GATutor: Intelligent Tutoring System for Greedy Algorithms

Dissertation Report

Submitted in the partial fulfillment of the requirements

for the degree of

Master of Technology

by

Mukund Lahoti

Roll No. 123050018

under the guidance of

Prof. Sridhar Iyer

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai
2014

Acknowledgement

I would like to express my gratitude to my guide **Prof.Sridhar Iyer**, for his constant motivation, supervision and guidance, constructive suggestions during the planning and development of this research work. His suggestions always helped me go forward in the right direction when the work was stuck. Advice given by Shitanshu Mishra has been a great help at certain times. Special thanks should be given to Meenakshi Verma, my research project team member for her professional assistance.

Mukund Lahoti
Mtech2
Computer Science and Engineering
IIT Bombay

Abstract

Algorithms are an important part of computer science course which have widespread applications. Given their importance it has become necessary to teach and learn them with sound clarity. There have been many attempts in the past to do so; primarily with animation. With many computer based tutoring systems already there, we have build an Intelligent Tutoring System for effective teaching and learning of Greedy Algorithms-GATutor. For this we also devised a general framework of teaching which can be applied to any Greedy Algorithm using guided discovery learning principles. System makes the user to develop an algorithm themselves by providing them stimulating questions and timely hints to real life scenarios. Users' data is analyzed by the system which then provides insights to the teacher.

In this report, we have first describe various other systems for teaching of algorithms. Then we explain our framework for teaching greedy algorithms. After that we explain in detail the software design of our system, its module details and the challenges faced in building them. Finally we describe the system evaluation based on usability, learning and attractiveness along with the future work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Need and Motivation

Programming techniques and algorithms are the essence of Computer Science. There have been research and work for the effective teaching and learning of them and because of which now instead of just listening to a live lecture, there are many ways for teaching and learning algorithms. One of these methods is showing the algorithm through animation and there are systems dedicated to it. But we found that algorithm visualization systems aid students merely in the understanding of the flow of algorithms. There does not seem to be any other pedagogical goal. If we look at some typical textbooks for Greedy Algorithms they have this structure: Greedy Algorithm technique, problem statement, optimal function and then a pseudo code. There has been some work for effectively combining the benefits of both. Still there are two questions at large-Is it possible to make it interesting rather fun without compromising the concepts? and can the role of instructor be obviated in doing so? We devised a strategy of allowing the students to play with the concepts, do things on their own, providing them stimulating questions, giving them timely and to the point hints. A Computer based Intelligent Tutoring System implementing the strategy is build in which students apply their cognitive skills and are more likely to retain the concepts. Student is made an active learner by fostering constructivist learning as much as possible through the system. Instructors can also effectively monitor the progress of students through such system and use the system whenever required.

1.2 Overview of work done

1.2.1 Final System

Final system (GATutor) is in the form of a website where both students and instructors can login. Students start learning Greedy Algorithms by first solving a puzzle without any knowledge of the underlying algorithm involved. Then

the system gives them a couple of questions in the answer of which the tiny bit of relevant information is there. Students try to answer the question where at each wrong answer system tells them why they are wrong and hints them to the correct answer. Ultimately students come up with the trick to solve the puzzle which is basically an algorithm. Instructor on the other hand can see the progress of the entire class in general as well as detailed progress of a student. He can also find out misconceptions of the entire class as well as of a student.

1.2.2 Educational Technology Involved

A Framework is designed keeping above goals in mind to facilitate the teaching as well as learning. The framework is such that it makes students to explore different options and come up with a solution (Constructivist) and gently pushes him/her to the correct answer (Guided Learning). Higher levels of Bloom's Taxonomy have been addressed. Real life puzzles to be solved within a time limit stimulates the mind. The entire algorithm is taught interactively. The tutor provides guidance whenever necessary. The feedback is input dependent means the system tells exactly what is the mistake done, if any. Thus a sort of Scaffolding is also done. All Greedy Algorithms have a correct Selection and Optimal function. The correctness of these is explained through contradiction. By learning small things initially the entire algorithm is known to a student on his/her own through a Discovery learning approach.

1.2.3 Walk-through the system

Our System is a website whose web pages are dynamic on server as well as client side. It can be explained through various modules

- **Login and Back-end**
  A student enters into the system with the user name and password. From there on his activity is being tracked. This activity can be seen by the administrator in the database or by the instructor in an instructor page. Instructor can draw meaningful conclusions from this. This is explained in later chapters of the report.

  Servlets are used for checking the credentials of a user through a jdbc connection with the mysql or for creating a new user account. Predefined sql queries are written for instructor to see meaningful data from database. Database is safe to my knowledge of SQL injection.

- **Algorithms**
  All the major greedy algorithms are covered viz. Scheduling, Kruskal's, Prim's, Dijktstra's and Knapsack. Knapsack and Kruskal's Algorithm are explained in later chapters of this report whereas Scheduling, Prim's and Dijktstra's are explained in Meenakshi's Report.

All the pages are jsp pages which store in the database its access time and logged-in user.

- **Real life puzzle**
  Each algorithm starts with a real life interesting puzzle to be solved within a time limit. It is unlikely that without the prior knowledge of the algorithm, the puzzle can be solved in the time limit. This is further explained in the later chapters.

  Puzzles are made using SVG and user plays with it using javascript.

- **Teaching by asking**
  Students are now asked some basic questions through which they develop the notion of the implied algorithm. Their correct answers are also visualized to them for their better understanding. This is further explained in the later chapters.

  Visualization is by javascript changing the attributes of SVG elements. Student's interactivity is also managed by javascript.

- **Wrong answers by students**
  Every wrong answer by student is dealt seriously. They are told exactly what is the mistake. Misconception is further removed by visualizing them a contradicting example. This is further explained in the later chapters.

  All this is done with javascript.

- **End quiz**
  At the end of the chapter their is a quiz too. The graphs and the knapsack involved in the greedy algorithms are modifiable so no extra paper is required on the part of student. Student's can track their progress in a progress bar and give multiple attempts for a question though by penalizing himself in the marks awarded.

  Quiz module is in jQuery on top of SVG graphs.

1.2.4 Experiments

We conducted many experiments at different stages of our system by actual users (first and second year undergrad students). System was modified to their appropriate feedbacks.
At the end, we conducted experiments for the effectiveness of the system on the lines of system's usability, learning and attractiveness.

1.2.5 Screen-Shots

Figure 1.1: Snapshot of Login page of system

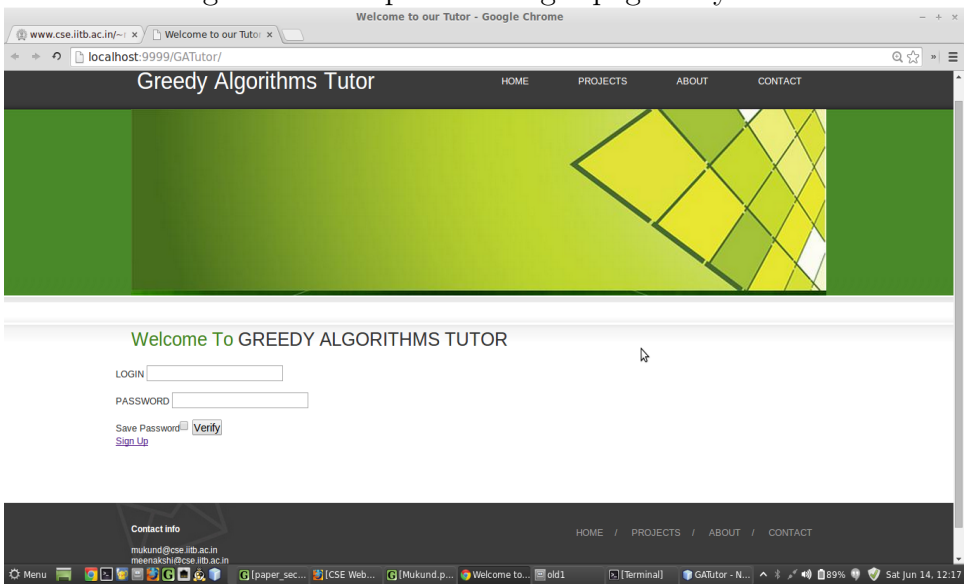
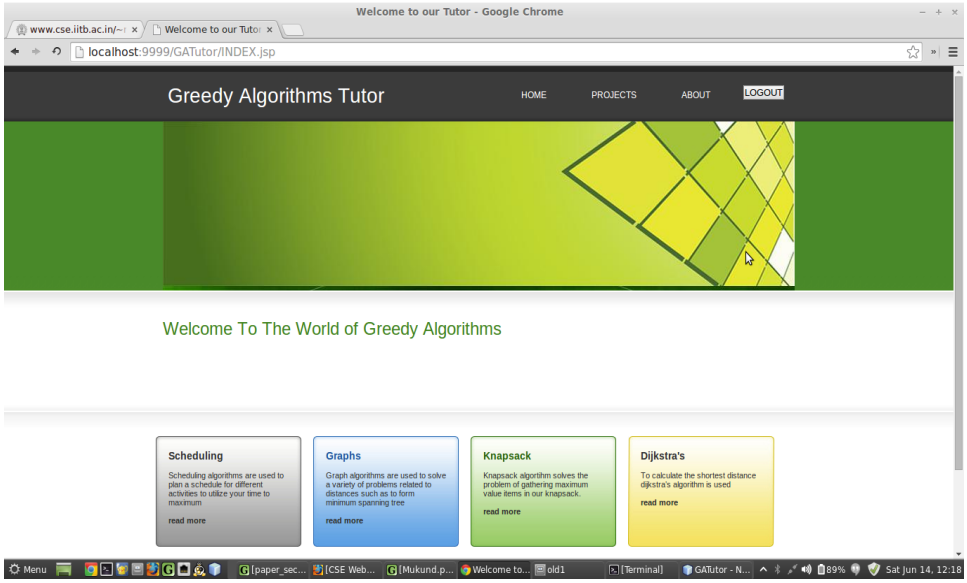


Figure 1.2: Snapshot of Index page



1.3 Organization of this report

Chapter 2 compares our system with various other systems already present on different parameters. Chapter 3 explains our strategy of teaching and gives the overview of our system. Chapter 4 explains the design details with explanation of various modules of our System. Chapter 5 consists of experimental results. Chapter 6 presents the future work.

This is a research work jointly done with Meenakshi. In this report the login module for the student and administrator and the analysis using the database is explained. Also Kruskal's and Knapsack algorithm are explained along with the proof of correctness and optimality. In Meenakshi's report [**?**] Scheduling, Dijkstra's and Prim's algorithms are explained.

Chapter 2

Related Work

2.1 Introduction

We extensively searched for Programming and Algorithm Tutoring systems. We found some very impressive programming tutors and then we set our target as Algorithm teaching and learning particularly Greedy Algorithms. We found that many systems are scattered over the Internet but none of the tutors satisfied educational theories to a reasonable extent and also not all are worth mentioning.

- Algo Tutor : AlgoTutor takes the concept of operation blocks, and uses that to help and teach students the process of designing correct algorithms. It includes a program pad to demonstrate to the students the conversion of an algorithm to a program. This visualization helps the students to not only understand how to materialize an algorithm when it is ready, but also give valuable first hand experience in doing the same. The visualization tool also helps students to debug the code at each level. This aid is re- quired for students to understand the thought process behind the process of debugging. It also has a drag and drop interface, which is very user friendly.

  The AlgoTutor, however, provides only low level programs. This makes its applicability limited to novice users. Advanced students would like to play with newer programs, to understand the process of building programs. It is also not open source, which refrains other instructors to develop new modules for their own use, and possibly contribute back to the software. Tutor has been tested on the university students using the concept of pre/post tests. Results of these experiments has been shown in the paper [**?**], briey.Different snapshots of algotutor has been shown in 2.1,2.2 and 2.3.

Figure 2.1: Snapshot of AlgoTutor

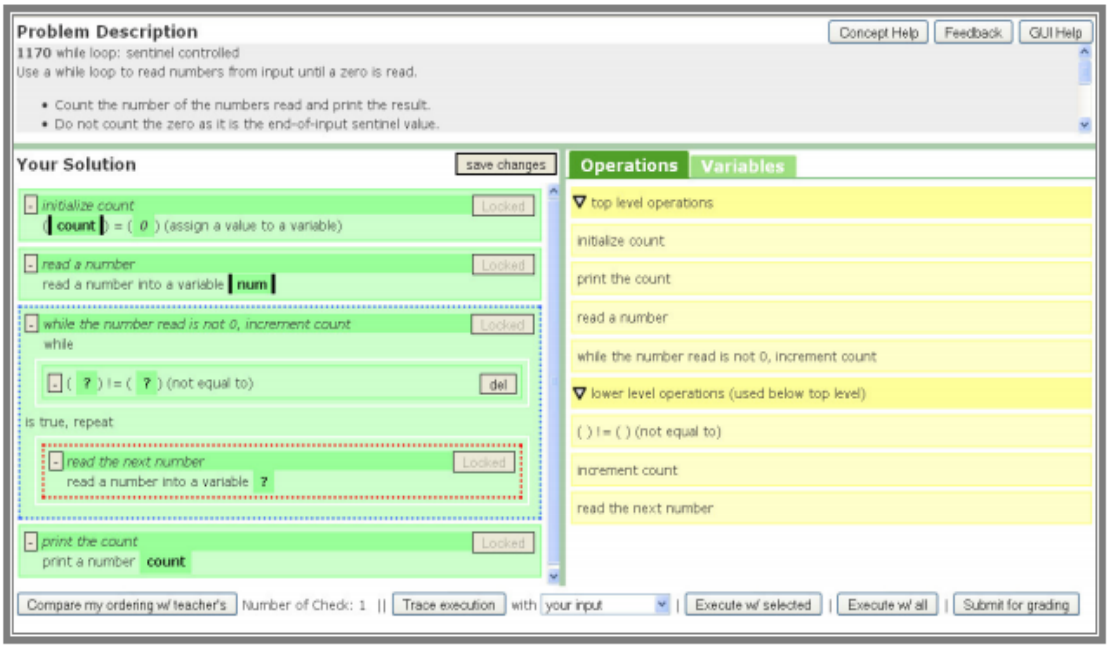


Figure 2.2: Snapshot of AlgoTutor

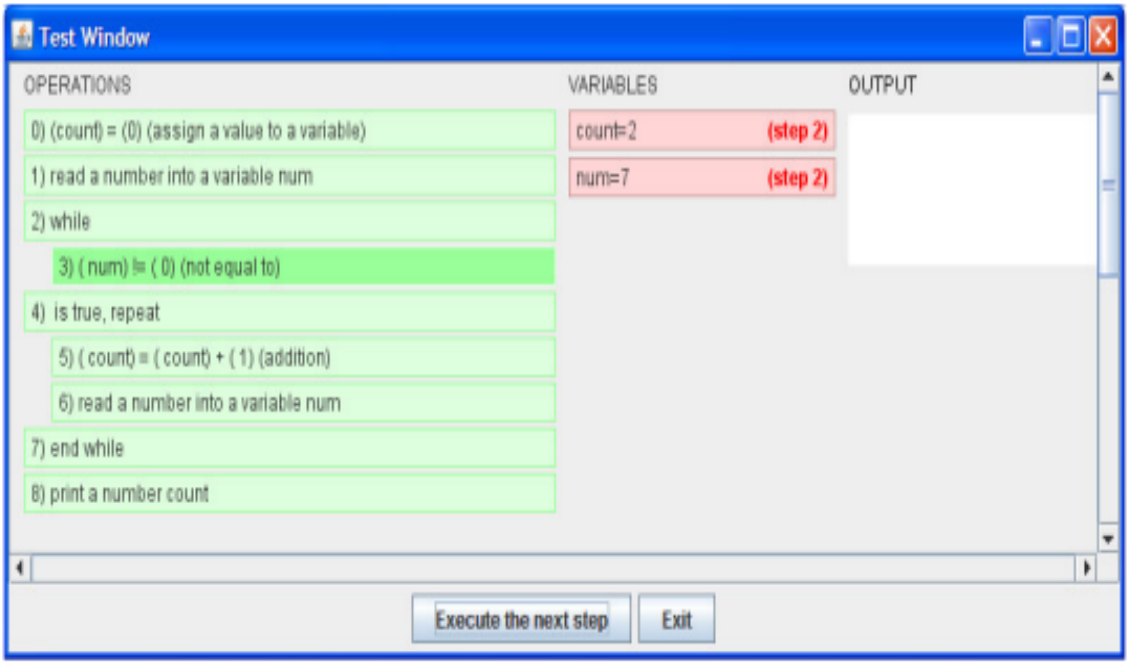


Figure 2.3: Snapshot of AlgoTutor

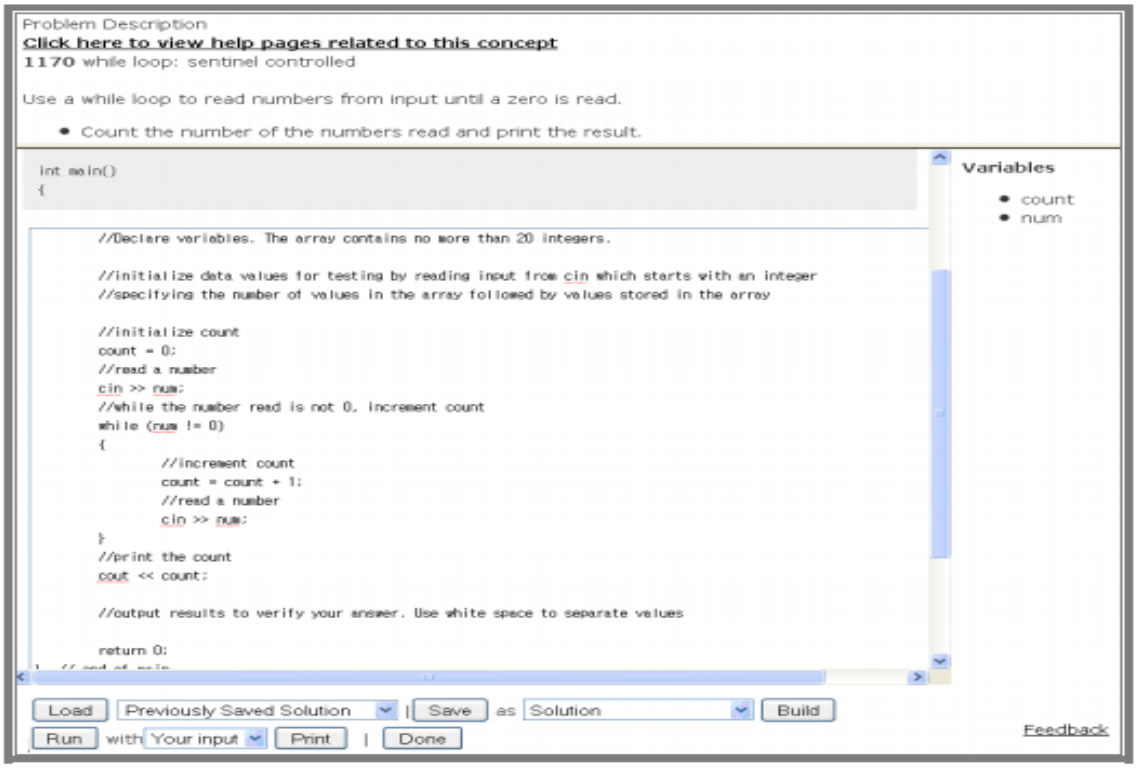
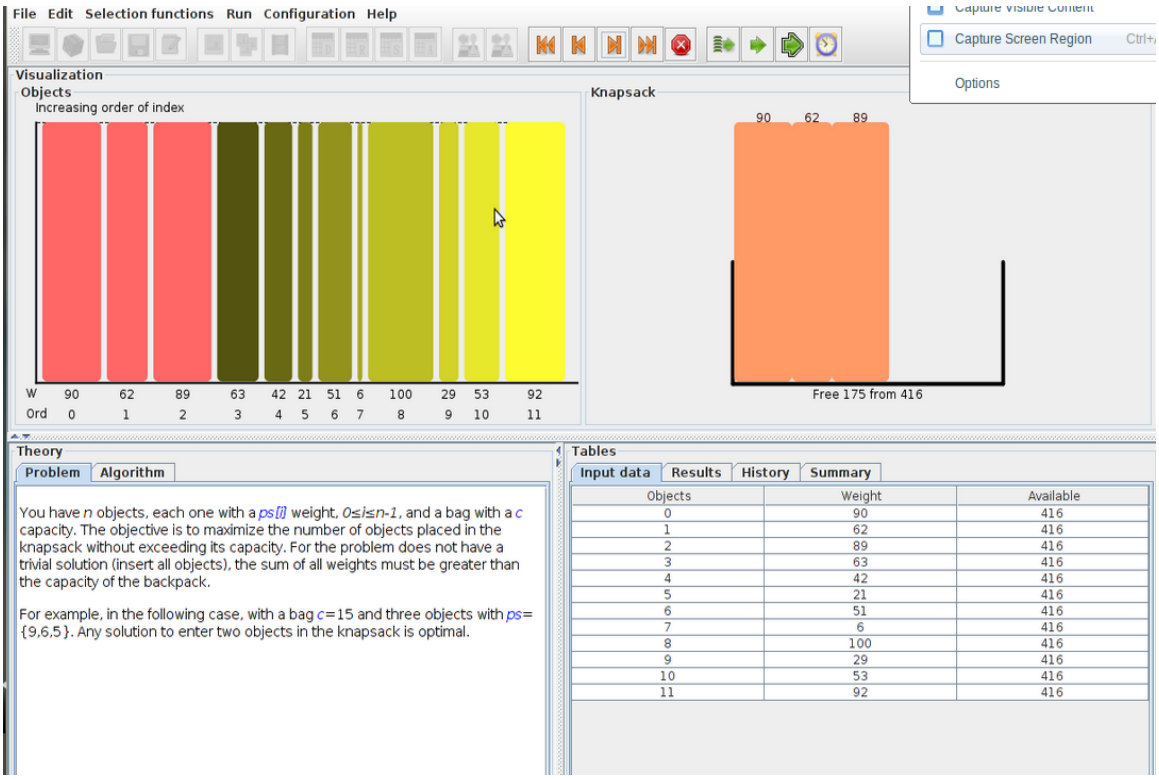


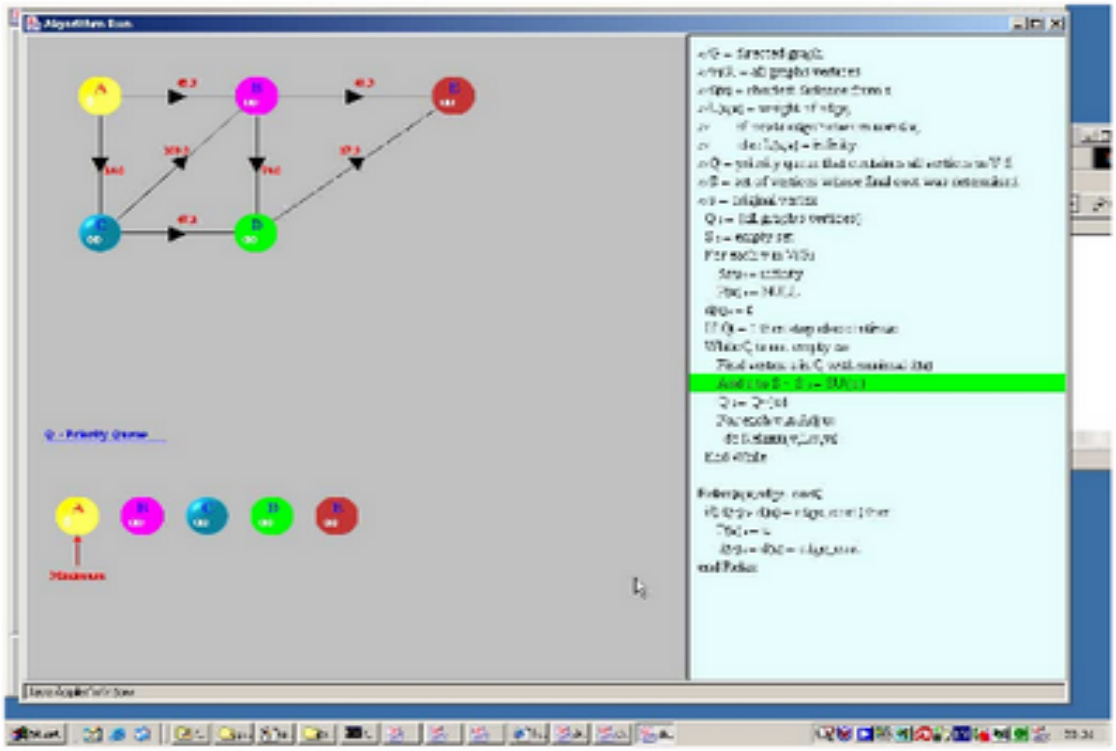
Figure 2.4: Snapshot of Greed-Ex system



- GREED-Ex: This system [?] shown in 2.4 is for teaching of Activity Selection and Knapsack problem only:two very common problems where Greedy approach is applied. Students can enter the input for the problem or system can generate random input. Depending upon the selection function (like increasing end time, decreasing start time,etc.) provided by the student, the result is visualized by the system. This means students "see" the progress of the algorithm. Pseudo-code is also provided. System also summarizes the result of different selection functions in a summary table. Experiments were performed on the undergraduate computer science students.

  They say that their system works on discovery learning principle but we feel that the user is still a passive entity though not entirely. He is not able to choose activities but is just shown an animation according to the selection criteria. Also no test or quiz is there to judge learning of student. Also there is no application of the problem.

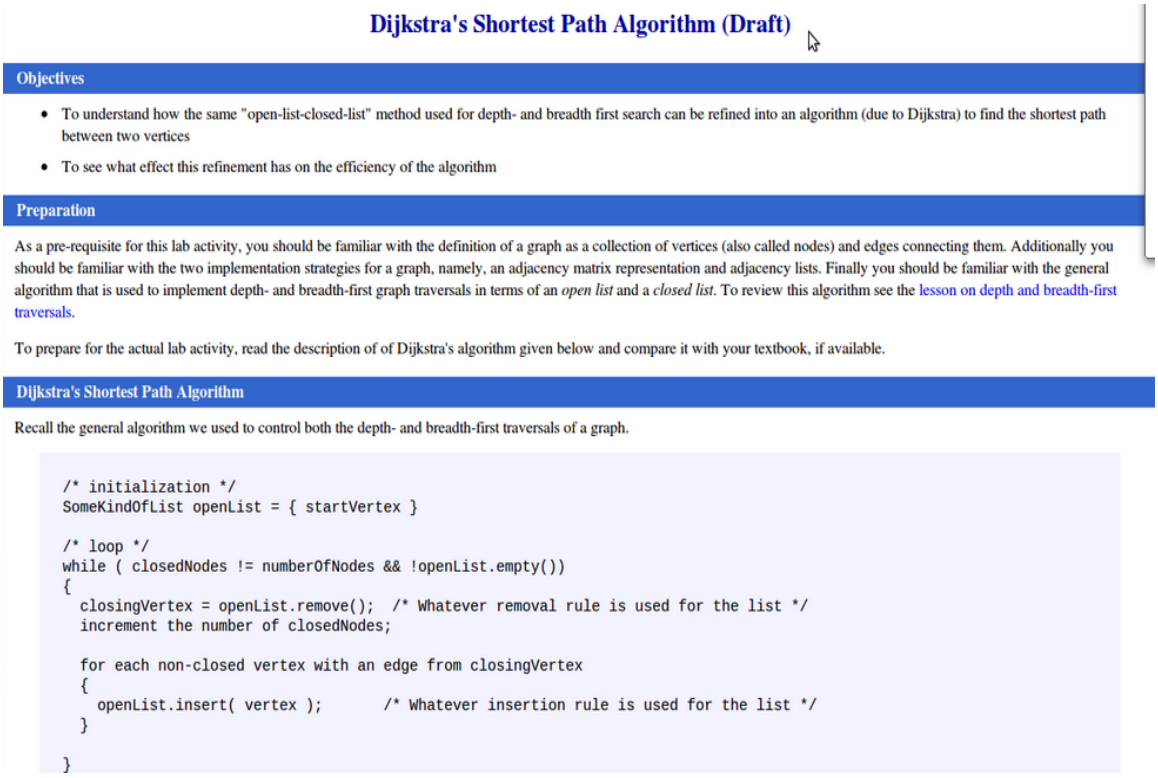
Figure 2.5: Snapshot of Animal system



- Animal [?]: They have a good collection of algorithms of different categories as shown in 2.5 which include backtracking, compression, cryptography, graph, searching & sorting, trees, hashing, etc. They also provide teacher to change the values and properties (like colour) of the animation (which they have called generator framework) which is good. System supports three languages with lots of control or settings like stuff which is impressive.

  The system is very good when student wants to learn a particular algorithm or is facing difficulty in that. But to teach an entire area of algorithm with this is not so engaging. User has control of the animation limited to its speed and zoom. User is just a passive learner.

Figure 2.6: Snapshot of Jhave system



Dijkstra's Shortest Path Algorithm (Draft)

Objectives

- To understand how the same "open-list-closed-list" method used for depth- and breadth first search can be refined into an algorithm (due to Dijkstra) to find the shortest path between two vertices
- To see what effect this refinement has on the efficiency of the algorithm

Preparation

As a pre-requisite for this lab activity, you should be familiar with the definition of a graph as a collection of vertices (also called nodes) and edges connecting them. Additionally you should be familiar with the two implementation strategies for a graph, namely, an adjacency matrix representation and adjacency lists. Finally you should be familiar with the general algorithm that is used to implement depth- and breadth-first graph traversals in terms of an *open list* and a *closed list*. To review this algorithm see the lesson on depth and breadth-first traversals.

To prepare for the actual lab activity, read the description of of Dijkstra's algorithm given below and compare it with your textbook, if available.

Dijkstra's Shortest Path Algorithm

Recall the general algorithm we used to control both the depth- and breadth-first traversals of a graph.

```
/* initialization */
SomeKindOfList openList = { startVertex }

/* loop */
while ( closedNodes != numberOfNodes && !openList.empty())
{
  closingVertex = openList.remove();  /* Whatever removal rule is used for the list */
  increment the number of closedNodes;

  for each non-closed vertex with an edge from closingVertex
  {
    openList.insert( vertex );       /* Whatever insertion rule is used for the list */
  }

}
```

- Jhave [**?**]: They are also content rich though not as Animal. They also have control and settings related stuffs. They have to tried to catch attention of the student with some pop-up questions which makes it different from others. Their system is very mature which has wiki and manuals. They have also developed scripting language for writing animations which basically is a series of snapshots.

  They also lack the feature learning-by-doing. Also some rich input dependent feedback to the answer given by student would make it better.

  There is also worth mentioning point here. Recently they have made JhavePop which is basically an EBNF parser used to animate the program given the source code in C++/Java for "linked list". It also shows a typical memory layout for such a program. We found this one of a kind unique feature in algorithm visualization community.It is shown in fig 2.6

- AlgoViz [?] : This is the most useful site for interested teachers and students in the field of algorithm visualization. Their site algoviz.org serves as a common gateway to algorithm visualizations scattered over the Internet.

- Others: There are also few US Universities who have hosted algorithm visualization content on their web-page.

2.2 Related Table

The following table is made in collaboration with Meenakshi [?] as it contains comparison of systems studied by both of us.

	System	Testing	Educational Theory	Strong Points	Weak Points	Supported Algorithms	Drill n Practice	Real-life example	Interactive
1	"AlgoTutor: AlgoTutor with Visual Algorithm Tracer and Program Pad embedded in it."	Tutor has been tested on the university students using the concept of pre/post test and results has been shown in the paper.	Concept of "operation blocks" has been used to design correct algorithm.	"1.It includes program pad so student can see how algo converts to program. 2. Its visualization tool helps student to debug code at each level. 3.Drag and drop interface is user friendly."	"1.Only for novice learners and does not include high level programs. 2.It is not open source."	basic level programs,not a particular algorithm support	not provided	not provided	No
2	JHAVE:Algorithm visualization system		Using animation to visualize algorithm	"1.Theory embedded. 2.quiz in between teaching."	Not interactive.	Many algorithms are supported including graphs,Sorting,Hashing and Miscellaneous.	not provided	not provided	NO
3	ANIMAL		Using animation to visualize algorithm	1.Animation provided with simultaneous code view.	Not interactive	All Major Algorithms provided	not provided	not provided	NO
4	www.algoviz.org-The Algorithm Visualization Portal		Animation of Algorithm and a portal where a collection of links to algorithm visualizations exists	Integrated many algo visualization systems.	Only animation is included. Not own any system.	Graphs, Sorting, Greedy, Compression, Numerical	Not provided	Not provided	No

Table 2.1: Comparison of Algorithm Tutor Systems

	System	Testing	Educational Theory	Strong Points	Weak Points	Supported Algorithms	Drill n Practice	Real-life example	Interactive
5	GREED-EX	participants were computer science majors enrolled at our university in a second-year mandatory course on design and analysis of algorithms,Results were better in experiment group in post test rather in pre-test.	Discovery learning approach	Student discover by himself,results table and history is provided to compare different approaches	No drill and practice included,not checking whether student has understood or not,he may end up with wrong approach in mind.	Activity Selection, Knapsack.	Not provided	Not provided	NO
6	http://www.-cs.usfca.edu/-galles/-visualization/	not available	Animation of algorithms	Visualization of Algorithm is provided	No examples are included,cannot change the desired input,	Graphs, Sorting, Greedy, Compression ,Numerical	No	No	NO
7	GAT-utor	A Pilot test has been conducted,results showed some improvement over the system along with some good comments	Guided Discovery Learning	Students can play with algorithm, They are guided if they are wrong, Proof of contradicting conditions are provided	Limited to greedy algorithms	Graphs, Scheduling	Test has been provided	Uses real life examples	Involvement of student 100 percent of time

Table 2.2: Comparison of Algorithm Tutor Systems

Chapter 3

Design of GATutor-Greedy Algorithms Tutor

3.1 Introduction

GATutor is an Greedy Algorithm tutoring system based on a rule based framework worked out by us. Every algorithm follows the same flow of steps to make students learn algorithms. It is a interactive system as it demands students intervention at every point.

3.2 Teaching of Greedy Algorithms

3.2.1 Teaching Strategies of other systems

Many algorithms tutor exist but some mainly focus on theoretical material(JHAVE) while other mainly focuses on visualization of the algorithm(ANIMAL).1There does not exist any system which checks learning of the student at each phase. The GreedEx system focuses on Discovery learning and uses the concept of discovering optimal selection function by experimentation approach but other learning goals have not been taught such as to give the proof of wrong choice.

Mainly systems focuses on animation of the greedy algorithms(algoviz). There exist no system which focuses on interactive learning of greedy algorithms and help in giving the proof of wrong selection functions. System also implement only understand and analyze level of greedy algorithm.

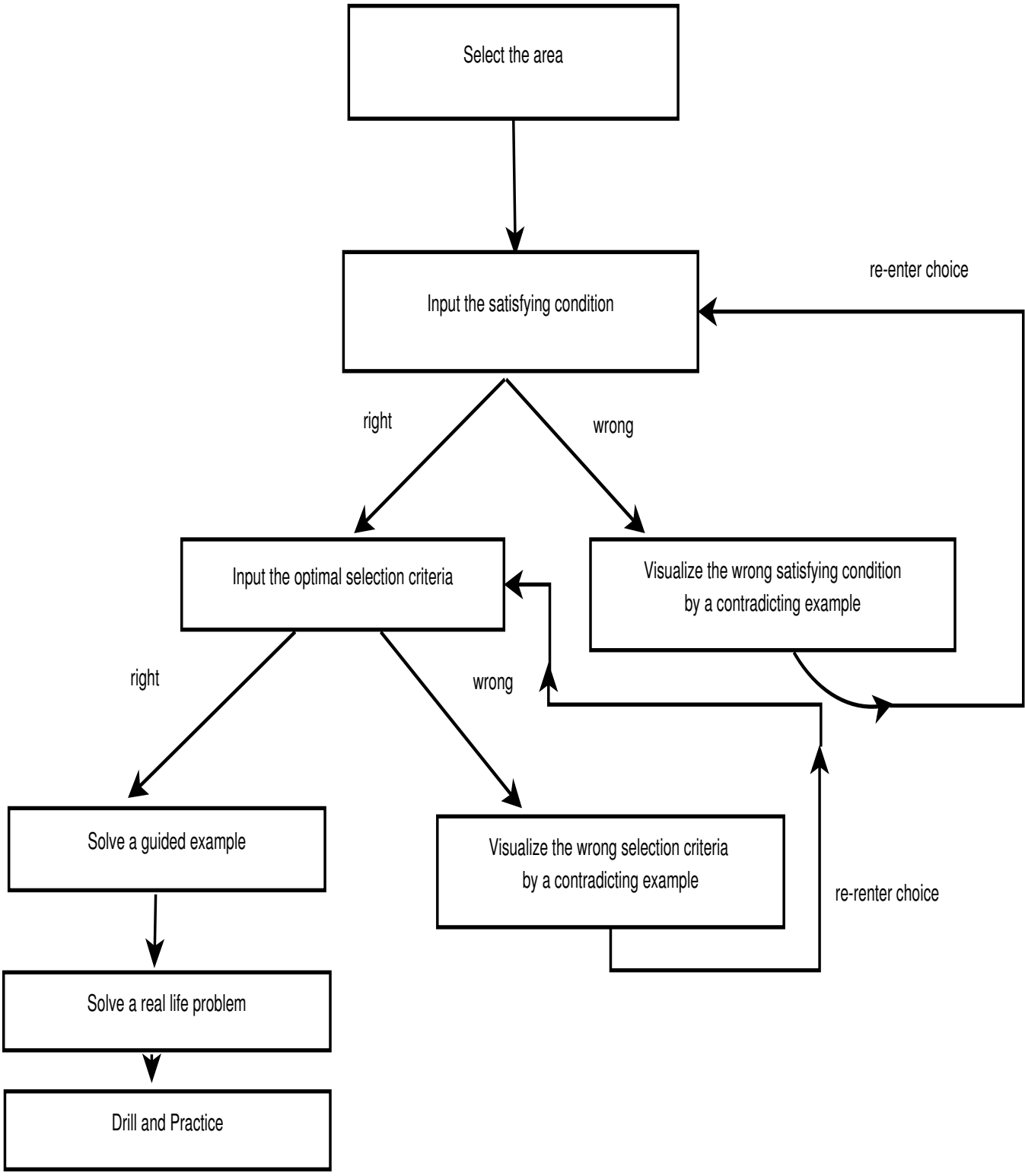
3.2.2 Teaching of Greedy Algorithms-Our Strategy

0

We have developed a rule based framework for teaching greedy algorithms. All greedy algorithms will follow the same set of rules. Our system follow recall, understand, apply, analyze and evaluate level of Bloom's taxonomy as compared with [**?**].

This is explained in more detail in Meenakshi's thesis [**?**]
The framework is shown below:

Figure 3.1: Framework for Building System



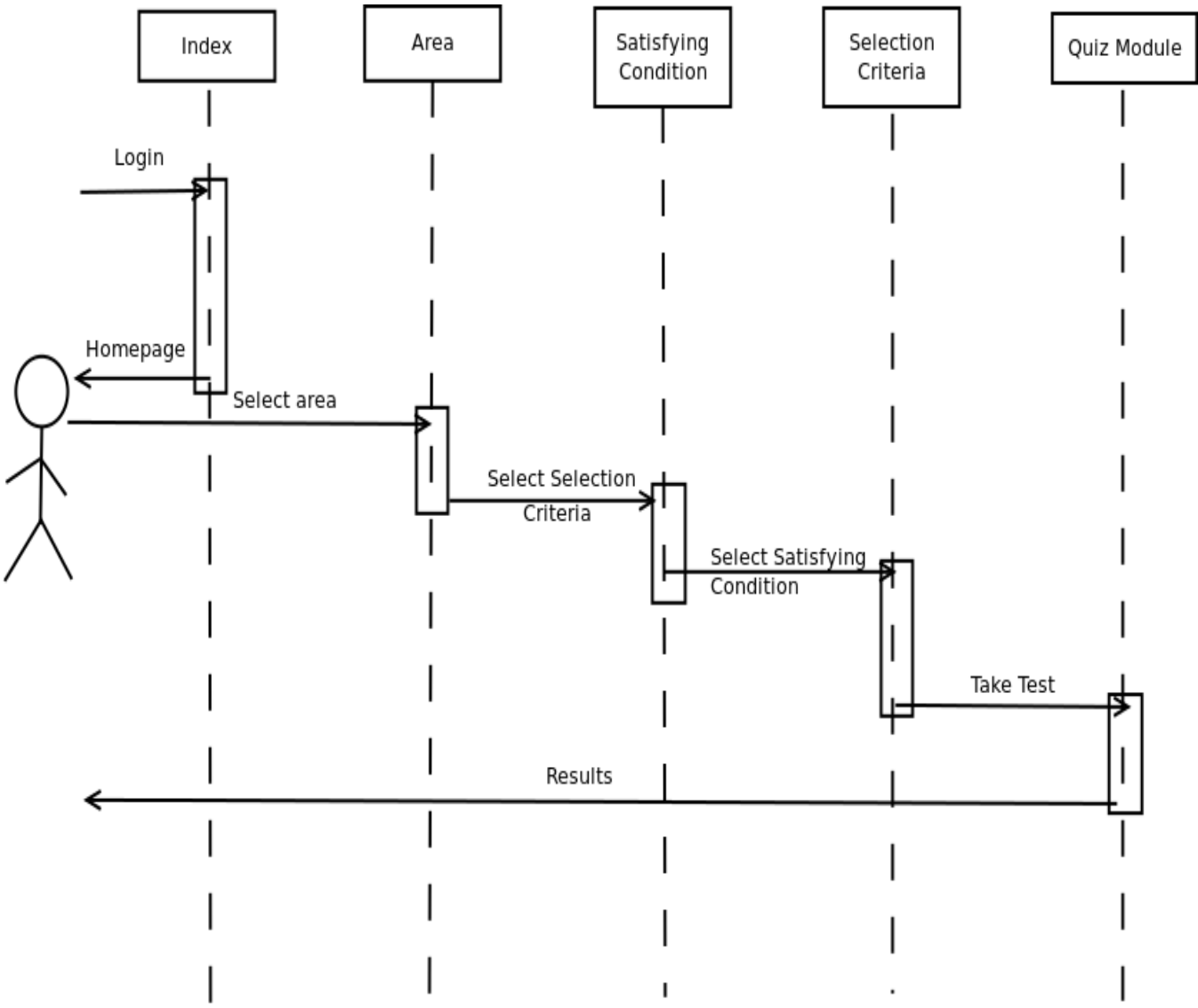
3.3 Learner's perspective

Learners can easily use this system as it is a learning object, so they will not have to aware about any specific technical knowledge to use this system apart from basics. Learners of this system will be basically 2nd year Computer Science Engineering students. Steps followed by learners are shown in this sequence diagram. They can judge their performance by taking the quiz.As the system is interactive at every stage,they will not get bore neither their mind will divert as without their interaction system will not move or produce the output.

Users are basically the instructors in various institutes. Users are also the one who can contribute to the content for the system. Users can also become developers if they have the knowledge of the above mentioned programming languages and can contribute to the content by doing a research on the particular algorithm which they want to insert.

The responsibility of user is to hand on this software to the students and let them study the particular algorithm by their own.

Figure 3.2: Sequence diagram of User Activities



3.4 Developer's perspective

The system has been developed in javascript,html and jquery.Being a developer of the system you will require all these skills. It took time to implement first algorithm in this system but later for other algorithms the time taken was less as we have the framework prepared for us. The system was divided into modules containing

- Cover different areas
- Capture different algorithms of a particular area
- A real life example
- Modules for proving wrong choice
- Modules for right choice
- Quiz module

Developing the content for a area was also a major task. To create example for every wrong choice included some research in that particular algorithm. To build the wrong choices which could be provided by the user also required a study. The various misconceptions a student can have was thought for.

3.5 User Interface

User Interface is the main part of any software system, it includes various parameters. It allows a user to interact with the system. It can irritate a user if it is not good and can prevent the user from using the system.

The theme color of the system is a main factor as it helps to attract the attention. We have used green color as it signifies correctness, we ignored blue color as many system are built in blue color so sometimes it becomes boring as it signifies material related to studies.

We have used click-able images everywhere so that user can play with the input and apply his algorithm to compare the results.

Buttons are provided at every step to jump to next stage. Tabs of various areas have been given so that user can jump to any area.

Chapter 4

Detailed Design

4.1 Introduction to system

The system is an Intelligent Tutoring System(ITS). An ITS is a system which teaches students according to their understanding and provides customized feedback. Various technologies have been used to build the system viz jsp, html, css, javascript, jquery, mysql. [**?**]There are many design decisions which we try to defend here-

4.1.1 Technology

All of our website technologies are supported by desktop browsers and current versions of Android browser, Blackberry browser, IE browser, iOS Safari and Opera Mini and others. Like some other systems no preference settings are required. Some browsers might ask you to allow to run javascript (if that feature is turned on), you allow it and that is all what is required.

4.1.2 Login

First page of the system is index.xhtml which is a login page. Student login is made for the purpose of tracking their progress. New user can also be enrolled. We know while logging-in, user credentials are checked from the database server. This is done in a straight-forward way as-
The user input is collected in a form in index.xhtml which is sent to java class at the server CheckUser. CheckUser does three things-
CheckUser connects to the database table and checks for a row with given username and password. Then it redirects to according pages.
If while logging-in user ticked the save password box, CheckUser also sets a cookie to be stored in user's browser.
It also sets a session attribute with user's username.
We have ensured not to make the input given by the user a direct part of the sql query for authentication, thus curbing a possible sql injection to an extent.
The password can be saved in the browser in the form of a cookie.

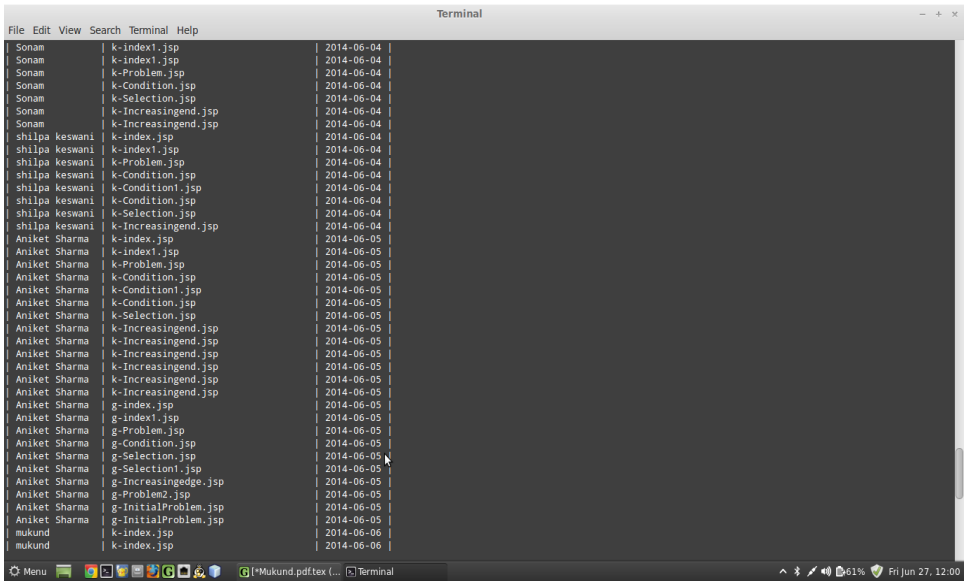
4.1.3 Tracking of student and its Analysis by Instructor

It is to be noted that student's browsing is being tracked and stored in a log table in database. This is implemented by first making a connection to the database and then page gathers session Attribute in which the username of the user was stored at the time of logging-in by the user. All of these information along with the date of access of this page is stored in the database; the code for which is-

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>

<%


String url = "jdbc:mysql://localhost:3306/";
String dbName = "gatutor";
String driver = "com.mysql.jdbc.Driver";
String userName = "sec_user";
String password = "wiki";
try {
Class.forName(driver);
Connection con = DriverManager.getConnection(url + dbName,
userName, password);
String pageID = request.getRequestURL().toString();
pageID = pageID.substring(pageID.lastIndexOf('/')+1);
String userID = (String) session.getAttribute("userName");
java.sql.Timestamp date = new java.sql.Timestamp( new java.util.Date().getTime());
String query = "insert into log (userid, page, access) " +
       "VALUES( \'"+ userID +"\', \'"+ pageID +"\', ?);";
PreparedStatement ps = con.prepareStatement(query);
ps.setTimestamp(1, date);
out.println(query);
ps.executeUpdate();
} catch (Exception ex) {
//ex.printStackTrace();
out.println( ex.getMessage());
}

%>
```

The screenshot of how this is stored in the database is in 4.1. This serves as a raw data of information for the instructor. By different sql queries we can extract meaningful information out of that. Three instructor pages have been created as shown in fig 4.2, 4.3, 4.4. Page-1 in fig 4.2 shows that how many students from the class registered themselves, how many attempted and how

Figure 4.1: Log table in Mysql



many actually completed the entire algorithm. It also gives details of name and email of registered students. We can know detailed information about a student by entering her username in textbox provided at the bottom of this page.

Page-2 shown in fig 4.3 tells the instructor how many areas are completed by this student and what was her browsing flow through the system. The names of the web pages are given in close resemblance of the context in it. So the instructor is able to see what answers this student gave to different questions asked by the system.
sql query for no. of areas student attempted is "select count(*) from (select distinct case when page like 'k%' then 'k' when page like 'g%' then 'g' when page like 's%' then 's' when page like 'd%' then 'd' end from log where userid=?) x;"

Page-3 shown in fig 4.4 tells the instructor how many times a particular page was visited (by the class). Instructor knows by the name of the page the contents of it as the names are given judiciously. Thus he is able to see how many students gave a particular answer to a question, what was the most popular answer to a question. This tells the instructor what is the common understanding of the class and if many students give a particular wrong answer to a question then instructor knows what is the misconception to be addressed in next classes.
Sql query for this is "select page, count(userid) as visited from log group by page;"

Figure 4.2: Class Overall Stats

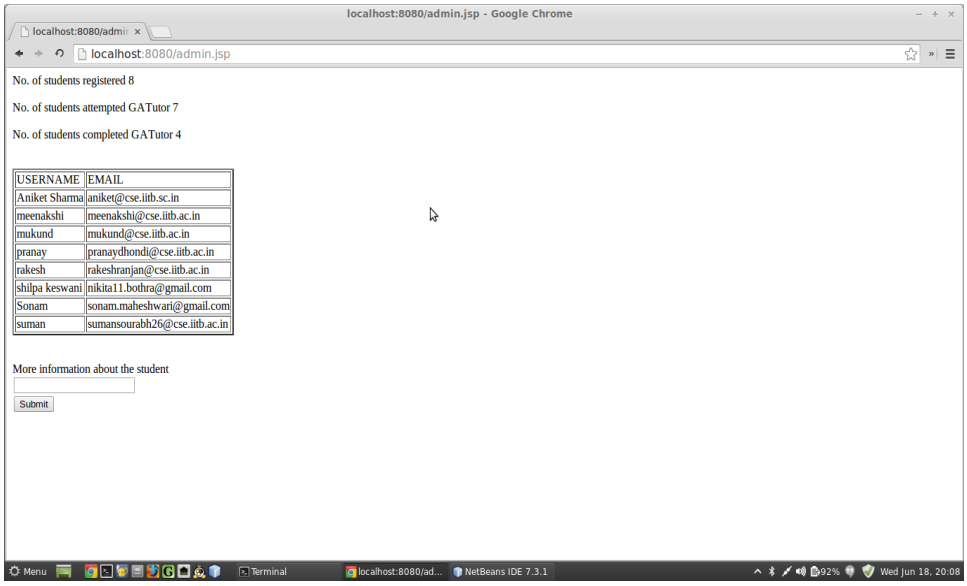


Figure 4.3: Individual Stats

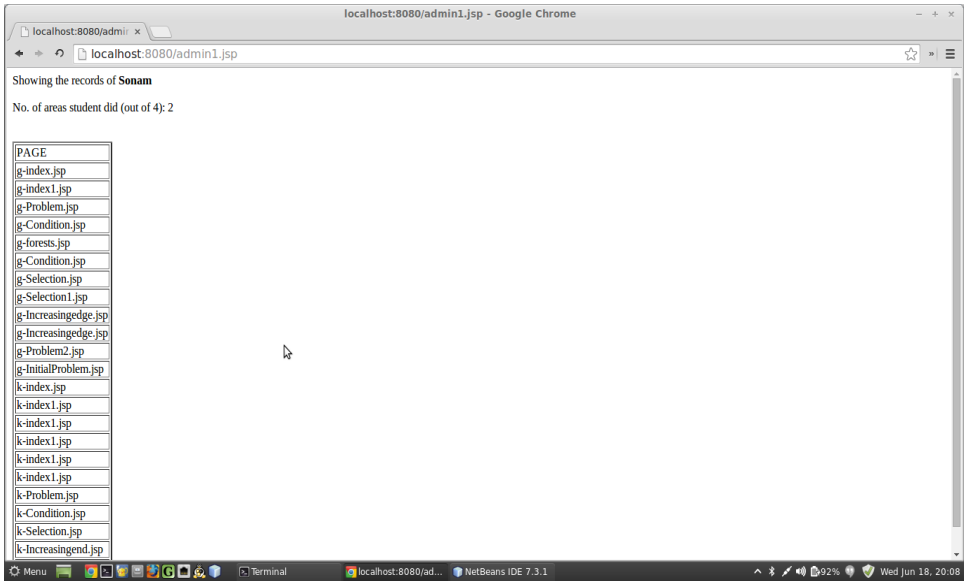
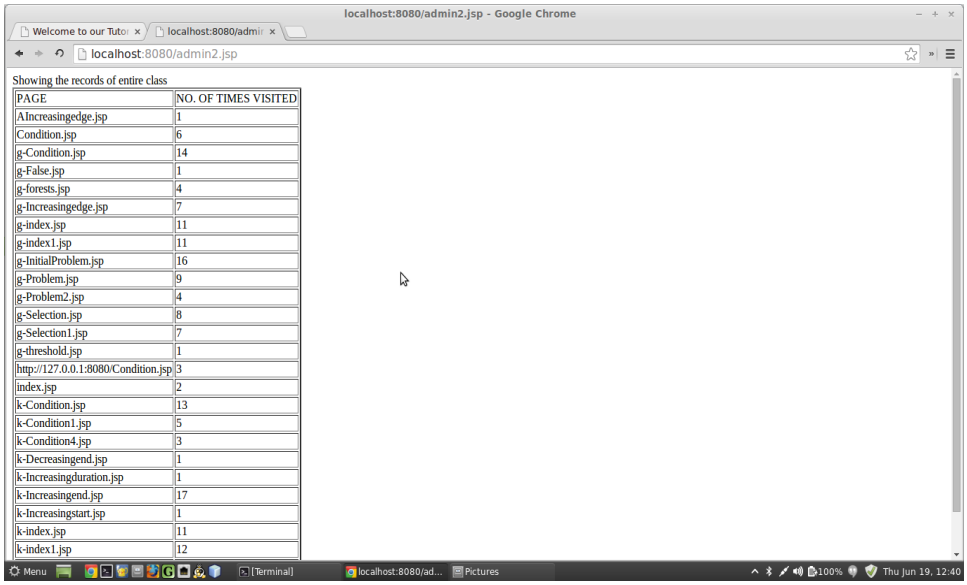


Figure 4.4: Class Misconception



4.1.4 Usage of system by Instructor

Though the system is designed keeping in mind to bypass the need of an elementary lecture, still it can be used by the instructor anytime before or after the class. If used before the class the lecture can start with a different level and also the instructor now has an idea of misconceptions in a class. After class usage of the system makes students actually appreciate the beauty of the algorithm and serve as a practice for further strengthening of concepts. Quiz module can be given as a homework too.

4.1.5 Dynamic web-pages

All the Scalable Vector Graphs are made click-able wherever possible. To come to an answer user has to click on appropriate parts of the graph and play with it. Thus user is not just sitting and watching some "movie" but is actually doing the things. Thus our system is interactive and demands attention. Also with this the study material becomes less boring.
Svg is a lightweight format and its various element's attributes are changed dynamically by the user which is supported by the javascript running in the browser. A code snippet from Knapsack page is-

```
var sum = 0;
var flag = 0;
var MAX = 5;
var i = 0;
function modify(e)
{
var t = e.target;
if(t=="[object SVGRectElement]")
{
if(flag==0){che_call();flag=1;}
if(t.getAttribute("fill")=="#FF0000")
{
sum+=33;
if(i>=MAX){sum-=33;alert("You are exceeding the holding capacity of knapsack");}
document.getElementById("box").textContent=sum+" Rs";
document.getElementById(t.id).style.visibility="hidden";
i+=1;
}
if(t.getAttribute("fill")=="#0000bf")
{
sum+=25;
if(i>=MAX){sum-=25;alert("You are exceeding the holding capacity of knapsack");}
document.getElementById("box").textContent=sum+" Rs";
document.getElementById(t.id).style.visibility="hidden";
i+=1;
```

```
}
if(t.getAttribute("fill")=="#ff7000")
{
sum+=10;
if(i>=MAX){sum-=10;alert("You are exceeding the holding capacity of knapsack");}
document.getElementById("box").textContent=sum+" Rs";
document.getElementById(t.id).style.visibility="hidden";
i+=1;
}
if(t.getAttribute("fill")=="#007f00")
{
sum+=40;
if(i>=MAX){sum-=40;alert("You are exceeding the holding capacity of knapsack");}
document.getElementById("box").textContent=sum+" Rs";
document.getElementById(t.id).style.visibility="hidden";
i+=1;
}
}
}
document.documentElement.addEventListener('click',function(e){modify(e);},false);
```

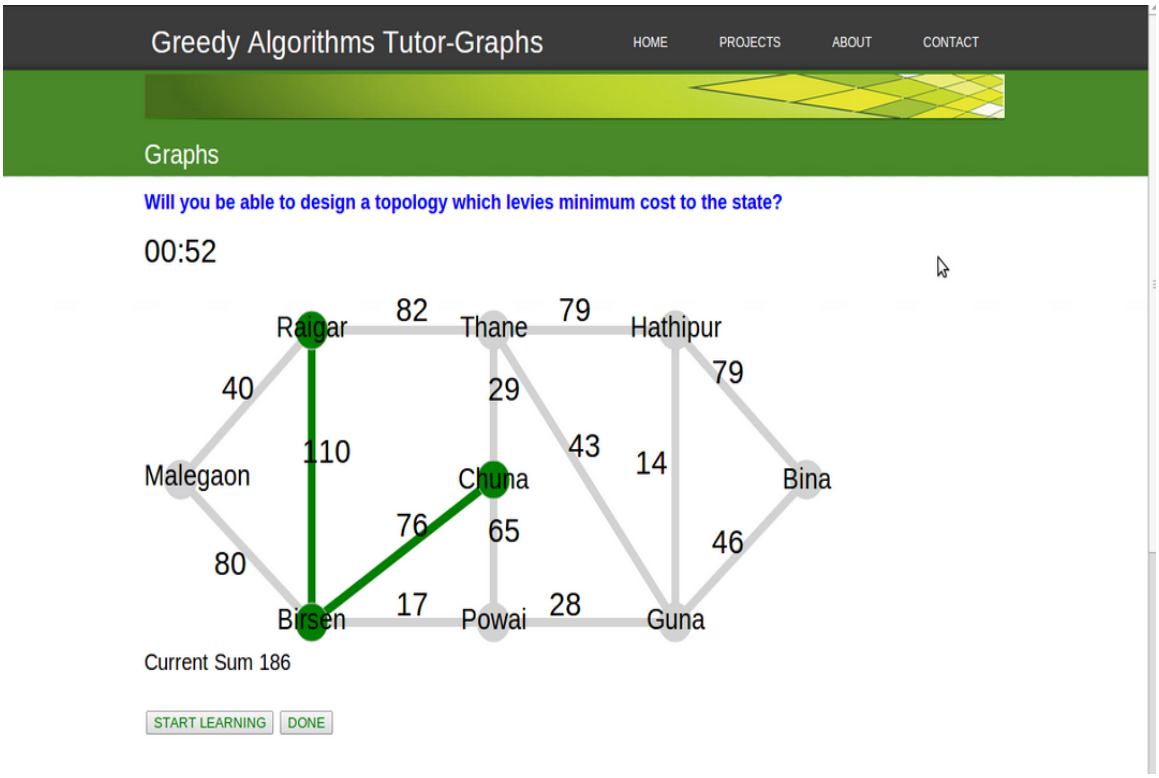
4.1.6 Real life puzzle with time limit

The starting of each algorithm teaching is with a real life puzzle or challenge. The real life feature makes the student interest and motivated to learn the algorithm. Puzzle comes with a timer only to show the student that how the algorithms make life easier and faster as shown in fig 4.5 There is also a counterpart that student may not feel good for not able to solve something and thereby demotivating him at the beginning itself but we stick our design only which got proved by conducting experimentation. Code snippet for this page is

```
var dict = {"AB":40,"AC":80,"BC":110,"BD":82,"CI":76,"CE":17,"DI":29,"IE":65,
"DG":43,"EG":28,"DF":79,"FG":14,"FH":79,"GH":46};
var dict1 = {"A":0,"B":0,"C":0,"D":0,"E":0,"F":0,"G":0,"H":0,"I":0}
var sum=0;
var flag = 0;
var modify = function (e){
var t = e.target;
if(t=="[object SVGPathElement]")
if(t.getAttribute("stroke")!="green")
{
if(flag==0){che_call();flag=1;}
sum = sum + dict[t.id];
```

```
t.setAttribute("stroke","green");
document.getElementById(t.id[0]).setAttribute("fill","green");
document.getElementById(t.id[1]).setAttribute("fill","green");
dict1[t.id[0]]+=1;
dict1[t.id[1]]+=1;
document.getElementById("val").innerHTML="Current Sum "+sum;
}
else
{
t.setAttribute("stroke","lightgrey");
if(dict1[t.id[0]]==1)
document.getElementById(t.id[0]).setAttribute("fill","lightgrey");
else if(dict1[t.id[0]] > 1) dict1[t.id[0]]-=1;
if(dict1[t.id[1]]==1)
document.getElementById(t.id[1]).setAttribute("fill","lightgrey");
else if(dict1[t.id[1]] > 1) dict1[t.id[1]]-=1;
sum -= dict[t.id];
document.getElementById("val").innerHTML="Current Sum "+sum;
}
}
document.documentElement.addEventListener('click',modify,false);
function che_call()
{
var ti = setInterval(function(){che()},1000);
var sec=59;
function che(){
document.getElementById("demo").innerHTML="00:"+sec;
sec=sec-1;
if(sec<0){
clearInterval(ti);
document.documentElement.removeEventListener('click',modify,false);}
}
}
```

Figure 4.5: Real life puzzle with time limit



4.1.7 Reasons for choosing SVG

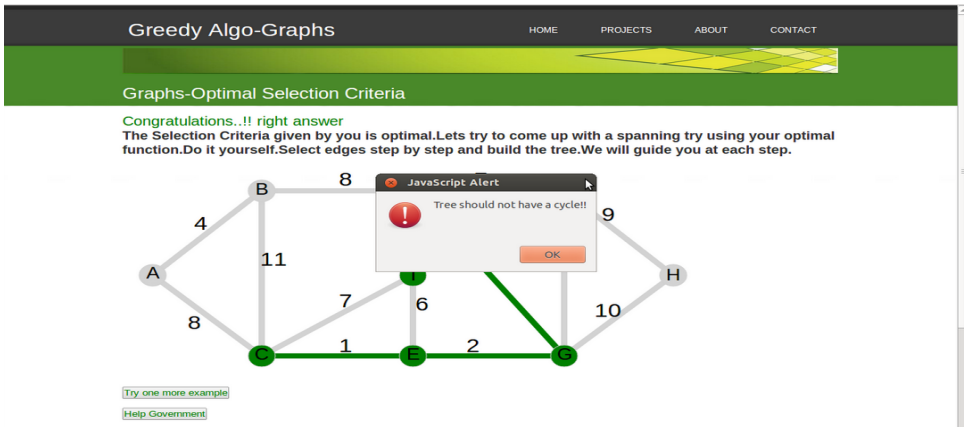
All the puzzle drawings and graphs were made using SVG (Scalable Vector Graphics). It has potential advantages-
1.It is a vector technology i.e. not a raster technology. This is one of the reasons why they are scalable. When we want to zoom in on such a vector image there is no distortion because the system is mathematical whereas in raster images we expose ourselves to the little tiny dots.
2.SVG is XML based text code and can be used within other language formats to polish it.
3.SVG can be easily edited precisely as it is based on mathematical coordinates.
4.SVG images are lightweight and this becomes more pronounced when we go for heavier images.
5.SVG images are composed of elements that can be easily accessed by scripting languages such as javascript like XML or html.
6.For other graph formats there is a ¡img¿ tag which sends another HttpRequest SVG doesn't send any other request.

4.1.8 Customized feedback

Since in our system all the way user is learning things by doing hence it is very possible that he commits some mistakes. Mistakes can be trivial or conceptual. System is intelligent enough to know student's mistake. Dynamic feedback is produced whenever he is wrong and exactly "what part" of the concept is wrong or what trivial mistake is incurred as shown in 4.6. This timely feedback prevents the student from committing further mistakes and keeps him on track.
Suppose in graphs topics if student does not selects edges according to the criterion he is given feedback.
If not edges to be selected are selected appropriate feedback is given. If while forming a spanning tree, a student tries to form a cycle he is given according feedback. A code snippet for detecting cycle is-

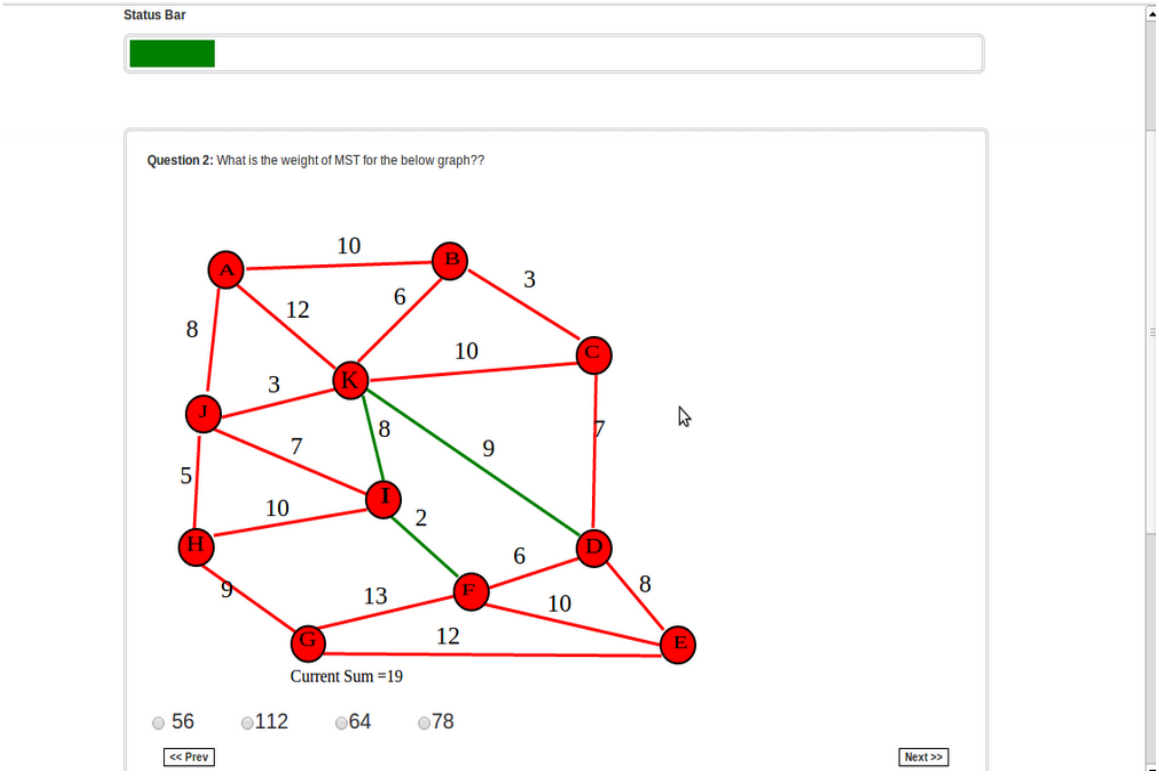
```
var parent = {"A":"A","B":"B","C":"C","D":"D","E":"E","F":"F","G":"G",
"H":"H","I":"I"};
var a,x,y;
function find(a)
{
if(parent[a]==a)return a;
return find(parent[a]);
}
function union(x,y)
{
    parent[ find(x) ] = find(y);
```

Figure 4.6: Feedback as a alert box




```
}
function check(e)
{
var t = e.target;
if(t=="[object SVGPathElement]")
if( find(t.id[0]) == find(t.id[1]) )return 1;
}
```

Figure 4.7: Quiz Page



4.1.9 Grading Scheme of quiz and progress bar

Grading scheme is such that marks awarded for a question depends on how many attempts it needed to solve that question. This is to ensure student's engagement as far as possible by not just signing off with saying that this question's answer is this. Progress bar increases whenever a question is answered and makes the user feel good as shown in 4.7.

4.1.10 Click-ability of diagrams and graphs by javascript

The graphs shown in the web-pages are click-able. On clicking different parts of the graphs they change. This is done as the graphs are avg, which is basically a XL based code, and javascript whiz is used for accessing XL elements. The reason for using javascript for this is that it is a client side code i.e. it is run on users' processors. This does not require extra bandwidth to send the data to server and fetch the response. This makes the code to run faster. From a user's perspective the system is highly interactive. A code snippet for this is-

```
var modify = function (e){
var t = e.target;
if(t=="[object SVGPathElement]")
if(t.getAttribute("stroke")!="green")
{
if(flag==0){che_call();flag=1;}
sum = sum + dict[t.id];
t.setAttribute("stroke","green");
document.getElementById(t.id[0]).setAttribute("fill","green");
document.getElementById(t.id[1]).setAttribute("fill","green");
dict1[t.id[0]]+=1;
dict1[t.id[1]]+=1;
document.getElementById("val").innerHTML="Current Sum "+sum;
}
else
{
t.setAttribute("stroke","lightgrey");
if(dict1[t.id[0]]==1)document.getElementById(t.id
[0]).setAttribute("fill","lightgrey");
else if(dict1[t.id[0]] > 1) dict1[t.id[0]]-=1;
if(dict1[t.id[1]]==1)document.getElementById(t.id
[1]).setAttribute("fill","lightgrey");
else if(dict1[t.id[1]] > 1) dict1[t.id[1]]-=1;
sum -= dict[t.id];
document.getElementById("val").innerHTML="Current Sum "+sum;
}
}
document.documentElement.addEventListener('click',modify,false);
```

Chapter 5

Experiment

5.1 Software Testing

A regressive testing has been conducted on the system to ensure that all link are working properly and the output provided at each stage is correct.

5.2 Learning and attractiveness

5.2.1 Implementation

We took 20 students, from B.tech 2nd year from IIT Bombay who were not knowing about this algorithm before.We asked them to fill a survey form to evaluate our system on different parameters.

5.2.2 Sample

Our sample is B.tech 2nd year students as design and analysis of algorithms is taught in 2nd year of B.tech.

5.2.3 Data Collection

-

The instruments of our data collection were

1. Survey
    - Test Score
    - Attempts
    - Open ended Feedback
    - 4 point likert scale
2. we interacted with student to know their general view about the system.

5.2.4 Data Analysis

The instruments of our data collection were

1. Survey
    - Test Score - is used to identify whether student has learned the algorithm or not.
    - Attempts - helpful to infer that in how many attempts was student able to learn and perform.
    - Open ended Feedback - we qualitatively analysed the text in consultation with education technology researchers to verify the quotes that we obtained from the qualitative analyses.
    - 4 point likert scale - We have used it to deduce inferences.
2. we interacted with student to know their general view about the system.

5.2.5 Results

Following are the inferences we have drawn by analysing the answer given by the students in the survey:

Learning

- Intelligent Tutoring System has been effective in learning as students were able to understand the algorithm as they answered and can be deduce from the test results.
- ITS was interesting as students find it to be fun working with it.
- It build up confidence in students to apply same algorithm in other example also.
- Student would like to study other topics also in these kind of ITS as it helps in fast learning, teaches more in less time, easy to learn and interesting.
- It increases the understanding of how to approach a problem.
- It was beneficial as it is good to learn new things from basics,better than class passive ways.
- Re-attempts embedded in the test has been proved useful as almost all have improved their score using that.
- **Average test score of the students is 135.67 out of 200 from whiz it can be concluded that the system effectively helped students to learn the algorithm.**

Attractiveness

Interface

- Interface is user friendly.

Content

- Content was crisp and attracting user's attention as well.
- Using it, understanding was developed in gradual manner giving answers to all the questions in mind.

The improvements suggested to us are:

- The arrows were taking time to click as they were thin.
- More theory should be embedded.
- In Quiz,option to jump to other questions should be provided.

download from net-datatable

Table 5.1: Learning and Attractiveness Evaluation

Question Asked	**Student-1**	**Student-2**	**Student-3**	**Student-4**
How many times you attempted the test?	2	2	2	3
What was your test score at first attempt?	180	150	170	130
What was your final score of the test?	200	175	190	180
(1)After Working with Computer Based Tutorial I have improved my understanding of the topic(Kruskals Algorithm) I studied.	4-Strongly Agree	2-Disagree	3-Agree	4-Strongly Agree
give reason for your answer(1)	Interface was very student family	i was in a hurry and the questions weren't as such clear from the graphs. I personally do not feel it to be of much use to put the main question ahead of theory. I feel the way the theory is taught conventionally needs to change. The interactiveness feature can be added in that.	Got the logic to solve such problem in an easy way	It started from the basics
(2)I would like to study more about Kruskal's algorithm and other related algorithm.	3-Agree	3-Agree	3-Agree	3-Agree
give reason for your answer(2)	Knowledge is never ending :-P	I always found graphs interesting	Improve more on general logical analysis	Improve more on algorithms
(3)I found studying on the CBT interesting.	4-Strongly Agree	3-Agree	4-Strongly Agree	4-Strongly Agree
give reason for your answer(3)	Fun, fast and my learning was fast	CBT can provide flexibility which conventional methods can't. The theory, the problems the order can all be modified by the user based on his comfort.	Easy and user friendly.	Interesting,not much efforts required.

Table 5.2: Learning and Attractiveness Evaluation

Question Asked	**Student-1**	**Student-2**	**Student-3**	**Student-4**
(4)(1)After Working with Computer Based Tutorial I have improved my understanding of the topic I studied.	1-Strongly Disagree	3-Agree	4-Strongly Agree	4-Strongly Agree
give reason for your answer(4)	Feeling confident now	Seriously? :p	Sufficient to understand the basics.	CBT taught it..
(3)I am confident that i can correctly apply the studied greedy algorithms on other similar examples.	3-Agree	3-Agree	4-Strongly Agree	3-Agree
2-CBT has increased my interest in studying	Some may not work as well, but this did splendidly.	Again, see flexibility	Teaches more in less time.	Interesting
(6)I would like to other courses on same kind of CBT?.	3-Agree	3-Agree	4-Strongly Agree	4-Strongly Agree
How did the user interface of the CBT helped you to learn better?	interactive stuff, could see results right away.	it helps in visual representation of algorithm and solving examples always helps in better learning	"1. Animations helped in interacting and learning efficiently 2. It was easy to use."	it was demanding user interference which i liked the most
How did the content of the CBT helped you to learn better?	Content was crisp and it helping in getting the attention well.	Content needs to be more! Much much more! More explaining in written is also required. Just illustrations won't do.	Easy to comprehend and understanding is developed in a gradual manner giving answers to all the questions in my mind.	It is easy to study from this,but it require more content and more algorithms to be embedded.

Table 5.3: Learning and Attractiveness Evaluation

How did you feel CBT to be frustrating-(Disadvantageous)?	The arrows took time to get selected. Also no de-select option. Please give a little more importance to shifting from one question to another quickly.	The current structure is forcing you to follow a predefined order of taking things. That should again be customizable. Users based on their comfort should be able to choose whether to see theory first and then do problems or to do problems first and then see theory! ;)	Not thrilling and exciting though very useful The color combination in he graph is too distracting. The red color distracts too much while solving problems. The edges should be more thick and sensitive. It require more number of clicks to click on an edge.	The problems are largely similar and it's frustrating to solve so many similar questions. Some of the answers are wrong. And the UI is not much good.
How did you feel CBT to be beneficial?	Very beneficial.	interactive tests is good, should be added more.	CBT is good to learn new things.	In any case , it is better than studying in class.

5.3 Usability evaluation

Details of usability evaluation has been explained in Meenakshi's Thesis [**?**]

Chapter 6

Conclusion and Future Work

Ours is a novel approach in the teaching of algorithms. We have started with Greedy Algorithms. The system is very interactive and teaches the user the entire algorithm through a guided discovery learning approach. The system can be given to the class by instructors also to get to know where the class stands in terms of the understanding of the algorithm as the system provides statistics by tracking every user. The system was also tested on a bunch of students to prove its efficacy.
In the future other algorithms can also be added. Also the statistics and misconceptions of the class can be shown to the instructor in nice graphical forms instead of a tabular form. Entire web pages can be restyled so that minimum scrolling is required. A progress bar can be added for students showing how much learning they have completed and reward points could be given for the same based on educational theories. An option for the student to be able to see what percentage of students gave what answer to a question can be added after researching whether this would be useful or not. Despite all these future work system is in a ready to be implemented state.