BT3051 - ASSIGNMENT 3

BE23B016

K.Varunkumar

12/09/2024

1 Problem 1

A pharmaceutical company has many drugs in its production pipeline and will select a drug for large scale production based on two criteria:

- Market Demand (High Demand = High Priority)
- Production Cost (High Production Cost = Low Priority)

The criteria can change with time.

The company's R&D team works on new drugs which can be added to the production pipeline. Choose a suitable data structure to represent the drugs in the production pipeline along with their market demand and production cost. Implement this as a class where you can add drugs to the production pipeline, change the priority of the company, and select a drug for production.

Simulate a company's one year production plan (Jan - Dec) based on the following conditions:

1. At the start of the year, the company had 10 drugs on the production list. In January, the production criteria was market demand. A drug was selected based on its market demand score.
2. Till April, the same drug was continued while 4 more drugs were added (1 in the beginning of each month) to the production list. In May, the company changed its criteria to production cost and an appropriate drug was selected from the list and was continued till the end of the year while new drugs were added to the list every month.
3. Print the drug with the highest priority at the end of each month (not the one in production).

1.1 Requirements

We require a Data structure that:

- Allows for efficient insertion of new data
- Allows for the efficient sorting of the data on 2 different parameters (Market Demand - MD, and Production Cost - PC).

The above requirements can be met using a Heap Data structure.

1.2 Functions Used:

1. `{insert(self, data, mode="MD")}`

   Purpose: To insert a new element into the heap, the heap is adjusted according to mode (either max-heap or min-heap).

   Parameters:
   data: The data to be inserted into the heap. Must be a list of the of form [Drug_Name, Drug_MD, Drug_PC].
   mode: The mode of the heap (either "MD" for max-heap using MD as the parameter or "PC" for min-heap) using PC as the parameter. Default is "MD".

2. `{heapify(self, database, mode="MD")}`

   Purpose: Creates a heap from the data in the database.
   This function repeatedly calls the insert function to build the heap from the list.

   Parameters:
   database: A list of elements to insert into the heap.
   mode: The mode of the heap, used to set the priority for the repeated call of the insert function.

3. `{breakdown(self)}`

   Purpose: Returns a list by the removing elements of the heap, from left to right, layer by layer. Used to reconstruct the heap for a change in priority.

   Parameters:
   None

4. `{priorchange(self, newmode)}`

   Purpose: Changes the mode of the heap based the priority of the company and re-heapifies the existing data.

   Parameters:
   newmode: A string that can be either:
   "MD" - if the companies priority is market demand
   "PC" - if the companies priority is production cost.

1.3 Running Test-cases

OUTPUT 1

```
The data of length 10 has been uploaded:
================================================================
Priority Drug at the start of the Year is: Metoprolol
================================================================
1 ) Priority Drug in January is: Metoprolol
2 ) Priority Drug in February is: Metoprolol
3 ) Priority Drug in March is: Metoprolol
4 ) Priority Drug in April is: Metoprolol
================================================================
THE PRIORITY HAS CHANGED.
================================================================
5 ) Priority Drug in May is: Propranolol
6 ) Priority Drug in June is: Propranolol
7 ) Priority Drug in July is: Omega-3-acid ethyl esters
8 ) Priority Drug in August is: Omega-3-acid ethyl esters
9 ) Priority Drug in September is: Omega-3-acid ethyl esters
10 ) Priority Drug in October is: Fluticasone/Salmeterol
11 ) Priority Drug in November is: Fluticasone/Salmeterol
12 ) Priority Drug in December is: Fluticasone/Salmeterol
================================================================
22
================================================================
[['Ramipril', 25, 62],
['Lorazepam', 69, 56],
['Salmeterol', 73, 98],
['Ibuprofen', 94, 35],
['Trazodone', 82, 98],
['Acetaminophen', 82, 38],
['Acyclovir', 47, 81],
['Carvedilol', 32, 100],
['Metoprolol', 94, 85],
['Morphine', 76, 68],
['Fluconazole', 97, 42],
['Fenofibrate', 7, 12],
['Loratadine', 75, 49],
['Gabapentin', 19, 30],
['Insulin', 74, 33],
['Hydrocodone', 39, 39],
['Amlodipine', 49, 84],
['Cephalexin', 40, 111],
['Propranolol', 89, 17],
['Atorvastatin', 93, 18],
```

```
['Omega-3-acid ethyl esters', 18, 9],
['Fluticasone/Salmeterol', 31, 1]]
================================================================
```

OUTPUT 2

```
The data of length 10 has been uploaded:
================================================================
Priority Drug at the start of the Year is:  Gabapentin
================================================================
1 ) Priority Drug in January is:  Pravastatin
2 ) Priority Drug in February is:  Simvastatin
3 ) Priority Drug in March is:  Simvastatin
4 ) Priority Drug in April is:  Simvastatin
================================================================
THE PRIORITY HAS CHANGED.
================================================================
5 ) Priority Drug in May is:  Spironolactone
6 ) Priority Drug in June is:  Dextromethorphan
7 ) Priority Drug in July is:  Dextromethorphan
8 ) Priority Drug in August is:  Dextromethorphan
9 ) Priority Drug in September is:  Dextromethorphan
10 ) Priority Drug in October is:  Dextromethorphan
11 ) Priority Drug in November is:  Dextromethorphan
12 ) Priority Drug in December is:  Prednisone
================================================================
The number of drugs at the end of the year in the database is: 22
================================================================
Printing the dataset at the end of the year for verification.
[['Omega-3-acid ethyl esters', 43, 100],
 ['Hydrocodone', 58, 68],
 ['Cephalexin', 53, 70],
 ['Doxycycline', 2, 62],
 ['Naloxone', 74, 70],
 ['Verapamil', 84, 54],
 ['Venlafaxine', 47, 93],
 ['Valacyclovir', 16, 96],
 ['Glipizide', 34, 99],
 ['Morphine', 37, 86],
 ['Ibuprofen', 41, 92],
 ['Tramadol', 83, 34],
 ['Amoxicillin', 67, 59],
 ['Levothyroxine', 20, 52],
 ['Hydrochlorothiazide', 27, 45],
 ['Pravastatin', 77, 50],
 ['Gabapentin', 72, 77],
 ['Spironolactone', 55, 31],
 ['Canagliflozin', 41, 36],
 ['Simvastatin', 96, 41],
 ['Dextromethorphan', 3, 15],
```

```
 ['Prednisone', 8, 12]]
==========================================================================
```

---

2 Problem 2

Two companies PiedPiper and Hooli are in their hiring process. They receive 10,000 resumes from the same set of applicants. Both the companies evaluate the resumes and score each of them between 0 to 1000. They sort the resumes from highest to lowest and add them to their database after which they call the top 100 applicants for the interview process. The person with the highest resume score will be the first one to be interviewed

Both companies use different strategies to sort the resumes:

- PiedPiper chooses a random resume and compares the others with that in a recursive fashion
- Hooli divides the resumes into two halves to sort them recursively

Simulate this interview process and check which company goes through the interview process quickly and explain why.

2.1 Requirements

2.1.1 PiedPiper

PiedPiper chooses a random resume from the list and uses that resume's score as the template for comparison.

This is essentially a Quicksort on the list of resumes using a randomly chosen pivot each time.

2.1.2 Hooli

Hooli chooses to divide the set of resumes into halves recursively and sort them as they are put back together.

This is essentially just a Mergesort.

2.2 Running Test-cases

The output time was averaged over 500 runs.

```
The average time taken by Hooli's system to sort is: 0.060301839399999495
The average time taken by Piedpiper's system to sort is: 0.045610383999998706
```

On Average Piedpiper's method to sort the candidates is faster than the approach followed by Hooli.
Eventhough both companies follow algorithmic approaches that have an average time complexity of $O(n \cdot \log(n))$

---

3 Problem 3

You are designing a digital note-taking system where you can add, update, retrieve and delete notes (heading and content). In this system, unique identifiers are generated from the note heading. Implement this using python.

Note: Python dictionaries are not allowed. You can use built-in hash functions

3.1 Requirements

The problem requires us to create a hashmap using python's inbuilt hashfunction and ensure it has the following functions.

- write/add
- append
- update
- read/retrieve
- delete (Notes and heading)

3.2 Functions Used:

1. `{write(self, heading, text)}`

   Purpose: Adds a new heading to the database with the corresponding notes. The position of the inserted data in the array is determined by it's the headings hash value.

   Parameters:
   heading: The heading or title under which the notes are to be stored. It is a string.
   text: The notes associated with the heading.

2. `{update(self, heading)}`

   Purpose: Updates (overwrites) the existing notes associated with a given heading.
   Can be used to delete only the notes of a particular heading without deleting the heading itself.

   Parameters:
   heading: The heading whose notes is to be updated.

3. `{append(self, heading)}`

   Purpose: Can be used to append additional notes to the existing notes associated with a given heading.
   The existing notes are displayed to ensure ease of appending.

   Parameters:
   heading: The heading to which new notes are to be appended.

4. `{read(self, heading)}`

   Purpose: Used to read the Notes associated with a particular heading.

   Parameters:
   heading: The heading to whose notes are required to be read.

5. `{delete(self, heading)}`

   Purpose: Used to drop a particular heading and it's associated notes from the database array.

   Parameters:
   heading: The heading which is to be dropped from the database.

3.3 Running Test-cases

The write function was used, and they were read. The output was as shown below.

```
Optimizing CRISPR-Cas9 Gene Editing Efficiency in Mammalian Cells
 CRISPR-Cas9 gene editing technology has revolutionized genetics by allowing precise alterations in mammalian DNA. Its optimiza
tion can enhance gene therapy to treat genetic disorders like cystic fibrosis and muscular dystrophy. By improving efficiency,
researchers can reduce off-target effects, ensuring safer therapeutic applications. In cancer research, CRISPR helps target mut
ations with more precision. Additionally, it holds potential for regenerative medicine, creating cells that can self-repair dam
aged tissues.
================================================================================================
Bioreactor Design for Enhanced Microbial Fuel Cell Performance
 Bioreactors optimized for microbial fuel cells (MFCs) can increase the efficiency of bioelectricity production from organic wa
ste. MFCs convert chemical energy into electricity through microbial metabolism, providing a sustainable energy source. Better
bioreactor designs enhance microbial growth, improving the electron transfer process. This technology can be applied to wastewa
ter treatment plants, generating renewable energy while purifying water. It has future potential for powering remote areas with
limited access to conventional electricity.
================================================================================================
Improving Drug Delivery Systems Through Nanoparticle Engineering
 Nanoparticle-based drug delivery systems allow for precise targeting of diseased tissues, such as tumors, while minimizing sid
e effects on healthy cells. By engineering nanoparticles to carry drugs to specific cells, the efficacy of treatments like chem
otherapy can be greatly improved. These systems can also be designed to respond to environmental triggers like pH or temperatur
e changes, releasing drugs only at the site of action. Nanoparticles enhance the delivery of vaccines, improving immune respons
e. This technology has broad applications in personalized medicine and targeted therapies.
================================================================================================
Synthetic Biology Approaches to Combat Antibiotic Resistance
 Synthetic biology provides tools to design new antimicrobial peptides or enzymes to target antibiotic-resistant bacteria. This
approach can create custom gene circuits in bacteria that sense and destroy harmful pathogens without harming beneficial microb
iota. Engineered bacteriophages can also be used to selectively kill resistant bacteria, offering an alternative to traditional
antibiotics. Synthetic biology can produce novel antibiotics that bypass conventional resistance mechanisms. These innovations
are critical in addressing the global crisis of antibiotic resistance in healthcare and agriculture.
================================================================================================
```

Demonstrating the usage of the append function. As shown below.

```
#Appending notes to a heading
#Trying to append when the heading is not present
library.append("Improving Stress Resistance in Crops Using Gene Editing Tools")

#Trying to append when the heading is present
library.write("Improving Stress Resistance in Crops Using Gene Editing Tools","Gene editing tools like CRISPR can enhance a
library.append("Improving Stress Resistance in Crops Using Gene Editing Tools")
library.read("Improving Stress Resistance in Crops Using Gene Editing Tools")
```

```
Invalid Heading, Not in Database
==========================================================================================
Enter the new notes:Gene editing tools like CRISPR can enhance a crop's ability to withstand environmental stresses such as dro
ught, heat, and salinity. By modifying specific genes involved in stress response, crops become more resilient, ensuring stable
yields under changing climate conditions. Applications include engineering rice, wheat, or corn to thrive in water-scarce or hi
gh-temperature regions.
Enter F to finish appending, any other key to continue:x
Enter the new notes:Enter the new notes:This technology contributes to food security and sustainable agriculture by enabling cr
ops to perform better under stress. It helps farmers adapt to climate change and extreme weather events.
Enter F to finish appending, any other key to continue:f
==========================================================================================
Improving Stress Resistance in Crops Using Gene Editing Tools
 Gene editing tools like CRISPR can enhance a crop's ability to withstand environmental stresses such as drought, heat, and sal
inity.Gene editing tools like CRISPR can enhance a crop's ability to withstand environmental stresses such as drought, heat, an
d salinity. By modifying specific genes involved in stress response, crops become more resilient, ensuring stable yields under
changing climate conditions. Applications include engineering rice, wheat, or corn to thrive in water-scarce or high-temperatur
e regions.Enter the new notes:This technology contributes to food security and sustainable agriculture by enabling crops to per
form better under stress. It helps farmers adapt to climate change and extreme weather events.
------------------------------------------------------------------------------------------
```

Demonstrating the use of the update function.As shown below.

```
#Updating the notes of a heading
library.write("Biosensor Platforms for the Detection of Biological Warfare Agents","Sample text that is to be removed in pla
library.read("Biosensor Platforms for the Detection of Biological Warfare Agents")

library.update("Biosensor Platforms for the Detection of Biological Warfare Agents")
library.read("Biosensor Platforms for the Detection of Biological Warfare Agents")
```

```
Biosensor Platforms for the Detection of Biological Warfare Agents
 Sample text that is to be removed in place of proper information.
==========================================================================================
The current stored data is:
Biosensor Platforms for the Detection of Biological Warfare Agents
 Sample text that is to be removed in place of proper information.
Enter the new notes:Enter the new notes:Biosensors can be engineered to detect the presence of biological warfare agents such a
s anthrax, smallpox, or nerve toxins in the environment. These platforms use biological components like antibodies, enzymes, or
engineered cells to detect pathogens or toxins with high specificity and sensitivity. Applications include military defense, pu
blic health monitoring, and environmental safety. Rapid detection of biowarfare agents allows for immediate response and contai
nment, preventing widespread harm. This technology is critical for national security and bio-surveillance.
Enter F to finish updating, any other key to continue:f
==========================================================================================
Biosensor Platforms for the Detection of Biological Warfare Agents
 Enter the new notes:Biosensors can be engineered to detect the presence of biological warfare agents such as anthrax, smallpo
x, or nerve toxins in the environment. These platforms use biological components like antibodies, enzymes, or engineered cells
to detect pathogens or toxins with high specificity and sensitivity. Applications include military defense, public health monit
oring, and environmental safety. Rapid detection of biowarfare agents allows for immediate response and containment, preventing
widespread harm. This technology is critical for national security and bio-surveillance.
==========================================================================================
```

Demonstrating the use of the delete function.As shown below.

```
#Deleting the data pertaining to a heading
library.delete("Microbial Synthesis of Industrial Enzymes for Green Chemistry")
library.delete("Synthetic Biology Approaches to Combat Antibiotic Resistance")
```

```
Microbial Synthesis of Industrial Enzymes for Green Chemistry
 Microorganisms can be engineered to produce industrial enzymes used in green chemistry processes, reducing the need for harmfu
l chemicals. These enzymes catalyze reactions under mild conditions, lowering energy consumption and minimizing waste productio
n. Applications include the synthesis of pharmaceuticals, biofuels, and biodegradable plastics. Microbial enzyme production sup
ports more eco-friendly industrial practices by replacing harsh chemical catalysts. It offers an alternative to traditional man
ufacturing methods that are harmful to the environment.

 The above data has been deleted.
==========================================================================================
Synthetic Biology Approaches to Combat Antibiotic Resistance
 Synthetic biology provides tools to design new antimicrobial peptides or enzymes to target antibiotic-resistant bacteria. This
approach can create custom gene circuits in bacteria that sense and destroy harmful pathogens without harming beneficial microb
iota. Engineered bacteriophages can also be used to selectively kill resistant bacteria, offering an alternative to traditional
antibiotics. Synthetic biology can produce novel antibiotics that bypass conventional resistance mechanisms. These innovations
are critical in addressing the global crisis of antibiotic resistance in healthcare and agriculture.

 The above data has been deleted.
==========================================================================================
```

It is also shown, eventhough chaining is performed to ensure collisions don't break the system. They rarely occur.

---

4 Problem 4

With a list of protein IDs, their names and function, construct a tree-like structure that you have studied which can efficiently.

1. Add proteins
2. Find proteins and their function through their IDs

Provide examples for adding and finding proteins.

4.1 Requirements

The question here requires the use of structure with efficient insertion and access algorithms. It is also stated to use a tree like structure.

- Heap-Insertion is $O(\log n)$, whereas accessing a data point is $O(n)$.
  However, Heap using memoization for location can be traversed in log(n) time complexity.
- Binary Trees - Provided we implement balanced Binary trees, Insertion is $O(\log n)$, similarly accessing an element through it's value is also $O(\log n)$.

Since both Adelson-Velsky and Landis tree (AVL Trees), or other similarly balanced Binary search trees and Heaps with memoization offer similar complexities, a Heap with memoization will be implemented due to it being easier to setup.

Thus we shall implement an Heap and define the following functions in it.

- insert(data)
- access(protein ID)

Where data is a list of the form : [⟨ protein name ⟩ , ⟨ protein function ⟩ , ⟨ protein ID ⟩]

And access returns the data for a given protein ID.

4.2 Functions Used:

1. `{insert(self, data)}`

   Purpose: Inserts a new node into the heap.
   If the node with the same ID already exists, it prints an error message and terminates. Otherwise, it finds the terminal node and adds the new node to the heap.The the data structure is adjusted post insertion to satisfy heap property.

   Parameters: data: A list containing the data to be inserted.

2. `{heapify(self, input)}`

   Purpose: Inserts a list of data into the heap by calling the insert function for all the elements in input. If there is a need to check if all the data points have been inserted, the function returns the total number of data points inserted as well.

Parameters:
input: A list containing the data_lists to be inserted.

3. `{access(self, ID)}`

Purpose: Given an ID, it accesses the corresponding node and prints its information (name and function). If the ID is invalid (not present in the database), it prints an error message.

Parameters:
ID: The unique ID of the node to be accessed.

4.3 Running Test-cases

The data in the jupyter notebook, was input into the database and following was the output obtained.

```
database=prodb()
database.heapify(data)

database.access(8247910365274)

database.access(9253184076591)

database.access(2143908576405)

database.access(1143908576405)

database.access(1295803745910)

database.access(1295803745989)
```

```
Name: Protein 95
Function Plays a role in blood clotting
==========================================================================================================
Name: Protein 51
Function Involved in muscle contraction
==========================================================================================================
Name: Protein 83
Function Plays a role in blood clotting
==========================================================================================================
Invalid ID
==========================================================================================================
Name: Protein 98
Function Provides structural support to tissues
==========================================================================================================
Invalid ID
==========================================================================================================
```

5 Problem 5

A large queue of people is waiting in a one-way passage to buy tickets for the newly released movie WOAT. Unfortunately, the show is canceled, and people need to go back in the reverse order they arrived.

Use appropriate data structure to represent this; add 1000 people (identified by unique IDs 1 to 1000) to the queue and remove them when they are leaving.

5.1 Requirements

The question requires us to use a data structure that is efficient at inserting at the last position and removing the person from the last position as well.

5.2 Functions Used:

1. `{push(self, ID)}`

   Purpose: Adds (pushes) a new element to the top of the stack.

   Parameters:
   ID: The customers in the Queue.

2. `{dump(self, data)}`

   Purpose: Used to dump a set of elements into the stack. Repeatedly calls push.

   Parameters:
   data: List of IDs of customers in the Queue in order (bottom to top).

3. `{pop(self)}`

   Purpose: Used to remove the last/topmost element in the stack.
   Returns the element removed from the stack.

   Parameters:
   None

5.3 Running Test-cases

Since the array of length 1000 cannot be displayed here, it is shown in the jupyter notebook itself.

Consideration was taken to avoid duplicates in customer IDs.
