# CHAPTER 11

# ORACLE FOR LONGEST PATH

The longest path problem determines whether a given path is longer than a path traversed by the user. We formulate the problem as a graph and traverse along a path of the graph. The graph is encoded as edges. The edges are encoded in binary. The distances for the edges are encoded as integer.

The longest path problem can be explained by illustrating a problem. In the graph given below, we have the initial node as N1 and the final destination node as N4.

We have the edges between nodes as e1, e2, e3, e4, e5. As there are 5 edges we can encode it using 3 bits. This method can be extended for ‘n’ edges. The distances for particular edges are also given.



Figure ‑. Graph for the Longest Path

The user can only traverse in the direction given in the figure above. The user just needs to give the edges to be traversed and the length with which he wants it to be compared. If the path to be traversed is e1, e3, e4 and the length to be compared with is 200. Hence, they would be traversed in the path shown in figure3.



Figure ‑ Example which shows how the node selection is done.

As the path traversed is in the right direction, the path lengths of the edges are added as 45 + 35 + 90 = 170, which is less than 200. Hence the output of this would be False as it is less than the given path length.

Black box for the oracle can be shown below.



**Figure 11‑3 Oracle for Longest Path**

If the edges selected are e1, e2, e3 and the path length to be calculated is 300. The oracle will check whether the direction is proper.



Figure ‑ Example where the edge selection is not done properly

The oracle does not even calculate the distance, because the direction traversed is not proper and does not reach the destination node. The output is ‘false’ in this case.

This problem for longest path can be generalized to any number of nodes. The same method can be used for finding the shortest path, equal path, by changing the comparator in longest path.

For any graph the generalized oracle would be as shown in the figure given below.

Figure ‑ Oracle for ‘n’ edges

**Node Representation**

The nodes are first encoded in the code. We have the following constraints for the nodes:

The path starts only at the first node and ends at the destination node.

We have to enter and leave a node only once.

The nodes are encoded in the following way:

Basing on the constraints given above the graph given is coded in the following way:

There are four nodes in total. Hence covering all nodes at maximum we have 3 edges to cover.

We can encode the edges using a binary number. We can have edge1 – 001, edge2 – 010, edge3 – 011, edge4 – 100, edge5- 101.

The first input can be either edge1 or edge2. The second input can be 000 or edge3 , the third input can be either edge4 or edge5.

The path lengths are initially given by the user and we add them once the edges are selected.

I have used a multiplexer for selecting the edges and adding the distances. A multiplexer is used to select the edges and if the edges are selected properly we add up the distances. The select signal would be from the nodes. The input will be distances.

By using these rules we can encode the nodes in the oracle. These rules are applicable to any number of nodes. I have illustrated it for the graph given in the figure1.

The nodes can be represented in the following way:

The equation for node1 can be represented as e1 + e2 = node1. The node1 can start either from edge1 or edge2. And it can be only 1st input as it is the starting node.



Figure ‑ Representation for Node1

The node2 is represented as (edge1.edge3) + (edge3.edg4) + (edge1. edge0) = Node2.

For the node2 we have three edges. So it can start either from one edge and move on to the other edge. Hence, we have three different entry and exit points.



Figure ‑ Representation for Node2

The node3 can be represented as edge1.edge2 +edge3.edge5+edge2.edge0 = Node3.

For the node3 we have three edges. So it can start either from one edge and move on to the other edge. Hence, we have three different entry and exit points.



Figure ‑ Representation for node3

The node4 can be represented as edge4 + edge5 = node4.

This is the destination point for the graph given. This would be the final input for the node calculation.



**Figure 11‑9 Representation for Node4**

Nodes are connected using multiplexers based on the connections between nodes. So, once an edge is selected, we have the output of that mux as ‘1’ and connections between the nodes are checked in this way.



Figure ‑ Connection between Nodes

**Path Length Calculation**

The algorithm for path length calculation is as follows:

* The path lengths and edges are already encoded.
* The user inputs edges through which we have to traverse the path.
* If the edges are traversed based on the constraints given above we have an output of ‘1’ from the node selection design.

Hence if the input1 = “001”, input2 = “011” and input = “100”. Then we have an output of ‘1’ from the node selection design.

If the input1 = “100”, input2 = “101” and input3 = “111”, we have an output of ‘0’ from the circuit. This input acts as a select signal for the distance calculation. If the nodes are selected properly we pass the total distance oracle output.

The distance is simply calculated based on the edges given by the user.

***Figure 8: Circuit for the Path length calculation***

ez1, ez2, ez3, ez4, ez5 are the outputs from the ‘and’ gate of input and the edge. So if input2 is given as ‘011’ then we have the ez3 as ‘1’ and the particular length of the edge which is represented as de3 is selected and added using the adder. The distances of edges are represented as de1, de2, de3, de4, de5. Depending on the edge selected we add the distances.

**Final Multiplexer:**

A multiplexer is used finally to represent the output. If the select signal is one, then the output form Longest Path oracle is transferred. If it is zero then a zero which represents the edges are not properly selected.



**Figure 12.9: Final Multiplexer for Longest Path Oracle**