Assignment-13 33119

Title : Travelling salesman Problem (Branch and Bound)

Problem Statement : Write a program to solve the trave--lling salesman problem and to print the path and the cost using BB.

Objective : To understand and implement least cost branch and bound algorithm for solving travelling salesman problem and slopy BB strategy.

Theory :

- Travelling Salesman Problem

Let $G = (V, E)$ be a directed graph defining an instance of TSP. Let C_{ij} be the edge $\langle i, j \rangle$, $|V| = n$ we may assume two starts and ends at 1. So solution space $S = \{-1, \pi, 1\}$, π is permutation $(2, 3, \ldots, n)\}$ S may be organized into space tree.

- Least Cost Branch and Bound.

In order to use LCBB to search TSP tree, we need to define cost function and two other function $c(\cdot)$ and $U(\cdot)$ such that $\hat{c}(R) < C(R) < U(R)$ in such that sol.

node with least c(.) corresponds to G.

Algorithm :

1. Read the no. of cities n and read the tsp-cost matrix.
2. Initialiare red-matrix to tsp_cost matrix.
3. Cost = reduce-matrix (tsp-cost-matrix) // on the // perform row and cost ~~mo~~ reduction and find cost matrix.
4. node.cost [] cost obtained in step 3.
- node.path [0] = 1 ; // no. of cities traversed equal to
node.path [1] = 1 ; // start from city 1.
node.matrix = reduced matrix obtained in step 3.
5. node = expand (node); // perform expansion of live node and get first solution.
6. IF node.cost < list [i]. cost, go to step 14.
7. else
8. While (i) do
9. If size of heap is 0 break;
10. node = delete (); // delete node from heap.
11. node = expand (node); // again start expansion.
12. If (node.cost < list [1]. cost) go to step 14.
13 end do
14. print the path using node.path
15. print cost at that node.
16. You can verify the cost
17. stop.

Function expansion (node)

```
1. while (1)
2. do
3.    count = node.path [0];
4.    k = count + 1
5.    cost = node.cost;
6.    store node matrix to some temp matrix.
7.    r = node path [count] ;
8.    for i = 1 to n set visited [i] = 0
9.      for j=1 to count set visited [path [i]] = 1;
10.   for j = 0 to n
11.   Begin for
12.     if (! visited [j])
13.   Begin if
14.     Copy the temp matrix to red-matrix.
15.     set-infinity (red-matrix, r, j)
16.     cost 1 = reduce-matrix (red-matrix);
17.     node. cost = cost + cost 1 + temp-matrix [i] [j];
18.     node . path [0] = k ; // one more city visited.
19.     node . path [k] = j ;
20.     node .matrix = reduced matrix obtained in step16
21.   insert (node)
22.  End If
23. End for
24. if (k == n) break;
25. node = delete ();
26. End while;
27. Return node
```

Insert and delete function to take care of min_heap, adjusting heap, modify the size of heap.

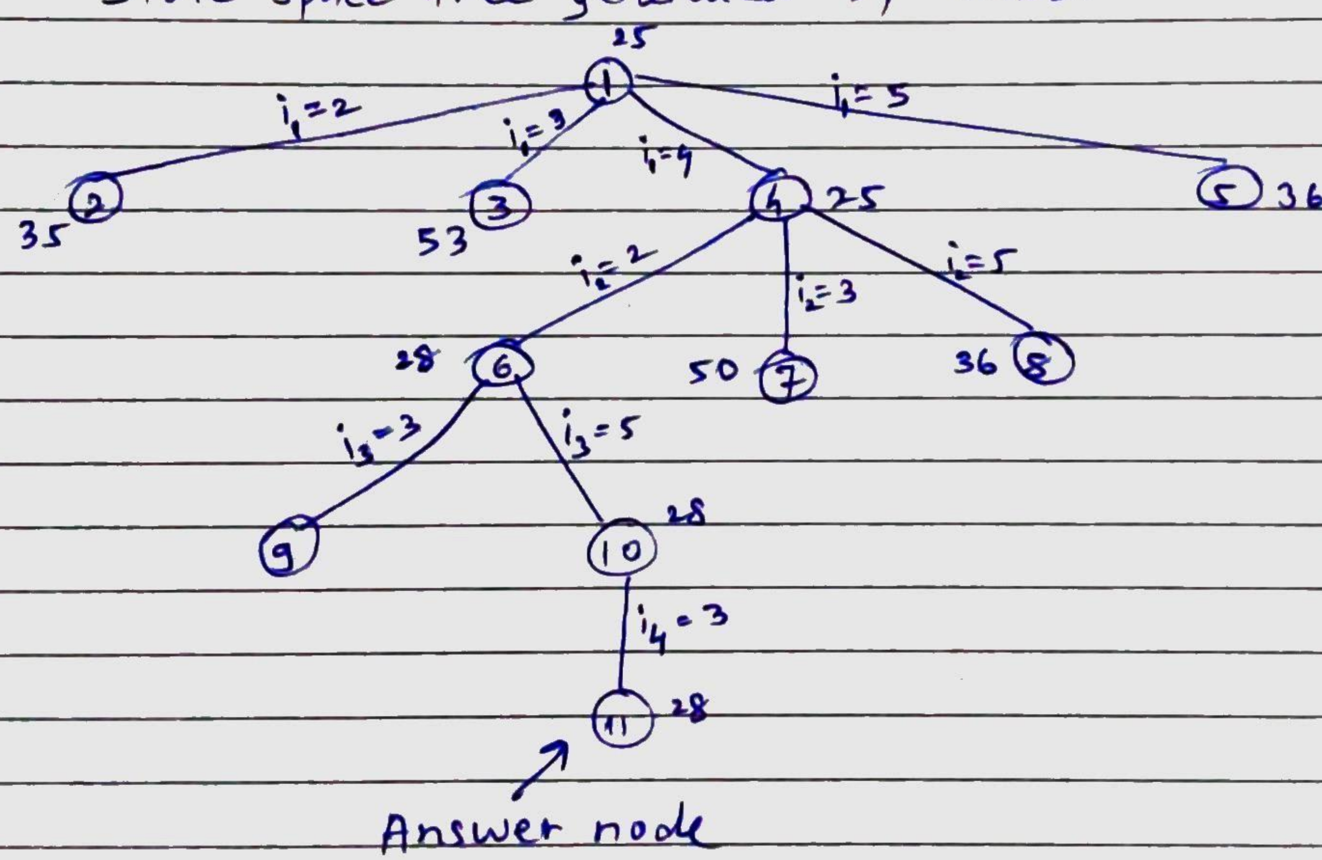
Example

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \longrightarrow \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

cost matrix

reduced cost matrix $(L = 25)$

State space tree generated by LCBB.



For node 2:
path (1,2)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

For node 3:
path (1,3).

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

For node 4:
path (1,4)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

For node (5):
path (1,5)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

For node (6):
path (4,2)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & \bullet \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$$

node 7:
path 1,4,3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}$$

For node 8:
path 1,4,5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 0 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix}$$

for node 9:
path 1,4,2,3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix}$$

node 10:

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

Input : Cost matrix TSP graph
Output : Reduced matrix showing obtained by applying LCBB.

Conclusion : The least cost Branch and Bound strategy for TSP is studied and implemented.