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

Problem 1.

just checking degree if it is 1, take edge we are going over vertexes to check their degree, if degree is 1, we take its edge then we go over higher degree vertices, if there aren't connected, we take their smallest cost edge

we can have a set S with unexplored vertices take one vertice, if it is degree=1 put the vertice + the edge in the graph

but we have to connect the vertices of degree 1 first

for proof: we need to show that in the MST, we have 6 unused edges from the original graph.

for vertices, their degree is bigger than 1, we could add other edges but algorithm chooses their min edge so if we add another edges, total cost would be higher.

Problem 2.

Problem 3.

here, I don't really get how to calculate the gain of reliability do we have to construct our own function to calculate it? in some sense, we need to bound it but I think it would be easier after asking

for example, we can assume only nodes access file can update them

Problem 4.

So for this problem, we can represent the matrix $n \times n$ as a bipartite graph where we have n nodes on one side, representing the rows and n other nodes on the other side, representing the column. A "1" represent an edge linking its column and row.
For a bipartite graph $G = (\{U, V\}, E)$, nodes in U representing the columns and nodes in V representing the rows. Let's denote a node U_i the one representing the i-th column and V_i the one representing the i-th row.
As we can see on Figure 1, to have a diagonalized matrix we need to have an edge linking

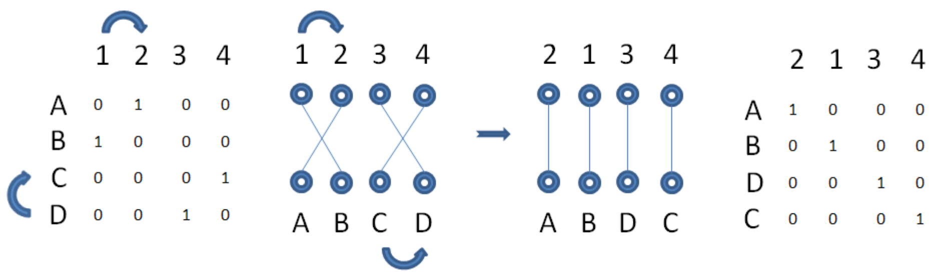

Figure 1: Illustration of swapping colums and rows

U_i and V_i, $\forall i \in \{1, 2, \ldots, n\}$. Plus swapping column i with column j means that U_i becomes U_j and U_j becomes U_i. Same logic can be apply to swap rows. So if we have n matching, it means we can diagonalize the matrix by swapping the columns and rows so we end up with n edges linking U_i with V_i, $\forall i \in \{1, 2, \ldots, n\}$. Thus our algorithm would just check if the matrix has n matchings.