1.The running time is O(V^2), because every element in adjacency matrix will be visited once, and there are $|V|^2$ elements in adjacency matrix, so the running time is O(V^2).

2. The ordering is p,n,o,s,m,r,y,v,x,w,z,u,q,t.

3. A vertex u can end up in a depth_first tree containing only u if all of its outgoing edges have been visted and then u is chosen to be visited.

4. Firstly, I will depth_first search the tree and calculate every node's dicoovery time and finishing time. If there exists u.d<v.d<v.f<u.f. or v.d<u.d<u.f<v.f, then the node u is a descendant or ancestor of node v.

5.If there is a back edge in a directed graph, there is acycle in the tree. Therefore, to detect a cycle in a directed graph is to detect a back edge. We can keep track of vertices currently in recursion stack of function for DFS traversal. This means that we put all the grey vertices in a stack, when the vertex in the top of the stack becomes black, we pop it out. But if we visit a vertex that is already in the recursion stack, then there is a cycle in the tree. The running time is O(|V|+|E|).

6.The sequence of courses is LA15, LA16, LA31, LA127, LA32,LA22,LA126,LA141, LA169.

7.First of all, I take the dungeon as a DAG, so there exists a topological sort. According to the topological sort, the initial reward for every node is 0, then start from the entrance, for every node which is connected with the previous node, calculate previous node's reward plus the edge's value, if the value is larger than the node's current reward then update the node's reward until the last one, then we will get the optimum path.

8.a) Because we only consider every martian can kiss the one to its left, even if there exists a back edge in the graph, we can get rid of it, then we can topological sort the graph.

b) We use topological_sort algorithm to obtain a topological sort of the grapg on the condition that we ignore all back edges.