a)

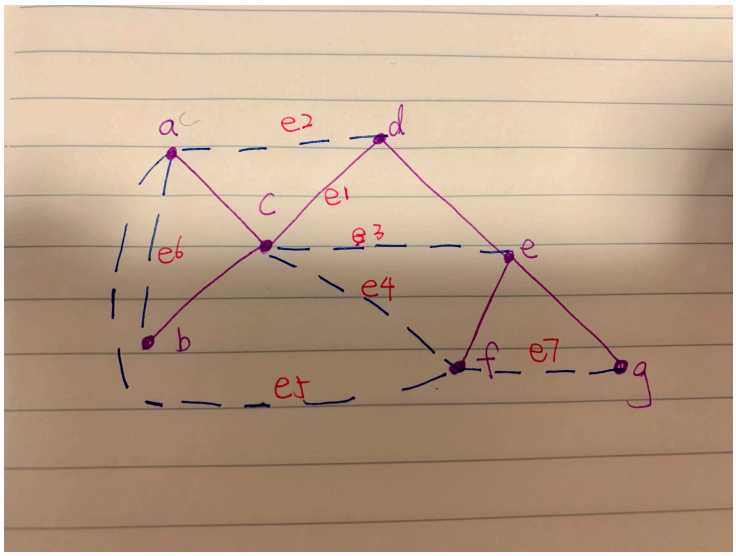
The way I choose to construct a maximum spanning tree is to use the technique in Kruskal's algorithm. In Kruskal's algorithm, the first step is to sort the edges. But instead scanning edges from the smallest to the biggest, I scan them for the largest to the smallest. One by one, I will check if its endpoints in different component. If yes, I will add that edge into my maximum spanning tree. if no, go on to the next edge. We continue this process until all the vertices are in one united component.

This will work because if the current added edge is not in the final maximum spanning tree, the currently united-two-vertices must be connect through another path. If we scan the edge in the order of from large to small, the another path, which comes later, precisely have a weight smaller or equal to to the current one. As a result, if it is smaller, we will replace it by the current edge to get a valid maximum spanning tree.

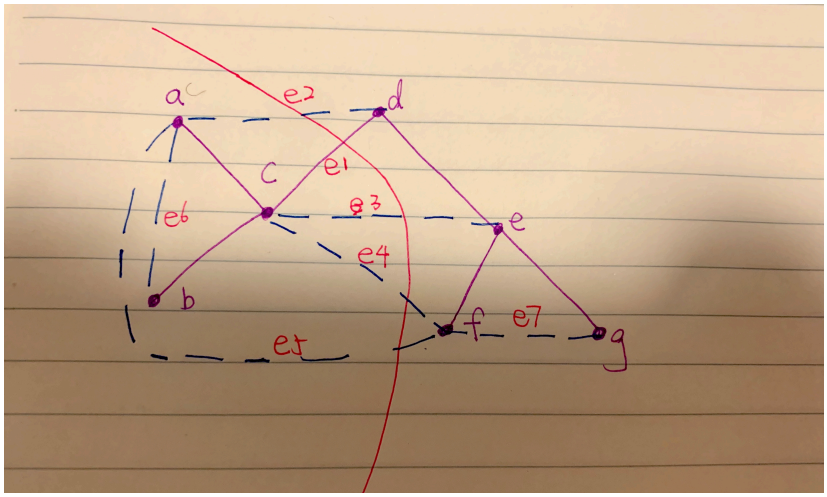
The time complexity is the same as the time complexity of Kruskal's minimum spanning tree algorithm: O(E log(V)) because the only thing different is we scan the edges from the largest to the smallest, which just needs to reversely iterate from he sorted edge list.

b)

Suppose in the following picture, purple line is the max spanning tree T, blue line are other edges in G not in the minimum spanning tree. We are trying to find that from vertex a(x) to vertex e(y), other paths from a to e other than the a-e path in the maximum spanning tree all have a edge no greater than the minimum edge in the maximum spanning tree.



Let's say e1 is the minimum edge in the maximum spanning tree T. e1 is precisely the only one edge connect the union of vertices set A {a, b, c} and the union of vertices set B {d, e, f, g} (tree property). So any other path from a to e must have one critical edge connect any vertices from set A to any vertices in set B. And all of these critical edges are the target edges that we want to prove as the one edge in any paths that no larger or than e1. To find these edges, we can use the technique edge cut to cut through the graph that ONLY cut through the minimum edge in T and all critical edges that connect set A and set B:



In the picture we can see e2 e3 e4 e5 are the critical edges while e6 and e7 are not. We need at least one of these edges in the path from a to e. They must be equal or less than e1 based on a prove on contradiction: if our premise is having a valid maximum spanning tree with e1 and e2 larger than e1. Then get rid of e1 and use e2 instead will give us a larger spanning tree, which contradicts to our premise before. As a result, every path from x to y in G contains an edge with a weight that is no greater than the smallest weight in the path from x to y in T.