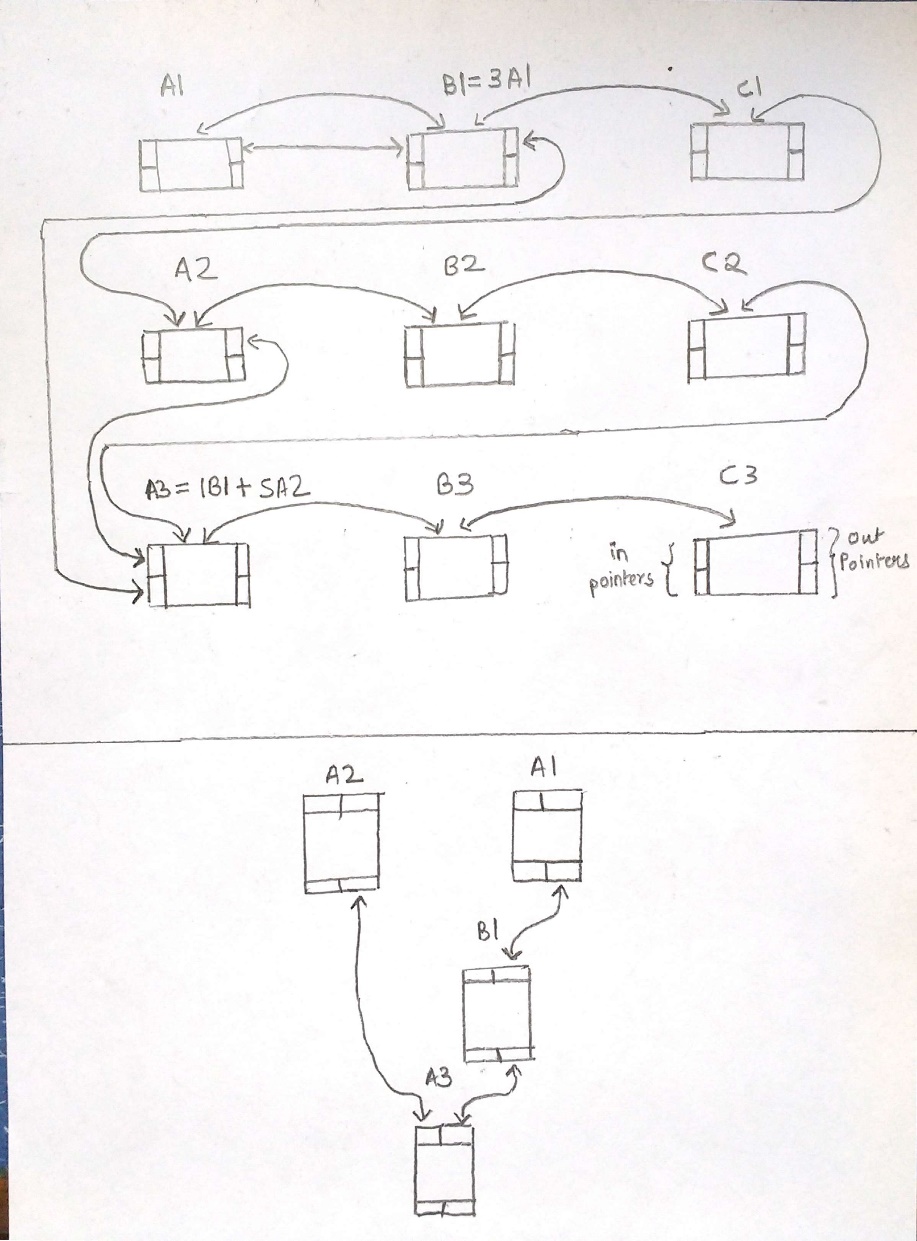
**Spread Sheet Algorithm**

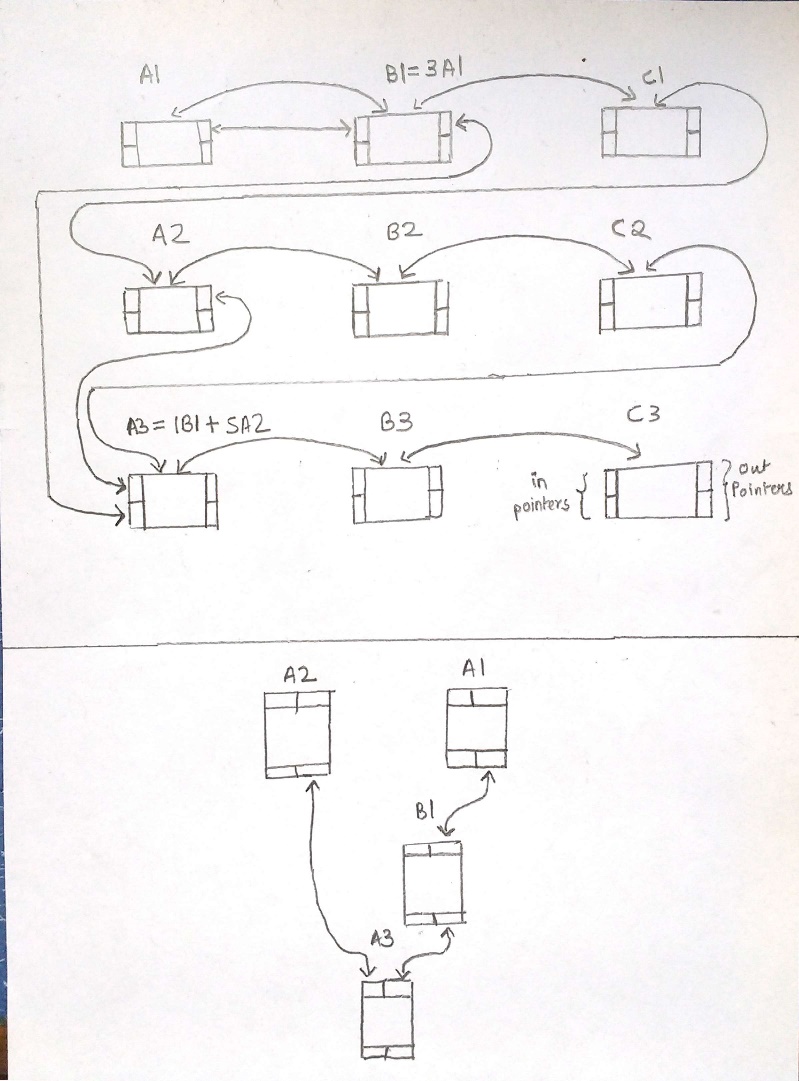
**Maitrey Mehta 1401040**

**Idea:**

The data structure used behind this approach is multi linked list. For testing purpose, I have taken into consideration a 3x3 sheet with at most two formulas where each formula can have at most two terms. Cells in the spreadsheetare represented as objects of a base class Node. Each node instance contains a unique index and a value it holds. Additionally, it contains N ‘in’ pointer, where N is the maximum number of terms in a formula (here 2). It also contains M ‘out’ pointers, where M is the total number of formulas that can exist in a spreadsheet. Next and Previous pointers are taken to maintain a doubly linked list. The overall structure of the spreadsheet will be as follows-



Here ‘in’ pointers contain addresses of nodes on which its value depends (as in A3’s in pointer will point A2 as A3’s value is dependent on A2) whereas ‘out’ pointers contain addresses of nodes that depends upon its value (as in A1’s out pointer will point B1 as B1’s value is dependent on A1). Note that if we separate each formula, it almost represents a dependency tree.

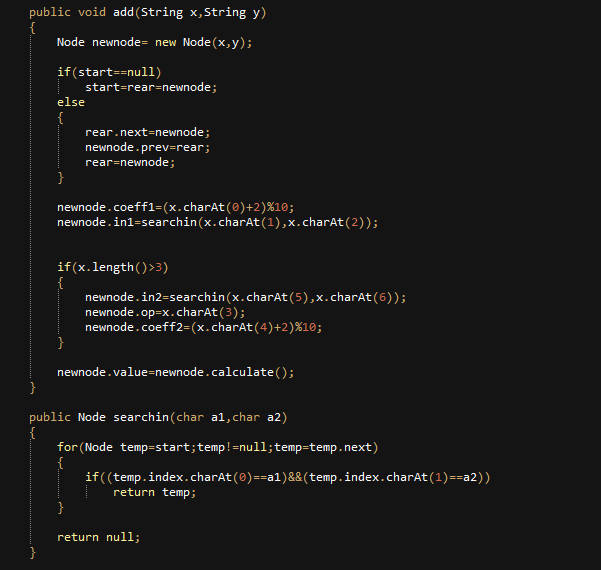


Hence, any change in value at a node is initiates recalculation of the values of the Nodes located at ‘out’ pointers.

**Analysis:**

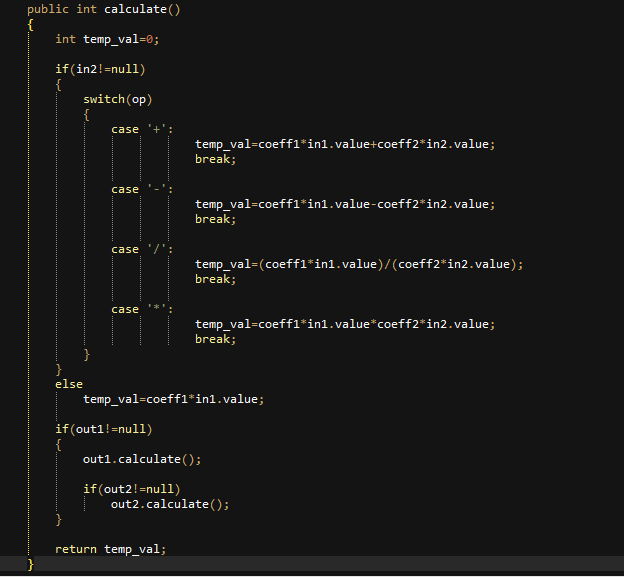
1)Adding a Node: It follows the same adding algorithm as a queue. So the time complexity is O (1).

2)Accessing the Formula:



Assuming a PxP matrix, the time complexity of assessing the formula can be defined as O (P2).

3)Calculation:



The complexity of calculate function is O(1).