Prob. 1	Prob. 2	Prob. 3	Prob. 4

Problem 1.

1. To decide whether there exists k edge-disjoint paths, we can define the capacity of each edge as 1. Thus the flow will only go through an edge at most 1 time. Then we push the maximum flow possible from s. It means a flow of $n = degree(s)$. So the maximum number of edge-disjoint paths is bounded by n. Then we see how many of the incoming edges of t will be used and check if it's bigger or equal to k. It means we pass through at most $|E|$ edges. So this algorithm runs in polynomial time.

2. To check if there are k node-disjoint paths, we can use the same algorithm as before but every time we reach a node (which is not t) by a given edge, we need to set the capacity of its other incoming edges (obviously if the in-degree of the node is bigger than 1) to 0. So a node can be accessed only 1 time. Then we see how many of the incoming edges of t will be used and check if it's bigger or equal to k. The algorithm runs in polynomial time since we visit at most $|V|$ nodes but we need also to change capacity of at most $|V|$ incoming edges.

3. Let's define E_1 as the set of all edges from s to u and E_2 the set of edges from u to t. Since it's a directed graph, E_1 and E_2 are disjoint. It means one of the k paths from s to u will not share any edge with one of the k paths from u to t. So from s to t we have at least k paths ("at least" because there may exist path from s to t that doesn't pass by u).

4. The counter example in Figure 1 shows that a directed graph may have k paths from s to u and k paths from u to t but there are less paths from s to t.

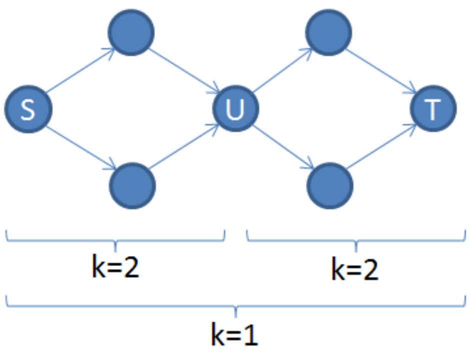

Figure 1: A counter example

Problem 2.

Problem 3.

Problem 4.

This problem is a simplification of *Convex Hull* problem because here we just need to traverse the points, no need to check if traversing is going clockwise or not. Algorithm is the following for n points:

- Find the initial point p_i which has *minimum* y, finding minimum of n points is bound by $O(n)$
- Sort the remaining points in ascending order according to polar angle that they made with p where it is bound by $O(n\log n)$ because we only spend $O(1)$ time for each polar angle calculation, in total, that results in $O(n)$ which is dominated by sorting n points.
- Traverse the points in the order and put an edge between the current point p_c and the following point p_f. Since we are traversing all points, this is $O(n)$.
- At the end, put one more edge between p_c and p_i to close the loop and create a polygon. This operation is just a constant time operation, $O(1)$.

In short, we have:

$$\begin{aligned} \textit{Total Cost} &= \textit{Find}\, p_i + \textit{Calculate polar angles with } p_i + \textit{Sort points} + \textit{Close loop} \\ &= O(n) + O(n) + O(n\log n) + O(1) \leq O(n\log n) \end{aligned}$$