A Training Report

On

SPREADSHEET CLONE

Submitted in partial fulfilment of requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science & Engineering

Submitted By

ANKIT CHAUDHARY

(00551202718)

Under the guidance of

Prateek Narang

(Co-founder, Coding Blocks)

Department of Computer Science & Engineering

Bharati Vidyapeeth's College of Engineering

A-4, Paschim Vihar, New Delhi-110063

May-June, 2020

CERTIFICATE

CODING BLOCKS
Code Your Way To Success

Certificate No: CBCER-20-21/1009

Date: 6th Oct, 2020

INTERNSHIP COMPLETITION CERTIFICATE

To Whom It May Concern

This is to certify that **Mr. Ankit Chaudhary**, student of **Bharati Vidyapeeth's College of Engineering, New Delhi** having College Roll No as **00551202718**, has completed **Online** course in **Competitive Programming** as part of the summer training program. The duration of the training was 4 weeks. It started on **12th May, 2020** and ended on **12th June, 2020.**

During this training period, the student has covered all the topics and solved all the assignments in the course.

For Coding Blocks Pvt. Ltd.

Coding Blocks Private Limited New Delhi

Priyanshu Agrawal
Business Head
Mail id: internship@codingblocks.com
Contact No: 9999579111

Regd Office: Coding Blocks Pvt. Ltd (CIN: U74140DL2014PTC266843)
First Floor, 47 Nishant Kunj, Main Pitampura Road,
New Delhi – 110034, India
GSTIN: 07AAFCC6244C1ZZ

CANDIDATE's DECLARATION

I hereby declare that the work presented in this report entitled **Spreadsheet Clone**, in partial fulfilment of the requirement for the award of the degree **Bachelor of Technology** and submitted in **Department of Computer Science & Engineering, Bharati Vidyapeeth's College of Engineering**, **New Delhi** (**Affiliated to Guru Gobind Singh Indraprastha University**) is an authentic record of my own work carried out during the period from June – July 2019 under the guidance of **Prateek Narang, (Co-founder, Coding Blocks).**

The work reported in this has not been submitted by me for award of any other degree of this or any other institute.

Name: Ankit Chaudhary
Enrolment No: 00551202718

ACKNOWLEDGEMENT

I express my deep gratitude to **Prateek Narang**, (**Co-founder**, **Coding Blocks)**, for his/her valuable guidance and suggestion throughout my training. We are thankful to **Dr. Vedika Gupta,** (Computer Science Engineering, 3rd year, Evening Shift) for their valuable guidance.

Sign: Ankit Chaudhary

Enrolment No: 00551202718

ORGANIZATION INTRODUCTION

Coding Blocks was founded in 2014 with a mission to create skilled Software Engineers for our country and the world. They are here to bridge the gap between the quality of skills demanded by industry and the quality of skills imparted by conventional institutes. At Coding Blocks, they strive to increase student interest by providing hands on practical training on every concept taught in the classroom.

In total, they provide over 20 different courses across categories and methodologies to students. These courses comprise the most relevant subjects and technologies that are being used in the industry these days. They boast of having the most comprehensive topic coverage, and of constantly updating with the rapidly changing times. In addition to theoretical teaching, they also offer hands-on experience to all their students. Every course has hackathons included in the course structure itself, so the student can develop from himself, and do while he learns. All their courses are designed only on the basis of interest and proficiency. There is no age barrier for any course. They have had school students enrol for their advanced courses as well as senior industry professionals enrol for their foundation courses.

They have a total of four centres for classroom teaching that cover the length and breadth of Delhi-NCR and two more centres which help them conduct courses in Punjab. They also conduct regular online bootcamps and webinars to make learning convenient. As of summer of 2017, they had also launched their online courses. Their team of teachers, administrators and developers have a combined experience that cannot be paralleled by any competitor.

Their founder Mr. Manmohan Gupta has already succeeded with ventures of global repute such as Vidya Mandir Classes and Nagarro. They have team members who have worked at global giants like Facebook, Sony, Cyanogen, American Express, Barclays and many more. With such a large amount of national and international experience, they promise only the best for the students, and nothing else. They also boast of 1000s of success internship and placement stories for their students.

Table of Contents

PREFACE

This report is prepared to fulfil the requirements of the Bachelor of Technology Program of "Bharati Vidyapeeth's College Of Engineering on “Spreadsheet Clone". I have chosen this topic as knowledge of data strictures and algorithms related to competitive programming which is needed for building this project. Data structures and algorithms play a major role in implementing software and in the hiring process as well. Data Structures are a crucial part of several computer algorithms as they allow programmers to do data management efficiently. A wise selection of data structures can improve the performance of a computer program or algorithm in a more useful way. For example, segment trees are used for range queries, similarly, graphs are used for making social networks. In this project, graphs are the most important data structure, behind the implementation of spreadsheets.

In Chapter 1, I have introduced the what is spreadsheet, features and functioning in spreadsheets and applicatons of spreadsheets in real-life. In Chapter 2, I have discussed the implementation of the stack and graph data structure, which is the building block of this application. Along with these, graph traversal algorithms like Depth-First search and cycle detection in directed graphs is also discussed. This application works very well for automatic recalculation and real-time updates. In Chapter 3, I have discussed the working methodology of these algorithms and how they are used to calculate formulas efficiently, when cells are updated by the user. In Chapter 4, I have concluded the whole report

LIST OF FIGURES

WEEKLY PROGRESS

S.No	Week	Topic
1.	1st Week	Introduction to Data Structures, Recursion, Bit Masking and STL.
2.	2nd Week	Mathematics, Number Theory, Divide and Conquer
3.	3rd Week	Graphs, Dynamic Programming
4.	4th Week	Project

1. Introduction

1.1 What is spreadsheet software?

Spreadsheet software is a software application capable of organizing, storing and analyzing data in tabular form. The application can provide digital simulation of paper accounting worksheets. They can also have multiple interacting sheets with data represented in text, numeric or in graphic form. With these capabilities, spreadsheet software has replaced many paper-based systems, especially in the business world. Originally developed as an aid for accounting and bookkeeping tasks, spreadsheets are now widely used in other contexts where tabular lists can be used, modified and collaborated. Spreadsheet software is also known as a spreadsheet program or spreadsheet application.

Data in spreadsheet is represented by cells, organized as rows and columns and can be text or numeric. Features like conditional expressions, functions to operate on text and numbers are also available in spreadsheets. Calculations can be automated, and spreadsheets are generally easier to use than other data processing applications. However, limitations of spreadsheet software include difficulty in identifying data errors, restriction of finite number of records, inefficient in handling large amounts of text, inability to scale for access and manipulating large data volumes, inability to create reports as in case of databases, high data storage requirement and unavailability of certain querying and sorting techniques.

1.2 Features of spreadsheets:

Some of the features of spreadsheet software which makes user experience easier are as follows -

a. Calculation and functionalities are easier to represent in spreadsheets than in word processors, and thus effective data handling is possible.

b. This software is capable of interacting with databases, can populate fields and can also help in automation of data creation and modification.

c. Spreadsheet software can be shared both online and offline and allows for easy collaboration.

d. Real time Update - This feature refers to updating a cell's contents periodically with a value from an external source-such as a cell in a "remote" spreadsheet. For shared, Web-based spreadsheets, it applies to "immediately" updating cells another user has updated. All dependent cells must be updated also.

e. Automatic Recalculation - this optional feature eliminates the need to manually request the spreadsheet program to recalculate values (nowadays typically the default option unless specifically 'switched off' for large spreadsheets, usually to improve performance). Some earlier spreadsheets required a manual request to recalculate since the recalculation of large or complex spreadsheets often reduced data entry speed.

f. Cell Formatting – It includes the attributes of either the content- point size, color, bold or italic or the cell, that is, border thickness, background shading, color. To aid the readability of a spreadsheet, cell formatting may be conditionally applied to data.

1.3 Applications of spreadsheets:

Although spreadsheets are most often used with anything containing numbers, the uses of a spreadsheet are almost endless. Below are some other popular uses of spreadsheets.

a. Forms - Form templates can be created to handle inventory, evaluations, performance reviews, quizzes, time sheets, patient information, and surveys.

b. School and grades - Teachers can use spreadsheets to track students, calculate grades, and identify relevant data, such as high and low scores, missing tests, and students who are struggling.

c. Lists - Managing a list in a spreadsheet is a great example of data that does not contain numbers, but still can be used in a spreadsheet. Great examples of spreadsheet lists include telephone, to-do, and grocery lists.

d. Sports - Spreadsheets can keep track of your favourite player stats or stats on the whole team. With the collected data, you can also find averages, high scores, and statistical data. Spreadsheets can even be used to create tournament brackets.

e. Finance - Spreadsheets are ideal for financial data, such as your checking account information, budgets, taxes, transactions, billing, invoices, receipts, forecasts, and any payment system.

2. Work Carried Out

2.1 Objective:

To develop a spreadsheet software by implementing their functionality using graph and other data structure. Use graph algorithms efficiently for automatic recalculation when numeric values or formulas get updated.

The main purpose is to develop a spreadsheet software which has the function of real time update and recalculating the cell values, when any of the cell value or formula is modified. This concept uses graph algorithms like Depth First Search to traverse the graph and Cycle detection in directed graph to validate formulas.

2.2 Technology Stack Used:

The spreadsheet application is a desktop application which is made using the electron framework of Javascript as well as other web development languages- HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and Javascript (JS).

a. HTML – It is the standard markup language for documents designed to be displayed in a web browser.

b. CSS - Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

c. JS - JavaScript is high-level, often just-in-time compiled, and multi-paradigm language. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

d. Electron - Electron is an open-source software framework developed and maintained by GitHub. It allows for the development of desktop GUI applications using web technologies: it combines the Chromium rendering engine and the Node.js runtime. Electron is the main GUI framework behind several notable open-source projects including Atom, GitHub Desktop, Light Table, Visual Studio Code, and WordPress Desktop.

2.3 User Interface:

The spreadsheet application developed consists of a grid of cells, with 26 rows (A-Z) and columns (1-100). It consists of a cell address and formula bar. On top, it consists of a menu bar, which has three main file operations – Open a New file, Open an existing file and Save the current file. The application also has a formatting bar which contains options for altering color of text, cell, and font-family. Text format can be changed to bold, italics, and Underline. Figure 1. shows the user interface of the spreadsheet application.

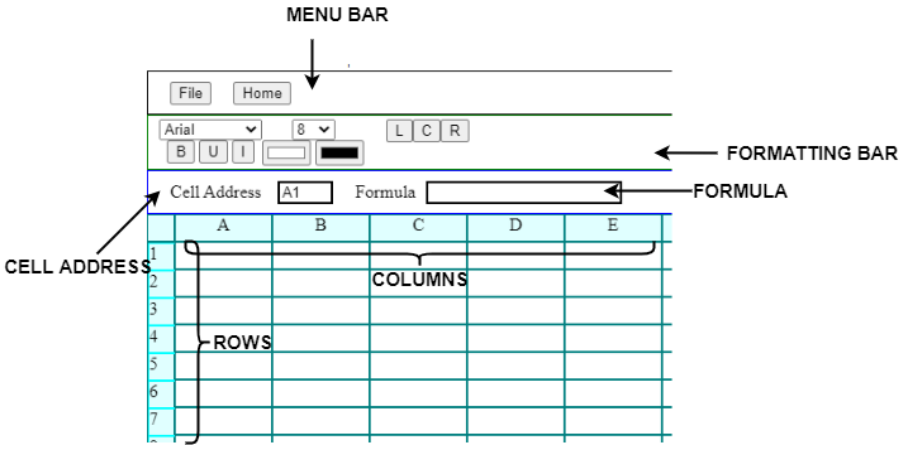


Figure 1. User Interface of the application

2.4 Data Structure used:

The grid cells in a spreadsheet contain either formulas or raw data, called values, in the cells. Values are general numbers, but can also be pure text, dates, months, etc. Formulas say how to mechanically compute new values from existing values. These formulas (infix expressions) in the cells are evaluated using stack data structure. They create dependencies between cells. These cells get interlinked and depend on each other's value. These relationships and dependencies are solved using graph data structure.

Following sections explains the graph and stack data structures, which is used in the spreadsheet application.

2.4.1 Stack Data structure:

Stack is a linear data structure (abstract data type) which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out). Here, the element which is placed (inserted or added) last, is accessed first. Basic operations performed in the stack are Push, Pop and Peek. Push denotes adding an item in the stack. If the stack is full, then it is said to be an Overflow condition while Pop denotes removing an item from the stack. The items are popped in the reversed order in which they are pushed. Peek means to access the top element of stack. Push and Pop operations are shown in figure 2.

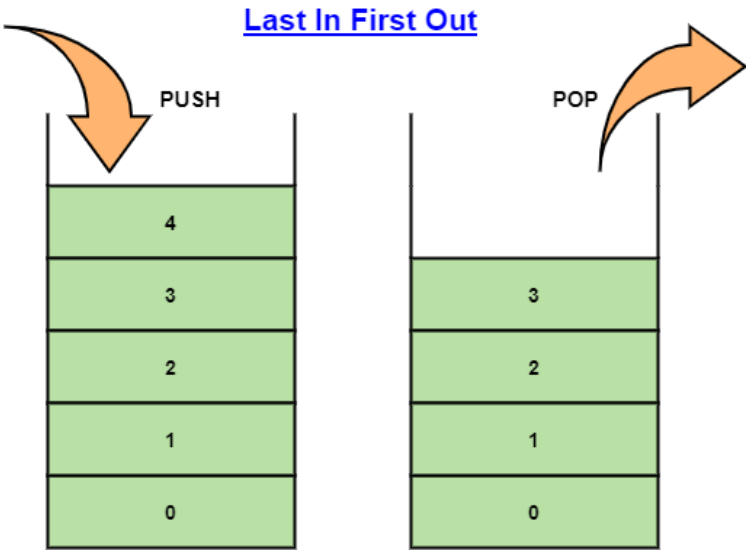


Figure 2. Push and pop operations on stack data structure

In this project, stacks are used to evaluate formulas in the cells. Formulas are infix expressions entered by the user to perform easier calculations and Depth First Search Algorithm.

2.4.2 Graph Data structure:

A graph (non-linear data structure) is a collection of nodes that have data and are connected to other nodes. It consists of a collection of vertices or nodes, V and a collection of edges E, represented as ordered pairs of vertices (u,v).

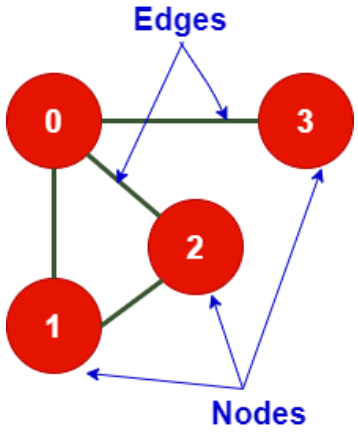


Figure 3. Representation of graph data structure

In the graph shown in figure 3,

V = {0, 1, 2, 3},

E = {(0,1), (0,2), (0,3), (1,2)},

G = {V, E}.

2.4.2.1 Graph Representation:

Graphs are commonly represented in two ways:

A. *Adjacency Matrix*

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w. The adjacency matrix for the above example graph is shown in figure 4.

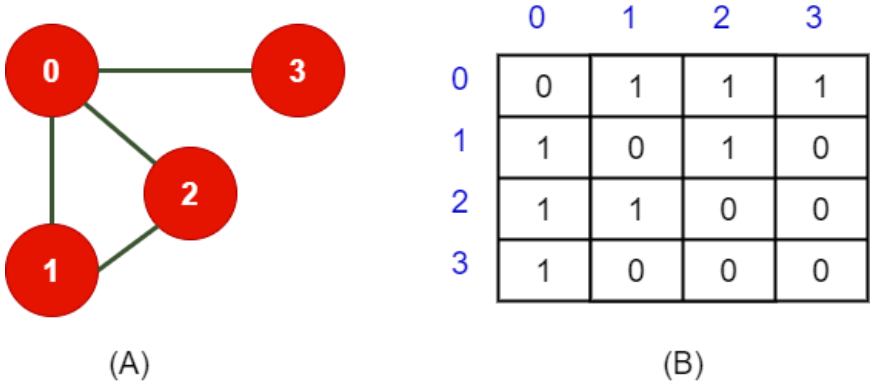


Figure 4. (A) denotes the graph and (B) denotes its adjacency matrix representation

Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done O(1). But it consumes more space, that is, O(V^2). Even if the graph is sparse (contains less number of edges), it consumes the same space. Adding a vertex is O(V^2) time.

B. *Adjacency List*

An array of lists is used. The size of the array is equal to the number of vertices. Let the array be an array[]. The index of the array, array[i], represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex. This representation can also be used to represent a weighted graph. The weights of edges can be represented as lists of pairs. Figure 5 is the adjacency list representation of the above graph.

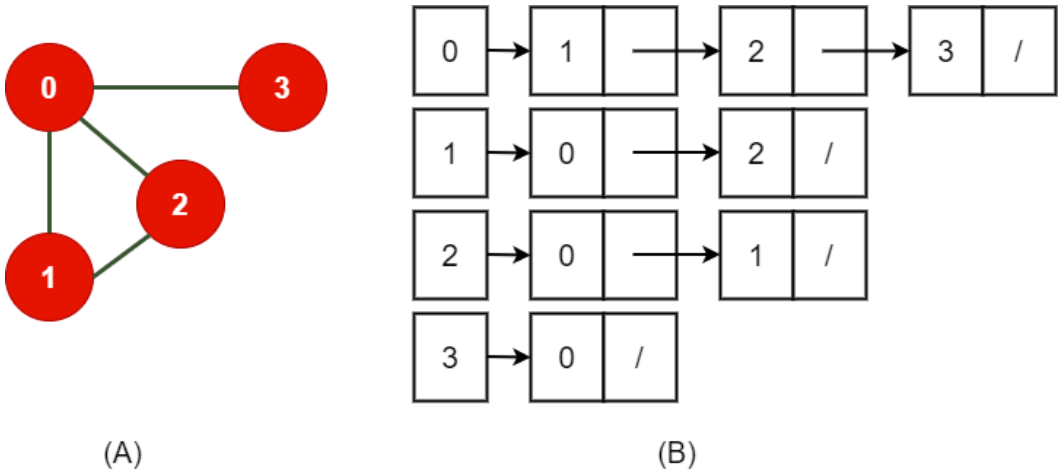


Figure 5. (A) denotes the graph and (B) denotes its adjacency list representation

An adjacency list is efficient in terms of storage because we only need to store the values for the edges. For a graph with millions of vertices, this can mean a lot of saved space.

Some of the best use cases for Graph Data Structures are in; Social Graph APIs such as Facebook's Graph API, Recommendation Engines such as Yelp's GraphQL API, Path Optimization Algorithms such as Google Maps Platform (Maps, Routes APIs) and Car Navigations, Web Analytics and Scientific Computations.

2.5 Representation of spreadsheet:

The spreadsheet is represented as a two-dimensional matrix at the user interface, which is a grid of cells, values, or formulas in the cells. A cell is referenced by its column and row index. Rows are referenced in decimal notation starting from 1, while columns are represented using letters A-Z. These column indices (A-Z) are referenced through American Standard Code for Information Interchange (ASCII) codes of English alphabets, that is, starting from A=65 up to Z=90. The ASCII codes are transformed to a scale of 0-25 so that column indices start from 0 instead of 65. For instance, cell B5 refers to the cell at row 5 and column 2, as ASCII code of B is 66, and it is transformed to a scale from 0-25. Therefore if mat[][] represents the spreadsheet, then B5 will be referenced as mat[5][2] instead of mat[5][66].

On the other hand, it is often convenient to think of a spreadsheet as a mathematical graph, where the nodes are spreadsheet cells, and the edges are references to other cells specified in formulas. This is often called the dependency graph of the spreadsheet. A dependency graph is a directed graph representing dependencies of several nodes towards each other. It is a graph that has a vertex for each object to be updated, and an edge connecting two objects whenever one of them needs to be updated earlier than the other. Dependency graphs without circular dependencies form directed acyclic graphs, representations of partial orderings (in this case, across a spreadsheet) that can be relied upon to give a definite result.

2.5.1 Directed Acyclic graphs

A Directed Acyclic Graph (DAG) is a graph that flows in one direction, where no element can be a child of itself. It is a graph that is directed and without cycles connecting the other edges. This means that it is impossible to traverse the entire graph starting at one edge. The edges of the directed graph only go one way. The graph is a topological sorting, where each node is in a certain order. In any directed acyclic graph, we define a vertex v to be a source, if there are no edges leading into v, and a sink if there are no edges leading out of v. Every finite DAG has at least one source, and at least one sink. In fact, given any vertex v, there is a path from some source to v, and a path from v to some sink. An example of directed acyclic graph is shown

in figure 6. Here, source node is A and sink nodes are E, F, and G. All the edges are directed and do not form a cycle.

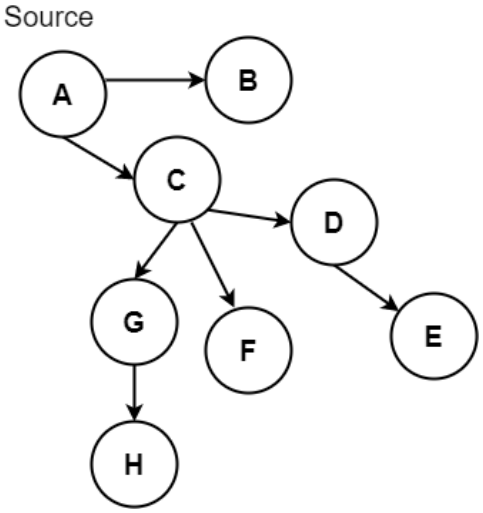


Figure 6. Directed Acyclic Graph

DAG's are used to represent one-way dependency between two nodes. Also, it overcomes the problem of circular dependencies. Cell values are dependent on each other, when formulas are used. In such a case, there should not be a cycle present in the graph. If there is a cycle then there will be an infinite loop over the graph which will never halts as shown in figure 7. Hence, DAG's are best suitable data structure for spreadsheets.

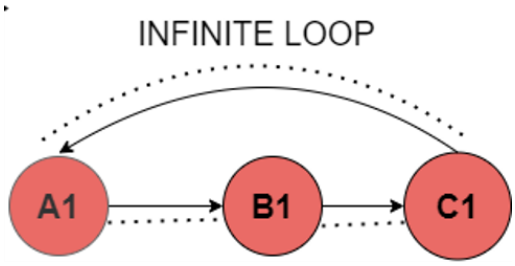
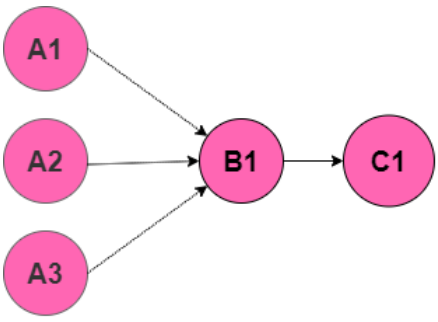


Figure 7. Problem of circular dependencies in spreadsheets

Now, to represent directed acyclic graphs as spreadsheets, graphs are represented as Adjacency list. Although, it seems to be an adjacency matrix to the user, but it uses list to store its parents and its children. Along with these, it also stores values and formula inserted in the cell. Therefore, a single node of the graph will contain four types of information –

1. its cell value – any numeric value present in the cell,
2. its children (as downstream) – in which this cell is present as formula,
3. its parents (as upstream), - on which this cell is dependent, and
4. formula for the cell.

Let's say, B1 = (A1+A2+A3)/3, and C1 = B1*100. Clearly, B1 is dependent on the values of A1, A2 and A3, and C1 depends on the values of B1. If we change any value of A1 or A2, it will directly impact the value of B1 and C1. This shows that B1 and C1 has dependency from A1, A2 and A3. A graphical representation of directed acyclic graph of above example and its adjacency matrix is shown in figure 8.



	A1	A2	A3	B1	C1
Value	1	2	3	2	200
Upstream	-	-	-	A1, A2, A3	B1
Downstream	B1	B1	B1	C1	-
Formula	-	-	-	(A1+A2+A3)/3	2*B1

Implementation of Graph as Adjacency List

Figure 8. Graph representation of a spreadsheets

2.6 Algorithms:

Since spreadsheets are represented as directed acyclic graphs, Graph algorithms like Depth-First Search and Cycle Detection in Directed Graphs can be used for automatically calculating values and real time updating of cells when values and formulas are altered.

2.6.1 Depth First Search (DFS) Algorithm:

Cells of the spreadsheets represented by nodes of the graphs are need to be updated, when cell values and formulas are altered. To update its dependent node values in the dependency graph, Graph traversal algorithm – Depth First Search is used.

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. So, the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally print the nodes in the path. Its Space and Time Complexity is O(n).

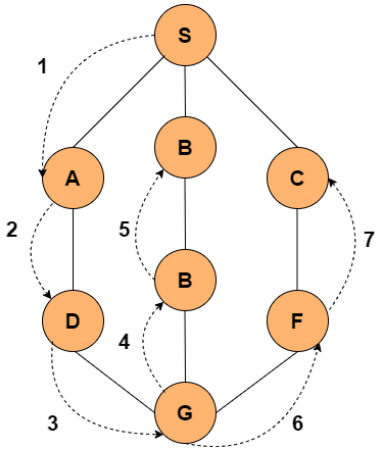


Figure 9. Visualization of depth first search algorithm

Figure 9, shows basic DFS of a graph. It traverses from S to A to D to G to E to B first, then to F and lastly to C. It employs the following rules.

1. Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
2. If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
3. Repeat Rule 1 and Rule 2 until the stack is empty.

DFS Algorithm:

```
1.  DFS(G, u)
2.     u.visited = true
3.     for each v ∈ G.Adj[u]
4.           if v.visited == false
5.              DFS(G,v)
6.           end if
7.     end for
8.  init() {
9.     for each u ∈ G
10.          u.visited = false
11.       end for
12.    for each u ∈ G
13.       DFS(G, u)
14.       end for
15. }
```

2.7 Functions Used:

Some of the functions used are:

I. UpdateCell

It is used for updating cells whenever a new value is entered and the cell is part of any graph. It is based on Depth-First Search Algorithm and recursively updates all the children of the node in the graph.

```
function updateCell(rowId, colId, nVal) {
    let cellObject = db[rowId][colId];                    // Get Adjcency list
    cellObject.value = nVal;
    // update User Interface
    $(`#grid .cell[r-id=${rowId}][c-id=${colId}]`).html(nVal);
    for (let i = 0; i < cellObject.downstream.length; i++) {
       let dsObj = cellObject.downstream[i];
       let ds_formula = db[dsocordObj.rowId][dsocordObj.colId];
       let dsonVal = evaluate(ds_formula);          // Function for evaluating formula.
       updateCell(dsObj.rowId, dsObj.colId, dsonVal);    // DFS call
    }
}
```

II. SetFormula

When a new formula is updated, SetFormula function is called. This function add new upstream in the child nodes and downstreams to the *parent node*.

```
function SetFormula(cellElement, formula) {
    formula = formula.replace("(", "").replace(")", "");
    let formulaComponent = formula.split(" ");
    for (let i = 0; i < formulaComponent.length; i++) {
      let charAt0 = formulaComponent[i].charCodeAt(0);
      if (charAt0 > 64 && charAt0 < 91) {
        let { r, c } = getParentRowCol(formulaComponent[i], charAt0);
        let parentCell = db[r][c];
        let { colId, rowId } = getrc(cellElement);        // Get rowID and ColumnID
        let cell = getcell(cellElement);                  // Get Cell Data
        parentCell.downstream.push({
          colId: colId, rowId: rowId
        })
        cell.upstream.push({
          colId: c, rowId: r
        })
      }
    }
  }
```

III. RemoveFormula

When a formula is deleted, that is, a new formula is created or a new cell value is added, previous formula is to be removed. Upstream and downstream related to that formula will be removed.

```
function RemoveFormula(cellObject, cellElem) {
    cellObject.formula = "";
    let { rowId, colId } = getrc(cellElem);
    for (let i = 0; i < cellObject.upstream.length; i++) {
      let uso = cellObject.upstream[i];
      let fuso = db[uso.rowId][uso.colId];
      let fArr = fuso.downstream.filter(function (dCell) {
        return !(dCell.colId == colId && dCell.rowId == rowId);
      })
      fuso.downstream = fArr;
    }
    cellObject.upstream = [];
  }
```

3. Working of the project

This section shows how the working of the algorithms and the ouputs generated in the spreadsheets when these cells are updated. Section 3.1 describes the flow how dircted acyclic graphs changed when a cell value or formula is replaced with numeric value and Section 3.2 presents the working of graphs and spreadsheets when formula is changed when either value or formula already exists in the cell. Following section represents the different cases when automatic cell calculation is done.

3.1. When value of a cell is updated

When value of a cell is changed, first of all, the algorithm will check whether there is a value present or a formula. If there is value present, then all its children need to be updated according to the new value. Therefore, DFS algorithm needs to be run from the updated node and end till all its children gets updated. If there is formula present in the cell, then all its edges from its parents needs to be removed and then updating of the nodes takes place using DFS algorithm. For example, in the given figure 10, given spreadsheet has cells with values (A1=10, A2=20, A3=30, A4=10), and B1, B2 and C1 has value which is derived from the formulas, B1=A1+A2, B=A3+A4, and C1=B1+B2 respectively. Now if value of cell A2 is changed to 50, then all its children (B1 and C1) should be updated as per A2. B1 value gets changed to 60 (as A1+A2 = 60) and in the same way C1 value gets changed to 130 (as B1+B2=130). Figure 10 represents running of DFS algorithm when value of A2 (20) is replaced with by 50.

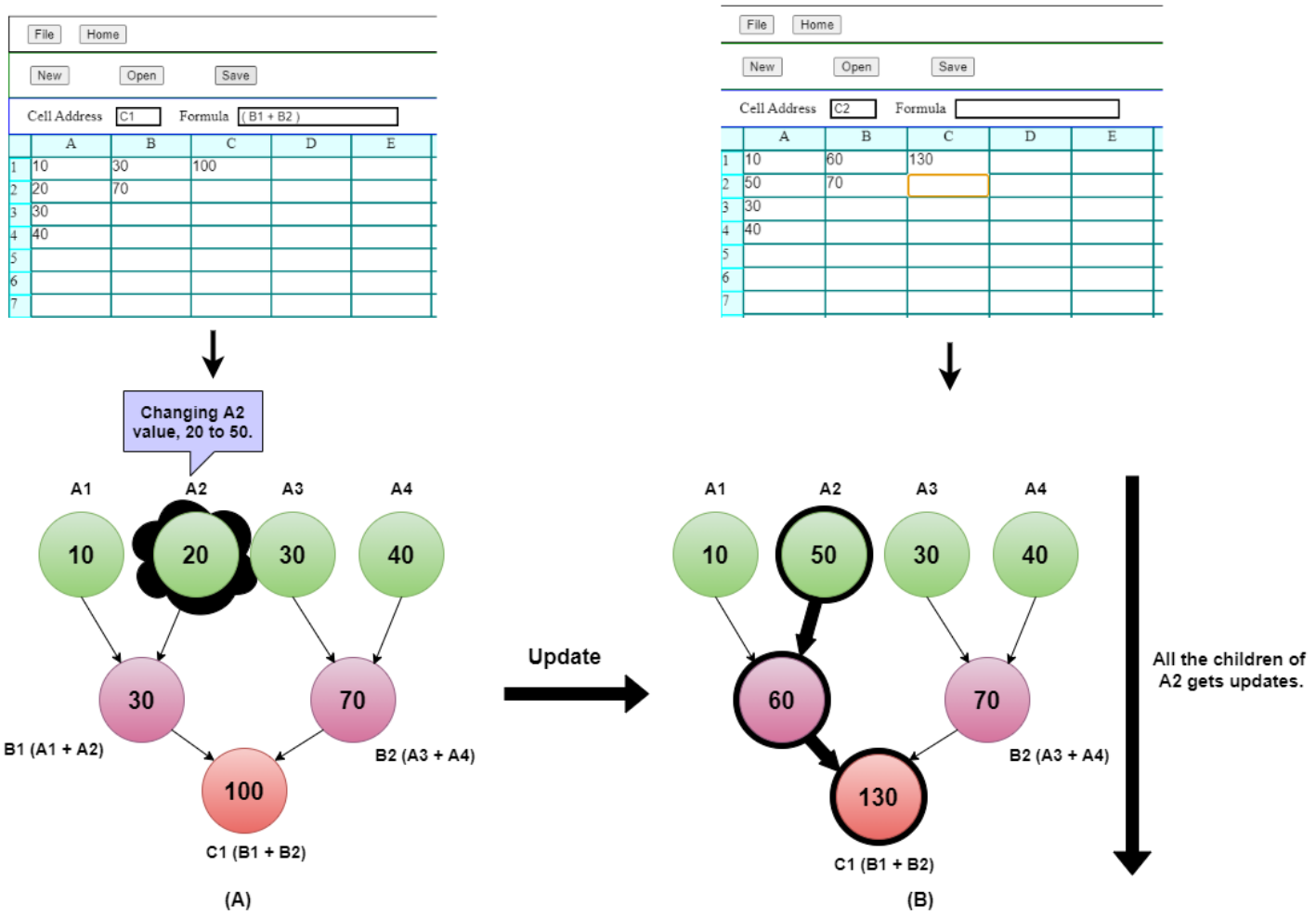
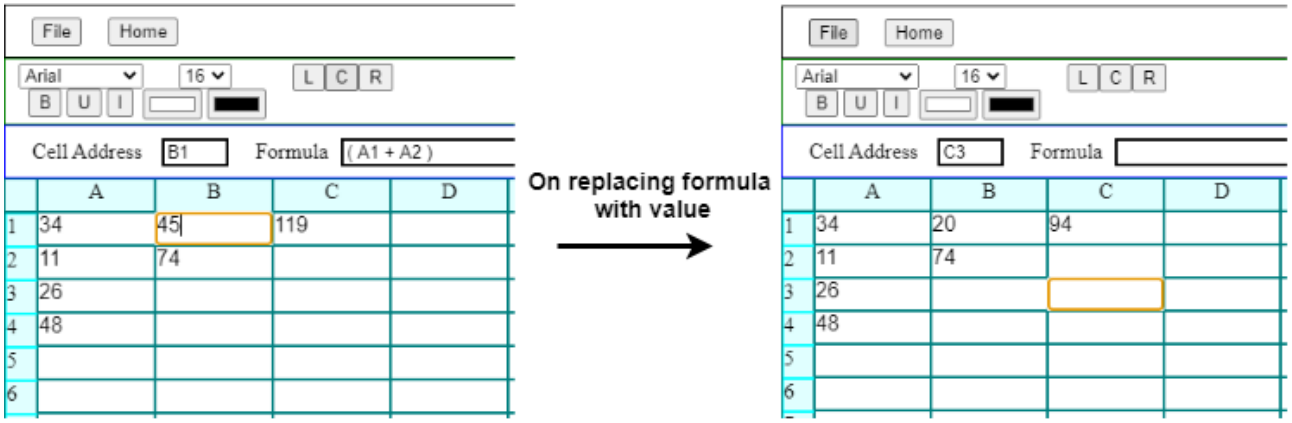
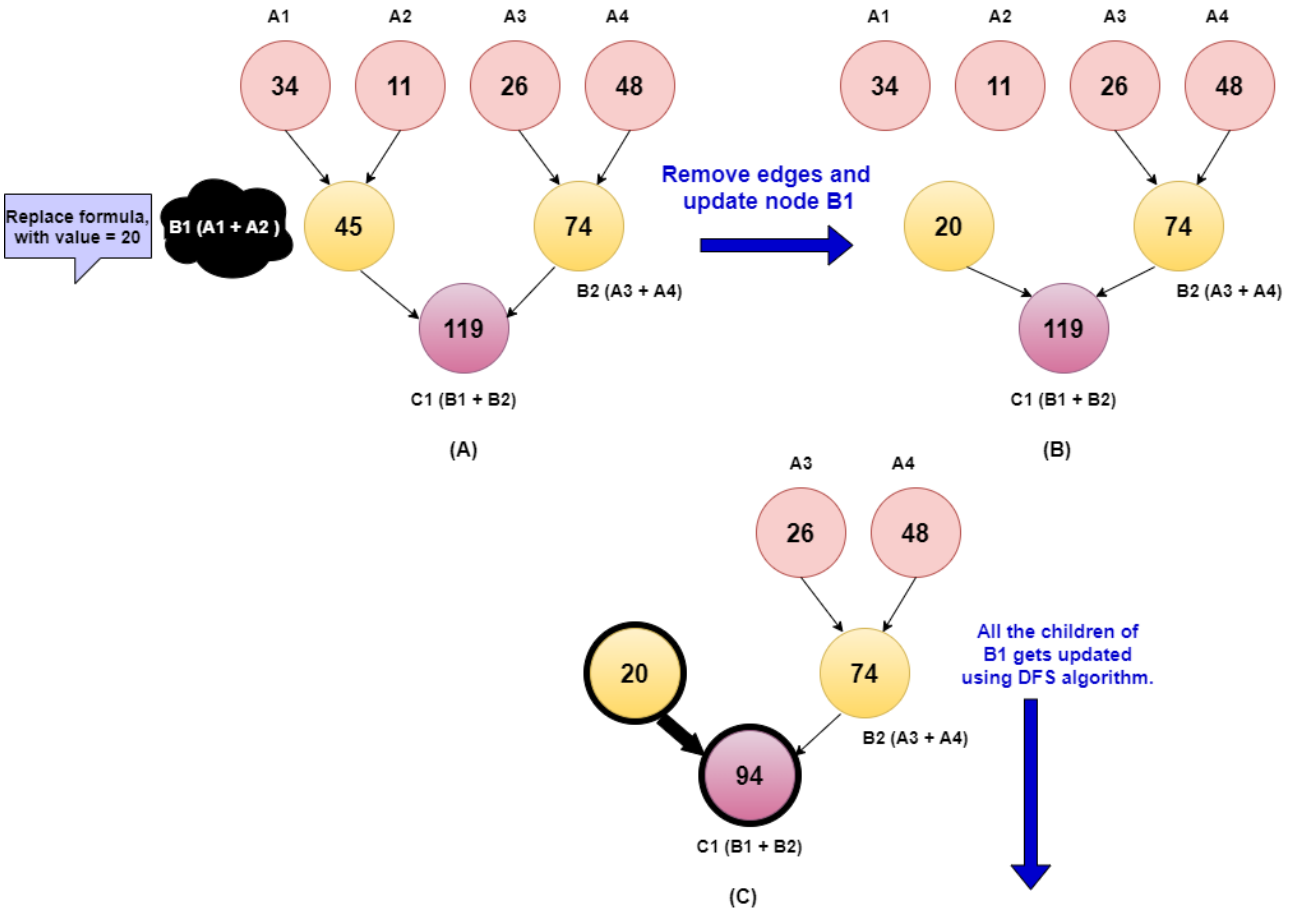


Figure 10. When cell value is replaced with value (A) Before update and (B) After Update

Similarly, when formula is deleted and a numeric value is added in place of the formula, First of all the edges according to the formula gets deleted. For the example shown in figure 11 (A-B), edges B1-A1 and B1-A2 are deleted first, that means value of A1 and A2 is deleted from the upstream of B1 and node B1 is deleted from downstream of A1 and A2. After this node is updated and DFS starts running from node B1 and its child C1 is updated as per its formula.



(I)



(II)

Figure 11. When cell formula is replaced with a value (I) Spreadsheet screenshot and (II) Graph visualization

3.2. When formula of a cell is updated

When a new formula is added in place of a numeric value or a formula, first of all, edges which are related to the previous formula are removed using RemoveFormula() function and then in the next step, cycle detection algorithm is used to check the whether there is cycle present or not in the new formula. If the cycle is present in the formula, then, the algorithm will not proceed and no updation will take place. On the other hand, if cycles are not present, then new upstream (parents) and downstream (childen) will be assigned as per the new formula. Then, the cell at which formula is updated, DFS algorithm will run and update all the cells.
For example, in the given figure 12, given spreadsheet has cells with values (A1=10, A2=20, A3=30, A4=10), and B1, and C1 has value which is derived from the formulas, B1=A1+A2, and C1=2*B1 respectively. Graphical representation of the spreadsheet and screenshot is shown in figure 12. Now if formula of cell B1 is changed toA3+A4, then firstly, edges A1-B1 and A2-B1 will be removed. That means B1 is deleted from downstream of A1 and A2 and similarly, A1 and A2 is deleted from upstream of B1. Now new formula is set using SetFormula() function and new upstream and downstream is allocated. B1 value gets changed to 70 (as A3+A4 = 70) and in the same way C1 value gets changed to 140 (as 2*B1=130).

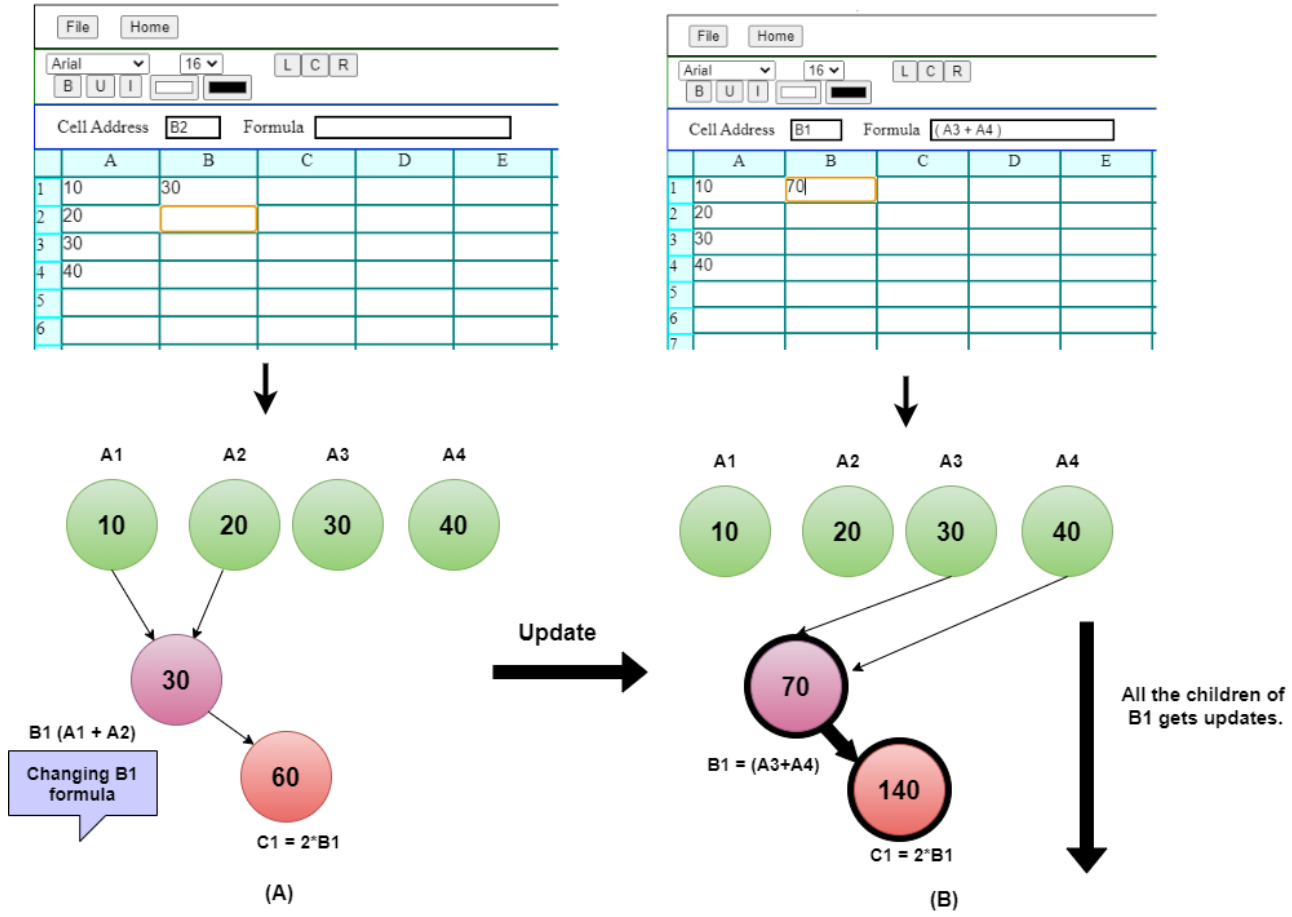


Figure 12. When formula value is replaced with another formula (A) Before update and (B) After Update

In the second case, when formula is added, when no existing formula is associated with the cell, firstly, cycle detection is done to prevent circular dependencies and then new upstream and downstream is added to the child and parent respectively. When value is updated according to the formula, DFS algorithm will run and update all its children. Graphical representation and spreadsheet screenshot is shown in figure 13.

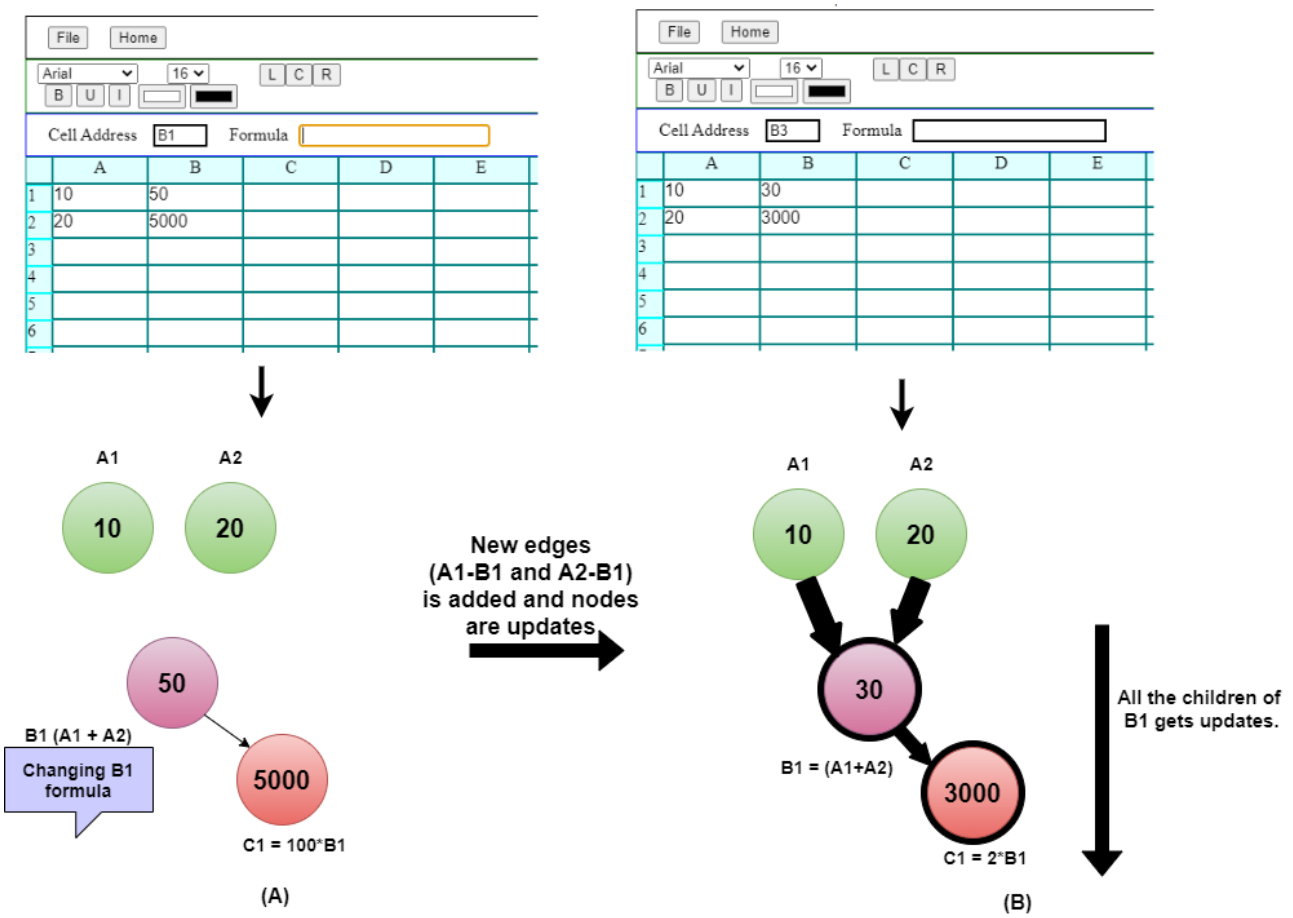


Figure 13. When formula value is replaced with numeric value (A) Before update and (B) After Update

4. Conclusion

I have successfully implemented the graph as a data structure in spreadsheet application. This project focused on the automatic recalculation and real time update of cell values, when any formula or value is modified. Directed acyclic graphs and adjacency list representation of graphs are the best suited for this project implementation. Graph algorithms make it easier to handle such calculations and implement spreadsheets as data structure. This project can be used for doing calculations, making spreadsheets, marksheets and more real-life applications.

In future it can be extended and enhanced in the following directions:

1. Its time and space consumption can be improved by using more efficient graph algorithms.
2. It can further be developed to work with dates, and textual data.
3. More enhanced user interface will be developed.
4. Authentiaction and database accesscan also be added.

5. References

1. https://en.wikipedia.org/wiki/Spreadsheet
2. https://en.wikipedia.org/wiki/Directed_acyclic_graph
3. https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
4. https://www.geeksforgeeks.org/detect-cycle-in-a-graph/
5. https://www.geeksforgeeks.org/expression-evaluation
6. https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
7. https://www.geeksforgeeks.org/stack-data-structure/

6. Appendix

https://github.com/Ankitchaudharyy/Spreadsheet-Clone