Ex 1. 3. The algorithm is shown below

Algorithm 1 Hamiltonian Path Finding

Input: A DAG $G < V, E >$

Output: Whether there is a Hamiltonian Path in G.

```
function TopologicalSorting(V,E)
    $n \leftarrow 0$
    $A \leftarrow \{\}$
    while $E \neq \emptyset$ do
        $v \leftarrow$ a vertex in $V$ with no edges pointing to it
        $n \leftarrow n + 1$
        $A[n] \leftarrow v$
        $V \leftarrow V \setminus v$
        for $u$ in $V$ do
            if $< v, u > \in E$ then
                $E \leftarrow E \setminus < v, u >$
            end if
        end for
    end while
    return $A$
end function
$A \leftarrow TopologicalSorting(V, E)$
for $i \leftarrow 1 \rightarrow |A|$ do
    if $< A[i], A[i + 1] > \notin E$ then
        return false
    end if
end for
return true
```

4. For the Topological sorting part, all vertices are traversed in the outer while loop. In the inner loop, vertices are traversed to delete the edges, so the compleity is $O(|V|^2)$

For the checking part after the topological sorting, the sequence A is traversed from head to tail, so the complexity is $O(|V|)$

So in conclusion ,the complexity is $O(|V|^2)$

5. It belongs to P

Ex 2. 1. It is not bounded by a polynomial.

2. Yes because the effect to reduce a number of log^* is much larger than that of log, so apply it to n first should have a more severe effect on reduction.

3. The algorithm is shown below.

Algorithm 2 Finding a letter ball among eight

Input: Eight balls, among them there exists a lighter one.

```
Divide the balls into two groups of 4, weight them, pick out the lighter group.
Divide the 4 balls into two groups of 2, weight them, pick out the lighter group.
Weight the two balls.
return the lighter one
```

Ex 3. Rubik's cube is a 3D puzzle game based on a cube sonsisting of 26 small blocks around and a rotating axis group in the center. Each surface of the cube is paited a certain color and the blocks on each side can be rotated together around its center using the inner rotating axes. After rotation the alignment of the colors might be messed, and the goal of this game is to rotate the cube axes to make the color alignment back to the original.

Now two of the algorithms to solve Rubik's cube probelem will be introduced. The first algorithm is the most common algorithm among cube beginners, which is called a "layer-by-layer" algorithm.

The first step is to choose a base surface. Then the first sequence of move is to put the blocks which should be on the base surface but not on the corners back to their place. This time the game is just on so you can do it freely and is it simple enoungh. Then the thing is to put the corner blocks of the base surface back.

Now actually one layer of blocks are placed. The next step is to restore the middle layer. A sequence of 8 moves is needed to place one middle layer block, and repeat this 4 times you will make two layers back.

Then the top layer at last. The first thing to do to the top layer is to make a cross on the top surface, which means filling the top surfaces non-corner blocks with its color. Then based on a sequence of 7 moves, you should make the top surface filled with its color(but might not in the correct alignment)

The final step is to restore the alignment. Based on the two types of different permutation of the blocks, a 9-step or a 11-step sequence is needed. Then the cube is restored.

Another method is called ZZ method. The first step of ZZ method is to place all the edge blocks to their right place.

Then, you should place four corner blocks on one surface to their place, and the last step is to get the last four corner blocks to their right place.

References

[1] `https://rubiks-cube-solver.com/how-to-solve/`
[2] `https://en.wikipedia.org/wiki/Rubik%27s_Cube`

Ex 4. 1. Let the certificate be a path in the graph. Then what the machine should do is to traverse the path wo see whether the edges exist in E and whether a point is passed twice. So it will be a $O(|V|)$ process and thus it is NP.

3. Let the certificate be k vertices in the graph. Then check all the edges to see whether each edge contains any one of such k vertices. So it will be a $O(k|E|)$ process, so it is NP.

Ex 5. No because the divide operation will have much more influence when the number grows big, so this operation should also be included. In fact the direct trial division method is not a proper algorithm to determine that it is a P problem. And when the number grows big, the number of digits is also a factor to be considered into consideration.