1. A directed graph is **acyclic** if it does not contain any directed cycles. Let \( D(V,A) \) be a directed acyclic graph (dag), where each directed edge has a non-negative weight. Assume that the vertices \( \{v_1, v_2, \ldots, v_n\} \) are ordered so that for each directed edge \((v_i, v_j)\), we have \( i < j \). (Thus, the vertex ordering is such that each directed edge of \( D \) goes from left to right.) For a directed edge \((v_i, v_j) \in A\), let \( w(v_i, v_j) \) denote its weight. The weight of any directed path from \( v_i \) to \( v_j \) is the sum of the weights of all the directed edges in that path.

Assume that the graph is represented in the following manner: for each vertex \( v_q \), there is a linked list \( L[q] \) which contains each vertex \( v_p \) such that \( v_q \) has an incoming edge from \( v_p \). (Thus, each entry of the linked list represents one directed edge; assume that the entry also provides the weight of the edge.)

Give a dynamic programming algorithm that computes the maximum weight of a directed path in \( D \). The running time of your algorithm must be a polynomial in \( |V| + |A| \).

You must clearly specify what information is stored in the table used for dynamic programming and how the entries in the table are computed. Be sure to include (high level) pseudocode for your algorithm and explain why the running time of your algorithm is a polynomial in \( |V| + |A| \). You won’t receive any credit if your algorithm is incorrect or its running time is not a polynomial in \( |V| + |A| \). (30 points)

2. Given an undirected graph \( G(V,E) \), a **dominating set** of \( G \) is a subset \( V' \subseteq V \) of vertices so that for every vertex \( u \) in \( V \) – \( V' \), there is a vertex \( v \) in \( V' \) such that \( \{u,v\} \) is an edge in \( E \) (i.e., every vertex outside \( V' \) is “dominated by” a vertex in \( V' \)). A **minimum dominating set** for \( G \) is a dominating set of minimum cardinality.

**Example:** In the following graph, both \( \{v_1, v_4\} \) and \( \{v_2\} \) are dominating sets. However, \( \{v_1, v_3\} \) is not a dominating set since neither \( v_1 \) nor \( v_3 \) dominates \( v_4 \). The size of a minimum dominating set for this graph is 1.

![Graph with vertices v1, v2, v3, v4 and edges v1-v2, v2-v3, v3-v4]
A greedy method which aims to obtain a minimum dominating set in a graph \( G(V, E) \) is the following. (Recall that the degree of a vertex is the number of edges incident on that vertex.)

1. Let \( V' = \emptyset \).
2. \( \textbf{while } V \neq \emptyset \textbf{ do} \)
   (a) Compute the degree of each vertex in \( V \).
   (b) Find a vertex \( v \) of maximum degree in \( V \). Break ties arbitrarily.
   (c) Add \( v \) to \( V' \).
   (d) Delete from \( V \), the vertex \( v \) and all the vertices that have an edge to \( v \). This process should also remove all the edges incident on the vertices removed.
3. \( \textbf{Output } V' \).

Show an example of a graph for which this greedy strategy fails to find a minimum dominating set. Your answer should indicate a dominating set picked by the greedy approach as well as a minimum dominating set. (15 points)

3. Suppose \( G(V, E) \) is a graph and \( V' \) is a subset of \( V \). The \textbf{induced subgraph} \( G'(V', E') \) on \( V' \) is defined as follows. The vertex set of \( G' \) is \( V' \). The edge set \( E' \) contains every edge \( \{v_i, v_j\} \) of \( E \) where \( v_i \) and \( v_j \) are both in \( V' \). (For example, in the graph shown in Problem 2, the induced subgraph on \( \{v_1, v_3, v_4\} \) contains just one edge, namely \( \{v_1, v_3\} \).) Consider the following problem.

   \textbf{Given:} An undirected graph \( G(V, E) \) and an integer \( k \leq |V| - 1 \).

   \textbf{Required:} Find a subset \( V' \subseteq V \) of maximum cardinality such that in the induced subgraph \( G'(V', E') \) of \( G \), every vertex has a degree of at least \( k \).

   Give a polynomial time algorithm for the above problem. Your answer must contain a clear description of the algorithm, a proof of its correctness and an analysis of its running time. \textit{You won’t receive any credit if your algorithm is incorrect or its running time is not a polynomial in} \( |V| + |E| \). (25 points)

4. Consider the following scheduling problem.

   \textbf{Given:} A set \( \{T_1, T_2, \ldots, T_n\} \) of \( n \) tasks, where each task has execution time \( t_i \) and deadline \( d_i \) (\( 1 \leq i \leq n \)).

   \textbf{Question:} Can the \( n \) tasks be scheduled (without preemption) on a single processor so that each task finishes no later than its deadline?

   Give a polynomial time algorithm for the above decision problem. Your answer must contain a clear description of the algorithm, a proof of its correctness and an analysis of its running time. \textit{You won’t receive any credit if your algorithm is incorrect or its running time is not a polynomial in} \( n \). (30 points)