Advance Data Structure and Algorithm

Dependency Tree

Maharsh Patel

Roll No:1401109

1. Problem Statement:

Design, implement and analyze a space and time efficient algorithm and necessary data-structure for the cells of M such that when a particular cell i0σ0 is updated there is least amount of computation required to update the cells depending on i0σ0 and thus producing an efficient matrix update mechanism. Here i0 is a letter and σ0 is a

1. Design:
2. Data Structure:

Here we will use Directed Acyclic Graph such that:

* All vertices of DAG denote a cell containing a value or a formula.
* Edges are created in such a way that if cell b1 contains the formula of cell a1 than a directed edge is created from cell a1 to b1 and is denoted by (Ca1,Cb1).

1. Algorithm:
2. Addition of new cell:

* A cell will first ask for value or formula.
* A vertex which contains above cell will created.
* According to value or formula, edges will be created as stated in above para.

1. Editing the value of cell:

* Depth First Search is implemented to get the topological ordering starting at the edited cell.
* The ordering is in such a way that is there is directed edge from a1 to b1 than a1 should come before b1 as if a1 is affected than b1 is affected hence bringing no errors as b1 depends in a1.
* Topological sorting will be implemented in a stack.
* A cell is pushed onto stack only if all of its neighbors are visited. This occurs right before a cell returns from dfs and hence a topological ordering is obtained.
* Once stack is obtained we will evaluate by poping it elements from top to bottom and evaluating the formula if required.
* Hence editing of cell and updating of dependent cells is done

1. Editing the formula of cell:

* Same method for Editing the value of cell can be used.
* Depth First Search will handle the according to the formula

1. Removing a formula:

* To remove the formula by replacing it value, we have to remove the directed edges.
* To remove all edges, we will point edges of the vertex of the above cell to null.
* Then create a new vertex of above cell and add its value.
* If we want to just remove the formula, we will just point edges of its vertex to null.

1. Implementation:
2. Create class graph containing:

* No. of vertices which can be calculated using (no of rows\*no of columns).
* Adjacency List containing the objects of class cell (implementation of graph).
* Function to implement Depth First Search

1. Create class cell containing:

* Value of cell
* Formula of cell
* Functions to implement the algorithms discussed under Design section.
* A stack that will have topological ordering of cells

1. Analysis:
   1. Addition of new cell:

* If new cell contains a value then no edge creation is required hence time complexity is O(1) = c.
* If new cell contains a formula than no of edge is equal to no of variable in formula which can at most be 15. Therefore time complexity of addition of formula is O(1) = c\*15 = c1.
  1. Editing a cell:
* Editing a cell affects the cell which are both directly and indirectly dependent on it.
* Let no of direct dependency be Nd and no of indirect dependency be Ni.
* So we need total (Nd+Ni) updates.
* Now, dfs will have time complexity of O(Nd+Ni).
* And Stack Operation will have to update (Nd+Ni) cells. Hence its complexity will be same O(Nd+Ni).
* Total time complexity will be O(Nd+Ni).
  1. Removing of cell:
* Let no of edges connected to cell be n.
* So according to the algorithm we will have to make every edge null.
* Hence time complexity of this algorithm will be O(n).