**SPREADSHEET ALGORITHM**

* A spreadsheet is a finite set of cells arranged as a matrix. A cell is a set that contains three elements of interest :

1) A value

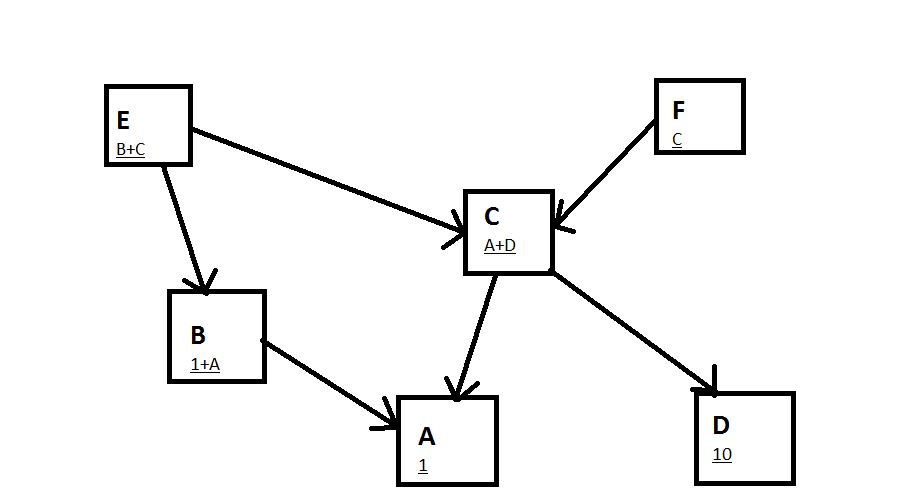
2) A formula

3) A cell reference

* A formula is an expression that defines a cell’s value as an expression f.
* A cell reference is an ordered pair that specifies a cell uniquely within the spreadsheet.
* For example, if cell M depends on N and P and N depends on Q and P depends on R, then the value of M is expressed as a composite function

M ( N (Q), P (R) ).

* For this expression to be fully resolved, the formulas for both Q and R must be constant formulas.
* A cell’s value is completely determined recursively by its formula.
* This recursion ends when a cell with a constant formula is reached.
* Each cell referred in the formula of a given cell Xis referred to as the child of X.
* The relation among the dependent cells in a spreadsheet may be represented as a weakly connected graph, G(V,E), where V is the set of vertices and E is the set of edges.



* Now if there is any change in one of the cells let’s say cell A, then there should be immediate change in those cells which are directly dependent on the cell, let’s say B and C.
* The algorithm for discovering the set of dependent cells of a given cell is thus recursive since there might be some cells on which the cells B and C depend.
* Dependency Set Generation may be recognized as an implementation of breadth-first search (BFS) algorithm.
* Following is the function to calculate the cells which are immediately dependent on a given cell.

Depsetgen(depset, m)

{

init\_size=depset.size();

for(k=m;k<init\_size-1;k++) // Timing : runs for n(size) times

{

for(j=0;j<sprdsheet\_size-1;j++)// Timing: runs for n(size) times

{

//Find all direct dependents of depset[k]

if(sprdsheet[j].formula contains depset[k].ref)

{

depset.add(sprdsheet[j].ref);

}

}

Depsetgen(depset, init\_size);

}

if(m==0)

{

depset.delete(depset[0]);

}

}

* The above algorithm takes O(n2) time to carry out the task.
* We now need to carry out the process of updating all the cells of a spreadsheet if one of the cells is updated.
* The process is an implementation of depth-first search (DFS) algorithm.
* The left-to-right, post-order tree transversal algorithm will be used to specify the order of evaluation of a cell’s descendents.
* Following is the code for the function of the same :

Input: original cell

Output : original cell with the updated value

depth=0; //initial value(global)

evalcell(cell)

{

for(i=0;i<cell.nchildren-1;i++) //Timing:depend on the number of children

{

depth++;

evalcell(cell.child[i]); // recursive call

}

evaluate();

if(depth==0)

{

cell.value = parse.(parser\_str);

}

depth--;

return cell;

}

* Following is the mechanism that has been used in order to carry out the process of evaluation of a cell.
* Note: This is the method which takes place when the function “evaluate()” is called.
* (I have attached a notepad file for the same which contains the function which carries out the task in an efficient way using stack method when a cell is updated.)