COMP-SCI 5592 0001- Dsng&Anal Algorithms FS2017

**Project 1 – Course Matching System**

(Using dynamic programming to allocate the best professors for courses based on the course content and the expertise of the professor)

By: Abdul Aleem Khusro

Arsalan Javaid

Bilal Ahmad

Syed Jawad Hussain Shah

Talha Javed

**Assumption:**

The first assumption team has is that the test data generated for this project will accurately mock the real-world data, so that the design used in this project can be extended to real world applications. Second assumption is n, which is the number of instructors that will be assigned to a course. The total number of courses is m and total number of topics are t. The team assumption is that there are more courses than instructors and there are more topics than courses, that is n ≤ m ≤ t. Further, the composition of courses was also assumed, and pseudo percentages of each individual topics was given to the courses. Lastly, pseudo scores between 0 and 5 were given to each professor for individual topics.

**Explanation of the Idea:**

The project started with creating data on text files, so that data is available for code execution later. For this purpose, five text files were created. Courses.txt file contained the data for course table, coursetopicpercent.txt file contained the information regarding the composition of the course. instructors.txt file contains the information of the instructors and their respective Id’s, topics.txt that contained the topics information and topicprofskill.txt file that contains professor ratings for each topic. There are various languages that are used for this project. One of the implementation of this project is executed in C#, while the other is conducted in JavaScript in NodeJS. The reason behind different implementation is that the team members expertise differed in various languages so a blend of both these languages was used. Further the UI (user interface) is created with AngularJs and Bootstrap. NodeJS was also used to enhance the UI.

The main Idea is that each professor was scored for each topic between 0 and 5 (0 means no knowledge of the topic and 5 means mastery knowledge) and this information was stored in topicprofskill.txt file. Further, the composition of each course was measured to see how much a course is made up of individual topics. For example, a course A is made of 50% of topic 1, 20 % of topic 2 and 30 % of topic 3 and this information was stored in coursetopicpercent.txt. For each course, the professor topic score was multiplied by the topic percentage composition for that course, to calculate a proportional relevance of the professor for that course, and all these individual relevancies were added to calculate an overall relevance score of a given professor for a given course. This information was stored in an array called “professorcourse”. After that, each course was aligned with that course’s professors’ relevance scores and was assigned to the professor with the highest score. As per our assumption that in modern day universities there are more courses (or sections of the courses) than professors, so using the same approach each professor is assigned two courses. There is also an array called “taken” that contains a Boolean value of true or false to indicate whether a certain course is already assigned to an instructor or not. In this way the algorithm keeps tracks of the course assignment and doesn’t assign an already assigned course to an instructor. Dynamic programing is used to generate the final out put of the algorithm to show courses with the assigned professor. Please see the diagram, screen shot, and the table below:

Step 1:

Score Each professor for all the topics

Step2:

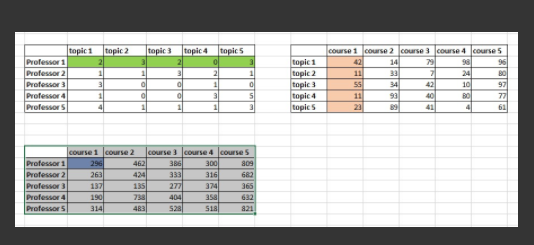
Find the topic composition of each course and assign each topic a percentage as per their respective contribution

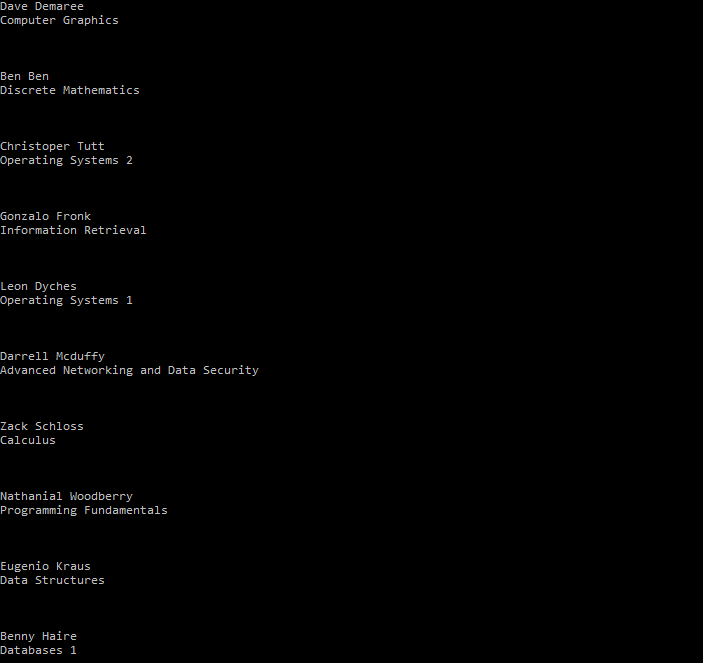
Step 3: calculate Professor Course Relevance score by multiplying each professor topic score by the topic percentage of that topic for that course and add all these proportional scores to get an overall course relevance score for each professor

Step4: Assign each course to the professor with the highest course relevance score.

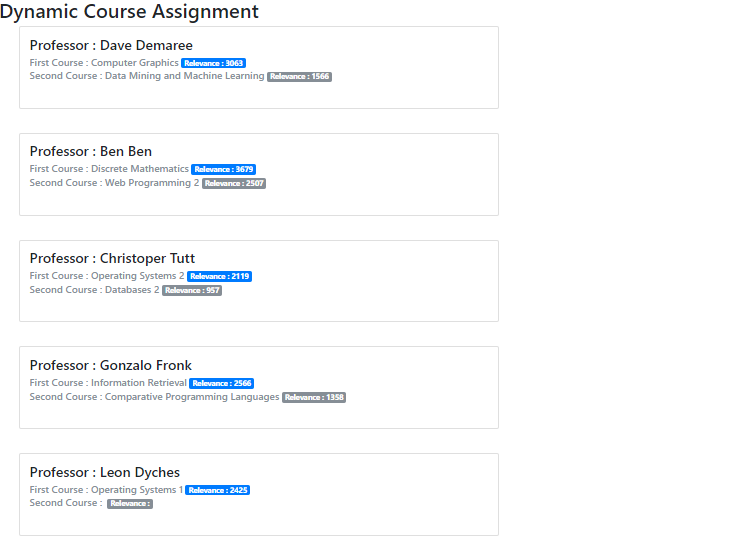
Diagram of algorithm Functionality:

**Screen Shots:**





After using NodeJS the above a better view of the above UI is created and is shown below



**Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Professor Name | 1st Assigned course | Relevance score for the 1st course | 2nd assigned course | Relevance score for the 2nd course |
| Dave Demaree | Computer Graphics | 3063 | Data Mining and Machine Learning | 1566 |
| Ben Ben | Discreate Mathematics | 3679 | Web Programming 2 | 2507 |
| Christoper Tutt | Operating System 2 | 2119 | Databases 2 | 957 |
| Gonzalo Fronk | Information Retrieval | 2566 | Comparative Programming Languages | 1358 |

**Efficiency of the Algorithm:**

The main functionality of this algorithm was to assign a course to the professor. For this purpose, it must work with the number of professors, number of courses and number of topics, as sated in assumption if we assume the number of professors to be n and number of courses to be m and number of topics to be t, then the complexity of this algorithm is n x m x t. Further, if we assume that roughly the magnitude of n, m and t is same then:

Big-O of this algorithm is roughly = (n3).

**GitHub URL for extra credit:**

<https://github.com/bilaleme/DynamicCourseAssignment>

Q) How would you modify your algorithm if a professor must teach two courses instead?

The code already performs this functionality.

**References:**

<https://support.microsoft.com/en-us/help/816149/how-to-read-from-and-write-to-a-text-file-by-using-visual-c>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/file-system/how-to-read-from-a-text-file>

<https://www.tutorialspoint.com/cprogramming/c_arrays.htm>

<https://www.programiz.com/c-programming/c-arrays/1?utm_expid=66600334-4.VfQskolZTeyX_ldg7j0KXQ.1&utm_referrer=https%3A%2F%2Fwww.google.com%2F>

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/bool>

<https://www.dotnetperls.com/bool>

<https://www.codeproject.com/Articles/73856/Csharp-4-0-Dynamic-Programming.aspx>

<http://scottdorman.github.io/2008/11/16/c-4.0-dynamic-programming/>

<http://www.rkinteractive.com/blogs/SoftwareDevelopment/post/2013/01/29/Algorithms-In-C-Dynamic-Programming-Introduction.aspx>

<https://www.codeguru.com/csharp/article.php/c17991/Dynamic-Programming-Using-C-40-and-Microsoft-Visual-Studio-2010.htm>

<https://stackoverflow.com/questions/22956683/dynamic-programming-recursion>

<http://www.benfarrell.com/2013/01/03/c-and-node-js-an-unholy-combination-but-oh-so-right/>

<https://stackoverflow.com/questions/25897951/is-there-any-standlone-gui-module-for-node-application>

<https://www.slideshare.net/RyanAnklam/modern-ui-development-with-nodejs>

<https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/>

<https://readwrite.com/2013/10/02/github-for-beginners-part-2/>