1(a) Pseudo code

\[ O(m+n) \]

// Compute degrees and set \( H[v] \leftarrow 1 \) if degree \( [v] \geq 10 \)

for each \( v \in V \) do

    count \( \leftarrow 0 \)

    for each neighbor \( w \) of \( v \) do

        count \( \leftarrow \) count + 1

        degree \( [v] \leftarrow \) count

    if degree \( [v] \geq 10 \) then

        \( H[v] \leftarrow 1 \)

\[ O(m+n) \]

// Neighbors of vertices with degree \( \geq 10 \) set their \( H[w] \) to 1

// Block 1

for each \( v \in V \) do

    if \( H[v] = 1 \) then

        for each neighbor \( w \) of \( v \) do

            \( H[w] \leftarrow 1 \)

\[ O(m+n) \]

// Vertices 2 hops away from vertices with degree \( \geq 10 \) set their \( H[w] \) to 1

// Block 2

for each \( v \in V \) do

    if \( H[v] = 1 \) then

        for each neighbor \( w \) of \( v \) do

            \( H[w] \leftarrow 1 \).
1(b) Note that the code in Block 1 and Block 2 are identical. Block 1 spreads $H[v] = 1$ to neighbor of vertices with degree $\geq 10$. Block 2 spreads $H[v] = 1$ to vertices 2 hops away from vertices with degree $\geq 10$.

To spread this to vertices 20 hops away, we just need to enclose the block of code in Block 1 in a loop that executes 20 times.

(Since each block runs in $O(m+n)$ time, 20 executions of this block also runs in $O(m+n)$ time.)

2(a) Delete $\epsilon[u,v]$ from $G$ and run the explore function with source $U$ on the resulting graph. If $v$ is visited by the explore function, then $\epsilon[u,v]$ is in a cycle in $G$.

(Explanation: If there is a path from $u$ to $v$ in the graph without edge $\epsilon[u,v]$, then adding $\epsilon[u,v]$ to that path gives a cycle containing $\epsilon[u,v]$.)
3(a) for all $u \in V$
- $\text{dist}[u] \leftarrow \infty$
- $\text{_prev}[u] \leftarrow \text{NIL}$
- $\text{dist}[s] \leftarrow c(s)$

H $\leftarrow \text{makeQueue}(V)$ (with dist-values as keys)

while H is non-empty do
- $u \leftarrow \text{deleteMin}(H)$

for all edges $(u,v) \in E$ do
  if $\text{dist}[v] > \text{dist}[u] + l(u,v) + c(v)$ then
    $\text{dist}[v] \leftarrow \text{dist}[u] + l(u,v) + c(v)$
    $\text{prev}(v) \leftarrow u$
    decreaseKey($H$, $(H,v)$)
3. (b) (i)  

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>distances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>After Iter 1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>After Iter 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>After Iter 3</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

(ii) An alternate edge ordering that allows Bellman-Ford to complete in 1 iteration:

(D, C), (C, B), (B, A), (D, A), (D, B)

4(a) A dominates 4 vertices, including itself. This is strictly more than the number of vertices dominated by any other vertex. So greedy picks A.

After that greedy is forced to pick 3 more vertices, one to dominate E, one to dominate F, and one to dominate G. So the greedy dominating set has size 4. Optimal solution = \{B, C, D\}.

4 (b)

A diagram is shown with points labeled A, B, C, D, E, F, and G. Intervals are given:

- \(I_1\) overlaps \(I_2\)
- \(I_3\) overlaps \(I_4\)
- \(I_5\)

<table>
<thead>
<tr>
<th>Intervals</th>
<th>(\omega(I)/\text{deg}(I))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I_1, I_5)</td>
<td>0.5/1 = 0.5</td>
</tr>
<tr>
<td>(I_2, I_4)</td>
<td>3/2 = 1.5</td>
</tr>
<tr>
<td>(I_3)</td>
<td>4/2 = 2</td>
</tr>
</tbody>
</table>

So greedy picks \(I_3\) and then \(I_1\) & \(I_5\). Total weight = 5

OPT = \{I_2, I_4\}. Weight of OPT = 6.