Q1. Let $S = \{a_1, \ldots, a_n\}$ be a set of intervals; $a_k = [s_k, f_k)$. Intervals a_i and a_j are compatible iff they do not overlap. Describe an algorithm that takes the intervals and returns a subset of S that maximizes the sum of the lengths of the intervals in S. The length of a_k is $f_k - s_k$.

Answer: For $a_k = [s_k, f_k)$, suppose $t(s_k)$ denote the weight of the MWIS of all intervals that close before s_k. Let $l(s_k)$ be the last interval of that MWIS. Then $\pi(k)$, the weight of "the MWIS with a_k as its rightmost interval" is $t(s_k) + (f_k - s_k)$. The predecessor of a_k in this MWIS would be $l(s_k)$.

Consider the closing of a_k at f_k; $t(f_k)$, the weight of the MWIS of all intervals that close on or before f_k, is the larger of $\pi(k)$ or the t-value of the previous endpoint; accordingly, $l(f_k)$ is k or the l-value of the previous endpoint. Thus the algorithm is

> Let $e_1 < e_2 < \ldots < e_{2n}$ be the endpoints of the intervals. Suppose the left-endpoints (LEP) and the corresponding right-endpoints (REP) have pointers to each other. These pointers can be obtained by sorting the endpoints and the intervals a few times.
>
> Initialize $t = 0$ and $l = 0$. Initialize, for $j = 1$ to n, $\pi(j) = 0$ and $pred(j) = NIL$.
>
> Do, for $i = 1$ to $2n$, { if ($e_i = s_k$ for some k) then set $\pi(k) = t + (f_k - s_k)$ and $pred(k) = l$; if ($e_i = f_k$ for some k) then if ($\pi(k) > t$) then set $t = \pi(k)$ and $l = k$; }

The time complexity is $O(n)$, once the endpoints are sorted.

M5. Suppose we are given an n-node rooted tree T, such that each node v in T is given a weight $w(v)$. An independent set of T is a subset S of the nodes of T such that no node in S is a child or parent of any other node in S. Design an efficient dynamic programming algorithm to find the maximum-weight independent set of the nodes in T, where the weight of a set nodes is simply the sum of the weights of the nodes in that set. What is the running time of your algorithm? **[12]**

Answer: The MWIS of T either contains the root r of T (in which case, it contains no child of r) or does not contain r (in which case, it contains one or more children of r). Moreover, the MWIS of T is a superset of the MWIS of the subtree rooted at each child of r.

This suggests the following algorithm.

Direct every edge of T towards the parent. Topologically sort T. For the vertices x of T taken in the topological order, do the following. (Maintain two arrays A and B. $A[x]$ is to contain the weight of the MWIS of the subtree rooted at x. Boolean $B[x]$ is to be 1 iff x is not in that MWIS.)

If x is a leaf, then set $A[x] = w(x)$ and $B[x] = 0$. Otherwise, first set $A[x] = A[v_1] + \ldots + A[v_k]$; and then if $B[v_1] \ldots B[v_k]$, set $B[x] = 0$ and add $w(x)$ to $A[x]$, else set $B[x] = 1$, where $v_1, \ldots, v_k$ are the children of x. The MWIS consists of every x such that $B[x] = 0$.

As every vertex and edge is considered once each, the time complexity is $O(n)$.

Q2.1. Suppose you are given a directed graph $G = (V, E)$ with costs on the edges c_e for $e \in E$ and a sink t (costs may be negative). Assume that you also have finite values $d(v)$ for $v \in V$. Someone claims that, for each node $v \in V$, the quantity $d(v)$ is the cost of the minimum-cost path from node v to the sink t. Give a linear-time algorithm (time $O(m)$ if the graph has m edges) that verifies whether this claim is correct. **[5]**

Answer: The the set of vertices from which t is reachable can be identified by inverting every edge, and then running DFS or BFS from t. The adjacency matrix of the inverted-edges graph can be obtained through radix sort. If t is not reachable from all vertices, then signal "false" and exit; this is because $d(v)$ is finite for all vertices. Otherwise, check if each vertex u has an out-neighbour v so that $d(u) = w(u, v) + d(v)$. This takes $O(m)$ time.

If t is reacahble from u, then some out-neighbour of u is on the shortest path from u to t.

Q2.2. Let $G = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative edge capacities $\{c_e\}$. Give a polynomial-time algorithm to decide whether G has a unique minimum $s - t$ cut (i.e., an $s - t$ of capacity strictly less than that of all other $s - t$ cuts). **[5]**

Answer: Find the maximum flow f using the Edmond-Karp Algorithm. If the graph has a unique $s - t$ cut, then the maxflow saturates exactly one cut. Consider the residual graph G_f. Let S be the set of vertices reachable from from s in G_f. Let T be $V - S$. Then f saturates multiple cuts iff t is not reachable in G_f from every vertex of T. This can be verified in linear time, as in the previous answer.