(Data Structures and Algorithms Design)

Academic Year 2022-2023

Assignment – PS12- [Weightage 25%]

1. Problem Statement –Question 1

Let us assume that a dictionary is implemented using an array data structure. Each entity in the dictionary consists of a word and its meaning. A file with a list of words and its meaning is given as input, which represents the array data structure. In-order to improve the time complexity, a Binary Search Tree is proposed for the dictionary implementation.

The aim is to implement the dictionary using Binary Search Tree. The data for the dictionary is read from the input file and should be inserted into the Binary Search Tree (BST), based on the alphabetical weightage of the value corresponding to each word. For example, a word starting with character 'A' have less weightage than 'B'.

Some examples are

Cat has less weightage than Dog
Cat has less weightage than Cell
Calendar has less weightage than Cat

The following functions are expected to be implemented:

1. **Create a dictionary**: the data need to be read from the file ArraydictPS12.txt one by one and inserted into the BST.
2. **Search a word** and display it's corresponding meaning if it is present in the dictionary. Read the search word from the file **"promptsPS12Q1.txt"** with the tag "SearchWord:". The search function should read each word and check whether the word is present in the BST dictionary or not. If yes it should add the word and its meaning to **outputPS12Q1.txt**. If not it should show the word name "not found" in place of the description. Repeat this for all the words in the file **promptsPS12Q1.txt**
3. **Substring search**: For a given sub string, list all the words with the sub strings in it. Program should read the input from the file "**promptsPS12Q1.txt**" with the tag "SubString:" and list all the words starting

with the given sub string. The result should be stored in a file **outputPS12Q1.txt**

For example: if the prompt file contains the line

SubString: ca

after executing the content of outputPS12Q1.txt looks like

------ Substring Search for ca----------

cat, calendar

Requirements:

1. Implement the above problem statement using Python 3.7
2. Create a dictionary with BST data structure. Read the words and its meaning from the file ArraydictPS12Q1.txt
3. Perform search operations in the resulting BST. The search words are in the file promptsPS12Q1.txt.
4. Perform substring search in the resulting BST. The search sub strings are in the file promptsPS12Q1.txt.
5. Perform an analysis for the features above and give the running time in terms of input size: n.

Sample input files

Provide the word and meaning of the word in the input file separated by a " / "

Sample ArraydictPS12Q1.txt

cat / small domesticated carnivorous mammal from cat family
camel / long-necked ungulate mammal
air / the invisible gaseous substance
sky / region of the atmosphere
dog / a domesticated carnivorous mammal
cell / smallest structural and functional unit of an organism

Sample promoptsPS12Q1.txt

SearchWord: cat
SearchWord: sky

SearchWord: shake

SubString: ca

Note that the input and output data shown here is only for understanding and testing, the actual file used for evaluation will be different

Sample output and output files

Below is a sample output after executing the program

Sample outputPS12Q1.txt

----------Reading from file ArraydictPS12.txt ----------
BST Created with 6 nodes

---------- Search words ----------
cat - small domesticated carnivorous mammal from cat family
sky - region of the atmosphere
shake - not found

---------- Sub String: ca ----------
cat, camel

Note that the input and output data shown here is only for understanding and testing, the actual file used for evaluation will be different

2. Deliverables

1. Word document **designPS12Q1_<student_id>.docx** detailing your design and time complexity of the algorithm.
2. **InputPS12Q1.txt and PromptsPS12Q1.txt** is file used for testing
3. **OutputPS12Q1.txt** file generated while testing
4. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

Zip all of the above files including the design document file in a folder with the name:

[Student id]_PS12Q1.zip .

3. Problem Statement –Question 2

There are N different models of drones manufactured at a drone manufacturing unit of the Air Force. Each drone must go through 2 major phases: 'manufacturing' and 'flight testing'. Obviously, 'manufacturing' must happen before 'flight testing'. The time for 'manufacturing' and 'flight testing' (pmi and fti for ith drone) for every drone may be different. If we have only 1 unit for 'manufacturing' and 1 unit for 'flight testing', how should we produce N drones in a suitable order such that the total production time is minimized and the drones are ready to deploy at the earliest?

Requirements:

1. Write a Greedy Algorithm to select the drone 'manufacturing' and 'flight testing' in such a way that total production time is minimized.
2. Analyse the time complexity of your algorithm.
3. Implement the above problem statement using Python 3.7.

Sample Input:

For example, if there are 6 different drones in total and time for each drone 'manufacturing' and 'flight testing' are given as shown

Drone i	pm_i (minutes)	ft_i (minutes)
1	5	7
2	1	2
3	8	2
4	5	4
5	3	7
6	4	4

Input should be taken in through a file called "**inputPS12Q2.txt**" which has the fixed format mentioned below using the "/" as a field separator:
<drone i> / < pm_i (minutes)> / <ft_i (minutes)>

Ex:

1 / 5 / 7

2 / 1 / 2
3 / 8 / 2
…

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

Sample Output:

Output the type of representation followed by a space-separated answer.

Drones should be produced in the order: 2, 5, 6, 1, 4, 3.
Total production time for all drones is: 28
Idle time of flight testing unit: 2

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

Display the output in **outputPS12Q2.txt**.

4. Deliverables

1. Word document **designPS12Q2_<student id>.docx** detailing your design and time complexity of the algorithm.
2. **inputPS12Q2.txt** file used for testing
3. **outputPS12Q2.txt** file generated while testing
4. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

   Zip all of the above files including the design document file in a folder with the name:

   [Student id]_ PS12Q2.zip

5. Instructions

1. It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
2. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.

3. For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.
4. Make sure that your read, understand, and follow all the instructions
5. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
6. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
7. Run time analysis is to be provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

Instructions for use of Python:

1. Implement the above problem statement using Python 3.7.
2. Use only native data types like lists and tuples in Python, do not use dictionaries provided in Python. Use of external libraries like graph, numpy, pandas library etc. is not allowed. The purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.
3. Create a single *.py file for code. Do not fragment your code into multiple files.
4. Do not submit a Jupyter Notebook (no *.ipynb). These submissions will not be evaluated.
5. Read the input file and create the output file in the root folder itself along with your .py file. Do not create separate folders for input and output files.

8. Deadline

1. The strict deadline for submission of the assignment is **<Refer E-learn Course page>.**
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will not be evaluated.

7. How to submit

1. All the deliverables(zip files from section 2 and 5) must be combined in one zip file and named as **<Student ID>.zip**

2. Assignments should be submitted via E-Learn > Assignment section. Assignment submitted via other means like email etc. will not be graded.

8. Evaluation

1. The assignment carries **13 +12 =25 Marks**.
2. Grading will depend on
    i. Fully executable code with all functionality working as expected
    ii. Well-structured and commented code
    iii. Accuracy of the run time analysis and design document.
3. Every bug in the functionality will have negative marking.
4. Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.
5. Use of only native data types and avoiding libraries like numpy, graph and pandas will get additional marks.
6. **Plagiarism will not be tolerated. If two different groups submit the same code, both teams will get zero marks**.
7. Source code files which contain compilation errors will get at most 25% of the value of that question.

9. Readings

Text book: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, WileyGoodrich, Roberto Tamassia, 2006, Wiley (Students Edition). **Chapters:** 2.3,5.1