## horizontal line



ADSA-Assignment 1

15.08.2016

**Table Updation Algorithm**

Krima Doshi

ICT Semester 5

1401035

# 

# 

# 

# Overview

Let M be an 26×10 matrix (think of Microsoft Excel file) with following structure

• Each cell of M is indexed by iσ, where i ∈ N \ {0} and σ is an alphabet

or set of alphabets.

• Every cell iσ is either a constant number or a formula whose variables are

other cells of M.

Clearly list down approach(s), required function(s) with inputs and outputs.

Make necessary assumptions (Size of the matrix: 100,000× 100,000, having at

most 10,000 formulas and every formula has at most 15 variables) for the same.

**Approach**

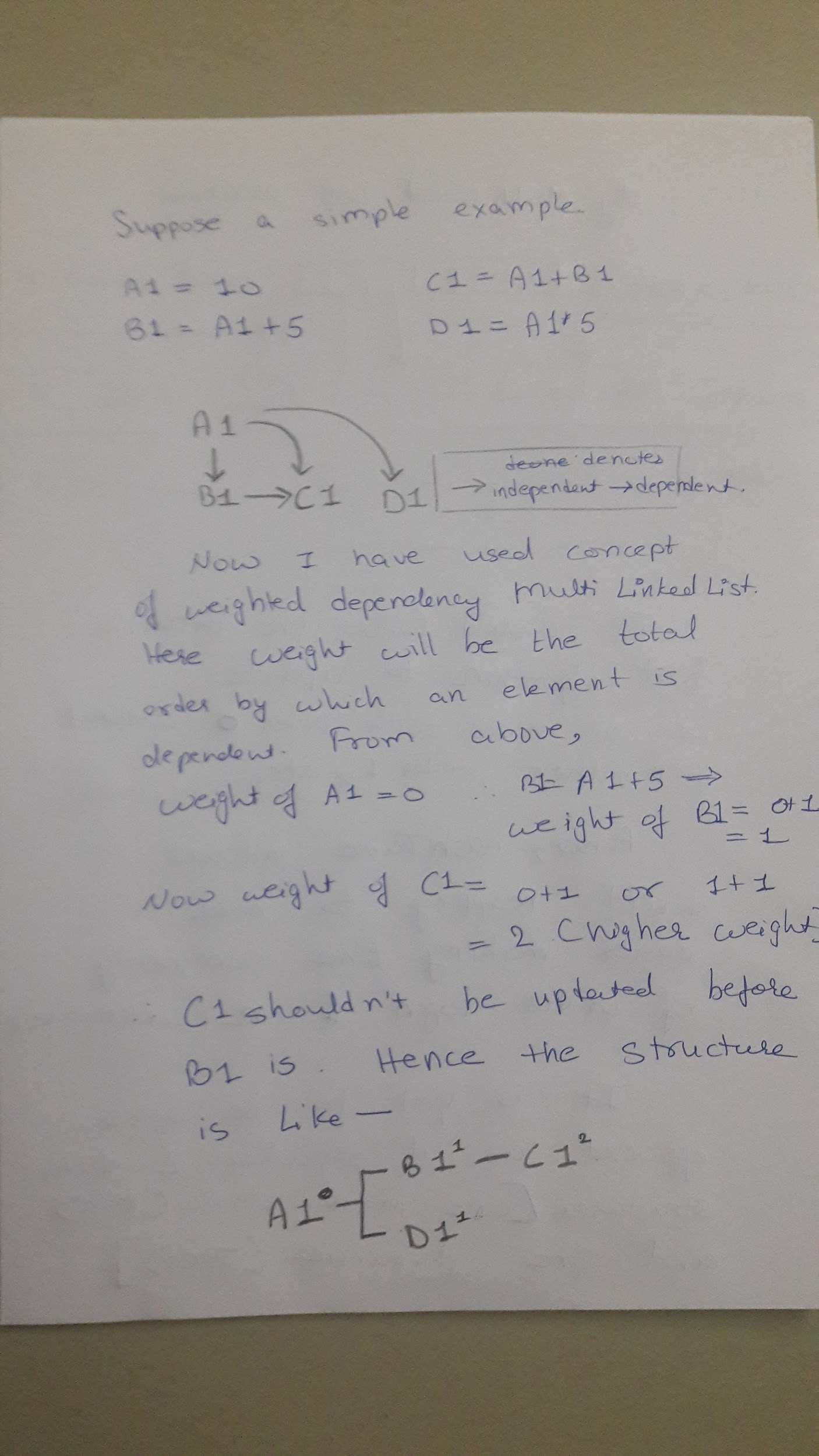
A hash map mapping between the cell address and Dependency Tree Node. A dependency tree with each node as a doubly multi-linked list for all the operations on the table.

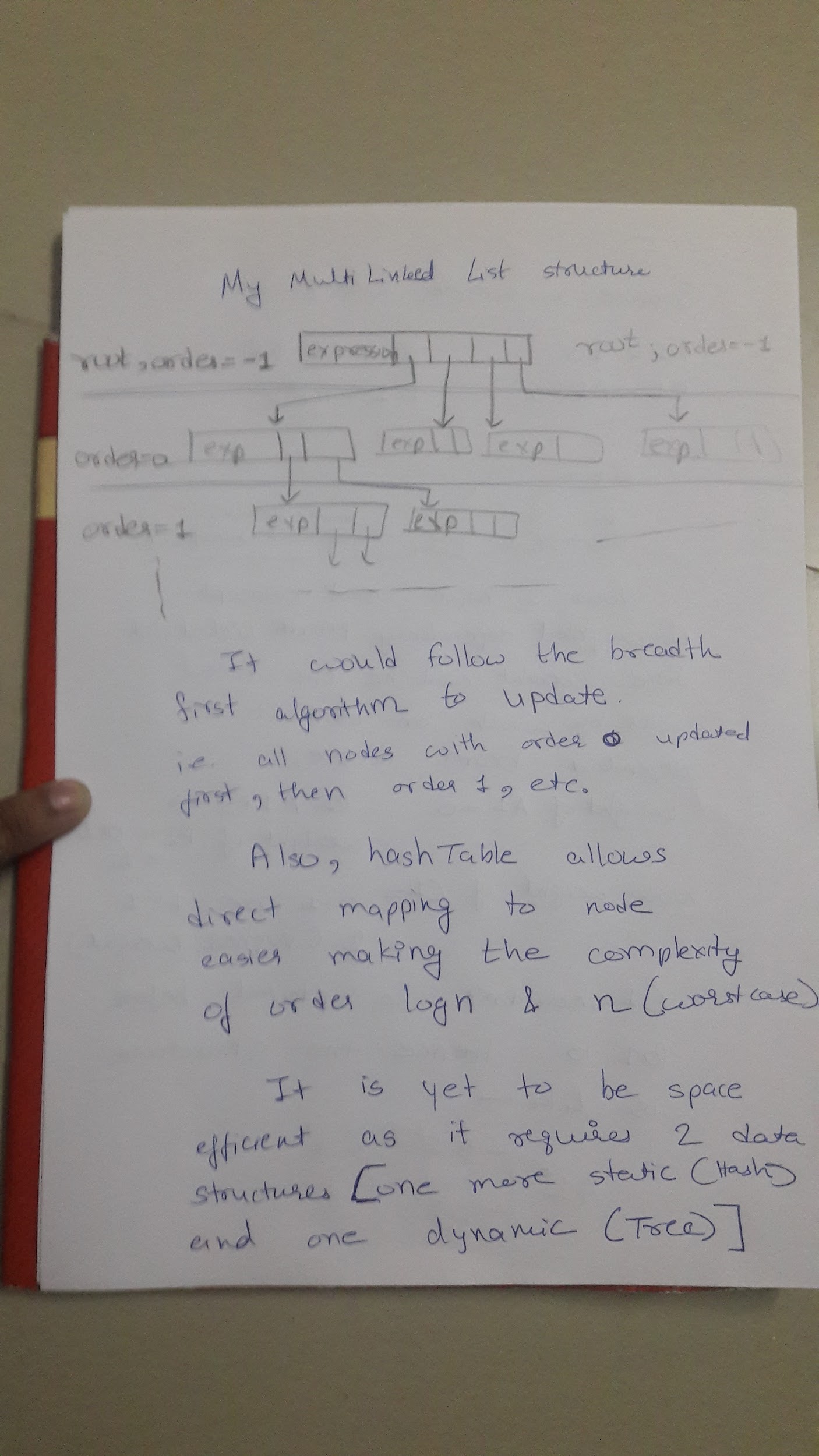
## Hash Table

The hash table is nothing but a 2 dimensional array (i.e. table) that maps each cell with its node in the dependency tree. The Hash Table will allow accessing the data in O(1).

## Multi-Linked Dependency Tree

The root of this tree will not contain anything. Each node contains a linked list of its dependencies, next node, value, expression as well as Hash Node of the cell. This algorithm will take O(log n) and at worst O(n). The algorithm followed:





# Important Functions

## Hash Table

**void display() -** display the whole table

**LLNode getNode(String name)** - get the index of the given cell { i.e convert A1 into [0][1]}

**void update(int i, int j, String exp)** - update a particular cell

## Hash Node

**void updateExpression(String exp)** - this function takes String for input, converts it into equation and stores the final value in VALUE.

**LLNode getLink()** - returns the corresponding Dependency Tree link to the cell.

## Multi-Linked Dependency Tree

**void decryptExpression()** - splits the given expression into constituents and then stores the value of the solution in VALUE.

**void addVar(LLNode var)** - add a variable link to the given cell node (max 15).

**void LLNodeAdd(LLNode dependent)** - add a linked list of the nodes dependent on the given node

**void update()** - update the given cell as well as the subsequent ones dependent on it.