VLSI CAD Mid Review Document

Author – Pratik Brahma and Ayush Agrawal

This mid-review shows the combined work done by Pratik Brahma and Ayush Agrawal

Bellman Ford Code is written by Ayush Agrawal

Data Parsing is implemented by Pratik Brahma

# Bellman – ford algorithm

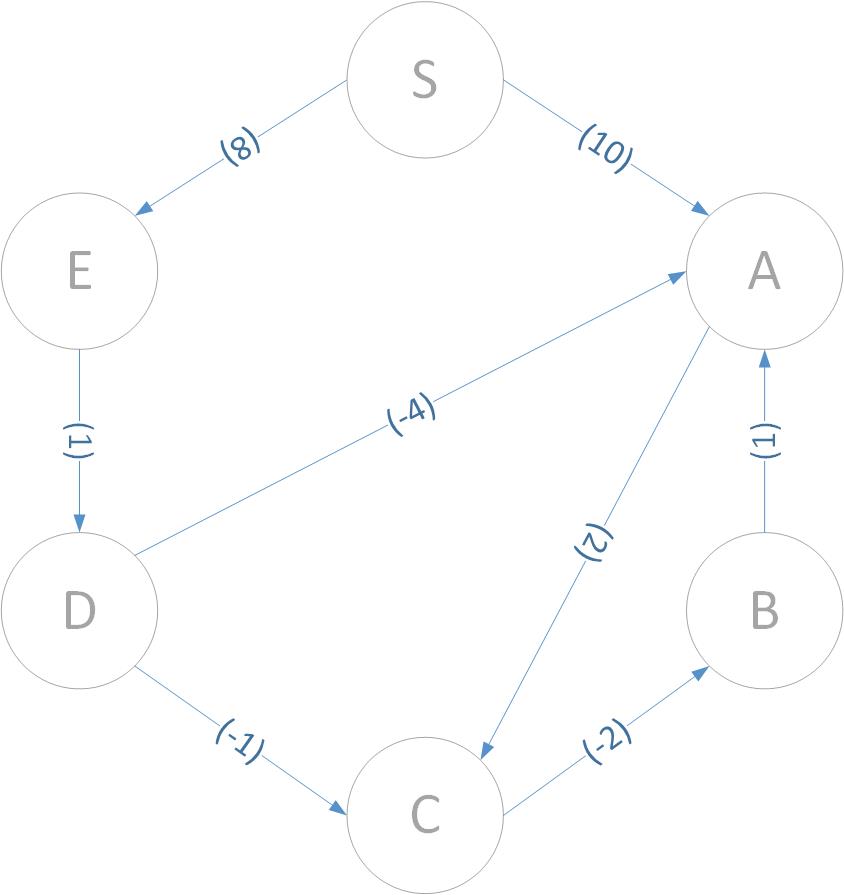
The **Bellman–Ford algorithm** is an algorithm that computes shortest paths from a single source vertex to all the other vertices in a weighted digraph. It is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution.

It takes in a weighted diagraph as an input with identified source vertex. It gives the output as a list of shortest paths from a single source vertex to all the other vertices in a weighted digraph and if a negative cycle is present in the graph.

Bellman–Ford runs in  {\displaystyle O(|V|\cdot |E|)}time, where {\displaystyle |V|} and {\displaystyle |E|} are the number of vertices and edges respectively.

For understanding the algorithm, consider the following:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | A | B | C | D | E |
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |

 INITIAL CONDITION:

ITERATION 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | A | B | C | D | E |
| 0 | 10 | 10 | 12 | 9 | 8 |

ITERATION 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | A | B | C | D | E |
| 0 | 5 | 10 | 8 | 9 | 8 |

ITERATION 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | A | B | C | D | E |
| 0 | 5 | 5 | 7 | 9 | 8 |

ITERATION 4:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | A | B | C | D | E |
| 0 | 5 | 10 | 7 | 9 | 8 |

ITERATION 5:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S | A | B | C | D | E |
| 0 | 5 | 10 | 7 | 9 | 8 |

In general, the algorithm requires a maximum of (number of nodes -1) iterations to get to the result. But in the above case we required only 3 iterations out of the total 5.

Click on the following to view the code: [Bellman Ford](bellman_ford.py)

# data structure formation

The data structure used throughout the code is a dictionary which has the following keys

|  |  |
| --- | --- |
| Key | Definition |
| costij | It gives the cost of the edge from node i to node j |
| capij | It gives the maximum amount of flow that can be passed through an edge (i->j) |
| flowij | It gives the current flow in the edge from node i to node j |

The function takes input in the from of a string which is of the type

**"(0->3,10,5);(0->1,10,5);(1->2,10,5);(3->4,10,5);(4->5,10,5)”**

This is of the form of a list of tuples. Each tuple definition is as given below

|  |  |
| --- | --- |
| Tuple Index | Definition |
| 0 | It defines the edge being added |
| 1 | It defines the cost for the current edge added |
| 2 | It defines the maximum capacity of the current edge added |

Click on the following to view the code: [Parsing](parsing.py)

**Future Task:**

1. Visual representation of graphs using pyEda