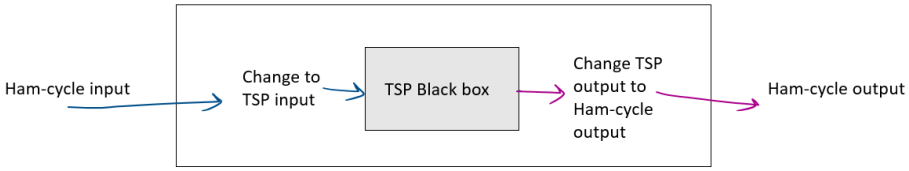
Solution: NP-Completeness and Heuristic Algorithms

1. **NP-Completeness:** Consider the Travelling Salesperson (TSP) problem that was covered in the exploration.
   Problem: Given a graph G with V vertices and E edges, determine if the graph has a TSP solution with a cost at most k.
   Prove that the above stated problem is NP-Complete
   Prove belongs NP:
   We need to show that it can be verified in poly time: If we are given a solution, we can verify that the solution covers all the vertices, and the cost is at most. All this can be done in polynomial time.

   Prove that it is NP-Complete:
   We pick a known NP-Hard problem, 'Hamiltonian Cycle' and reduce it to TSP problem.



   Change Hamiltonian Cycle input to TSP input:
   Let G = (V,E) be in an instance of Hamilton cycle. We construct an instance of TSP input as follows :
   - Let weight of edges be 0.
   - Now, add edges E` to G to make G` a complete graph. Make weight of these newly added edges to be 1.
   - This can be done in polynomial time.

   Our TSP problem input can be: "Given a graph G` is there is TSP solution on G` with cost of at most 0?"

   - If there is a solution, the Blackbox will return "yes"
   - If there is no solution, the Blackbox will return "no"

Change TSP output to Hamiltonian Cycle output:

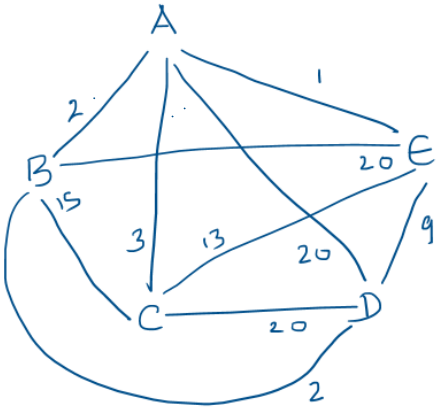
- If G' has a cycle with cost 0, every edge in that cycle should have a cost 0, thus we can say G has a Hamiltonian cycle.
- If there is no solution, that means there is no solution that visits every edge only once. Hence, the solution to the Hamiltonian cycle is also 'No'.
- This can be done in polynomial time.

We have reduced Hamiltonian cycle problem to TSP problem in polynomial time. Since Hamiltonian circuit is NP-Complete, TSP is also NP-Complete.

2. Implement Heuristic Algorithm:

a. Below matrix represents the distance of 5 cities from each other. Represent it in the form of a graph

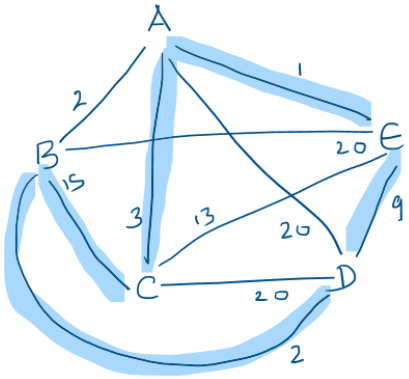
	A	B	C	D	E
A	0	2	3	20	1
B	2	0	15	2	20
C	3	15	0	20	13
D	20	2	20	0	9
E	1	20	13	9	0



b. Apply Nearest-neighbour heuristic to this matrix and find the approximate solution for this matrix if it were for TSP problem.

We will follow these steps:

- Pick an arbitrary starting point
- Pick next closest un-visited vertex
- Repeat until all the vertices are visited and return to the starting point.



Approximate TSP solution is highlighted in the graph. It has a cost of: 30.

c. What is the accuracy ratio of your approximate solution?

The optimal solution is: AB – BD – DE – EC -CA; this has a cost of 29.

Accuracy ratio = C/C* = 30/29 = 1 (approximately)

d. Write the pseudocode for the nearest neighbour heuristic.

```
def tsp(G):
    solution = {}
    current_vertex = pick a arbitrary vertex in G
    Add current_vertex to solution
    while (length of solution = total vertices in G):
        new_vertex = Find shortest edge connecting to
current_vertex that is not in solution

        Add new_vertex to solution

        current_vertex = new_vertex

    return solution
```