] < d[v] \), which is a contradiction. We conclude that no such vertex \( w \) can exit.

This does not prove that every vertex applies as a non-root node in some DFS tree of \( G^T \). For that we need to form \( G^T \) and perform DFS of \( G^T \).

Running time

Running time is \( O(V+E) \) time.

The loops on lines 1-4 and lines 8-10 of DFS_MIN take time \( \theta(V) \), exclusive of the time to execute the calls to DFS_MIN_VISIT. Line 6 also take time \( \theta(V) \) if using counting sort (sorting in linear time).

Phase 1: The procedure DFS_MIN_VISIT is called exactly once for each vertex \( v \in V \), DFS_MIN_VISIT is invoked only white vertex and the first thing it does is paint the vertex grey. During an execution of DFS_MIN VISIT(\( v \)), the loop on lines 13-18 is executed \( |R[v]| \) times, which is \( \theta(E) \). Thus, the total running time of DFS_MIN is \( \theta(V+E) \), and the running time of this problem is also \( \theta(V+E) \). Phase 2: Same time as phase 1.
4. (25 points) Minimum Spanning Trees: For the undirected, weighted graph G3 in Figure 3:

a) (12 points) Show the Minimum Spanning Tree resulting from executing Kruskal’s algorithm on G3. What is the total weight of the edges in this Minimum Spanning Tree?

\textbf{Answer)}
1: AC, EF
2: AG, CB
3: BF
5: BD
7: CF
8: GD

\textit{Kruskal’s Minimum Spanning Tree: See attached paper after the last problem.}
\textit{Total weight: 14}

b) (13 points) Show the Minimum Spanning Tree resulting from executing Prim’s algorithm on G3. (Use vertex A as the root.) What is the total weight of the edges in this Minimum Spanning Tree?

\textbf{Answer)}
\textit{Prim’s Minimum Spanning Tree: See attached paper after the last problem.}
\textit{Total weight: 14}
5. (25 points) For the directed, weighted graph G4 in Figure 4, execute Dijkstra's algorithm using vertex A as the source. Each time a vertex is removed from the priority queue, for each vertex v show its distance value d[v]. At the end of the algorithm, for each vertex v list the vertices along the shortest path from A to v.

**Answer**

Dijkstra's Minimum Spanning Tree: See attached paper after the last problem.

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[v]</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[v]</td>
<td>0</td>
<td>∞</td>
<td>1</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[v]</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>∞</td>
<td>∞</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[v]</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>∞</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[v]</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>∞</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>d[v]</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

**Note:** Each time a vertex is removed from the priority queue.

- A→ B: A→ C→ B, shortest path: 3
- A→ C: A→ C, shortest path: 1
- A→ D: A→ C→ B→ D, shortest path: 8
- A→ E: A→ C→ B→ F→ E, shortest path: 7
- A→ F: A→ C→ B→ F, shortest path: 6
- A→ G: A→ G, shortest path: 2
4-a) Kruskal's algorithm.

Total weight = 1+1+2+2+3+5 = 14.
4-b) Prim's algorithm

Total weight: $1 + 2 + 2 + 3 + 1 + 5 = 14$
Dijkstra's algorithm.

5