Algorithms and Problem-Solving

TEACHSYNC
EFFICIENT TEACHER LOAD ALLOCATION

SUBMITTED BY:

MANYA JAIN 20103161 B6

AVNI 20103153 B6

SAKSHI KUKAL 20103172 B6

DIVYANSH GOEL 20103174 B6

ACKNOWLEDGEMENT

We express our sincere regard and indebtedness to our project internal guide Dr. Hema N and Mrs. Jyoti, for their valuable time, guidance, encouragement, support and cooperation throughout the duration of our project. We would sincerely like to thank CSE Department for giving the opportunity to work on enhancing our technical skills while undergoing this project. This project was done under the guidance of Dr. Hema N and Mrs. Jyoti.

This project helped in understanding the various parameters which are involved in the development of a software application and the working and integration of the concepts such as backtracking algorithm to create a fully functional software application.

We would like to thank DEF, Head of Department and whole of department for their constant support.

PROBLEM STATEMENT

In the teacher load assignment problem, we are given a set of **N** teachers and a set of **M** subjects. Each teacher has an integer workload capacity, which you can think of as the number of hours that the teacher is available to teach. Each teacher also has a list of subjects they have the background to teach. Each subject has an expected number of hours it takes to teach.

INTRODUCTION

We would like to propose a program which solves the above discussed problem. It is complied in Code::Blocks with GCC complier where the concepts of Graphs, Ford Fulkerson Algorithm for Max Flow and File Handling is used. The source code for this program is around 200 lines. This system is designed to help an institution allocate the subject hours to teachers in an efficient and equally distributed way.

OBJECTIVE

Consider the following example:

Teachers:

- Capacity of T1: 10, background for S1, S2, S3.
- Capacity of T2: 10, background for S1, S3, S4.
- Capacity of T3: 10, background for S2, S3, S5

Subjects:

- S1: 7 Hours
- S2: 5 Hours
- S3: 5 Hours
- S4: 2 Hours
- S5: 3 Hours

In the example above, we might assign teacher T1 to subjects S2 and S3 (maximizing T1's workload), teacher T2 to subject S1 (the larger of the two tasks still available to it), and teacher T3 to subject S5 (the only subject still available to it), leaving T4 unassigned. In this model, we complete 22 of the 20 hours available.

Using the same mentioned approach, we determine the maximum flow for each teacher and we reach with a residual capacity graph. Using the residual capacity graph, we can determine the number of subject hours a teacher will teach. The following is a formula for computing the same:

$$No.\ of\ Subject\ Hours = Total\ Capacity - Residual\ Capacity$$

WORKING

To solve the teacher load assignment problem using the Ford-Fulkerson algorithm, you would need to construct a bipartite graph with teachers on one side and subjects on the other. You would then add edges from the source to each teacher with capacity equal to the teacher's workload capacity, and edges from each subject to the sink with capacity equal to the expected number of hours it takes to teach the subject. Finally, you would add edges from each teacher to the subjects they can teach with infinite capacity.

The code starts by defining several global variables and arrays to represent the graph, the residual graph, the parent array used by the BFS function, and a visited array to keep track of visited nodes during the BFS and DFS traversals.

The **BFS** function takes as input the source and sink nodes and performs a breadth-first search on the residual graph to find an augmenting path from the source to the sink. If an augmenting path is found, the function returns true and updates the parent array to store the path. Otherwise, it returns false.

The **DFS** function takes as input a node "S" and performs a depth-first search on the residual graph starting from node S. It marks all nodes reachable from S as visited.

The **maxFlow** function takes as input the source and sink nodes and implements the Ford-Fulkerson algorithm to find the maximum flow in the graph. It starts by initializing the residual graph to be equal to the original graph. Then, it repeatedly calls the bfs function to find an augmenting path from the source to the sink. If an augmenting path is found, it updates the residual graph by subtracting the flow along the path from the forward edges and adding it to the reverse edges. It also increments the maximum flow value by the flow along the path. The function continues until no more augmenting paths can be found, at which point it returns the maximum flow value.

The **main** function starts by reading in the number of teachers and subjects from standard input. It then sets the source node s to be 0 and the sink node t to be nTeachers + nSubjects + 1.

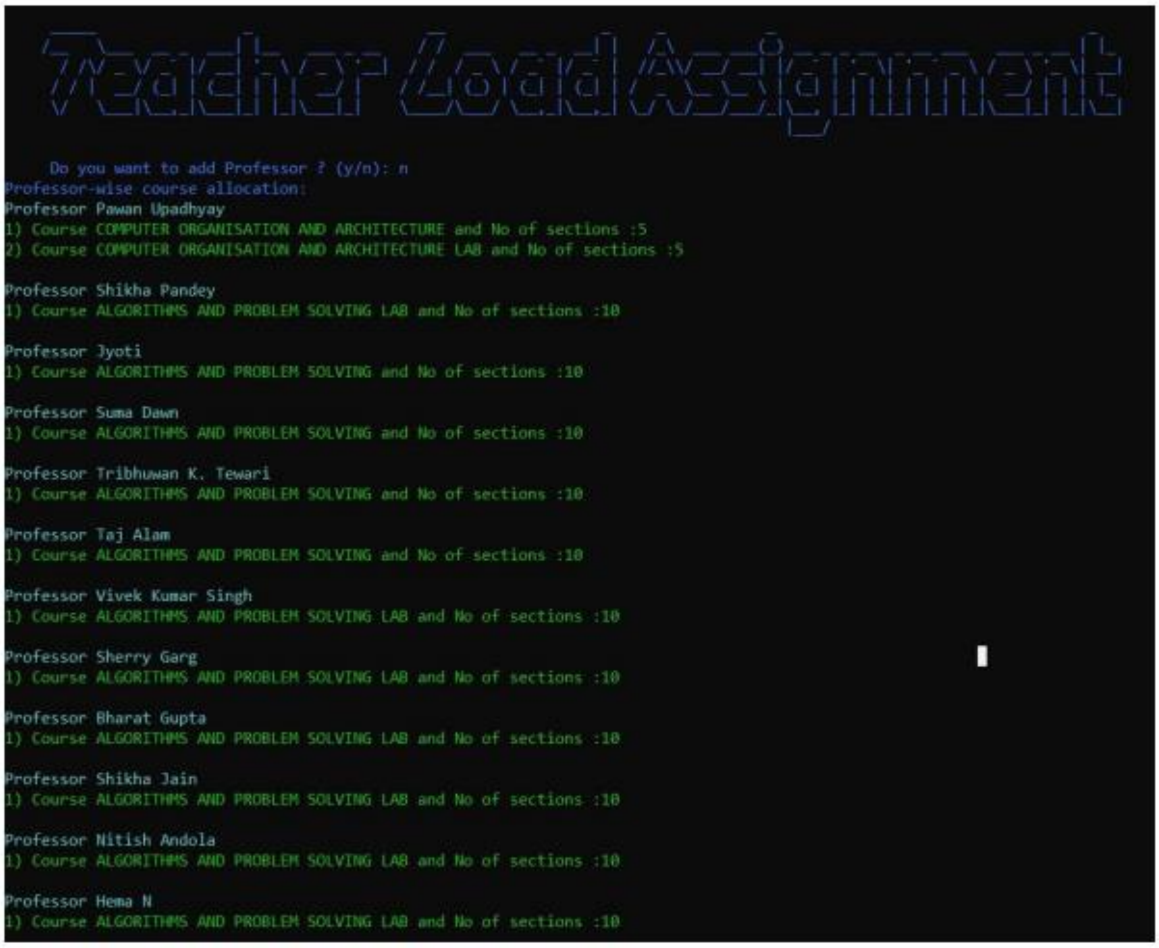
Next, it reads in the workload capacity of each teacher and adds an edge from the source to each teacher with capacity equal to their workload capacity.

Then, it reads in the expected number of hours it takes to teach each subject and adds an edge from each subject to the sink with capacity equal to this value.

Next, it reads in the subject expertise of each teacher. For each teacher, it reads in a list of subjects they can teach and adds an edge from that teacher to each subject with infinite capacity.

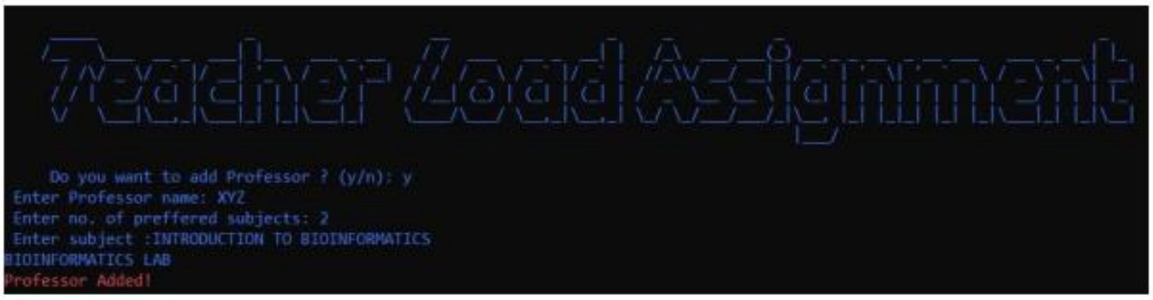
Once all input has been read in and processed, it calls the maxflow function to find the maximum flow in the graph. It outputs this value as well as any assignments of teachers to subjects based on positive flow in these edges.

OUTPUT



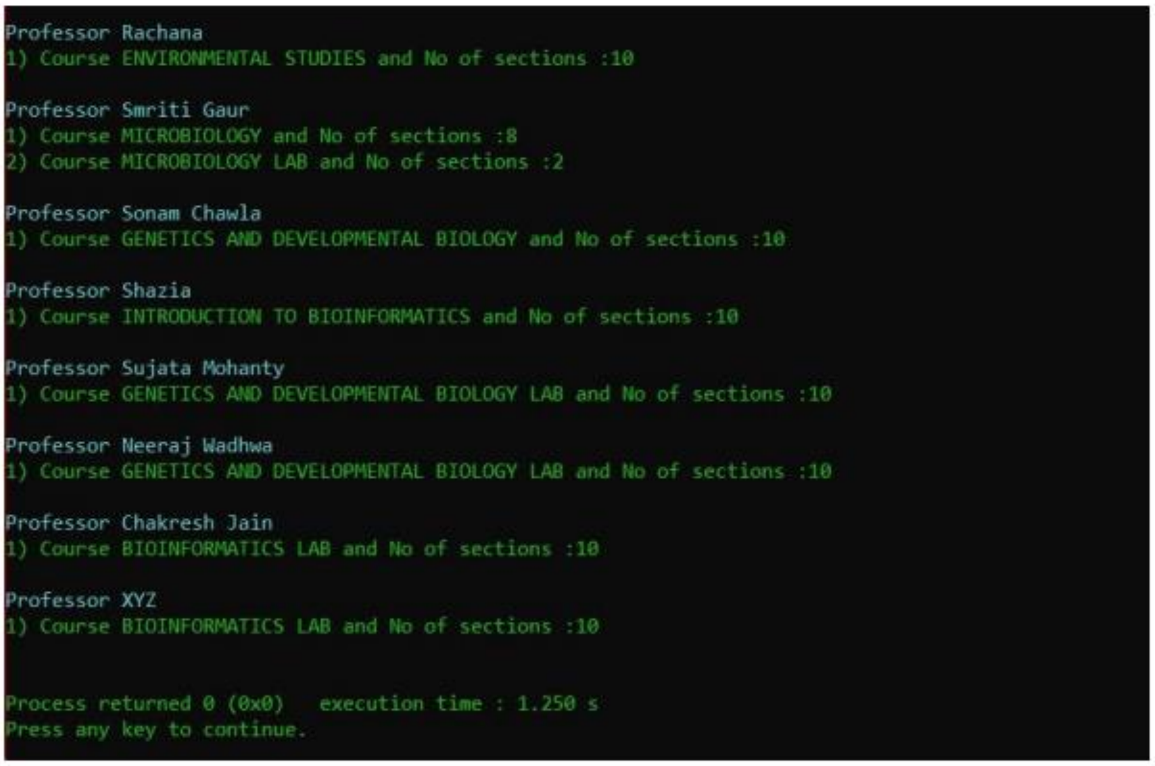
```
    Do you want to add Professor ? (y/n): n
Professor-wise course allocation:
Professor Pawan Upadhyay
1) Course COMPUTER ORGANISATION AND ARCHITECTURE and No of sections :5
2) Course COMPUTER ORGANISATION AND ARCHITECTURE LAB and No of sections :5

Professor Shikha Pandey
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10

Professor Jyoti
1) Course ALGORITHMS AND PROBLEM SOLVING and No of sections :10

Professor Suma Dawn
1) Course ALGORITHMS AND PROBLEM SOLVING and No of sections :10

Professor Tribhuwan K. Tewari
1) Course ALGORITHMS AND PROBLEM SOLVING and No of sections :10

Professor Taj Alam
1) Course ALGORITHMS AND PROBLEM SOLVING and No of sections :10

Professor Vivek Kumar Singh
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10

Professor Sherry Garg
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10

Professor Bharat Gupta
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10

Professor Shikha Jain
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10

Professor Nitish Andola
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10

Professor Hema N
1) Course ALGORITHMS AND PROBLEM SOLVING LAB and No of sections :10
```

ADDING A NEW PROFESSOR



```
    Do you want to add Professor ? (y/n): y
 Enter Professor name: XYZ
 Enter no. of preffered subjects: 2
 Enter subject :INTRODUCTION TO BIOINFORMATICS
BIOINFORMATICS LAB
Professor Added!
```

UPDATED ASSIGNMENT OF COURSES



```
Professor Rachana
1) Course ENVIRONMENTAL STUDIES and No of sections :10

Professor Smriti Gaur
1) Course MICROBIOLOGY and No of sections :8
2) Course MICROBIOLOGY LAB and No of sections :2

Professor Sonam Chawla
1) Course GENETICS AND DEVELOPMENTAL BIOLOGY and No of sections :10

Professor Shazia
1) Course INTRODUCTION TO BIOINFORMATICS and No of sections :10

Professor Sujata Mohanty
1) Course GENETICS AND DEVELOPMENTAL BIOLOGY LAB and No of sections :10

Professor Neeraj Wadhwa
1) Course GENETICS AND DEVELOPMENTAL BIOLOGY LAB and No of sections :10

Professor Chakresh Jain
1) Course BIOINFORMATICS LAB and No of sections :10

Professor XYZ
1) Course BIOINFORMATICS LAB and No of sections :10


Process returned 0 (0x0)   execution time : 1.250 s
Press any key to continue.
```

CONCLUSION

From the output screenshots above, it is evident that we have reached to a solution to optimise the teacher workload assignment.

Further, we would like to add that this problem also holds great opportunities for further exploration such as increasing the teacher intake limit etc.

REFERENCES

E-book

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein , Introduction to Algorithms, MIT Press, 3rd Edition, 2009

[2] Steven Skiena ,The Algorithm Design Manual, Springer; 2nd edition , 2008

Internet sites

[1] https://github.com/Sebastiao-Assuncao/Process-Processer_Assignment

[2] https://www.cs.toronto.edu/~jepson/csc373/tutNotes/tutEx_NetFlow2_2018S_Soln.pdf

[3] https://www.geeksforgeeks.org/

[4] https://stackoverflow.com/

Lecture

Suma Dawn. CSE. Class Lecture, Topic: “Max Flow Using Ford Fulkerson”

Jaypee Institute of Information Technology

Software

[1] Code::Blocks

THANK YOU