REPORT OF PROFESSORASSIST APP

FINAL REPORT OF CIS 152

Liang(Rebecca) Tang

The picture is adapted from https://www.dreamstime.com/illustration/professor-computer.html

TABLE OF CONTENTS

CHAPTER ONE. FORMALIZED PROPOSAL

Section one. Why is the ProfessorAssist App needed?

Being professors are well respected jobs since they shape future generations and serve as expert in their fields and disciplines. Professors usually face pressures everyday since they manage multiple teaching and research assignments simultaneously. In order to improve efficiency and guarantee all the assignments done by the deadline, professors need assistance to manage the work assignments. For example, Dr. Shoemaker is a professor in the department of Marketing at the University of Iowa. In his scope of work in the academic year of 2021-2022, he focuses on four grant-support projects, write four journal papers and four conference proceedings, teach four undergraduate courses and prepare four new undergraduate courses. Everyday when Dr. Shoemaker reviews his pending work in the plate, he needs to think carefully what sequence he would like to follow for the day's work. Based on the Theory of Planned Behavior in Social Psychology, it is highly possible that Dr. Shoemaker would choose one of the three modes: 1) choose one specific category (and subcategory) of work (e.g., research→grant-support projects in research→agriculture-related grant-→USDA grant project); 2) based on the urgency (i.e., priority) of the work, Dr. Shoemaker chooses the work with highest priority since the deadline is coming or it is extremely important; 3) like a stack of books, the previous work is put at the bottom, the newest work is located on the top. Dr. Shoemaker just grabs the newest work on the top of the work stack and then move down to the oldest work.

From the descriptions above, the professor really needs assistance in the complicated multi-task management. The ProfessorAssist App could help Dr. Shoemaker to achieve the goals and make choice of any of the three work styles aforementioned. In the following section, I would explain how the detailed assignments are managed and selected in the daily schedule of Dr. Shoemaker.

Section two. Binary Search Decision Tree

Dr. Shoemaker walks in the office in the morning and wants to decide the sequence of work he needs to do today. Today he has a preference on a specific category of work (i.e., teaching or research), and then he would like to further make choices of subcategory of work (i.e., grant-support projects in research) and "sub-sub-category" of work (i.e. agriculture-related grant), and even further (i.e., USDA grant). Binary search decision tree guides Dr. Shoemaker to keep making the decision of what work he would like to do until he finds his task today. Figure 1 shows the scenario of binary search decision tree used in the project.

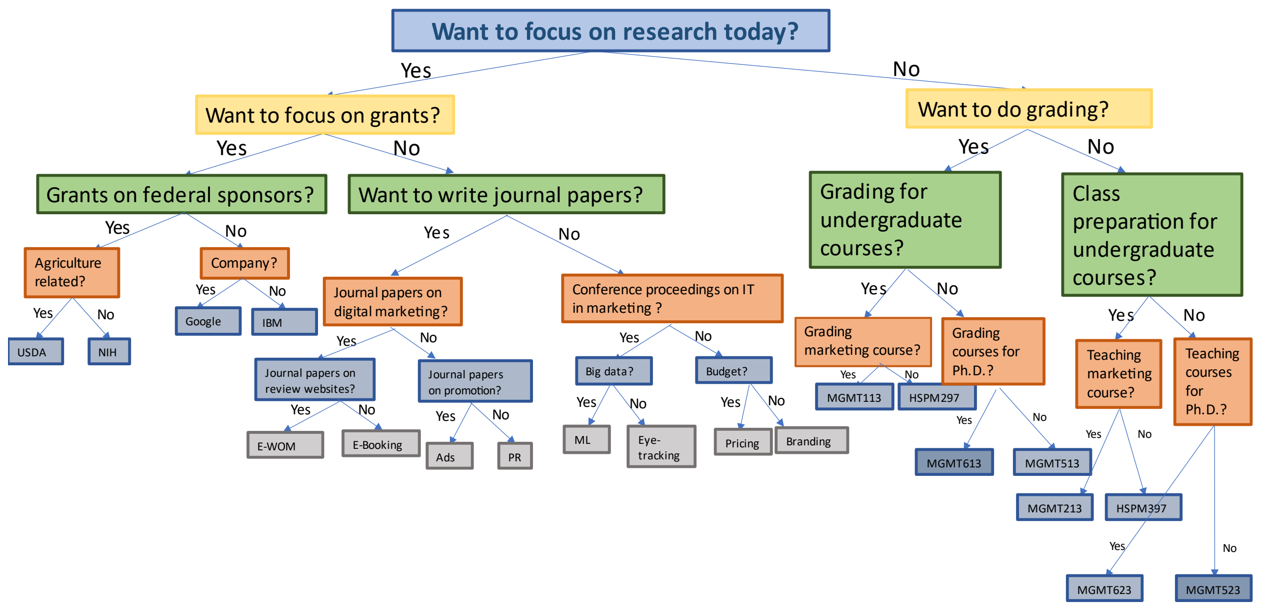
Want to focus on research today?
Yes
No
Want to focus on grants?
Want to do grading?
Yes
No
Yes
No
Grants on federal sponsors?
Want to write journal papers?
Grading for undergraduate courses?
Class preparation for undergraduate courses?
Yes
No
Yes
No
Agriculture related?
Company?
Journal papers on digital marketing?
Conference proceedings on IT in marketing ?
Yes
No
Yes
No
Grading marketing course?
Grading courses for Ph.D.?
Yes
No
Teaching marketing course?
Teaching courses for Ph.D.?
USDA
NIH
Google
IBM
Journal papers on review websites?
Journal papers on promotion?
Big data?
Budget?
MGMT113
HSPM297
E-WOM
E-Booking
Ads
PR
ML
Eye-tracking
Pricing
Branding
MGMT613
MGMT513
MGMT213
HSPM397
MGMT623
MGMT523

Figure 1. Dr. Shoemaker's Binary Search Decision Tree

Section Three. Stack of Work

As shown in Figure 1, the lowest level of the tree demonstrates the detailed work content (e.g., USDA, E-WOM). The detailed work content is pushed into the stack of work when Dr. Shoemaker just receives it. For example, when Dr. Shoemaker received the USDA grant on Feb 12, 2016, he pushed it in the stack. When Dr. Shoemaker got the notification from the department head on Jan 11, 2022 that he would teach MGMT 213 starting on Fall 2023, he pushed the preparation of MGMT 213 into the work stack on the same day. The following table is a demo that when the individual work assignment was pushed into the work stack. Since the purpose of the project is to demonstrate the use of stack rather than sorting by dates, in coding part, I already pushed them in the stack (a new assignment is pushed via GUI in the domo). In real life, Dr. Shoemaker pushes a new assignment when he receives the new task.

Table 1. Work Assignment and their Adding Date

Work Assignment	**Adding Date**
USDA grant	11/26/2015
NIH grant	2/34/2017
Google grant	4/11/2019
IBM grant	2/25/2020
E-WOM journal paper	4/20/2020
E-booking journal paper	9/12/2020
Ads journal paper	3/22/2021

PR journal paper	4/11/2021
ML conference proceeding	5/23/2021
Eye-tracking conference proceeding	6/22/2021
Pricing conference proceeding	6/25/2021
Branding conference proceeding	7/01/2021
Grading MGMT113	8/12/2021
Grading MGMT297	8/23/2021
Grading MGMT613	9/12/2021
Grading MGMT513	9/22/2021
Preparing MGMT213	10/23/2021
Preparing MGMT397	11/22/2021
Preparing MGMT623	12/12/2021
Preparing MGMT523	3/2/2022

Section Four: Priority Queue with Bubble Sort

As shown in Figure 1, the lowest level of the tree demonstrates the detailed work content (e.g., USDA, E-WOM). The work assignments have different priority. The linkedlist with bubble sort is used to achieve the goal. Please note in real life, we expect that the professor input priority of a specific assignment when he/she pushes the work assignment into both stack and priority queue. For example, on April 11, 2021, Dr. Shoemaker started to prepare a PR journal paper. He pushes this assignment into both stack and priority queue with the priority level of 8. In the demo, these assignments are already in the queue (one more assignment is added to the queue).

Work Assignment	**Priority (1-20)**
USDA grant	17
NIH grant	14
Google grant	15
IBM grant	20
E-WOM journal paper	12
E-booking journal paper	18
Ads journal paper	19
PR journal paper	8
ML conference proceeding	16
Eye-tracking conference proceeding	9
Pricing conference proceeding	7
Branding conference proceeding	11
Grading MGMT113	13
Grading MGMT297	10
Grading MGMT613	3
Grading MGMT513	5
Preparing MGMT213	1
Preparing MGMT397	6
Preparing MGMT623	2
Preparing MGMT523	4

Chapter Two. Time/Change Logs

I kept the weekly log as shown in Table 3. Generally, I worked on the final project on Thursday and Friday every week since Mar 1, 2022. In week 1 (Mar 1-Mar 7), I focused on writing the proposal of final project. I decided the purpose of ProfessorAssist app and the data structure and sorting algorithm used in the project. In week 2 (Mar 8-Mar 14), I took a break to fresh my mind during Spring Break. In week 3 (Mar 15-Mar 21), I designed the details of Dr. Shoemaker's binary decision tree, drafted it with powerpoint, and wrote the rough code for it. In week 4 (Mar 22-28), I finished the rough code of stack. In week 5 (Mar 29-April 4), I wrote up the rough code of priority queue with bubble sort. In week 6 (April 5-April 11), since I haven't used GUI for a long time, I reviewed the content of GUI by reading textbook and checking online information. Accordingly, I prepared the rough code of GUI. In week 7 (April 12-April 18), I wrote up unit tests and merged all the components together into an integral product. In week 8 (April 19-April 25), I prepared the final report and powerpoint for the project. In the final three days, I wrapped up and submitted the project out.

Table 3. Weekly Log of Final Project

Tus	Weds	Thurs	Fri	Sat	Sun	Mon
Mar 1	Mar 2	Mar 3	Mar 4	Mar 5	Mar 6	Mar 7
Preparation of final project proposal						
Mar 8	Mar 9	Mar 10	Mar 11	Mar 12	Mar 13	Mar 14
Spring Break						
Mar 15	Mar 16	Mar 17	Mar 18	Mar 19	Mar 20	Mar 21
Write up the rough code of binary search decision tree						
Mar 22	Mar 23	Mar 24	Mar 25	Mar 26	Mar 27	Mar 28
Write up the rough code of stack						
Mar 29	Mar 30	Mar 31	April 1	April 2	April 3	April 4
Write up the rough code of priority queue						
April 5	April 6	April 7	April 8	April 9	April 10	April 11
Review GUI and prepare the rough code of GUI						
April 12	April 13	April 14	April 15	April 16	April 17	April 18
Write up Unit Tests and integrate all the code and GUI into one complete product						
April 19	April 20	April 21	April 22	April 23	April 24	April 25
Prepare final report and ppt						
April 26-April 29						
Wrap up						

Chapter Three. Lessons Learned

In my initial proposal, I indicated that Dr. Shoemaker mangers 10 projects ongoing. For each project, the status and priority are kept updating. Besides the regular teaching and research work, he also has many miscellaneous, such as writing reference letters, attending department faculty meetings, replying emails, etc. However, in the process of preparing the codes, I found that my goal is too big. My original proposal is not a final project of a class, but designing an app for commercial purpose. And many functions in my original blueprint can't be implemented with my knowledge by far. Therefore, I have kept shrinking down my goals of the project.

The second blocker is how to use the data structure/algorithm to solve the practical program. For example, how does Dr. Shoemaker decide what work assignment he wants to do? In what scenario does he input priority of individual work assignment? What content is included into GUI which demonstrates the I/O? I hope to solve a practical issue with the specific data structure/algorithm as the best solution, rather than stiffly create a practical program with the purpose of applying the data structure/algorithm. To solve the program, I have kept revising my proposed ideas, codes, and GUI simultaneously. I made multiple revisions of all these documents. Although it takes much time, the product that comes out makes more sense and is applicable and practical.

The third blocker is how to merge three pieces (i.e., binary search decision tree, stack, and priority queue) as an integral product. For example, the user input the content of an individual assignment (i.e., title, priority). The title is used for both the stack and priority queue. The priority is used for priority queue. The input is implemented with GUI for one time. So I need to share some input between the two data structure coding. My solution is to write the codes of three components separately first. And then when designing GUI, I have more clear idea of what the user needs to input, and what information is shared among the three components. And then I returned to revise the codes of three pieces and merged them together into an integral product.

Chapter Four. Code & Unit Tests

The link to my github repo is: https://github.com/LiangTang888/CIS152

The package of the final project includes 16 classes (five components):

Component 1. GUI

1) Action.java;
2) Controller.java;
3) Decision.java;
4) GUIApp.java;
5) Input.java;
6) Model.java;
7) ProfessorAssistFrame.java;
8) View.java;
9) WorkMethod.java;

Component 2. Decision tree

1) DecisionTree.java;
2) Node.java;

Component 3. Priority queue with bubble sort

1) LinkedListAndBubbleSort.java;

Component 4. Stack

1) StackImplement.java;

Component 5. JUNIT tests

1) DecisionTreeTest.java;
2) LinkedListAndBubbleSortTest.java;
3) StackImplementTest.java;

Chapter Five. User's Manual

Step 1. The following screen shows up. The purpose is to ask the user to select one of three choice-making options.

Good morning, Dr. Shoemaker!

How do you want me to help you choose the work today? (Please only choose one)

A. Decision Tree
B. Priority Queue
C. Stack

Step 2. If the user clicks A. Decision Tree, he will be asked a series of questions until find the specific work assignment. A good example is as follows:

Want to focus on research today?

A. Yes
B. No

If the user chooses A. Yes, he/she will be directed to the following screen:

Want to focus on grants?

A. Yes
B. No

If the user chooses A. Yes, he/she will be directed the following page:

Grants on federal sponsors?

A. Yes
B. No

If the user chooses B. No, he/she will be directed the following page:

IT company?

A. Yes
B. No

If the user chooses B. No, he/she will be directed the following page:

Your choice for today's work is IBM

Exit

When the user clicks the exit button, the program ends.

Step 3. If the user clicks B. priority queue in step 1, he/she will be shown the following screen,

If need to add a new assignment, please input title and priority (accept integer, 1-100 is preferred). If need to remove an existing assignment, please input title of the assignment that needs to be removed.

What do you want to do with assignment?

A. Display the work assignments
B. Add a work assignment Title_______ Priority______
C. Remove the work assignment Title______

After clicking ok, the following screen shows up. The priority queue is ranked in priority order.

The work assignments in order are:

Step 4. If the user clicks C. Stack, he/she will be shown the following screen. If need to add a new assignment, please input title and priority (accept integer, 1-100 is preferred). If need to remove an existing assignment, please input title of the assignment that needs to be removed. The reason why priority level is needed since the input is shared between priority queue and stack.

What do you want to do with assignment?

A. Display the work assignments
B. Add a work assignment Title_______ Priority______
C. Remove the work assignment Title______

After clicking ok, the following screen shows up. The stack is from the most outdated to most updated (the most updated is shown at the bottom of the list).

The work assignments in order are:

Chapter Six. Conclusion/Summary

Section 1. Demonstration of MERUSE

My project reflects the MERSUE principles of good programming from the following perspectives:

1. *Modularity*: The project is composed of three parts (i.e., binary search decision tree, stack, and priority queue), which demonstrates the top-down design, which is re-suable, especially more parts are easily added the project. I adopted object-oriented design for individual parts, which use classes and encapsulation.
2. *Efficiency*: For each of the three parts (i.e., binary search decision tree, stack, and priority queue), I used the method with relative low complexities. The detailed information is listed as follows:

Data Structure/algorithm	Average			
	Access	Search	Insertion	Deletion
Stack	O(n)	O(n)	O(1)	O(1)
LinkedList(to implement priority queue)	O(n)	O(n)	O(1)	O(1)
Bubble sort	O(n)	O(n^2)	O(n^2)	O(n^2)
Binary search tree	O(log(n)	O(log(n)	O(log(n)	O(log(n)

3. *Robustness*: I tested the program considering kinds of corner situations and handles different errors when possible. The program could finish the fundamental functions expected in the project objectives.
4. *Usability*: The GUI design shows usability. It directs the user to get the answer he/she wants. For example, after the user chooses to add a new element into the stack. The GUI further asks whether the user would like to review all the elements in the stack with the added element? And for each of the three options (i.e., binary search decision tree, stack, and priority queue), the user could exit the program at different steps, which increases usability.
5. *Readable*. I formatted the codes on eclipse and named the variables properly which makes the other programmers easily guess what they refer, and include detailed comments throughout all the program.
6. *Elegant*: I keep the consistent style of the program which leads to readable code. Moreover, I used shorter code (i.e. directly adopted library if available) and efficient algorithms.

Section Two: Project Summary

The project aims to create an initial version of the ProfessorAssist App. The app has three options to help professors to identify the work: binary search decision tree, stack(last in first out), and priority queue (pick the one with highest priority out). Bubble sort algorithm is embedded in priority queue function. GUI is used to achieve the goal of computer-human interactions or I/O functions in the project. The app could serve as a multi-task management platform that satisfy the basic needs of professors.

Section Three: Future Versions

Due to the limitation of data structure/algorithm used and workload of the class project, the app has some functions that need to be improved in future reversions, although it has many merits. Future version of ProfessorAssist App could supplement the following functions:

1. This version only allows the stack and priority queue to add/delete new elements. However, binary search decision tree does not include these two functions. One reason is that the class project requires two data structure. The binary search tree is an extra one. The second reason is that it is complicated to add/delete elements in the binary decision tree. For example, a leaf of the binary search tree is USDA. If I want to add a child of USDA, I need to consider several issues to make it reasonable: 1) change USDA to a question, since in binary search decision tree, users are kept asking questions; 2) binary search tree has balance issue; 3) the user may want to add three or more children of a parent node, which makes the tree "un-binary". Therefore, the decision tree needs much more work to make it practicable and applicable.
2. Since the project includes binary search decision tree, many miscellaneous (e.g., write reference letters, reply emails) that don't fit to the binary structure aren't included in the app. However, these miscellaneous can't be ignored in a professor's daily work schedule.
3. Future versions could include adding the schedule of meetings in. Moreover, each work assignment needs to further classified into distinct parts and sub-parts, and the estimated time period for finishing each part/sub-part is also included in the app. And when a work assignment is pull out but the professor doesn't finish it today. How can he/she pull back "remaining part" of the work?
4. The computer-human interaction of the ProfessorAssist App needs to be further improved to make it more user-friendly. Customer surveys and experiments could be used to provide suggestions on how to make it happen.