Khrystian Clark

Assignment 9, CS325

08MAR2022

Assignment: NP-Completeness and Heuristic Algorithms

Note: You will discuss Question 1 as part of the Group Assignment. (Check this week's Group Assignment on Canvas for details).

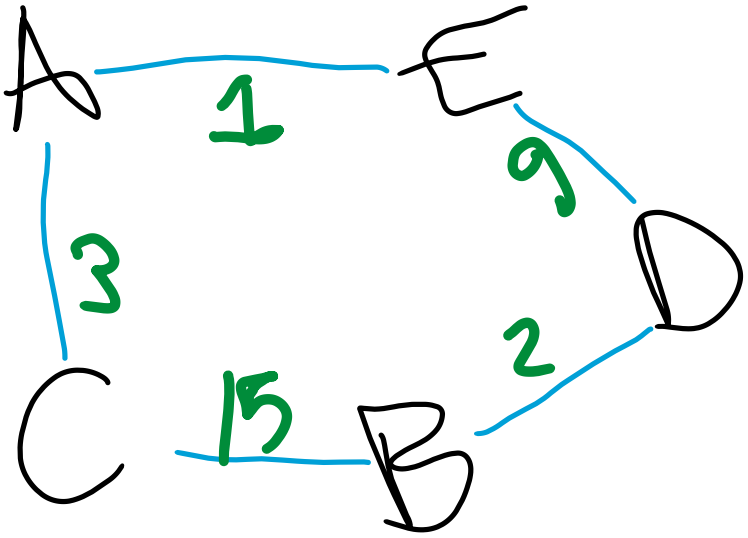
---

1. **NP-Completeness:** Consider the Travelling Salesperson (TSP) problem that was covered in te exploration.
   Problem: Given a graph G with V vertices and E edges, determine if the graph has a TSP solution with a cost of at most k.
   Prove that the above stated problem is NP-Complete.
   - Group part:
     - The steps I have:
       - A is NP
       - Find a similar NP-complete: shows B <=p A
       - Solve B with algorithm to solve A
       - Proof the solution is correct for all instances.
   - My developed answer:
     - We must prove: TSP is NP
       - The tour(G) contains each V once.
       - Minimum cost is the sum of edges, found in polynomial time, therefore TSP is NP.
     - Hamilton cycle(cycle that passes through all V in G once) $\leq_p$ TSP (this is an idea I learned during the group portion of this assignment)
       - Reduce the Hamilton Cycle to a known NP-hard problem.
         - Form a complete graph by adding edges connecting all vertices.
         - Give each added edge a value of 1
         - Original edges = 0
       - If we can find a Hamilton cycle in the updated graph that equals 0, then the graph has a Hamilton cycle.
       - Since the Hamilton cycle has been reduced to TSP, then it is shown that TSP is NP-hard. Since we've proven each step, then we can conclude that TSP is 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.
   i. A(1)→E(9)→D(2)→B(15)→C(3 to A)
   ii. 30 is the total cost

c. What is the approximation ratio of your approximate solution?
   i. Since the nearest neighbour and the optimal solutions are the same, then the approximation solution is 100%

d. Implement the nearest neighbour heuristic for TSP problem. Consider the first node as the starting point. The input Graph is provided in the form of a 2-D matrix. Name your function **solve_tsp(G)**. Name your file **TSP.py**

   Sample input
   G: [[0,1,3,7], [1,0,2,3],[3,2,0,1], [7,3,1,0]]
   Output: 11

   - Pseudo code
     - Traverse all nodes
     - Store the minimum weighted nodes
     - Store visited in order to avoid latency
   - Update the total cost as you visit the smallest nodes
   - Print the final cost of the traversal.

```
def solve_tsp(G):
    visited = [] # list of visited nodes
    y = 0    # holder values for graph traversal
    x = 0    # holder values for graph traversal
    cost = 0 # store the total cost

    while x not in visited:
        edges = G[x]  # the node placement that we update
for traversing through
        min_cost = -1 # storage for the updating of the
minimum cost in the node
        min_node = -1 # storage for smallest node
        visited.append(x)

        # find the smallest unvisited edge
        for (i, j) in enumerate(edges):
            if (min_cost == -1) and (i not in visited):
                min_cost = j
                min_node = i
            elif (min_cost != -1) and (j < min_cost) and (i
not in visited):
                min_cost = j
                min_node = i
        if min_node != -1: # find the next edge to check
            x = min_node
        if min_cost != -1: # add to the total cost
            cost += min_cost

    # adding the cost of final node to starting point
    cost += G[x][y]
    print("Output: ", cost)

# example graph
G = [[0,1,3,7],[1,0,2,3],[3,2,0,1],[7,3,1,0]]
solve_tsp(G)
```