GROUP MEMBERS:

MISRA YAVUZ, 2016400135

ALGI KANAR, 2016400123

PROBLEM DESCRIPTION

Problem: Writing a crawler which will take Bogazici University's OBIKAS registration pages and extract course offering information. The crawler needs to be able to obtain information from 1998 to 2019.While crawling we need to arrange the information step by step through departments and each department's courses according to their instructors, undergraduate course counts and graduate course counts. Then displaying all the attained values at console in csv format. Even though these mentioned operations are straightforward, overcoming certain possibilities' adverse conditions and selecting proper design choice are obligatory to manage a healthy result.

PROBLEM SOLUTION

The plausible way to solve this task is to separate it into 2 main parts which can be named as processing the input, then at last giving the output.

The former one is basically relying on using dictionaries and sets. At the beginning since the semesters and departments determined beforehand, we wrote the departments names and addresses' name parts manually. For the semesters we wrote a for loop to store the semesters as both console format and url address formats in arrays. After setting the start and finish semester we commenced to process them. Instead of traversing semesters and then departments, we proceed as taking the departments first and then semesters. This helped assigning the evaluated instructor count, undergraduate course count and graduate course count easily for the proceeding accesses to those values. While traversing a department's semesters (which is the second segment of the overall dictionary), we fill the dictionaries other segments in the order of deptname, coursecode, coursename and semester. For the sake of obtaining these segments we updated the values in bottom up manner. All segments are representing a separate dictionary doing the task indicated in the name of that dictionary. We added the deptname segment to separate the same department coded but different named departments such as ASIAN STUDIES and ASIAN STUDIES WITH THESIS. Furthermore, about the naming problems, we add an if condition to prevent losing the info of a same code but different named individual course of a department. We also put if blocks to prevent unnamed instructors to get into set_inst which shown as STAFF STAFF. Our evaluation of undergraduate and graduate counts happens in the first part last section. We fetch the codes of courses and we add them as graduate course if they are start 5 or above. All other courses are treated like an undergraduate course. For example in the situation of SFL P01, we take it as an undergraduate course. We assign the counts in separate dictionaries; first for each distinct semester, second

for all semester added and lastly, via for looping set_courses we get the total semester count without any duplicate values.

For the second part, we have the all departments info filled dictionary mydict_dept and undergraduate and graduate course info filled dictionaries which are countlist, totaldict and tt. In our design choice we are evaluating the instructors and the line of existence of courses (ex: ,x, , ,x,)right before the display. After traversing till the bottom of the information, the first action is to attain the line of existence by holding prevsemester information which generates the exact spaces and crosses. The second action is, by using the instructor sets which are separated in terms of the time of holding and releasing the data. The set of sa is to take same semesters' courses instructors, therefore it should be cleared in each semester for a lesson but then need to be taken out again with the help of separator dictionary in order to get the column semester instructor info. On the other hand, s is just responsible for each course in all semesters and stotal will not be cleared till the new department comes since it is a set for all instructors of a department in general. For the sake of pointing out one more possibility, we add the if block **if len(instructor)==0** in case of an instructor data loss because of an empty set.

After gathering all the information that we wanted, we first put the course line results in an array. Secondly, we output the headers containing the instructor, undergraduate and graduate course related line and then the arrays result values for each department in csv format. The csv format is gained by putting the comas for each new information.

CONCLUSION

While doing our project, we first understood the basics of web crawling and studied the registration page in detail. As our knowledge about the topic build up, we begin to write the code to achieve some tasks. Our algorithm is pretty straightforward and proceeds step by step. After completing the basics, we tested our program a lot to recognize if some uncommon possibilities occur. We encountered the departments with same code but with different names and treated them as distinct departments. Another issue was the course name modifications throughout the years. In our design choice, we treated courses according to their course codes; so to solve this problem, we kept the latest name of the course.

In the end, our code works fine and gives the information and statistics as asked. All departments and courses are sorted alphabetically. We used a significant amount of data structures, all are explained in the comments.