

(a) The graph on nodes  $v_1, \dots, v_5$  with edges  $(v_1, v_2), (v_1, v_3), (v_2, v_5), (v_3, v_4)$  and  $(v_4, v_5)$  is such an example. The algorithm will return 2 corresponding to the path of edges  $(v_1, v_2)$  and  $(v_2, v_5)$ , while the optimum is 3 using the path  $(v_1, v_3), (v_3, v_4)$  and  $(v_4, v_5)$ .

(b) The idea is to use dynamic programming. The simplest version to think of uses the subproblems  $OPT[i]$  for the length of the longest path from  $v_1$  to  $v_i$ . One point to be careful of is that not all nodes  $v_i$  necessarily have a path from  $v_1$  to  $v_i$ . We will use the value " $-\infty$ " for the  $OPT[i]$  value in this case. We use  $OPT(1) = 0$  as the longest path from  $v_1$  to  $v_1$  has 0 edges.

```

Long-path(n)
  Array  $M[1 \dots n]$ 
   $M[1] = 0$ 
  For  $i = 2, \dots, n$ 
     $M = -\infty$ 
    For all edges  $(j, i)$  then
      if  $M[j] \neq -\infty$ 
        if  $M < M[j] + 1$  then
           $M = M[j] + 1$ 
        endif
      endif
    endfor
     $M[i] = M$ 
  endfor
  Return  $M[n]$  as the length of the longest path.

```

The running time is  $O(n^2)$  if you assume that all edges entering a node  $i$  can be listed in  $O(n)$  time.