CALIFORNIA STATE UNIVERSITY, SAN BERNARDINO

Department of Computer Science
And Engineering
CSE 455

AlgorithmA 2010

Software Requirement Specification (SRS)
Second Iteration

CSE 455, Inc.
CEO: Dr. Concepcion
Project Manager: Patrick O'Connor
Assistant Managers: Danny Vargas and Abdel Kamel
Documentation Team: Erick Behr, Tyler Cannon,
Charles Korma, Kathleen Daugherty

Published: March 03, 2010

Contents

1. Introduction

The CEO of CS455 Inc. has requested to increase the number of animations for the second iteration to bring a total of 20 animations. The following document is a list of the proposed solutions for the CEO's expectations.   Upon the approval from the CEO of this document, development will commence immediately.

1.1 Purpose

The main goal of AlgorithmA 2010 is to continue to provide an interface to the academic community, in particular new students interested in the field of Computer Science and Engineering, to explore and learn about various algorithms and data structures. End users will have the ability to view a step-by-step animation for each of the algorithms and data structures that illustrate how they operate. AlgorithmA 2010 will continue to build these new algorithm animations with the same quality and didactical resolution, providing graphics where appropriate and refining the already existing explanations to better suit the end user, AlgorithmA 2010's second iteration will address the following requests:

- Create new seven animations
- Completion of end user context-sensitive help for each algorithm
- Completion of the application user's manual
- Completion of the source code technical documentation

1.2 Scope

The main goal of AlgorithmA is to provide step-by-step visualization of the various types of algorithms and data structures already implemented in the preceding AlgorithmA versions.

AlgorithmA 2010 will continue to move forward with the progress of the earlier iteration by utilizing JavaScript language and JavaScript libraries widely available on the Internet. Due to the flexibility and portability JavaScript is easier for the software engineer to use and reduces the system requirements needed to run the applications.

The following algorithms will be re-engineered:

Trees
    -AVL Trees
    -2-3-4 Trees
    -Red-black Trees
Graphs
    -Djsktra Shortest path
    -Critical path analysis
    -Kruskal's MST [minimum spanning tree]
    -Prim's MST

1.3 Definitions and Acronyms

AlgorithmA

PattE's sister project, which stands for "Algorithm Animation."

Animation

A visual display that depicts selected algorithms being studied. The display is based on the Cartesian x-y coordinate system.

Computer Science

Study of information and computation.

CS

Acronym for Computer Science.

Deployment Diagram

Deployment diagrams serve to model the hardware used in system implementations and the associations between those components.

Fault

An abnormal condition or defect at the component, equipment, or subsystem level, which may lead to failure. It is informally linked with "bug."

Graphical User Interface

A GUI is a method of interacting with a computer through a metaphor of direct manipulation of graphical images and widgets in addition to text.

HTML

Acronym for Hypertext Markup Language, the authoring language used to create documents on the World Wide Web.

HTTP

Acronym for Hypertext Transfer Protocol. It is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web Servers and browsers should take in response to various commands.

IDE

Acronym for Integrated Development Environment. IDEs assist computer programmers in developing software.

Interface

The communication boundary between two entities such as software and its users.

Iteration

The repetition of a process.

Java

A high-level programming language developed by Sun Microsystems.

Model-View-Controller

A software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modification to one component can be made with minimal impact to the others.

Mozilla Firefox

A free, cross-platform, graphical web browser that complies with many of today's standards on the World Wide Web. The most important standards for AlgorithmA 2010 will be the W3C web standards.

Open Source Software

Software whose source code is published and made available to the public, enabling anyone to copy, modify and redistribute the source code without paying royalties or fees. Open source code evolves through community cooperation.

PHP

Self-referential acronym for PHP: Hypertext Preprocessor, an open source, server-side, HTML embedded scripting language used to create dynamic Web pages. In an HTML document, PHP script (similar syntax to that of Perl or C) is enclosed within special PHP tags.

Software Requirements Specification

An SRS is used to describe all the tasks that go into the instigation, scoping, and definition of a new or altered computer system.

SRS

An acronym for Software Requirements Specification.

W3C

An acronym for the World Wide Web Consortium.

World Wide Web Consortium

An international organization that works to define standards for the World Wide Web.

1.4 References

Wikpedia, http://en.wikipedia.org/

Bruegge, Bernd, and Allen H. Dutoit. Object-Oriented Software Engineering. 3rd ed. New Jersey: Pearson Prentice Hall, 2010.

W3schools, http://www.w3schools.com/jsref/default.asp

"AlgorithmA 2008 SRS Prototype 2." CS455 Inc. Management Team. February 4, 2008.

Fowler, Martin. UML Distilled Third Edition. A Brief Guide to the Standard Object Modeling Language. Boston: Pearson Education, Inc. 2009.

IEEE SRS Std. 830-1998.

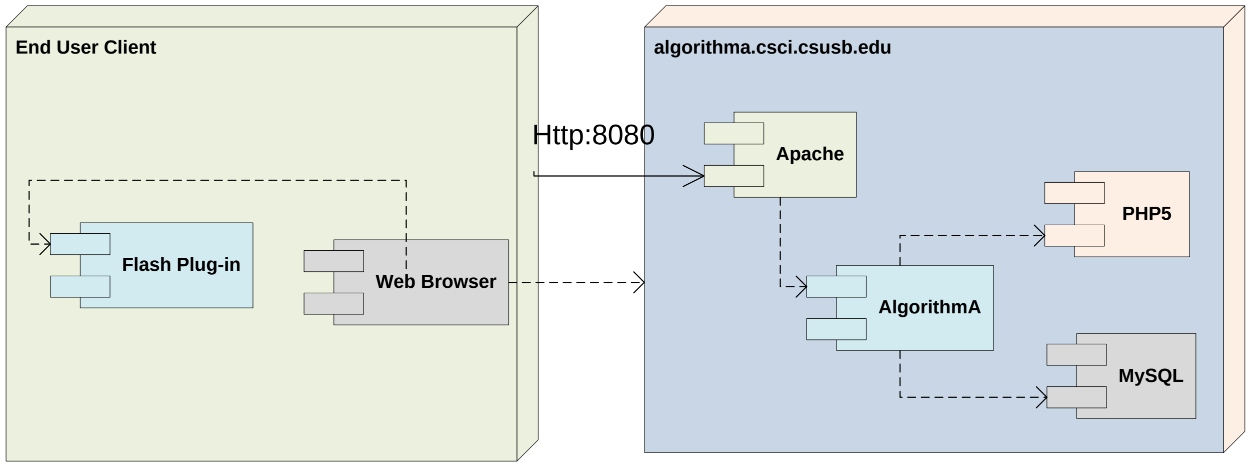
http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel4%2F5841%2F15571%2F00720574.pdf&authDecision=-203

2. OVERALL DESCRIPTION

2.1 Product Description

2.1.1 System Interfaces

The deployment diagram pictured below interprets the system interactions for iteration #2 of AlgorithmA 2010, which will be continue in use during iteration # 2.



Deployment Server Diagram

When the iteration # 2 reaches its completion, the Subversion repository will be exported to a production server. The project will be given a professional web address that users can access through HTTPS. Other browsers will continue to be supported. The development server will now only contain the latest milestone release. A milestone release will be considered a completed iteration. We expect to give our final presentation from this server.

2.1.2 User Interfaces

AlgorithmA 2010 will continue to consist of four basic user interfaces. The first interface will consist of the website itself and will serve to provide the navigation between all of the other interfaces in AlgorithmA 2010. The second interface will be an animation interface, which will provide a graphical animation of an algorithm. The third interface will be a walkthrough interface, which will consist of two panes, one for the animation of an algorithm, and the other for a walkthrough of pseudo code that corresponds to the animation. The fourth interface will provide specific context-sensitive help for hovering and operating each animation. In addition, a structured non-technical user's manual will be provided within AlgorithmA 2010 explaining mathematical concepts for each algorithm.

2.1.2.1 Website Interface

The primary interface will consist of the website itself and will serve to provide the navigation between all of the other interfaces in AlgorithmA 2010.
In iteration 2, the website interface will change, reflected in the following ways:

- Redesigning of the splash screen.
- Incorporation of AlmA, AlgorithmA mascot.
- The rest of website interface will remain the same as it was incorporated in iteration 1.

2.1.2.2 Animation Interface

The animation interface will remain the same as in the previous iteration. Our primary focus continues to is to maintain an educational overview of how a specific algorithm by making the process as simple as possible.

2.1.2.3 Walk-Through Interface

We shall also be including a walk-through interface that was founded back in 2008 AlgorithmA . We will carry this interface through to our reconstruction in JavaScript by making the code found within to be easier for those without a programming background can observe and understand easily.

2.1.2.4 General Information Interface

Along with the animation in this iteration will include two pages in the form of Help and About which will provide general information about the development team and aspects about AlgorithmA 2010.

2.1.3 Software Interfaces

AlgorithmA 2010 requires a web browser to be viewed. Any web browser may be used that follows the W3C web standards. For most of the web site, no other implements will be required. Two browsers are supported, Microsoft Internet Explorer and Mozilla Firefox.

2.1.4 Communication Interfaces

AlgorithmA 2010 will be implemented using JavaScript and AJAX. All client-side communication with the application shall use HTTP from the client's internet browser. Server-side communication will be controlled by the web-server. AJAX ran components will receive information directly from the server either using PHP or MySQL to generate content on screen in real time.

2.1.5 Memory Constraints

AlgorithmA 2010 will require 128 MB of RAM to be viewed.

2.1.6 Operations

AlgorithmA 2010 will be maintained during the winter quarter (January to March of 2010) and operated 24 hours a day, 7 days a week throughout the year. The maintenance will be conducted by CS455, Inc., and the hosting will be conducted by the Network Administration Group at California State University, San Bernardino.

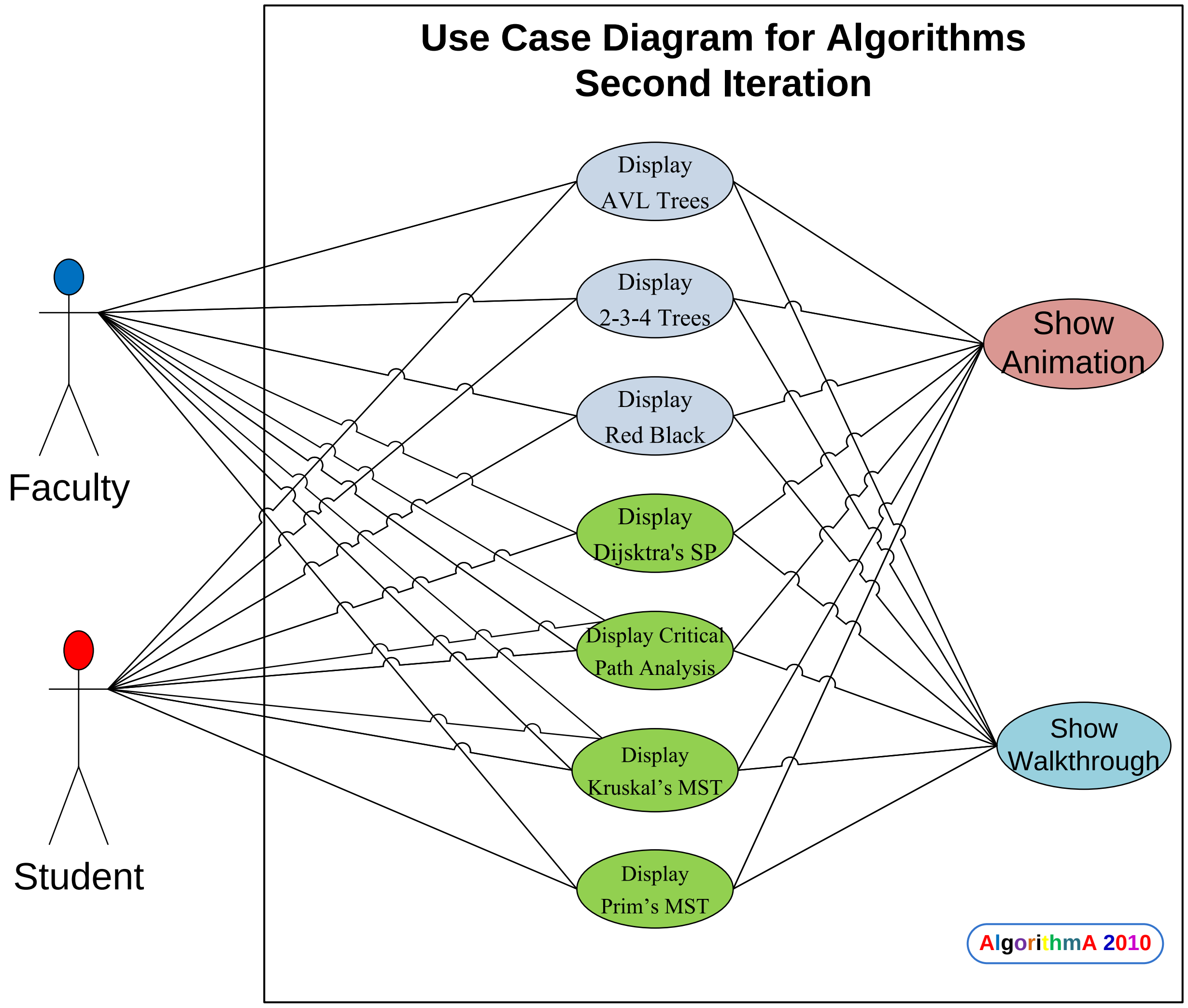
2.1.7 Site Adaptation Requirements

The on-campus workstations meet the current requirements of the JavaScript, and Java components. Home workstations will need to ensure that the requirements in section 2.1.5 are met.

2.2 Product Functions

The Use Case Diagram presented below illustrates which functions an end-user may choose to do in AlgorithmA 2010 for the second iteration. Walkthroughs and animations will be ported into iteration 2, which include the additions and enhancements of the first iteration.
Animation will maintain the same didactical aspect to further ensure that the user understands the meaning behind the animation. Such enhancements include:

1. Show true pseudo-code by default
2. Highlight in a different color the text to make them easier to read



2.3 User characteristics

AlgorithmA 2010 intention is to target students that attend Computer Science 201 and 202 courses (Introduction to C++ and Intermediate C++ respectively). These users are expected to have no prior knowledge in programming languages. AlgorithmA 2010 ultimate goal is meant to visually display algorithms and common data structures in order to better facilitate a means of learning for these students.

2.4 Constraints

AlgorithmA 2010 and algorithms module shall be functioning and deployed for presentation for the client after the week of Finals of the winter quarter, 2010. All animation should free of identified faults prior to the final demonstration.

2.5 Assumptions and Dependencies

AlgorithmA 2010 assumes that all previous modules and documentation are available and working at an acceptable level.

3. SPECIFIC REQUIREMENTS

3.1 External Interfaces Requirements

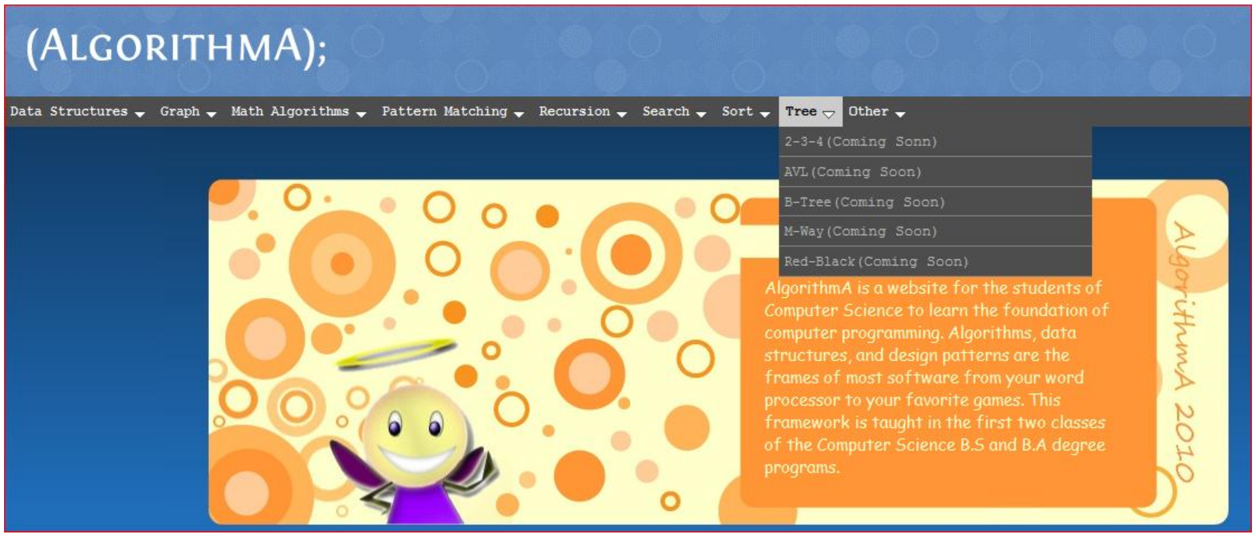
The logo designed for AlgorithmA 2010 will continue to be used along with AlmA the mascot:

AlgorithmA 2010 Logo

AlmA – AlgorithmA Mascot

3.1.1 User Interfaces

When the user first visits the AlgorithmA website, the main webpage will remain the same showing new algorithms enabled and it also display two a navigation bar on top, header, footer, and the main frame at the center of page. The navigation bar will have topics of the different algorithms associated with computer science. Then each of those topics will feature a drop menu showing the algorithms associated with that topic. Below is a concept of the introduction page of AlgorithmA 2010.



AlgorithmA is a website for the students of Computer Science to learn the foundation of computer programming. Algorithms, data structures, and design patterns are the frames of most software from your word processor to your favorite games. This framework is taught in the first two classes of the Computer Science B.S and B.A degree programs.

AlgorithmA 2010 Main Screen

3.1.1.1 General Interface

When the end-user first visits the AlgorithmA 2010 website, a newly redesigned splash screen will be introduced, which also contains two set of pull-down menus. In addition, this page will contain the logo and a synopsis of AlgorithmA 2010 capabilities and functions.

3.1.1.2 Navigation

The layout of the current navigation system will follow the format of the previous iteration, with the right frame containing the information and explanations and the left frame containing the links.

3.1.2 Hardware Interfaces

a. Server side
The web application will be hosted on one of the departments' Linux servers and connecting to one of the school Internet gateways. The web server is listening on the web standard port, port 80.

b. Client side
The system is a web based application; clients are requiring using a high speed Internet connection and using a up-to-date web browser such as Microsoft Internet Explorer 7.0 or greater or Mozilla Firefox 3.5 or greater. In addition, cookies acceptance must be enabled.

3.1.3 Software Interfaces

3.1.3.1 JavaScript

JavaScript was implemented throughout the website in order to display the correct feature the user requested. Once the user makes a request to see a specific walkthrough or animation, JavaScript sends a request to provide that particular feature.

3.1.3.2 Java

Java runtime will be needed to execute the old version of AlgorithmA, which will be available for future developments teams. In order for the user to use it, Java 2 Standard Edition must be installed on the user's system.

3.1.3.3 PHP

PHP is already implemented throughout the website in order to display the correct feature the user requested. Once the user makes a request to see a specific walkthrough or animation, PHP sends a request to provide that particular feature.

3.1.3.4 Communication interfaces

AlgorithmA 2010 is designed to be viewed on any internet browser, provided that:

1. JavaScript is enabled
2. Images are enabled
3. Java installed and enabled (Optional)

Performance may vary slightly between browsers. However, the functionality of the site should not be impaired.

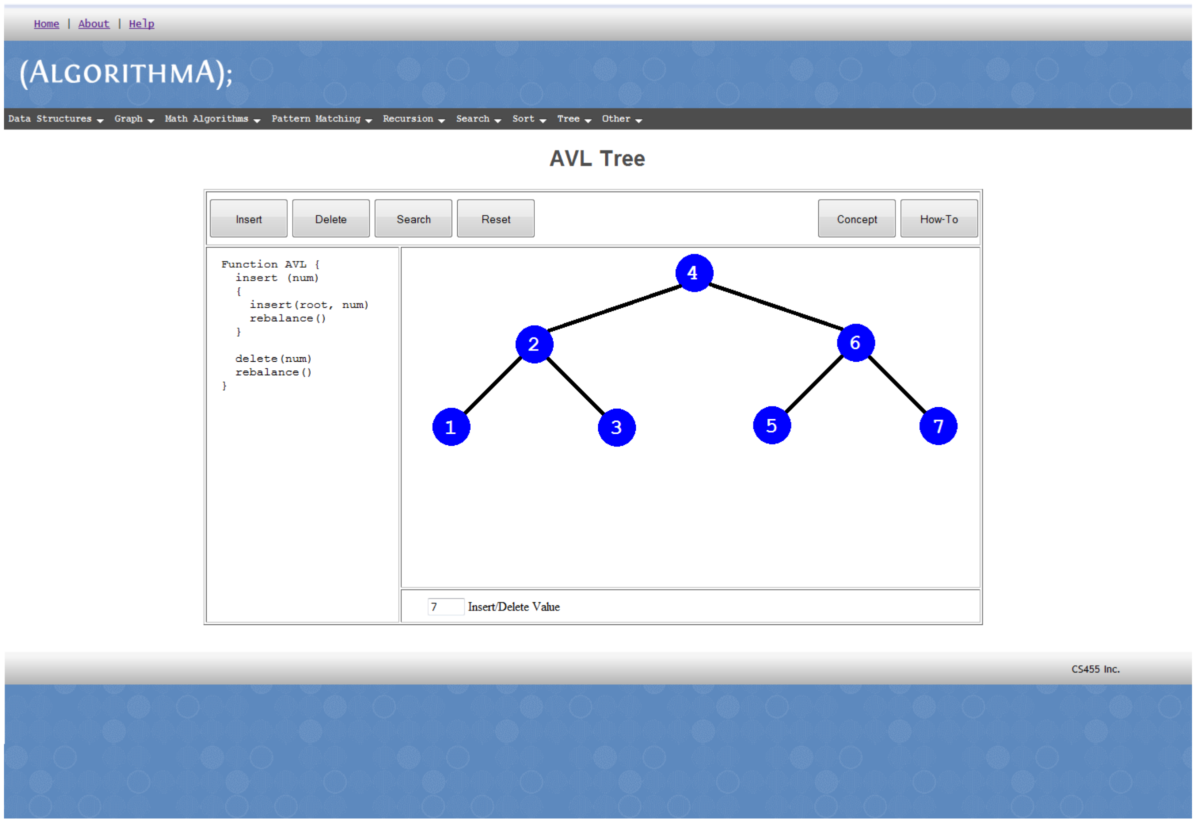
3.2 Functional Requirements

The functions specified in this section directly correspond to work that will be conducted on the AlgorithmA 2010 project. If this is the case, the {AlgorithmA 2010 team will perform whatever measures are necessary to complete the specified new animations and render them in a functional state. This means constructing modules that follow the appropriate architecture while avoiding technical, artistic, and time consuming elements that are implied given the nature of the project. More on this is specified in the SPMP and SQAP documents.

3.2.10 AVL Trees

3.2.10.1 Overview

The second iteration AlgorithmA 2010 will focus on the re-engineering of graphical AVL Tree animations. The four cases of tree balancing will be presented in a colorful animation using numbered circles. The animation will be divided into four separate animations using the root of the tree as the parent and the pivot as the child. The four cases will be animated as follows: Left Left Case, Right Right Case, Left Right Case, and Right Left case. To improve the animation, there will be "Tree-like" animation to relate the information to the user of the concept of insertion and removal of parent and child objects in a tree.



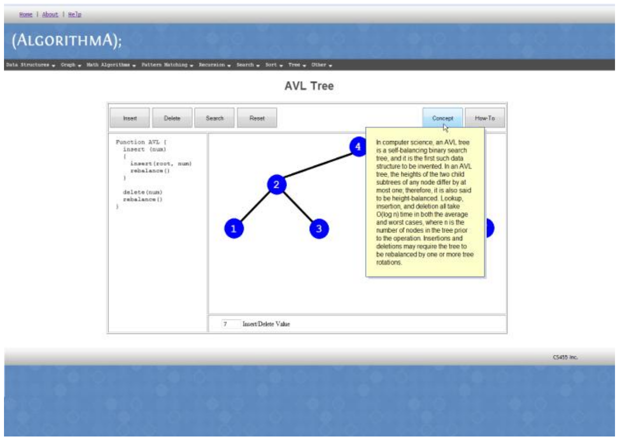
```
Function AVL {
  insert (num)
  {
    insert(root, num)
    rebalance()
  }

  delete(num)
  rebalance()
}
```

AVL Animation UI

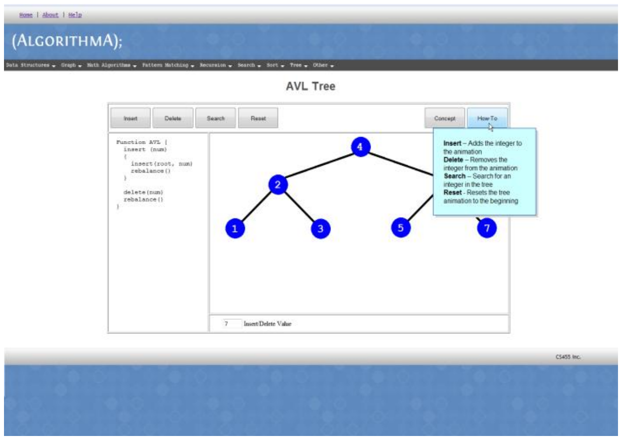
3.2.10.2 Layout

Six buttons will be available as executing commands for each animation. Each action button will be labeled as “Left Left Case”, “Right Right Case”, etc. The animation will consist on rearranging the tree to achieve balancing for each individual case.

3.2.10.3 Functionality



In computer science, an AVL tree is a self-balancing binary search tree, and it is the first such data structure to be invented. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; therefore, it is also said to be height-balanced. Lookup, insertion, and deletion all take O(log n) time in both the average and worst cases, where n is the number of nodes in the tree prior to the operation. Insertions and deletions may require the tree to be rebalanced by one or more tree rotations.



Insert – Adds the integer to the animation
Delete – Removes the integer from the animation
Search – Search for an integer in the tree
Reset - Resets the tree animation to the beginning

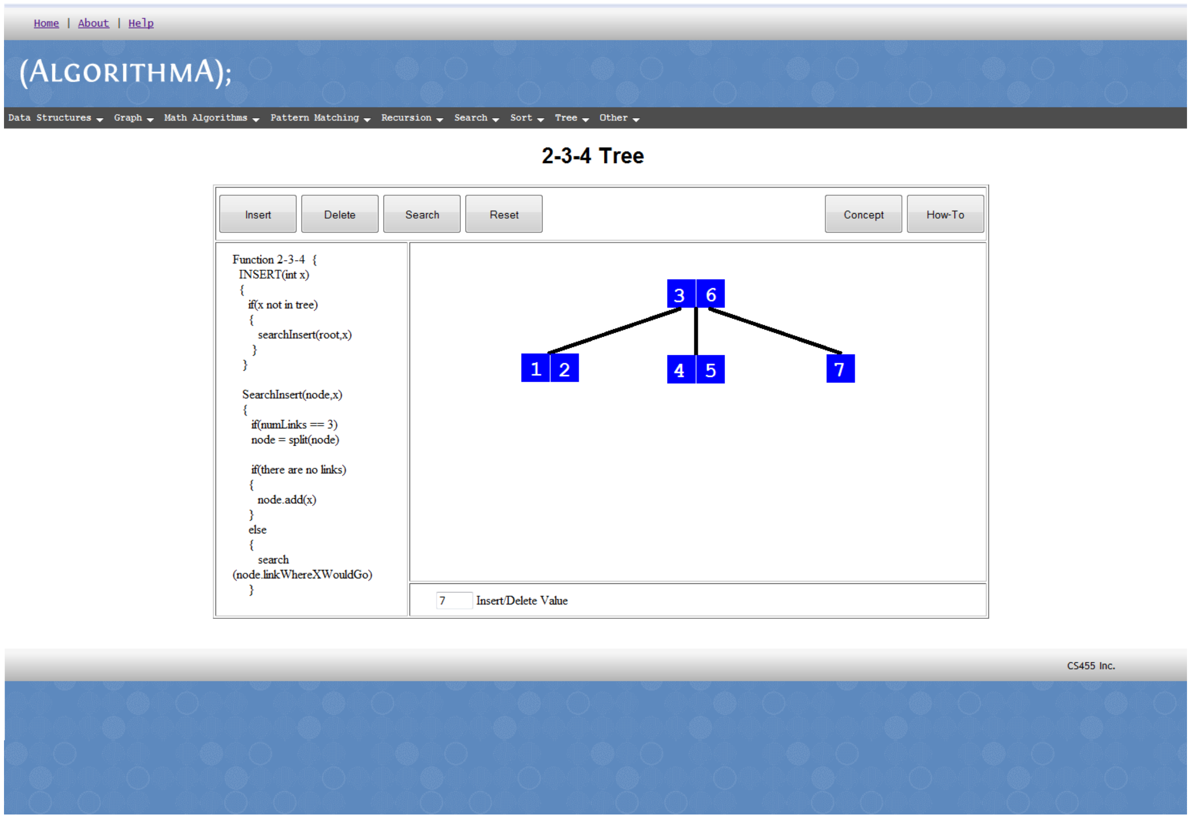
Actions buttons will allow user to execute each iteration individually.

Insert – Adds the integer to the animation
Delete – Removes the integer from the animation
Search – Search for an integer in the tree
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.2.11 2-3-4 Trees

3.2.11.1 Overview

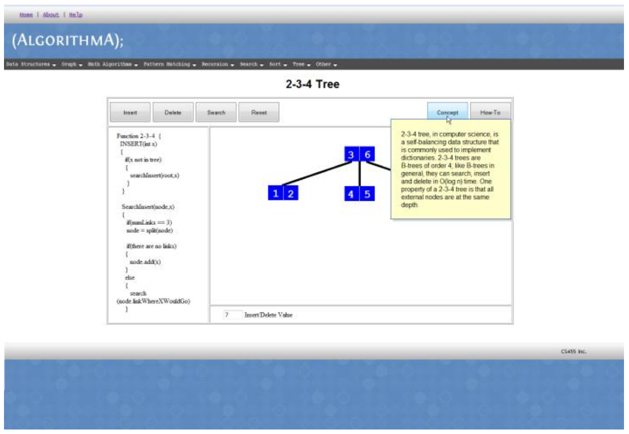
This animation will allow the user to insert an element into the tree and also have the ability to remove it from the tree. When the Start button is pressed, the animation will start executing the pseudo code on the left pane.



```
Function 2-3-4  {
  INSERT(int x)
  {
    if(x not in tree)
    {
      searchInsert(root,x)
    }
  }

  SearchInsert(node,x)
  {
    if(numLinks == 3)
    node = split(node)

    if(there are no links)
    {
      node.add(x)
    }
    else
    {
      search
(node.linkWhereXWouldGo)
    }
```

2-3-4 Tree Animation UI

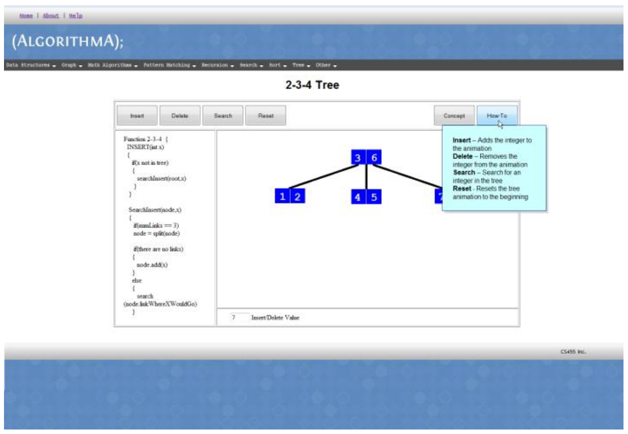
3.2.11.2 Layout



```
Function 2-3-4 {
  INSERT(int x)
  {
    if(x not in tree)
    {
      searchInsert(root,x)
    }
  }

  SearchInsert(node,x)
  {
    if(numLinks == 3)
    node = split(node)

    if(there are no links)
    {
      node.add(x)
    }
    else
    {
      search
(node.linkWhereXWouldGo)
    }
```

2-3-4 tree, in computer science, is a self-balancing data structure that is commonly used to implement dictionaries. 2-3-4 trees are B-trees of order 4; like B-trees in general, they can search, insert and delete in O(log n) time. One property of a 2-3-4 tree is that all external nodes are at the same depth.

7 Insert/Delete Value



```
Function 2-3-4 {
  INSERT(int x)
  {
    if(x not in tree)
    {
      searchInsert(root,x)
    }
  }

  SearchInsert(node,x)
  {
    if(numLinks == 3)
    node = split(node)

    if(there are no links)
    {
      node.add(x)
    }
    else
    {
      search
(node.linkWhereXWouldGo)
    }
```

Insert – Adds the integer to the animation
Delete – Removes the integer from the animation
Search – Search for an integer in the tree
Reset - Resets the tree animation to the beginning

7 Insert/Delete Value

3.2.11.3 Functionality

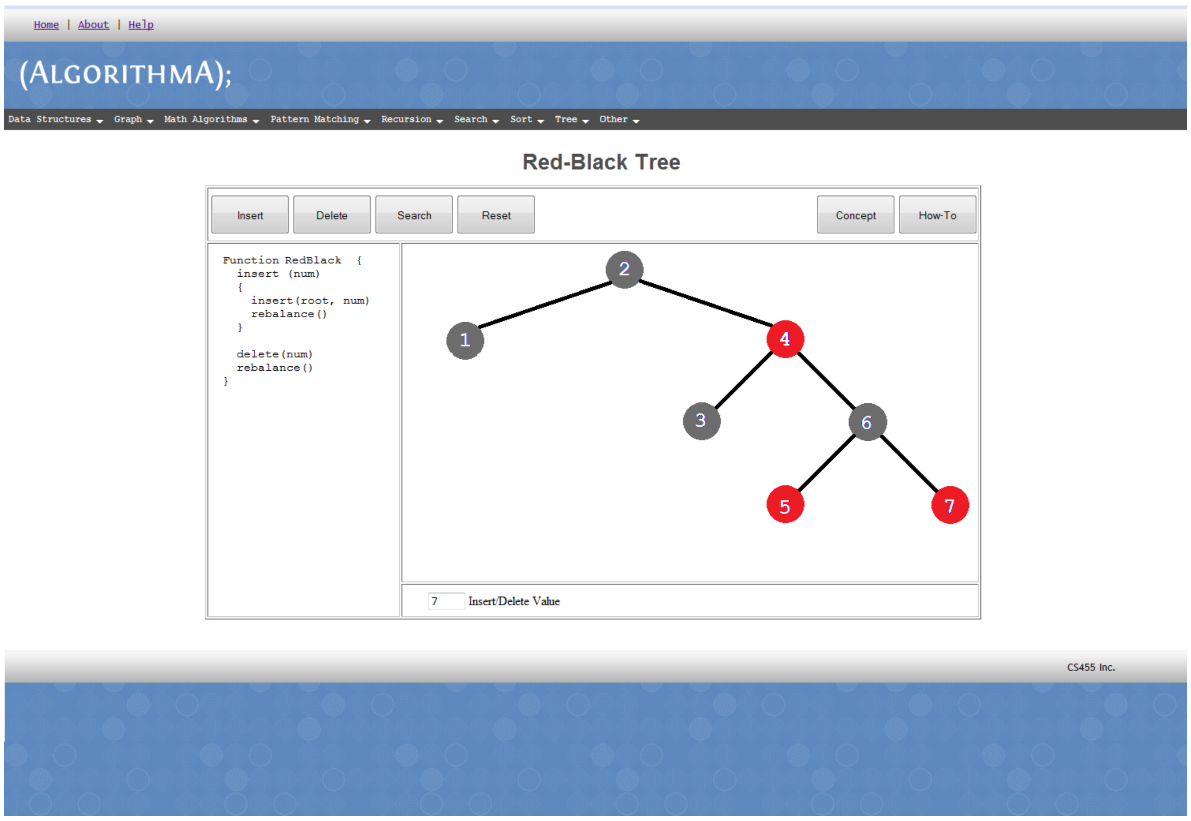
Actions buttons will allow user to execute each iteration individually.

Insert – Adds the integer to the animation
Delete – Removes the integer from the animation
Search – Search for an integer in the tree
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.2.12 Trees – Red Black Tree

3.2.13.1 Overview

This implementation will allow the user to insert an element to the tree and delete an element from the tree. The user will input a value and select to insert or delete. Then hitting the start button will initiate the execution of the code in left pane. By selecting step option she will be able to follow the pseudo - code along with animation line by line. For the each step forward button need to be pressed. When the user deselects the show code option, left pane disappears leaving the animation. User will be able to input a value and select insert or delete. Then hit start, Red Black Tree will be animated.

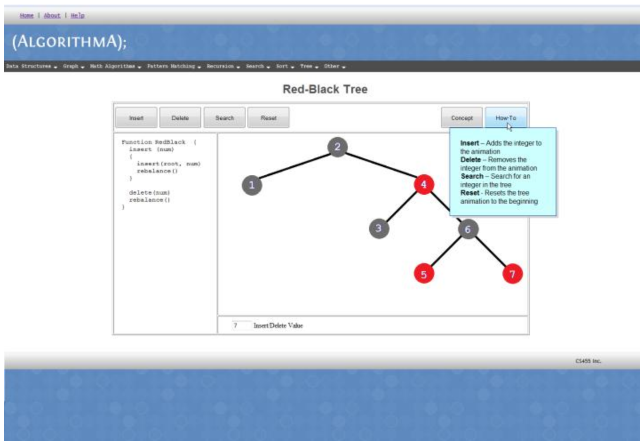


```
Function RedBlack  {
  insert (num)
  {
    insert(root, num)
    rebalance()
  }

  delete(num)
  rebalance()
}
```

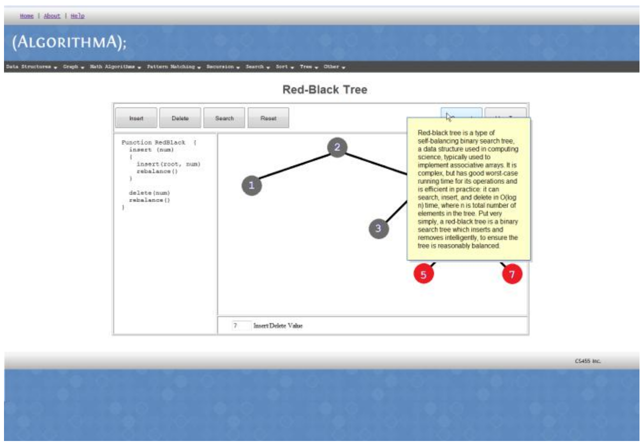
Red-Black Tree Animation UI

3.2.13.2 Layout

The contents of the introduction page to the Red-Black tree will remain unchanged. The animation insertion and deletion will be redesigned to show the tree's connection. The walkthrough will be fixed to show the algorithm with walking indicator. The authoring system will be designed and implement in the iteration.



Insert – Adds the integer to the animation
Delete – Removes the integer from the animation
Search – Search for an integer in the tree
Reset - Resets the tree animation to the beginning



Red-black tree is a type of self-balancing binary search tree, a data structure used in computing science, typically used to implement associative arrays. It is complex, but has good worst-case running time for its operations and is efficient in practice: it can search, insert, and delete in O(log n) time, where n is total number of elements in the tree. Put very simply, a red-black tree is a binary search tree which inserts and removes intelligently, to ensure the tree is reasonably balanced.

3.2.13.3 Functionality

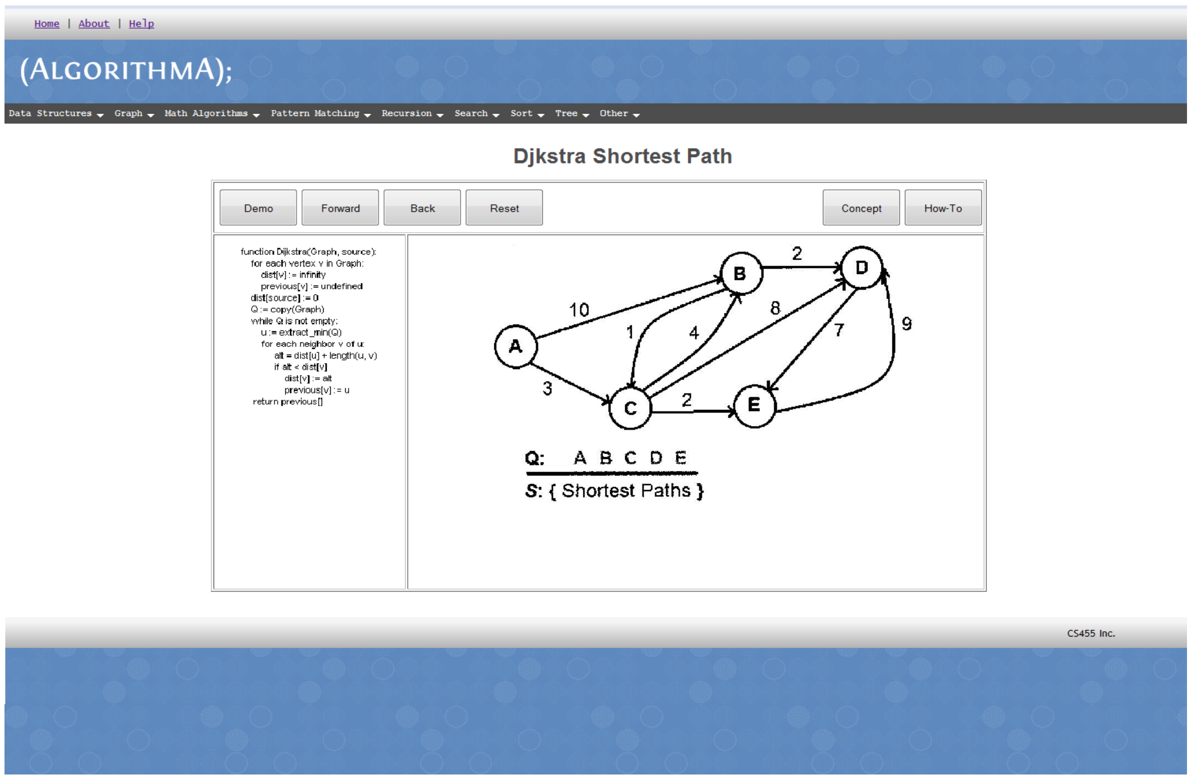
Actions buttons will allow user to execute each iteration individually.

Insert – Adds the integer to the animation
Delete – Removes the integer from the animation
Search – Search for an integer in the tree
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.2.14 Graph - Dijkstra's Algorithm

3.2.14.1 Overview

Iteration 2 of AlgorithmA will focus on Improvements to the Walkthrough Applet of Dijkstra's Algorithm. We will add pseudo code to go along with the animation. This pseudo code will run parallel to the animation so users can see exactly what is occurring when the algorithm is run.



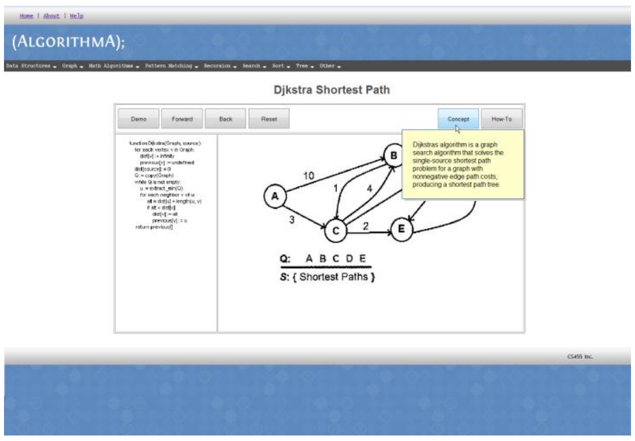
```
function Dijkstra(Graph, source):
    for each vertex v in Graph:
        dist[v] := infinity
        previous[v] := undefined
    dist[source] := 0
    Q := copy(Graph)
    while Q is not empty:
        u := extract_min(Q)
        for each neighbor v of u:
            alt = dist[u] + length(u, v)
            if alt < dist[v]
                dist[v] := alt
                previous[v] := u
    return previous[]
```

Dijkstra's Animation UI

3.2.14.2 Layout

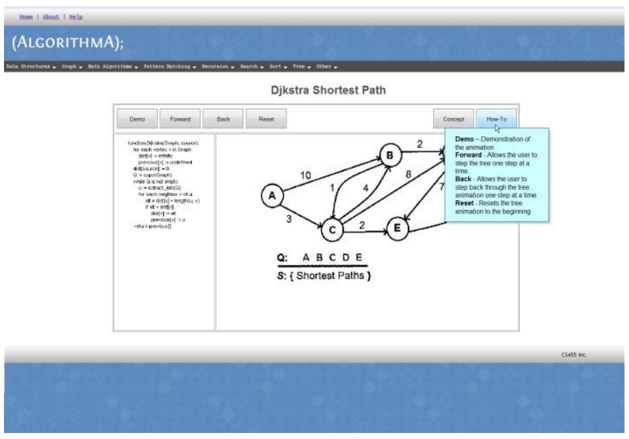
The contents of the introduction page to Dijkstra's Algorithm will remain unchanged. The Animation will also remain unchanged. The walkthrough will be modified to show a box of pseudo code on the left side of the applet. This way the user can see the algorithm working in action.

3.2.14.3 Functionality



```
function Dijkstra(Graph, source):
  for each vertex v in Graph:
    dist[v] := infinity
    previous[v] := undefined
  dist[source] := 0
  Q := copy(Graph)
  while Q is not empty:
    u := extract_min(Q)
    for each neighbor v of u:
      alt = dist[u] + length(u, v)
      if alt < dist[v]
        dist[v] := alt
        previous[v] := u
  return previous[]
```

Dijkstras algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree.



```
function Dijkstra(Graph, source):
  for each vertex v in Graph:
    dist[v] := infinity
    previous[v] := undefined
  dist[source] := 0
  Q := copy(Graph)
  while Q is not empty:
    u := extract_min(Q)
    for each neighbor v of u:
      alt = dist[u] + length(u, v)
      if alt < dist[v]
        dist[v] := alt
        previous[v] := u
  return previous[]
```

Demo – Demonstration of the animation
Forward - Allows the user to step the tree one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

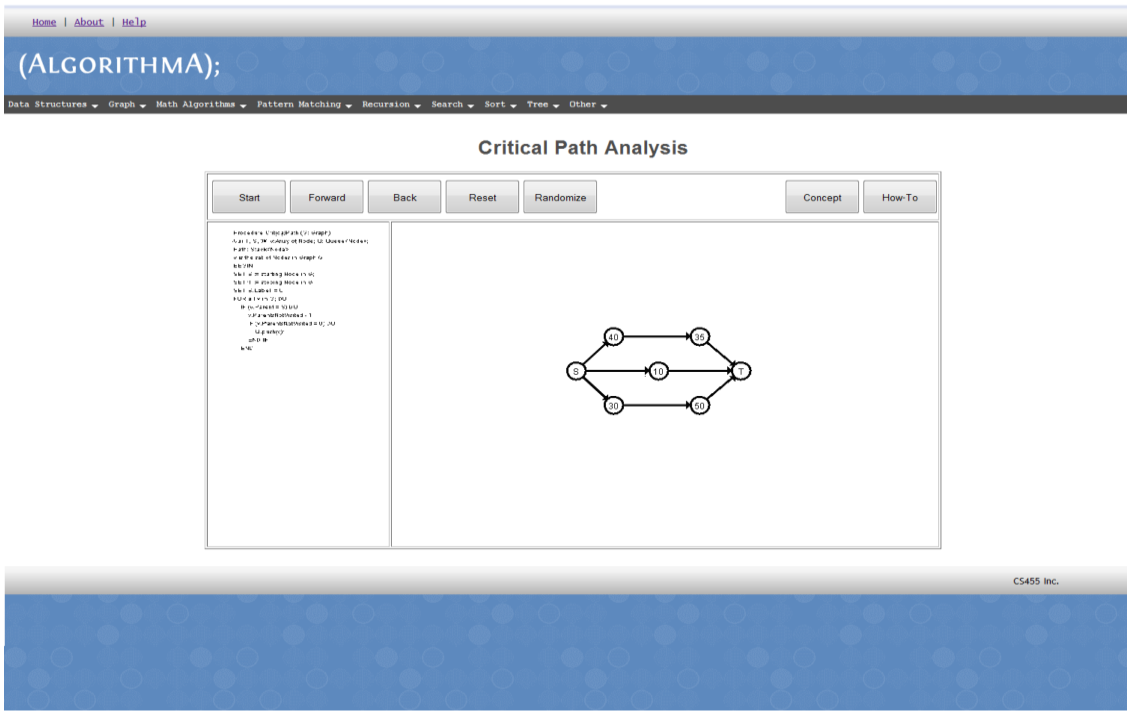
Actions buttons will allow user to execute each iteration individually.

Start – Starts the animation of what the pseudo code executes at run-time.
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.2.15 Critical Path

3.2.15.1 Overview

The Critical Path algorithm will be demonstrated on this page using both documentation and an interactive applet.

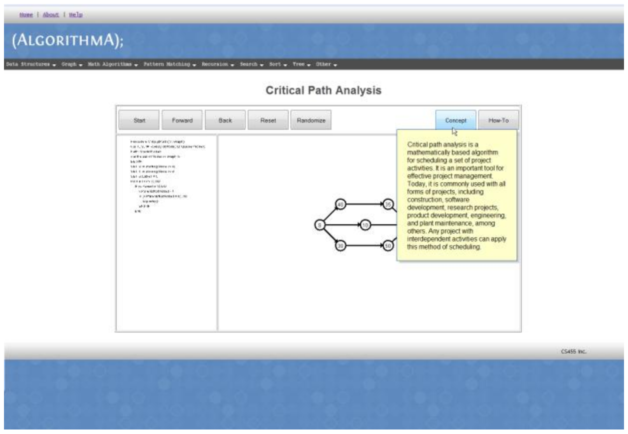


```
Procedure CriticalPath (V: Graph)
Var I, S, [illegible] v:Array of Node; U: Queue<Node>;
Path: Stack<Node>
V is the set of Nodes in Graph G
BEGIN
Set S = starting Node in G;
Set T = ending Node in G
Set S.Label = 0
FOR all v in V; DO
  IF (v.Parent = S) DO
    v.ParentsNotVisited - 1
    IF (v.ParentsNotVisited = 0) DO
      U.push(v);
    END IF
  END
```

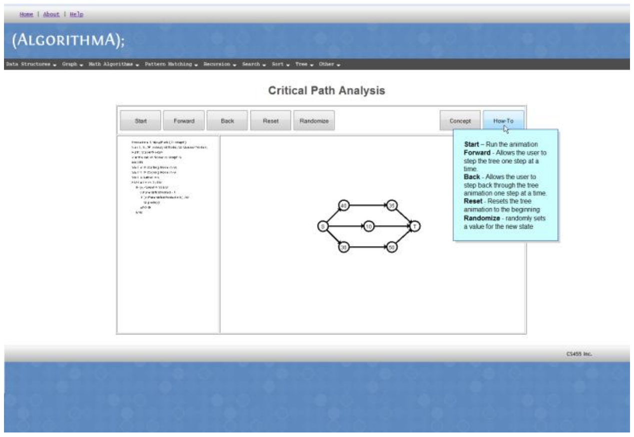
Critical Path Animation UI

3.2.15.2 Layout

The Critical Path page will display documentation concerning the Critical Path algorithm. This documentation will be followed by a center-aligned animation, which demonstrates the concept. The left pane will have a left pane containing pseudo code and a right pane containing a graph to represent the Critical Path algorithm. Buttons, checkbox options, and a slide bar will be centered at the bottom of the applet. Color and overall visual appearance will be selected to match the applets for other algorithms.



Critical path analysis is a mathematically based algorithm for scheduling a set of project activities. It is an important tool for effective project management. Today, it is commonly used with all forms of projects, including construction, software development, research projects, product development, engineering, and plant maintenance, among others. Any project with interdependent activities can apply this method of scheduling.



Start – Run the animation
Forward - Allows the user to step the tree one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning
Randomize - randomly sets a value for the new state

3.2.15.3 Functionality

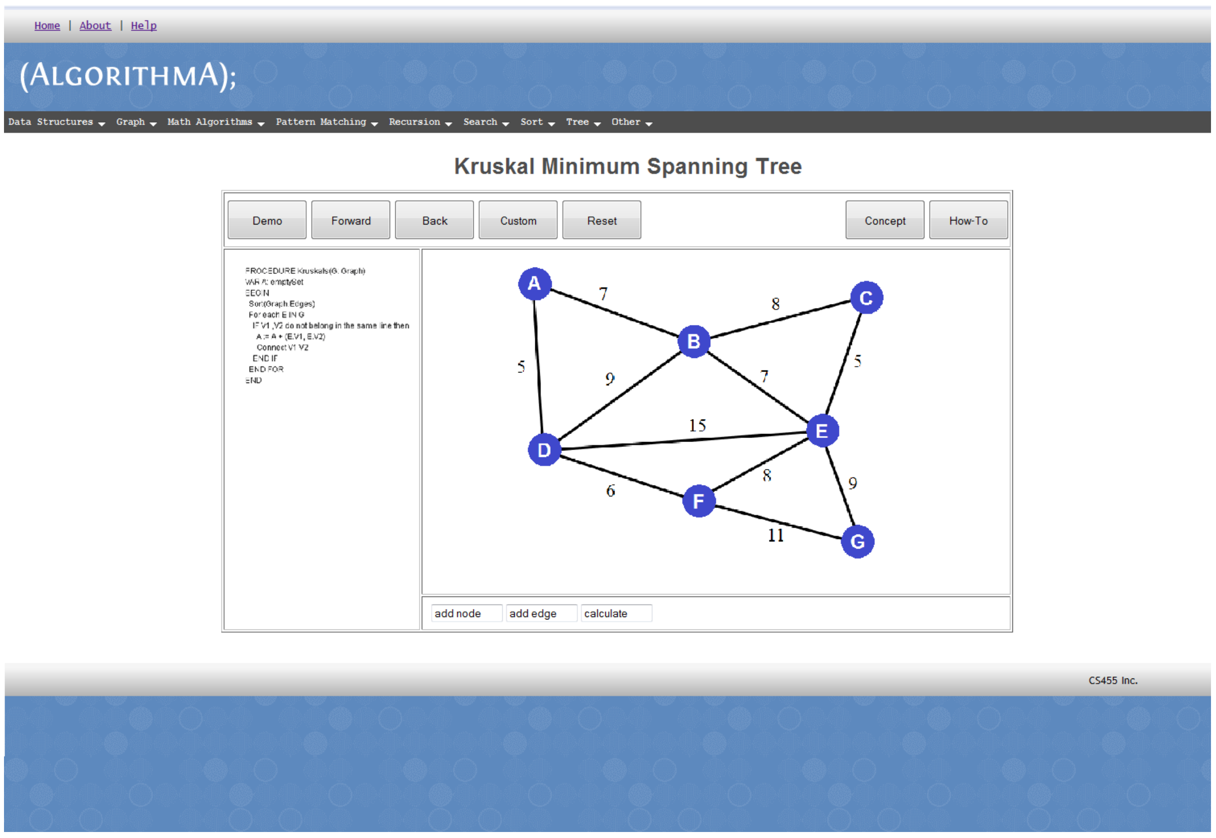
Actions buttons will allow user to execute each iteration individually.

Start – Starts the animation of what the pseudo code executes at run-time.
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.2.16 Kruskal's Algorithm

3.2.16.1 Overview

The minimum spanning tree (MST) of a weighted graph is a spanning tree whose sum of edge weights is minimal. The minimum spanning tree describes the cheapest network to connect all of a given set of vertices. Kruskal's algorithm for minimum spanning tree works by inserting edges in order of increasing cost, adding as edges to the tree those which connect two previously disjoint components. Here we see Kruskal's algorithm at work on a graph of distances between various cities. Almost imperceptivity at first, short edges get added all around the area, slowly building larger components until the tree is completed.

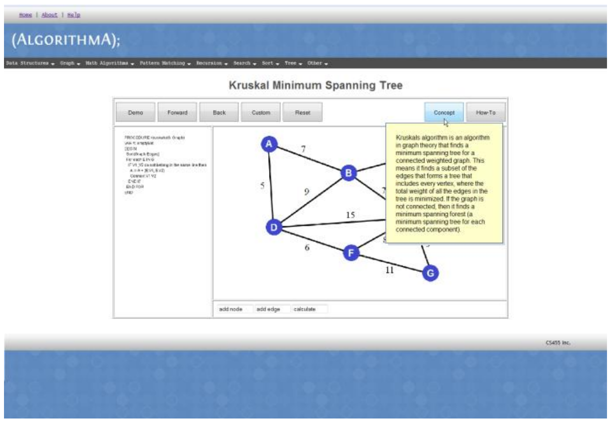


```
PROCEDURE Kruskals(G: Graph)
VAR A: emptySet
BEGIN
 Sort(Graph.Edges)
 For each E IN G
  IF V1 ,V2 do not belong in the same line then
   A := A + (E.V1, E.V2)
   Connect V1 V2
  END IF
 END FOR
END
```

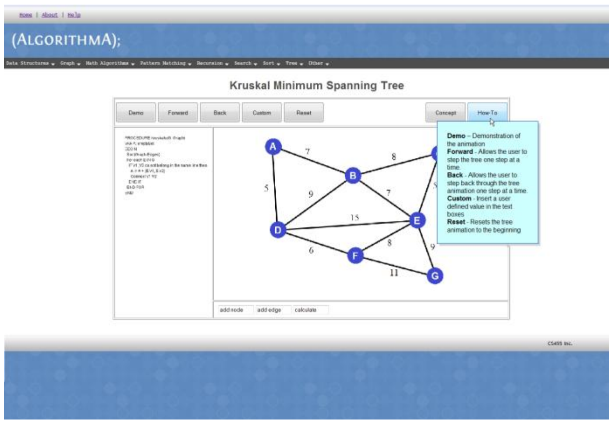
Kruskal’s Animation UI

3.2.16.2 Layout

Kruskal’s algorithm will be classified under graphs. Once selected, an introduction main page is shown that defines the algorithm, describes its uses, and displays a graphic that represents the algorithm. From this page, a user is able to view a walkthrough animation of the pattern.


Home | About | Help
(ALGORITHMA);
Data Structures Graph Math Algorithms Pattern Matching Recursion Search Sort Tree Other
Kruskal Minimum Spanning Tree
Demo
Forward
Back
Custom
Reset
Concept
How-To
add node
add edge
calculate
CS455 Inc.

Kruskals algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).


Home | About | Help
(ALGORITHMA);
Data Structures Graph Math Algorithms Pattern Matching Recursion Search Sort Tree Other
Kruskal Minimum Spanning Tree
Demo
Forward
Back
Custom
Reset
Concept
How-To
add node
add edge
calculate
CS455 Inc.

Demo – Demonstration of the animation
Forward - Allows the user to step the tree one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Custom - Insert a user defined value in the text boxes
Reset - Resets the tree animation to the beginning

3.2.16.3 Functionality

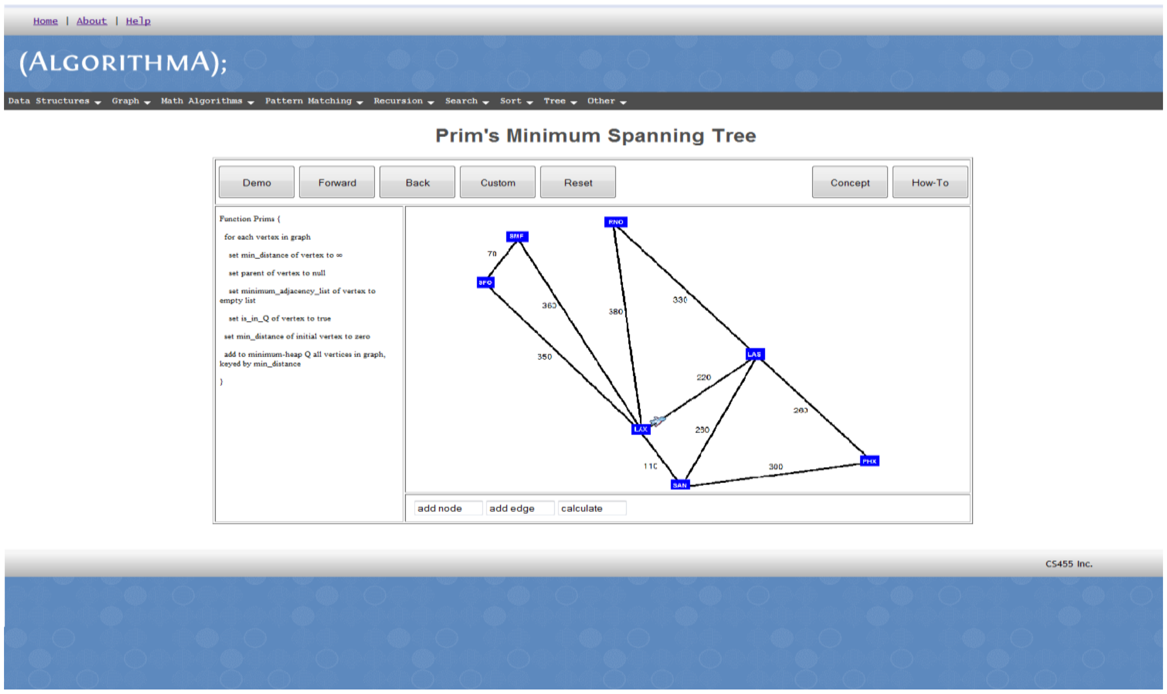
Actions buttons will allow user to execute each iteration individually.

Start – Starts the animation of what the pseudo code executes at run-time.
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.2.17 Graph - Prim's Algorithm

3.2.17.1 Overview

Prim's Algorithm will be classified under the Graphs pull-down menu. It will consists of two animation, one is the a walkthrough and the other that does a straight animation through it. No further intervention from the user is required as the animation will start and end automatically once the start button is pressed.

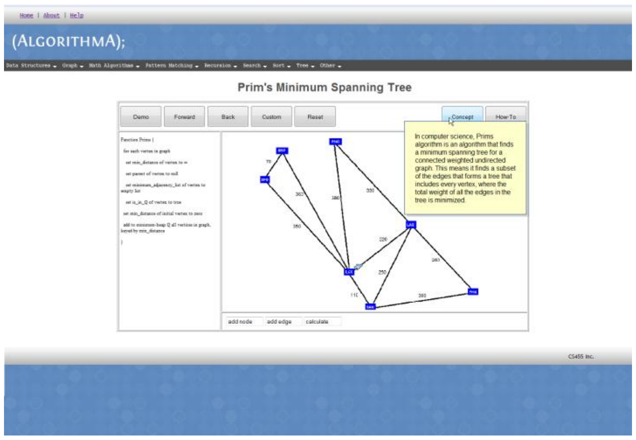


```
Function Prims {
  for each vertex in graph
    set min_distance of vertex to ∞
    set parent of vertex to null
    set minimum_adjacency_list of vertex to
empty list
    set is_in_Q of vertex to true
  set min_distance of initial vertex to zero
  add to minimum-heap Q all vertices in graph,
keyed by min_distance
}
```

Prim's Minimum Spanning Tree Animation UI

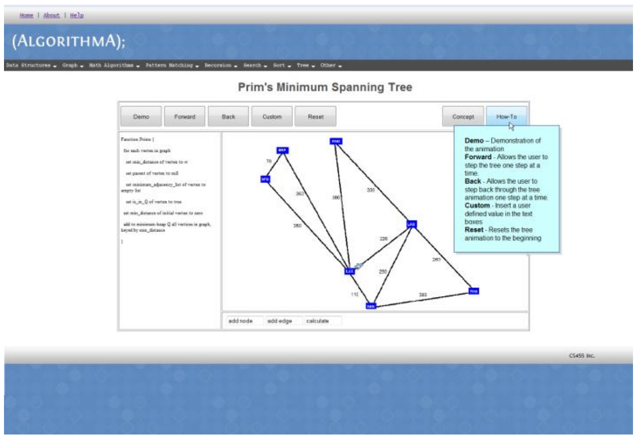
3.2.17.2 Layout

The animation and walkthrough applets will have a similar layout to the following figures.


Home | About | Help
(ALGORITHMA);
Data Structures Graph Math Algorithms Pattern Matching Recursion Search Sort Tree Other
Prim's Minimum Spanning Tree
Demo
Forward
Back
Custom
Reset
Concept
How-To
add node
add edge
calculate
CS455 Inc.

```
Function Prims {
  for each vertex in graph
    set min_distance of vertex to ∞
    set parent of vertex to null
    set minimum_adjacency_list of vertex to empty list
    set is_in_Q of vertex to true
  set min_distance of initial vertex to zero
  add to minimum-heap Q all vertices in graph, keyed by min_distance
}
```

In computer science, Prims algorithm is an algorithm that finds a minimum spanning tree for a connected weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.


Home | About | Help
(ALGORITHMA);
Data Structures Graph Math Algorithms Pattern Matching Recursion Search Sort Tree Other
Prim's Minimum Spanning Tree
Demo
Forward
Back
Custom
Reset
Concept
How-To
add node
add edge
calculate
CS455 Inc.

Demo – Demonstration of the animation
Forward - Allows the user to step the tree one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Custom - Insert a user defined value in the text boxes
Reset - Resets the tree animation to the beginning

3.2.17.3 Functionality

The animation applet will contain the following functions:

Actions buttons will allow user to execute each iteration individually.

Start – Starts the animation of what the pseudo code executes at run-time.
Forward - Allows the user to step forward through the tree animation one step at a time.
Back - Allows the user to step back through the tree animation one step at a time.
Reset - Resets the tree animation to the beginning

3.3 Walkthroughs

The walkthrough interface allows the user the interactively step through an algorithm animation.

3.4 Animation

The animation interface animates an algorithm giving the user the ability to obtain an overall perspective of an algorithm's functionality.

3.5 Performance Requirements

Unlike previous versions of AlgorithmA, performance continues to be the key to our version. The move from Java to JavaScript is based almost entirely on the fact that Java is not meant to be used in such an environment that we are using it for. Animations must be smooth, data retrieval must be fast, and navigation should be a breeze.

3.6 Design Constraints

Since the new animation are being re-engineered from scratch, having to port legacy code like in previous AlgorithmA projects will not be performed as it was the case in the first iteration. But we must ultimately conform to an easy to document and understand standard. We must also constrain ourselves to the JavaScript 1.6 standards.

3.7 Software System Attributes

The legacy for the future generations of AlgorithmA, otherwise known as AlgorithmA 2010, will be an implemented architecture that will bring simplicity to maintaining and extending the current code base. Besides coding, a much larger emphasis will be placed on documenting the iteration, by providing a knowledgebase of information. Such documentation include: JavaScript doc, detailed descriptions of code in the actual source code, and a comprehensive log of all activity via the wiki dedicated to AlgorithmA 2010.

3.8 Other Requirements

AlgorithmA 2010 must be supportable: It has to be maintained well enough to be smoothly taken over by the next CS455, Inc. AlgorithmA 2010 must be easy to use for instructors and students interested in learning about design patterns. AlgorithmA 2010 must be reliable: It cannot crash. Moreover, AlgorithmA 2010 must continue to run on the computer science school web site well after development for future project groups to access. It should be available 24 hours a day, 7 days a week, and 365 days a year. The only time when the site will become unavailable is during short maintenance periods.

3.9 Documentation

Three levels of documentation implemented in iteration # 1, will continue and insuring completion of each artifact for the delight of the users. A hyper context-sensitive help pop up screen will be also implemented throughout each algorithm. A detailed installation technical guide will be developed as supporting documentation for the next generations of AlgorithmA. And finally, a User's Manual will be completed based on the implementation of the previous iteration and the new animations. The User's manual will cover explanation of the algorithms behind the theory of each algorithm as well as how to execute and manipulate each control presented to the user when executing each module. The final documentation will be available in a PDF format.