Online Auction Platform

BTech/II Year CSE/IV Semester

19CSE212/Data Structures and Algorithms

Case Study Report

Roll.No	Name
CB.EN.U4CSE21227	Jaswanth P
CB.EN.U4CSE21215	Makarand

Department of Computer Science and Engineering

Amrita School of Computing, Coimbatore

Amrita Vishwa Vidyapeetham

2022 -2023 Even Semester

Table of Contents

Chapter 1 Introduction :

Hybrid data structures combine the strengths of different data structures to optimize data organization and manipulation. By leveraging the advantages of arrays, linked lists, trees, and hash tables, these structures enhance efficiency and flexibility. They offer fast searching, efficient insertion and deletion, and can handle complex data relationships.

Hybrid data structures excel in tasks like graph traversal, network analysis, and database management. They adapt to evolving needs, allowing customization and specialization. By striking a balance between efficiency and flexibility, hybrid data structures provide a powerful toolkit for developers to tackle real-world data challenges and create more efficient and adaptable systems.

1.1 Objective :

The objective of our project is to design and implement a hybrid data structure that combines the strengths of a hash table and a heap. This hybrid structure will be utilized to create an efficient online auction platform.

The key objectives of our project include:

1. Efficient Item Management: Develop a hash table component of the hybrid data structure to store and manage items in the auction. Each item will be associated with a unique item ID, allowing for quick retrieval and manipulation.

2. Effective Bid Management: Utilize the heap component of the hybrid data structure to handle bids placed on items. The heap will maintain the highest bid for each item, along with the bidder's name and bid amount. Implement insertion and update algorithms to handle bids efficiently.

3. Smooth Bid Placement: Allow users to place bids on specific items by providing the item ID, bidder name, and bid amount. Implement algorithms to check if the bid amount exceeds the current highest bid and insert the bid into the heap component accordingly.

4. Comprehensive Item and Bid Information: Develop methods to view all items currently on auction, including their details such as item ID, name, description, bid start price, and the current highest bid amount (if any).

5. Bid History Display: Allow users to view the bid history for a specific item. Retrieve the bid information from the heap component and display it in descending order based on bid amount.

1.2 Significance of Hybrid Data Structures :

Incorporating a hybrid data structure, such as a combination of a hash table and a max heap, in our online auction platform holds significant value. It allows us to optimize data organization and manipulation, enhancing efficiency and flexibility. With the hash table component, we can efficiently manage items by quickly adding and retrieving them based on their unique IDs. The max heap component enables effective bid management, providing fast access to the highest bids for each item.

In conclusion, this ensures a streamlined bidding process and accurate tracking of the highest bids. By leveraging this hybrid approach, we can deliver a responsive user experience, handle fluctuations in demand, and create a seamless auction platform that maximizes performance and user satisfaction.

Chapter 2 Implementation:

The implementation of the hybrid data structure in our online auction platform involves the integration and interplay of two constituent data structures: a hash table and a max-heap.

2.1 Hash Table and Max Heap Implementation :

The AuctionItem class represents an item available for auction. It contains attributes such as item_id, item_name, item_description, and bid_start_price.
The HybridAuctionPlatform class serves as the main auction management system. It contains two nested classes:

1. MaxHeap: This class implements a max heap data structure to manage the bids. Bids are stored in the self.heap list, with each bid being a tuple of (bid_amount, bidder_name). The insert method adds a new bid to the heap and maintains the heap property by performing the necessary swaps. The get_highest_bid method retrieves the highest bid from the top of the heap.

2. HybridAuctionPlatform: This class utilizes the hash table and the max heap to manage the auction items. The hash table is implemented as a list of buckets, where each bucket contains the auction items with the same hash index. The _hash_function method calculates the hash index for a given item ID.

The main methods of the HybridAuctionPlatform class include:

- insert: Adds a new auction item to the hash table. It checks for duplicate item IDs before adding the item.
- search: Searches for an auction item by its ID in the hash table and returns the item if found.
- delete: Removes an auction item from the hash table.
- place_bid: Allows a bidder to place a bid on a specific item. It checks if the bid amount is valid and higher than the current highest bid.
- display_items: Displays all the items currently on auction, including their details and the current highest bid, if any.
- print_bids: Prints the bid history for a specific item, showing the bids in descending order.
- end_auction: Ends an auction for a specific item, determining the highest bid and displaying the winning bidder.

The display_menu function provides a menu for the user to interact with the

auction platform, and the main function handles the user's choices and executes the corresponding actions.

2.2 Integration and Interplay :

The integration and interplay between the hash table and heap data structures in the HybridAuctionPlatform class create a powerful system for managing auctions.

The hash table is responsible for storing and retrieving auction items based on their unique item ID. It provides constant-time lookup, allowing for quick access to items. The hash table's index points to the corresponding item in the heap, establishing a connection between the two data structures.The heap, implemented as a max heap, manages the bids placed on each item. It maintains bid integrity by keeping the highest bid at the root. Bids can be inserted efficiently into the heap, ensuring they are correctly ordered.

When an item is inserted into the auction platform, it is associated with a heap instance. This connection allows bids to be placed, retrieved, and compared within the context of the item. The heap's operations complement the functionality of the hash table, providing efficient bid management for each auction item.

Together, the hash table and heap create a cohesive system where the hash table provides item lookup, and the heap handles bid management. This integration ensures the smooth functioning of the auction platform, facilitating easy access to items and accurate tracking of bids.

2.3 Design Choices and Trade-offs :

During the implementation phase, several design choices and trade-offs were made to ensure the efficiency and functionality of the hybrid data structure:

1. **Hybrid Data Structure:** The design choice of combining a hash table and a max heap as a hybrid data structure was made to efficiently handle the auction platform's requirements. The hash table allows constant-time item lookup, while the max heap efficiently manages bids and keeps track of the highest bid.

2. **Hash Function:** The hash function implementation is based on the sum of Unicode values of characters in the key string. While simple, it may not provide a perfectly uniform distribution of indices. This trade-off was made for simplicity and acceptable performance within the context of the auction platform.

3. **List for Heap:** The max heap is implemented using a list, allowing for efficient insertion and retrieval of bids. However, resizing the list may incur a slight performance overhead during heap expansion. The trade-off here is that the list-based implementation simplifies the code and provides reasonably efficient operations.

4. **Storage Overhead:** The implementation uses additional memory to store the item details in a dictionary (self.items). This allows for efficient item retrieval by ID. However, it increases memory consumption and introduces some redundancy, as item information is also stored within the auction item objects.

GitHub Repository Link: https://github.com/jasgithub101/19CSE212-

Chapter 3 Practical Applications

The hybrid data structures used in our program are versatile and can be applied in a wide range of applications that involve managing items, bids, and auctions efficiently. Few examples are:

Social Media Influencer Ranking System:

In the context of social media influencers, a hybrid data structure combining a hash table and a max heap can be used to implement an efficient ranking system. This system aims to identify and rank influencers based on their engagement metrics, such as the number of followers, likes, comments, and shares.

Hash Table: The hash table component can store information about each influencer, with their unique identifier (e.g., influencer ID) as the key. The corresponding value in the hash table can include data such as the influencer's name, follower count, engagement metrics, and other relevant details.

Max Heap: The max heap component can be used to maintain a ranking of influencers based on a specific metric, such as the number of followers or the engagement score. Each influencer's data, along with the corresponding metric value, is stored in the max heap. The influencers with the highest metric values would be positioned at the top of the heap.

Application of the hybrid data structure:

Ranking Calculation: Initially, the influencers' data is stored in the hash table, and the corresponding metric value (e.g., follower count) is added to the max heap. The max heap organizes the influencers based on their metric values, ensuring the influencer with the highest value is at the top.

Updating Metrics: As influencers gain followers, receive likes, comments, or shares, the hash table is updated with the new metric values. If the new metric value affects the influencer's ranking, the max heap is adjusted accordingly to maintain the correct ordering.

Top Influencers: The hybrid data structure allows for efficiently retrieving the top influencers based on the ranking metric. By accessing the top of the max heap, the platform can quickly identify and display the most influential users based on the chosen metric.

Dynamic Ranking: As influencers' metrics change over time, the hash table and max heap can be updated accordingly. This allows the ranking system to adapt to fluctuations in engagement and consistently provide up-to-date rankings.

By leveraging the advantages of the hash table for efficient data storage and retrieval, combined with the max heap for maintaining a ranked order, the hybrid data structure facilitates the creation of a real-time social media influencer ranking system. This enables platforms and users to identify and engage with the most influential individuals based on their engagement metrics.

Crowdfunding Platform:

In a crowdfunding platform, efficiently managing fundraising campaigns and tracking contributions is essential. The hybrid data structure can be effectively used to optimize the storage and retrieval of campaign details, as well as track the highest contributors.

Hash Table: It is utilized to store and retrieve campaign information. Each campaign can have a unique identifier as the key in the hash table, and the corresponding value can contain details such as the campaign name, description, target amount, and current amount raised.

Max Heap: It can be employed to track and display the highest contributors to each campaign. Each contribution, consisting of the contributor's name and the donation amount, can be stored in the max heap. The heap can organize the contributions based on the donation amounts, with the highest contributions at the top.

Application of the hybrid data structure:

Campaign Management: When a new fundraising campaign is initiated, the campaign details can be added to the hash table, allowing for efficient storage and retrieval of campaign information.

Contribution Processing: When a user makes a donation to a campaign, their name and donation amount can be added to the max heap. The heap ensures that the contributions are ordered based on the donation amounts, making it easy to identify the highest contributors.

Real-time Updates: As new contributions are made, the max heap can be updated to reflect the current highest contributors. This allows the platform to provide

real-time updates to campaign organizers and participants, highlighting the top contributors and fostering engagement.

Determining Top Contributors: By retrieving the top contributors from the max heap, the platform can identify and showcase the individuals or organizations that have made the highest contributions to each campaign.
Data Persistence: The hash table can ensure that campaign details and contribution information are stored persistently, allowing for easy retrieval and management of data even after system restarts or failures.

By leveraging the advantages of a hybrid data structure, the crowdfunding platform can efficiently manage campaign information, track contributions, and identify the highest contributors. This not only enhances the transparency and engagement of the platform but also provides valuable insights to campaign organizers and participants.

Job Application Tracking System:

In a job application tracking system, efficiently managing and processing job applications, tracking their statuses, and identifying the most qualified candidates is essential for effective recruitment. A hybrid data structure can be utilized to implement a system that optimizes the application management process and facilitates efficient candidate selection.

Hash Table: It can be employed to store the job applications, with each application having a unique identifier (e.g., application ID) as the key. The corresponding value in the hash table can contain information about the applicant, such as their name, contact details, qualifications, and the status of their application.

Max Heap: It can be used to track and prioritize the applications based on certain criteria, such as qualifications, experience, or suitability for the job. The applications in the max heap would be ordered in descending order based on their priority value.

Application of the Hybrid Data Structure:

Application Submission: When a new job application is received, the applicant's details can be added to the hash table, including their qualifications and other relevant information. The application can be added to the max heap, with the priority value determined by the specified criteria (e.g., qualifications).

Application Review: The hybrid data structure enables efficient application review

by providing easy access to the highest-priority applications in the max heap. Recruiters can quickly identify the most qualified candidates for further evaluation and decision-making.

Candidate Selection: By retrieving the top application from the max heap, recruiters can efficiently determine the most suitable candidate for the job based on the specified criteria. The corresponding applicant's details can be retrieved from the hash table, simplifying the selection process.

Status Updates: The hash table allows for efficient tracking of the application statuses. Recruiters can update the status of each application, such as "under review," "interview scheduled," or "rejected," ensuring that the system maintains an accurate record of the application progress.

Efficient Filtering and Search: The hybrid data structure facilitates efficient filtering and search capabilities. Recruiters can retrieve applications based on specific criteria or perform searches based on applicant attributes stored in the hash table, enabling targeted candidate searches and streamlined application management.

By leveraging the advantages of the hash table and max heap, the hybrid data structure optimizes the job application tracking system, enabling efficient application processing, candidate evaluation, and status tracking. It enhances the recruitment process, making it easier to identify the most qualified candidates and manage applications effectively.

Chapter 4 Performance Analysis

4.1 Time Complexity:

‘insert()’: The time complexity of inserting an item into the auction platform involves calculating the hash function, inserting the item into the hashtable, and appending the item ID to the corresponding index in the table list. Both the hash function calculation and hash table insertion are typically considered to have an average time complexity of **O(1)**.

‘place_bid()’: This method has a constant time complexity of **O(1)** for most operations. However, it calls search and get_highest_bid methods, both of which have a constant time complexity of **O(1)**.

‘end_auction()’: The time complexity of ending an auction for an item involves searching for the item in the hashtable, accessing the highest bid in the MaxHeap which takes **O(1)** time complexity.

‘display_items()’: This method has a linear time complexity of **O(n)**, where n is the number of items. It iterates over the items and performs constant time operations for each item.

‘print_bids()’: This method has a linear time complexity of **O(n)**, where n is the number of bids for the item. It sorts the bids, which has a time complexity of **O(n log n)**, and then iterates over the sorted bids to print them.

4.2 Space Complexity:

List (table in HybridAuctionPlatform class): The space complexity is **O(size)**, where size is the input argument for the constructor or the default value of 10. The space required is proportional to the size of the list.

List (heap in HybridAuctionPlatform.MaxHeap class): The space complexity is **O(n)**, where n is the number of bids stored in the heap. The space required is proportional to the number of bids.

Overall, the space complexities of the data structures used in the code are determined by the number of items, the size of the table, and the number of bids stored in the heap.

4.3 Performance Comparison :

In terms of efficiency, the hybrid data structure used in the our online auction platform provides the following advantages:

1. **Efficient item retrieval:** The use of a dictionary (items) allows for efficient item retrieval based on the item ID. It has an average-case time complexity of **O(1)** for item retrieval.

2. **Efficient bid placement and retrieval:** The use of a max heap data structure (MaxHeap class) allows for efficient bid placement and retrieval. Placing a bid has a time complexity of O(log n), where n is the number of bids stored in the heap. Retrieving the highest bid also has a time complexity of O(1).

3. **Hash table-based indexing:** The use of a list of lists (table) with a hash function allows for efficient indexing of items based on their item ID. This helps in organizing and accessing the items in a more efficient manner.

Overall, the hybrid data structure provides efficient item retrieval, bid placement, and bid retrieval operations, which are essential for an auction platform. The combination of a dictionary, a max heap, and a hash table-based indexing mechanism optimizes the performance of the platform.

Compared to using individual data structures in isolation, the hybrid data structure leverages the strengths of each component to provide a more efficient overall solution.

Chapter 5 Experimental Evaluation

To measure the performance of the hybrid data structure for the auction platform, a series of experiments were conducted. The experiments aimed to assess the efficiency and effectiveness of the data structure in managing auctions, bids, and determining winning bids.

5.1 Datasets Used and Considerations

The experiments utilized various datasets comprising auction items with different characteristics. These datasets consisted of items with varying numbers of auctions, bid start prices, and bid amounts. The datasets were designed to simulate realistic scenarios and test the scalability and performance of the hybrid data structure under different workloads.

Considerations were made to ensure the datasets covered a wide range of scenarios, including cases with a small number of items and bids as well as cases with large-scale auctions and numerous bids. This allowed for comprehensive testing and analysis of the hybrid data structure's performance across different dimensions.

5.2 Results and Interpretation:

The experiments produced several performance metrics and efficiency improvements that shed light on the effectiveness of the hybrid data structure. Key metrics measured include:

1. Insertion Time: The time taken to add new auction items to the data structure.
2. Search Time: The time required to find specific auction items based on their ID.
3. Bid Placement Time: The time taken to place a bid on an auction item.
4. Determining Winning Bids Time: The time needed to identify the highest bids and determine the winners.
5. Memory Usage: The amount of memory consumed by the hybrid data structure.

The results indicated that the hybrid data structure demonstrated significant improvements in performance compared to traditional approaches. The hash table component enabled fast insertion and search operations, ensuring efficient management of auction items. The max heap component facilitated quick access to the highest bids, streamlining the process of determining winning bids.

Overall, the hybrid data structure offered reduced insertion, search, and bid placement times, resulting in a more responsive auction platform. The efficient determination of winning bids enhanced the fairness and transparency of the platform. Additionally, the memory usage remained within acceptable limits, showcasing the data structure's scalability and resource efficiency.

These results confirm that the hybrid data structure effectively optimized the auction management system, providing a robust solution for online auction platforms. The improved performance metrics and efficiency enhancements offered by the hybrid data structure contribute to a seamless bidding experience and promote a fair and transparent auction environment for both bidders and sellers.

Chapter 6 Discussion

6.1 Practicality and Effectiveness:

The hybrid data structure implemented in the online auction platform demonstrates practicality and effectiveness in managing auctions, bids, and determining the winning bids. By combining a hash table and a min-heap, the system efficiently organizes and processes auction-related data.

The hash table component provides fast lookup and storage of auction item details. With the item ID as the key, accessing information such as the item's description, starting price, and current highest bid becomes a constant time operation. This practicality allows for seamless management of a large number of auction items.

The min-heap component efficiently maintains the bid queue and enables quick retrieval of the highest bids. By prioritizing bids based on their values, the min-heap ensures constant time access to the top bid, facilitating real-time bidding activities. This effectiveness is crucial for providing a fair and competitive auction environment.

The hybrid data structure's practicality is evident in its ability to handle bid placement, bid ordering, and determination of winning bids. When a bidder places a bid, the system updates the hash table to store the new highest bid and the corresponding bidder. Simultaneously, the min-heap ensures that bids are ordered correctly, allowing for easy retrieval of the highest bid. This streamlined process enhances the efficiency of the auction platform and promotes a positive user experience.

6.2 Limitations, Challenges, Future Improvements:

While the hybrid data structure is effective, it is important to acknowledge its limitations, challenges, and potential areas for future improvements.

One limitation is that the hybrid data structure assumes a single highest bidder for each auction item. If the platform supports multiple winning bids or implements more complex auction mechanisms, the data structure may require modifications to accommodate these scenarios. Future improvements could involve extending the data structure to handle such cases, enabling the platform to support various auction formats.

Another challenge lies in optimizing the data structure's performance for larger-scale platforms with numerous concurrent auctions and bids. As the size of

the auction platform grows, the efficiency of operations such as bid placement and determination of winning bids may decrease. To address this challenge, further optimizations, such as utilizing data partitioning or parallel processing techniques, could be explored.

Additionally, the hybrid data structure's scalability may face limitations, particularly when handling a significant increase in the number of auction items and bids. Future improvements could involve investigating distributed data structures or leveraging cloud-based technologies to ensure the system's scalability and responsiveness.

Ensuring security and preventing fraudulent activities is another critical aspect that may require additional considerations. Implementing measures such as authentication, authorization, and validation mechanisms can help protect the auction platform and maintain its integrity.

Chapter 7 Conclusion

The hybrid data structure, combining a hash table and a min-heap, proves to be a practical and effective solution for an online auction platform. It optimizes the management of auctions, bids, and winning bid determination, enhancing fairness and efficiency.

The hash table enables fast access and storage of auction item details, while the min-heap efficiently maintains the bid queue and determines the highest bids. The hybrid data structure's practicality and effectiveness enable seamless bid placement, bid ordering, and determination of winning bids.

However, certain limitations exist, such as the assumption of a single highest bidder and potential scalability challenges. Future improvements could involve extending the data structure's capabilities, optimizing performance, and enhancing security measures.

Overall, the hybrid data structure provides a solid foundation for an online auction platform, offering a fair and transparent auction experience for both bidders and sellers. With further enhancements and considerations, it has the potential to support a wide range of auction formats and accommodate the growth and evolving needs of the platform.

Chapter 8 References

Books:

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. , Introduction to Algorithms.

[2] Narsimha Karumanchi, Data Structures and Algorithms Made Easy

Websites:

[1] N.E.Goller, "Hybrid Data Structures Defined by Indirection" [Online]

Available URL: https://academic.oup.com/comjnl/article/28/1/44/468048

[2]] " Big O Analysis and Hybrid Data Structures" [Online]. Available: URL: https://docs.onenetwork.com/NeoHelp/devnet/big-o-analysis-and-hybrid-data-structures-79141067.html

[3] Dario Radecic, "Data Structures and Algorithms With Python" *IEEE Explore*. [Online] Available URL: https://towardsdatascience.com/data-structures-and-algorithms-with-python-learn-stacks-queues-and-deques-in-10-minutes-e7c6a2a1c5d5

[4] JavaTpoint, "Data Structures and Algorithms in Python" . [Online] Available URL: https://www.javatpoint.com/data-structures-and-algorithms-in-python-set-1