TEXTBOOK PROBLEM DEPENDENCY WEB

An Undergraduate Research Scholars Thesis

by

JOSEPH MARTINSEN

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Philip B. Yasskin

May 2019

Major: Computer Engineering

TABLE OF CONTENTS

ABSTRACT

Textbook Problem Dependency Web

Joseph Martinsen
Department of Computer Engineering
Texas A&M University

Research Advisor: Dr. Philip B. Yasskin
Department of Mathematics
Texas A&M University

After a textbook has been written and published, one may want to customize it for a particular audience; it may be desirable to delete or reorder some of the chapters or sections. However, there may be dependencies among the chapters, sections, examples and exercises which make it very tedious to rearrange the order of not only the chapters but also, the associated exercises. Martinsen is building a graph database to describe the dependencies among the chapters and sections in a portion of the online Calculus book, MYMathApps Calculus, being written by Yasskin. In addition there will be a database of examples and exercises which describes their dependencies on the chapters and sections. Further, he is building a GUI for an instructor or institution to reorder the chapters and sections by drag and drop and have the examples and exercises automatically reorder.

DEDICATION

To. . .

ACKNOWLEDGMENTS

To...

NOMENCLATURE

Consumer	Adopting institution or professor who wants to reorder the textbook
DAG	Directed Acyclic Graph
Tree	A form of a Graph that has direction and no cycles
Topic	
Unit	Chapter/Section/Page
Exercise	A problem that is given to the user to solve

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Background Info

Textbooks go through a long and arduous process before a student or professor is able to view and use it. This process requires much work and effort into not only validating the content of the textbook but also validating the ordering of the textbook as whole. Much like a jigsaw puzzle, each chapter fits one after another based on the dependency of the topics being taught. On top of these chapters being ordered, the exercises must also be placed in the correct place in order to not give an exercise that is based on a topic that has not been presented to the reader previously.

After all this work on ordering has been completed (among other meticulous things), the textbook may finally ready for publication. As with many good textbooks, many professors and institutions may enjoy the content within the textbook but would prefer delivering the content to a student in a different order than what the current order the textbook is in currently. Most of the time, the work and effort that has gone into meticulously ordering the chapters and exercises must now be revisited again and modified. This process is nearly as time consuming and laborious as the first going through this process.

1.2 Existing Technology

In the current field, most re-orderings of chapters and exercises are done by hand with human intervention by either textbook authors or commercial institutions. An exception is TopHat (TM) who, for their interactive textbooks, has the ability to reorder in a drag and drop manner but one lacking feature is any sort of feedback of any sort being provided to the user. I was able to test drive this product and it felt much like a changing the order of a power point presentation as it is in its current state. Besides the technology features that

TopHat offers on the rest of their platform, which is outside the scope of this discussion, much of what their reordering seems to achieve can simply be done by reordering the pages of the textbook in some sort of PDF viewer.

From speaking with several Math professors at the Joint Math Meeting 2019, this idea of being able to reorder structural components of their textbook is an issue they commonly face and is not brand new. From each professor I interacted with, there were not aware of a simply and useful that would help them complete this task. There was one platform for identifying dependencies between subjects and textbooks that then allows a professor to generate a curriculum based off these topics, but this is done on a much higher textbook level, not by chapter, section or exercise.

1.3 Approach

My approach to this situation wants to take on a similar approach to what TopHat has done but builds heavily on the fact that the original author has an idea of the flow of a textbook, including changes to the order the repercussions that may result. While it is not feasible for a professor to provide this feedback in person, the realtime feedback aspect can be preserved by creating a platform that *understands* the original authors concerns and worries and then provide them to the consumer of the textbook the moment a change they make may result in possible conflict of dependencies.

2. THE ALGORITHM

2.1 Introduction

The core portion of the platform that will allow reordering of aspects of the textbook can simply summed up as *The Algorithm*. This underlying algorithm is independent of technology stack or implementation. It is simply a high level description and analysis of the proposed solution to the given problem.

With simply an understanding of the algorithm, the implementation should follow in a natural and simple manner.

2.2 Motivation for the Algorithm

The algorithm has other items to consider outside simply just solving the given issue. As with most algorithms, there is a thought and focus on completing the desired task in an efficient and optimal manner. This means that if the algorithm is able to provide feedback but if it is done in an clunky manner, the algorithm is still considered a failure and not useful.

In addition to efficiency, correctness is another major point of focus. In this context, correctness relates to properly conveying the thoughts and concerns of the original author to the professor or institution trying to modify the ordering of a textbook. As a result of a particular action or modification by the consumer, there should never arise a situation where a suggestion or warning provided by the algorithm conflicts with the thoughts or viewpoints of the original author of the textbook. These messages provided by the algorithm should be simply an extension of the original if not exactly the same as if the author was at the computer sitting with the user of the platform.

Correctness also relates to truly reordering the textbook as desired and specified by the

one who modified the order of the textbook. After a change by the consumer, they should be able to clearly understand what type of change they are proposing, how this affects the textbook as a whole, how nearby sections or chapters may be affected and finally after committing these changes, the actual textbook should properly reflect these changes as desired by the consumer.

[1] [2] [3]

2.3 The Design

2.3.1 Structure

The design first has to answer the question of what. What information is necessary to properly accomplish the given task. There are N items that are being considered. At the root of everything, there is a topic. This is the core dependency upon which all other dependencies are built upon or contrived from. All the following data types all directly or indirectly are tied to a particular topic or group of topics. A unit is very generic and could be broken up into three different types. A unit is either a chapter, section or page. For the purpose of dependency mapping, there has not been a use case that distinctly separates one of these from another for the purpose of dependency mapping so therefore they have been simply grouped together into type unit. The final data type is exercise. An exercise is different than a unit because of how it is presented and how the underlying content is used. A unit is simply a presentation of content and/or topics to the reader. An exercise is formal way of testing the consumers knowledge and practice their skills. Another special consideration of exercises is their dynamic and flexible nature. Unit structure is what the consumer will likely spend a majority of time reordering and tweaking. Once the unit structure is in place, the exercises should automatically populate and relate to the correct units given the new unit dependency mapping.

The next topic of focus is how to properly structure the necessary information to com-

plete the given task. Since this is in fact a hierarchical dependency mapping problem at its core, a directed acyclic graph (DAG) is the format was chosen to map and store the dependencies. The reason being because a given topic can have one of two true relations and one metarelation with any another topic. This given topic may depend on another topic because material must first be introduced in the latter topic in order to properly deliver the content in the former topic. This is an example of a child relationship. The reverse relation is also a relationship. This reverse relationship is an example of a parent relationship 2.1.

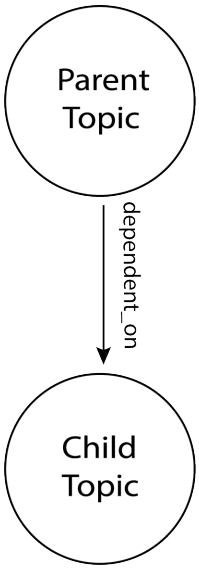

Figure 2.1: Parent child topics.

Finally, the metarelation is the situation where two topics are located on similar levels of the hierarchy in the dependency mapping and do not have a parent or child relationship with one another. These topics are independent of one another, meaning that the order of these two topics in no way affect one another 2.2.

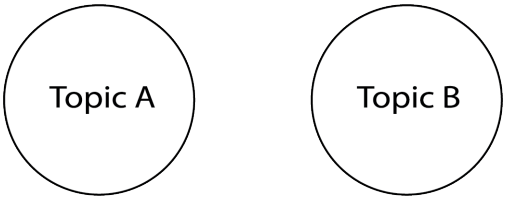

Figure 2.2: Unrelated topics. Can have any number of parents and children but do not relate to one another.

The previous dependency mapping completely describes the requirements for mapping topics but units and exercises require more mapping. Units have two relationships to map too. Like topics, units have a dependency mapping between the same types as themselves. Units are also tied to topics because units are the form in which topics are introduced in the textbook. A particular unit can introduce one more topics. For the purpose of relating these two, an unique id is assigned to each topic. For a given chapter then depends on N topics, a list of unique ids of each of the N topics are stored on the chapter node 2.3.

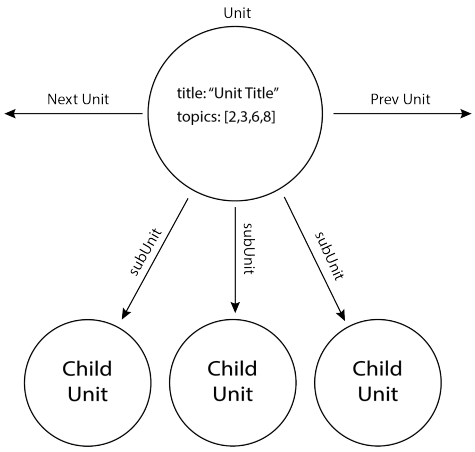

Figure 2.3: Unit mapping.

The final mapping is for the exercise data type. Exercises have three different possible

data types it can depend on. They can depend on a topic, a unit or another exercise. The topic and unit dependency should ideally be the same but for our purposes, both must be satisfied in order for this dependency to be satisfied. In the absence of one, specifically the topic dependency, the other can be used. This is because if the topic dependency is missing for an exercise but unit dependencies are listed, the topic dependency can be generated by traversing the unit dependency mapping for the given units the exercise depends on. The final dependency on exercises is independent of topic or unit. The situation for exercise dependency arises when the results of one exercise are used in another exercise. In this situation, a given exercise should follow immediately after the dependent exercise or after another exercise that shares the same dependency on the same exercise.

Outside of dependency mapping, there is the actual order of the textbook and order of exercises. They can be contained with a DAG much like all the other topics. These are what actually describes how the textbook will actually be structured. These mappings are the only ones that are mutated for the sole purpose of reordering a textbook. There is a base mapping that is initially defined by the author for the first publication of the book. These base mappings should be structured in a way such that all dependencies are satisfied.

Finally it must be formally defined what *satisfying all dependencies* mean. It short, a unit or exercise satisfy all dependencies when all of its dependencies' dependencies are satisfied. For an entity to depend on another entity, it simply means that the parent entity must occur or appear in an earlier point in the dependency mapping tree. Since this is a recursive definition, it must terminate with some sort of base case. This base case is a base unit or topic which is a unit or topic that has no other dependencies. If all other topics, units and exercises that a particular entity has depends on has appeared earlier on in the dependency mapping tree, then it can be said that this entity has all dependencies satisfied.

2.3.2 Flow

For a textbook that is going to be integrated into this dynamic modification flow, several dependency trees/DAGs need to be laid out describing the textbook. Specifically the different orderings as defined by the original author and a consumer and how they interact with one another must be laid out. They are as follows.

1. Tree of unit ordering
2. Tree of exercise ordering
3. Tree of unit interdependency + topic dependency 2.3
4. Tree of exercise interdependency + topic or unit dependency
5. Tree of topic interdependency

Trees 1 and 2 are initially created by the original author in order to define the original ordering of the given textbook. These trees (1 and 2) are the only ones that will ever be modified by the consumer in order to reorder a given textbook. Trees 3, 4 and 5 are defined by the author at the completion of writing the textbook and should not be modified for the current version of the textbook. A modification of trees 3, 4 or 5 should be done to correct a dependency mistake, to add a new entity or to remove an entity. These modifications should result in a new edition of the textbook. Trees 3, 4 and 5 are used in order to validate and confirm that all dependencies are met in trees 1 and 2 for both the original 1 and 2 trees created by the author as well as for when they are modified by the consumer.

The actual workflow is as follows.

1. Original author creates trees 1, 2, 3, 4 and 5. Trees 1 and 2 may be generated if trees 3, 4 and 5 are completed.

2. Adopting institution or instructor specifies the desired ordering of the textbook by modifying tree 1.

    (a) During this process warnings may be provided to the user if any of their modifications to tree 1 cause a dependency to not be met as defined in trees 3, 4 and 5

3. Using trees 3, 4 and 5 as well as the user modified tree 1, tree 2 is generated defining the order of the exercises

    (a) The adopting institution or instructor can preview the exercise and modify the order in which they may appear.

    (b) During this process warnings may be provided to the user if any of their modifications to this newly generate + modified tree 2 cause a dependency to not be met as defined in trees 1, 3, 4 and 5

4. The newly generated order in the form of trees 1 and 2 can then be utilized to modify the text and links in the given textbook

3. THE IMPLEMENTATION

3.1 The Textbook

For the implementation of the design, I have gone with proof of concept model. Taking a single textbook and applying this workflow for the purpose of working out any workflow issues and testing the algorithm in a real world situation. The textbook that will first be integrated is Dr. Philip B. Yasskin and Dr. Meade Mathematics textbook *MYMACalc: A Web Based, Interactive, Multimedia Calculus*. This textbook was initially written in LaTeX but has since been converted into a web based textbook.

3.2 Textbook Build Process

As aforementioned, the textbook is web based and as such, is developed to conform to web standards. Each page is written in Hypertext Markup Language (HTML). In order to better clarify as well as lighten the load on the author(s) of the textbook, a JavaScript object notation (JSON) file defines the order of the textbook. This order is similar to tree 1 as previously presented. From this ordering, the textbook is then built and structured. This JSON structure allows means that tree 1 is completed. Within tree 1, topics are then manually added to each unit in order to begin the process of building the other trees. Once these topics have been added, tree 3 and 5 can be generated by the implicit ordering defined by tree 1. The topics names are then replaced with ids of the topics. The author is then given the opportunity to then tweak trees 3 and 5 to indicate the correct dependency mapping of these trees.

3.3 The Tech Stack

3.3.1 Database

A graph database is utilized for storing the five different trees. The particular database used was Neo4j. This database allowed for DAGs to be stored in it's entirety. It also allowed for each querying and finding of nodes by simply information about the node. There was also an interface provided by this service that would visually show nodes that are matched by a given query as well as the relationship between each node.

3.3.2 Server

The purpose of the server was to fetch and store data to the database and perform the computational portions of the algorithm. This included identifying warning messages and ensuring that all dependencies are met for each change or modification that a consumer makes.

3.3.3 Client

The client utilized ReactJS. This is front end library that had existing modules that allowed for easily modifying and reordering a JSON tree. This was not a simple implementation so it did require a good amount of tweaking in order for it to fit the necessary use case.

3.4 Integration with Textbook

Full integration with the textbook involves two different parts. One is adding all 5 textbook trees for use by the platform and the other is being able to export the necessary data in order to actually modify the textbook. As mentioned previously, the textbook being used had a good portion of tree 1 completed. With some effort and additional mapping, trees 2, 3, 4 and 5 can be constructed.

3.4.1 Import Process

The import process put simply requires the original author to upload all five trees. From there the server is then able to validate and identify if there are any possible dependencies not satisfied in trees 1 and 2. If there are, the author is then provided messages as to what dependencies are not met and allows the autor to modify them using the graphical user interface. This graphical user interface is the same as the one the consumer uses for reordering latter on.

3.4.2 Reorder Process

This is the process where the consumer has the control to modify the structure of the units and the exercises. There are two views for accomplishing this. One is focused on simply the unit and the other focuses more on the exercises and the the units they are directly tied to. During each modification or tweak that lasts for longer than two seconds, the new and updated tree is sent to the server for verification. The server then verifies dependencies and identifies if any have been violated. Depending on the order of the violation and where in the tree the violation occurred, a message is generated to inform the consumer of what the violations are and possible suggestions as to how to fix the ordering.

After the consumer has modified the ordering of the units, they are able to review these changes. This review screen highlights changes, additions or deletions of units. This screen allows the user to see all the changes they are attempting to do before they commit them. Once the user approves these changes, this new ordering is then stored as the new ordering for the particular consumer. Along with storing these changes, the consumer is given the option to allow tree 2 to be automatically generated. Regardless of if the user allows for this to happen, they are then taken to the exercise reordering screen. The same reordring for units then follows with warnings and messages appearing as well as

the review screen showing before a user fully commits their ordering changes.

3.4.3 Export Process

Due to the structure and modifications to the build process of the textbook in use, the export process is rather trivial. It is simply an export of trees 1 and 2. With the textbook in use, especially with tree 1, these trees in JSON format are then placed in the appropriate location in the build process. This allows for the existing build process to generate the new ordering of the textbook as specified by the consumer.

4. CONCLUSION

4.1 Results

Images of the application and textbook?

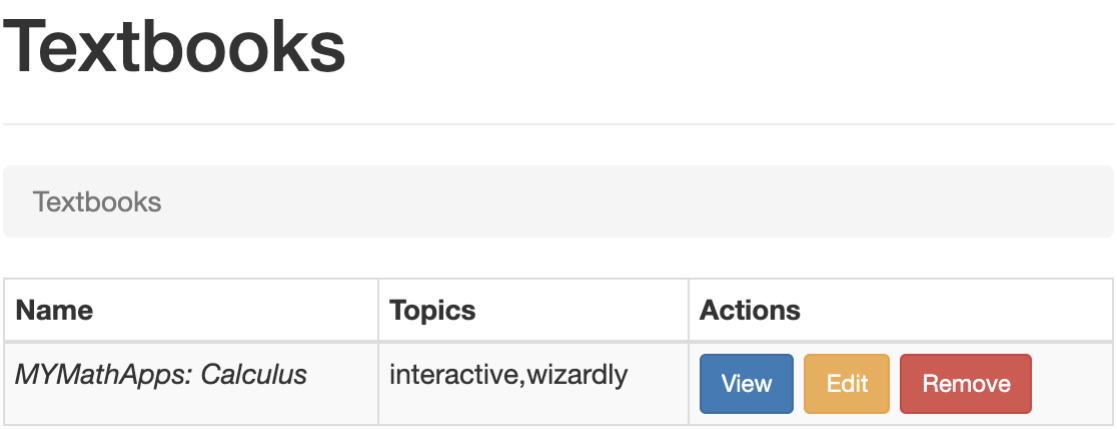

Figure 4.1: Unrelated topics. Can have any number of parents and children but do not relate to one another.

4.2 Challenges

Time, takin 23 credit hrs ect. . . ? Tweaking existing build process of textbook for integration.

4.3 Broader Impact

Allow other authors to utilize this design + implementation?

4.4 Future Plans

Get this functional for entire MYMACalc book

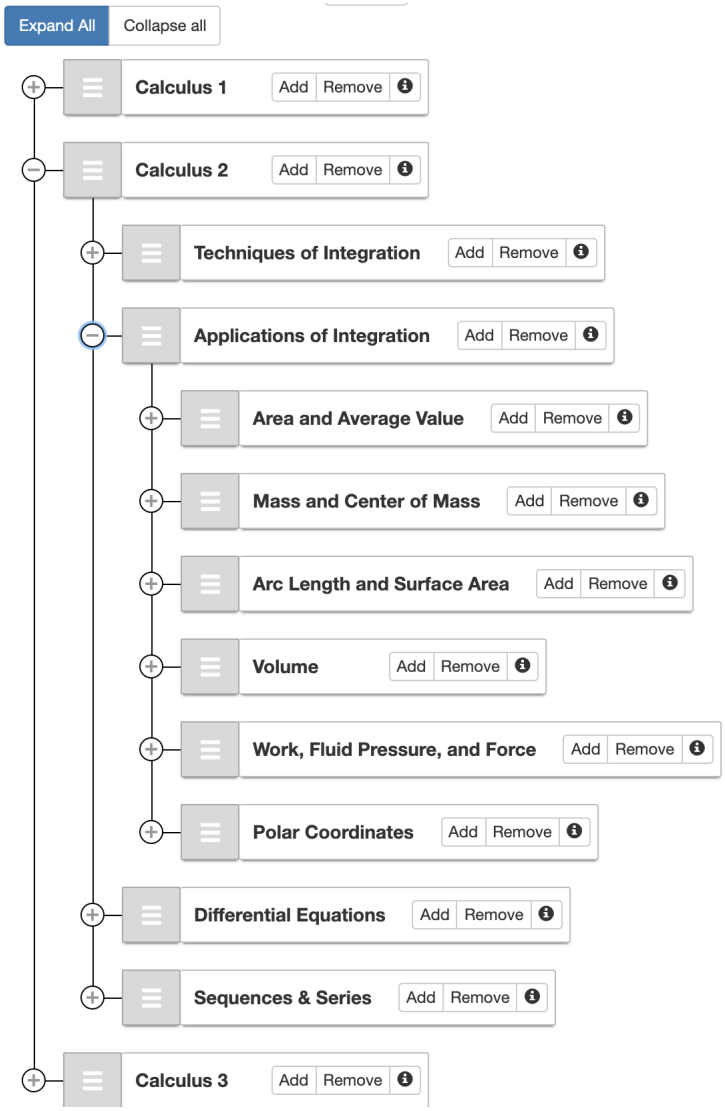

Figure 4.2: Unrelated topics. Can have any number of parents and children but do not relate to one another.

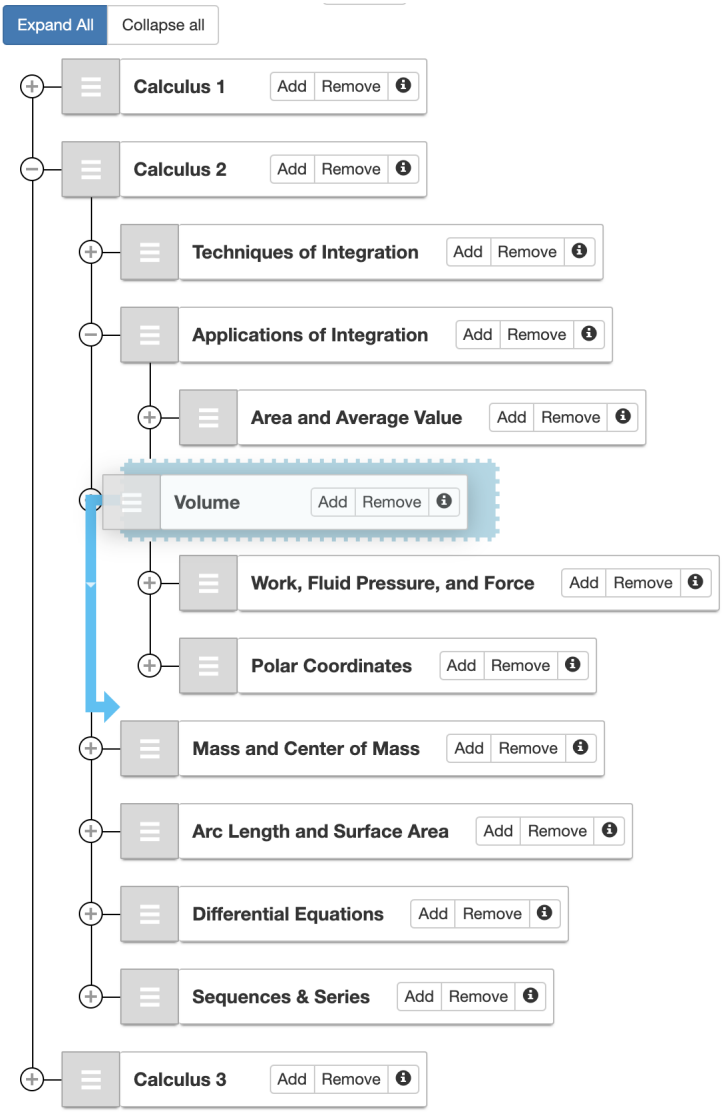

Figure 4.3: Unrelated topics. Can have any number of parents and children but do not relate to one another.

REFERENCES

[1] P. Bille, "A survey on tree edit distance and related problems," *Theoretical Computer Science*, vol. 337, no. 1-3, pp. 217–239, 2005.

[2] D. Tsur, "Faster algorithms for guided tree edit distance," *Information Processing Letters*, vol. 108, no. 4, pp. 251–254, 2008.

[3] B. Vaughn, "Reconciliation." Web, 2019.