# Data Structure Fall 2020, programming Assignment #1 README

B06502009 廖偉霖

*The following shows the simple description, rough time complexity and rough space complexity of functions in the source code in order.*

1. getpage():

*This function will return the page rank matrix as a shape of (501,501) numpy.ndarray and the # of connections of each page as a dictionary.*

Time complexity → O(n) read n files and store into len(n) list and dictionary

Space complexity → O($n^2$) make $n \times n$ matrix

2. page_rank(d, DIFF, page_mat):

*This function will return the latest update and the its rank based on the d and DIFF, where d should be a float from 0~1, and DIFF is recommended to be a float under 0.1.*

Time complexity → O($n^2$) due to the calculation of inner product of nxn matrix

Space complexity → O($n^2$) same reason with time complexity

3. write_Q1_ans(rank, update, connect, d, DIFF):

*This function will write our result (the rank of the pages, connections of each pages and their importance value) into the corresponding files.*

Time complexity → O(n)

Space complexity → O(n) rank and update are both list with length n; connect is a dictionary with length n

4. Q1():

*This function will make files with some combinations of d and DIFF by executing functions: write_Q1_ans(), page_rank(), getpage().*

Time complexity → O($n^2$) the combination of the functions involved

Space complexity → O($n^2$) the combination of the functions involved

5. getword():

*This function will return a word_list (list) and a word_dictionary (dict). word_list contains all the unique words contained in every page (this web), and word_dictionary contains page numbers as its keys with type of str and the words in that page as its value with type of list.*

Time complexity → O(n)

Space complexity → O(n) both list and dict are in size len(n)

6. find_page(word_list, word_dict):

*This function makes a reverseindex.txt file, which contains every word in the web and the pages that contain those words.*

Time complexity → O(n)

Space complexity → O(n)

7. Q2():

*This function simply executes functions: getword(), find_page().*

Time complexity → O(n)

Space complexity → O(n)

8. search_engine(d, DIFF, page_mat, input_list, word_list, word_dict):

*This function will find the top 10 hit pages of the words contained in the input_list. If the item in input_list is a single word input, then it will just print the reuslt (top 10 page). If the item in input_list are multiple words input, then it will print two results, which are AND and OR. AND means the page should contain all the input words, while OR means the page can contain any of the input words. Lastly, the function will write the result to the corresponding file.*

Time complexity → O($n^2$) actually, it is based on the size of the input file and the # of pages (501). In this case the input file is "list.txt". If the size is very large then the time complexity would be O($n^2$); if not, then it can be viewed as constant size, like single input for extreme case, then the time complexity would be O(n).

Space complexity → O(n)

9. Q3():

*This function execute search_engine() with some combinations of d and DIFF, and execute getpage(), getword() for required input for search_engine().*

Time complexity → O($n^2$) the combination of the functions involved

Space complexity → O($n^2$) the combination of the functions involved

---

*Functions which are not presented above are used for searching words manually. There are 2 options provided, one allows the users to fix d, DIFF before searching; the other needs to assign d, DIFF before searching every time. The searching accepts single and multiple inputs (with space separated) as well. The time/space complexity of both options are O ($n^2$), but obviously the latter option cost more time.*

** There is only 1 source code file, so just run the code with python hw.py on terminal, then it will work right away with instruction printed on the terminal.*