Design and Analysis of Algorithms

Lecture-41

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Branch and Bound

Branch and Bound

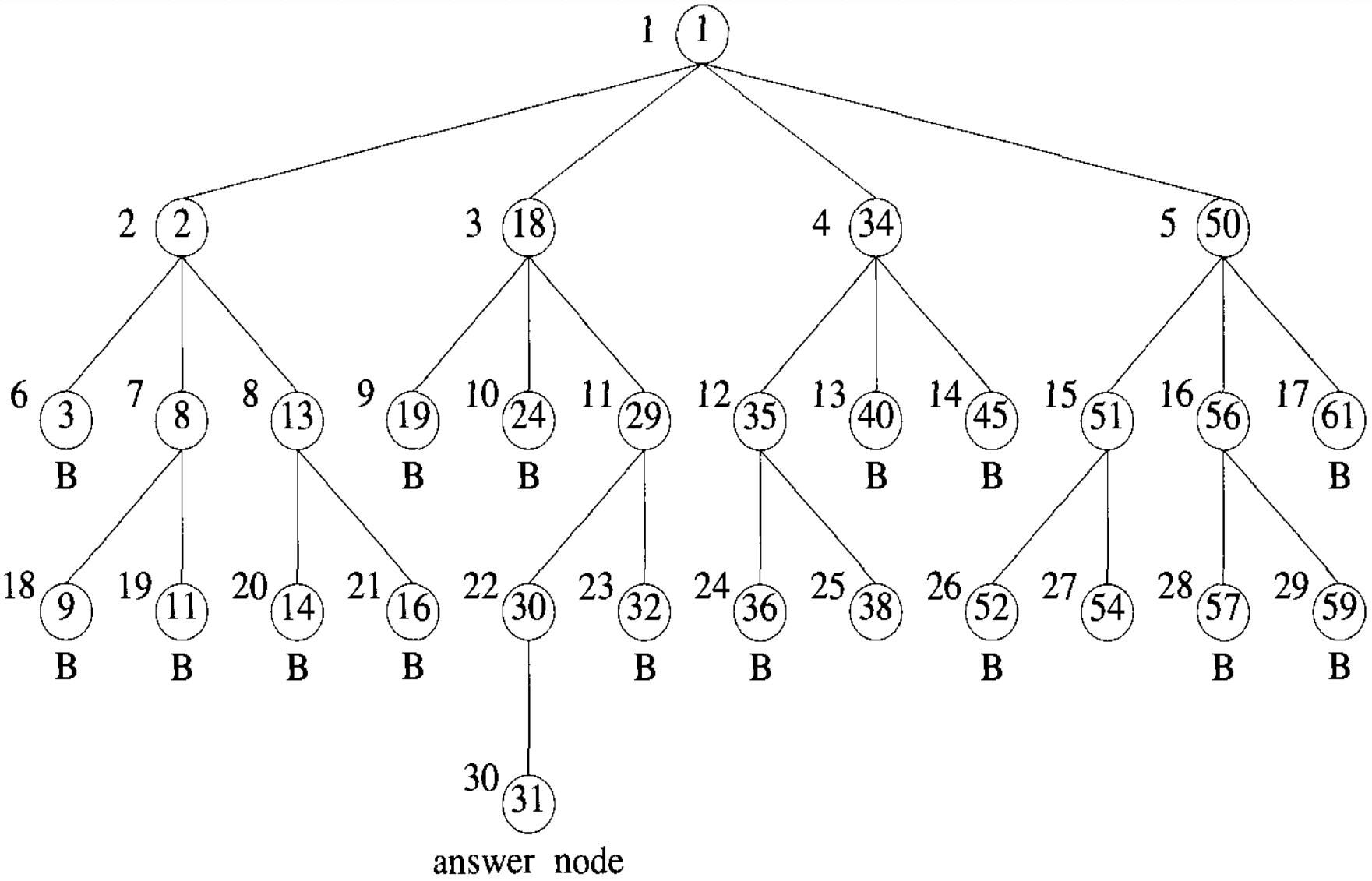
- A branch and bound algorithm is an optimization technique to get an optimal solution to the problem. It looks for the best solution for a given problem in the entire space of the solution.
- Branch and bound is a systematic method for solving optimization problems. B&B is a rather general optimization technique that applies where the greedy method and dynamic programming fail. However, it is much slower. Indeed, it often leads to exponential time complexities in the worst case.

Branch and Bound

- Branch and bound refers to all state space search methods in which all children of the E-node are generated before any other live node can become the E-node.
- There are two graph search strategies, BFS and D-search, in which the exploration of a new node cannot begin until the node currently being explored is fully explored.
- In branch and bound terminology, a BFS-like state space search will be called FIFO(First In First Out) search as the list of live nodes is a first-in-first-out list (or queue).
- A D-search-like state space search will be called LIFO(Last In First Out) search as the list of live nodes is a last-in-first-out list (or stack).

Example: Consider the 4-queen problem.

FIFO branch-and-bound algorithm to search the state space tree for this problem.



Least Cost (LC) Search Branch and Bound(LCBB)

- In both LIFO and FIFO branch-and-bound the selection rule for the next E-node is rather rigid and in a sense blind.
- The selection rule for the next E-node does not give any preference to a node that has a very good chance of getting the search to an answer node quickly.
- The search for an answer node can often be speeded by using an "intelligent" ranking function c’ for live nodes.
- The next E-node is selected on the basis of this ranking function.
- A search strategy that uses a cost function c’(x) to select the next E-node would always choose for its next E-node a live node with least c’. Hence, such a search strategy is called an LC-search(Least Cost search).

Travelling Salesperson Problem

In this problem, we have given a directed graph with cost matrix.

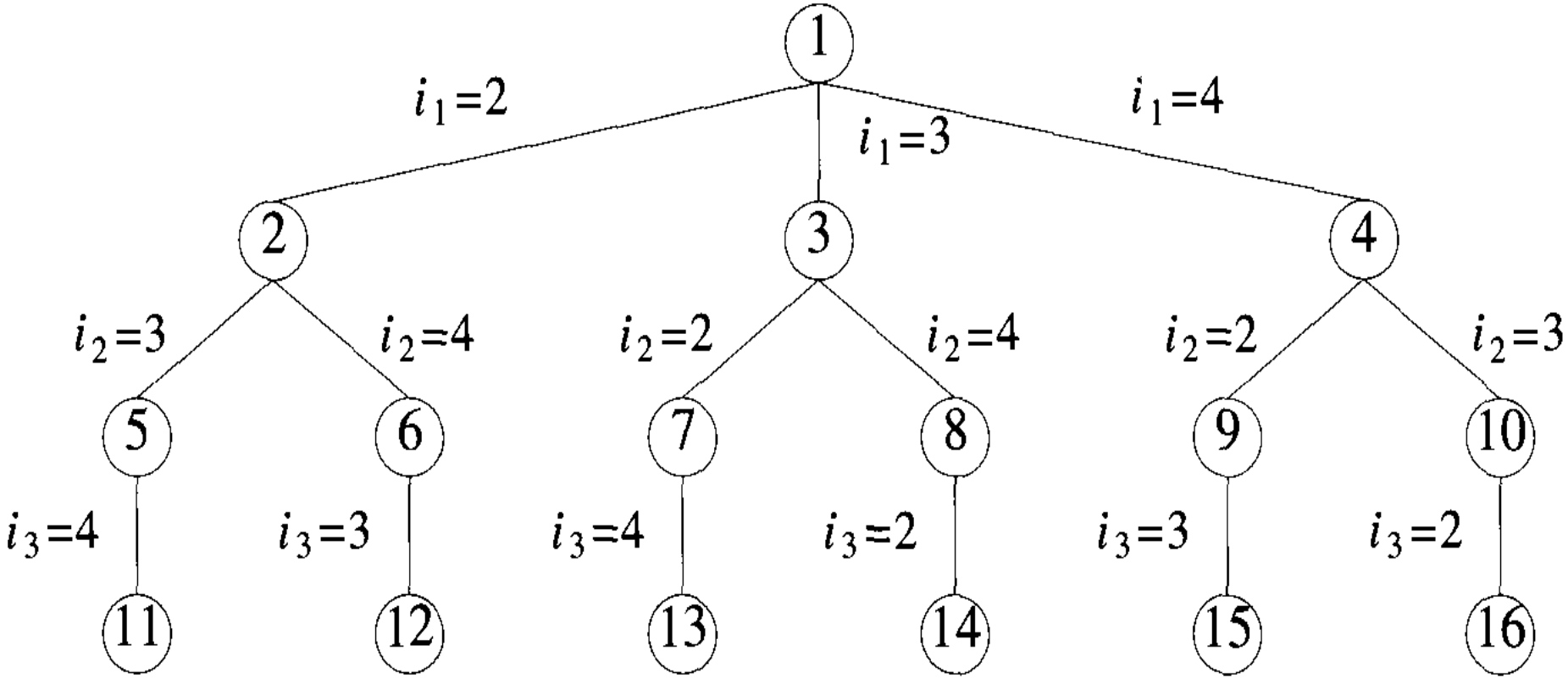
We have to find a shortest tour of the graph which contains all the vertices of the graph with no vertex repeated except first and last vertex.

Travelling Salesperson Problem

- Let G = (V, E) be a directed graph defining an instance of the traveling salesperson problem. Let c_{ij} denote the cost of edge(i, j), $c_{ij} = \infty$ if $(i,j) \notin E$, and let $|V| = n$.
- Without loss of generality, we can assume that every tour starts and ends at vertex 1. So, the solution space S is given by S= {1, **π**, 1 ! **π** is a permutation of (2,3,… , n)}.Then |S|= (n-1)! .
- The size of S can be reduced by restricting S so that $(1, i_1, i_2, \ldots\ldots, i_{n-1}, 1) \in S$ iff $(i_j, i_{j-1}) \in E$, $0 \leq j \leq n-1$ and $i_0 = i_n = 1$.
- S can be organized into a state space tree.

Travelling Salesperson Problem

Following figure shows the tree organization for the case of a complete graph with |V| = 4.



State space tree for the traveling salesperson problem with n = 4 and $i_0 = i_4 = 1$

Travelling Salesperson Problem

- Each leaf node L is a solution node and represents the tour defined by the path from the root to L. Node 14 represents the tour $i_0 = 1$, $i_1 = 3$, $i_2 = 4$, $i_3 = 2$, and $i_4 = 1$.

Reduced row or column:

A row (column) is said to be reduced iff it contains at least one zero and all remaining entries are non-negative.

Reduced cost matrix:

A matrix is said to be reduced iff every row and column is reduced.

Travelling Salesperson Problem

- To solve the travelling salesperson problem using LCBB method, we construct state space tree.
- We compute reduced cost matrix at each node of the state space tree. And also we compute a cost function c' at each node.
- Reduced cost matrix of root node is computed directly from given cost matrix of the graph.
- If C is the cost matrix of the graph, then reduced cost matrix of C is the reduced cost matrix corresponding to root node.
- If c' represent the cost function at each node to determine the optimal tour, then

c' for root node = Total amount subtracted in determining reduced cost matrix at root node

Travelling Salesperson Problem

Reduced cost matrix of the remaining node is computed in the following way:-

Let A be the reduced cost matrix for node R. Let S be a child of R such that the tree edge (R, S) corresponds to including edge (i,j) in the tour. If S is not a leaf, then the reduced cost matrix for S may be obtained as follows:

(1) Change all entries in row i and column j of A to ∞.

(2) Set A(j,1) to ∞.

(3) Reduce all rows and columns in the resulting matrix except for rows and columns containing only ∞.

If r is the total amount subtracted in the step (3), then

$$c'(S) = c'(R) + A(i,j) + r$$

Travelling Salesperson Problem

Example: Solve the following travelling salesperson problem whose cost matrix is the following:-

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

Solution: The reduced cost matrix corresponding to root node will be the following:-

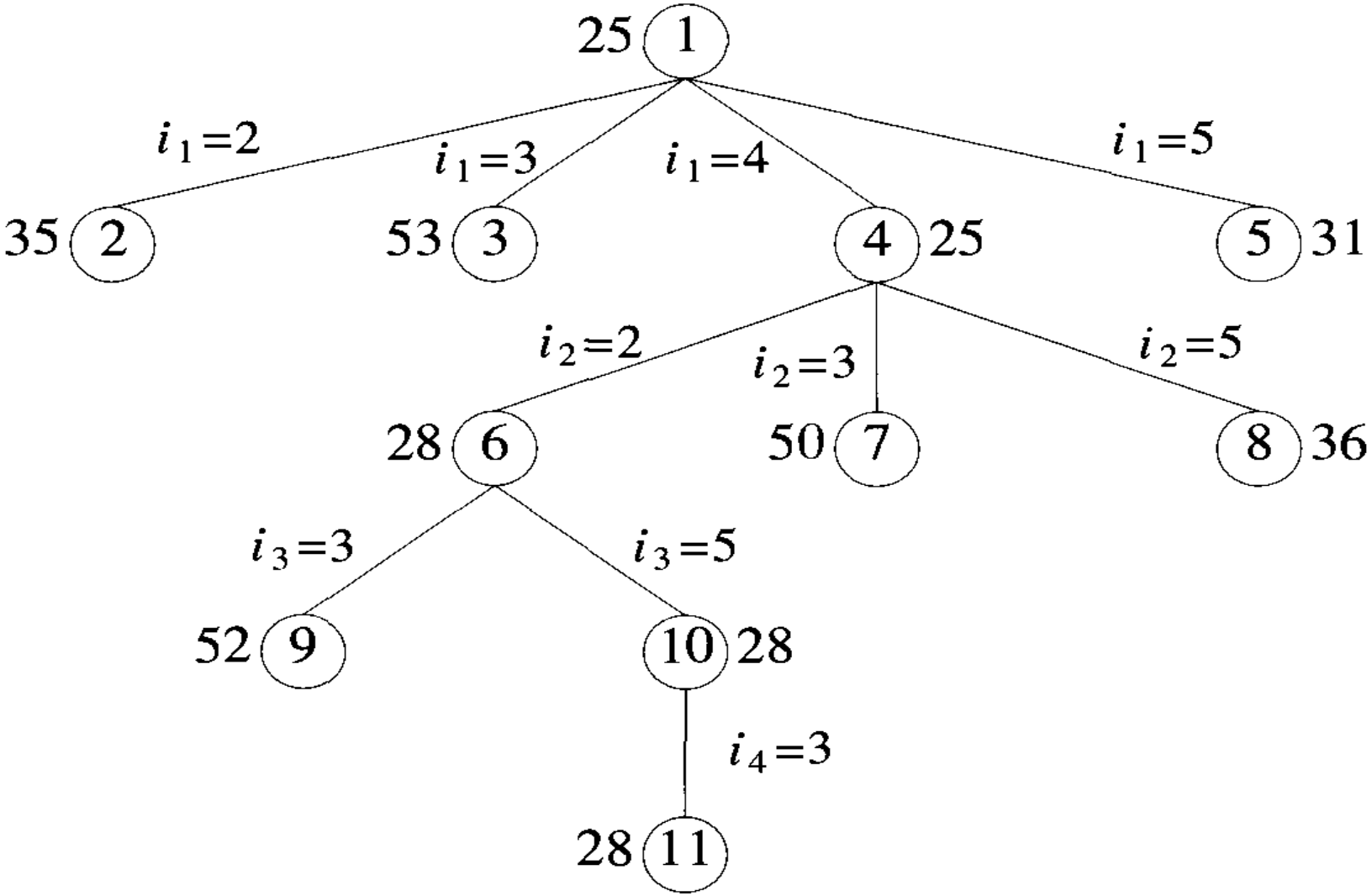
$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

And value of cost function at root node is

$$c'(\text{root}) = 25$$

Travelling Salesperson Problem

The portion of the state space tree that gets generated is



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

(a) Path 1,2; node 2

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

(b) Path 1,3; node 3

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

(c) Path 1,4; node 4

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

(d) Path 1,5; node 5

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

(e) Path 1,4,2; node 6

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

(f) Path 1,4,3; node 7

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

(g) Path 1,4,5; node 8

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

(h) Path 1,4,2,3; node 9

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

(i) Path 1,4,2,5; node 10