Online Auction Section

Project Name: Unique Bid Blind On-line Auction System

Version: 1.0

Date: 2025-03-10

Author: Emma Mattie

# 

# Table of Contents

1. Introduction  
2. System Overview  
3. Design Considerations  
4. Part A: Complexity   
 4.1 Time Complexity Analysis  
5. Part B: Data Structures  
 5.1 Proposing a Solution for Storing and Processing Bids Efficiently  
 5.2 Changes Needed for Handling Tie Situations  
 5.3 Solution Implementation  
6. Tie Resolution Logic

7. Performance Considerations

8. Conclusion

# 1. Introduction

This document explains the design and functionality of the Unique Bid Blind Online Auction System. The system allows users to place bids on items, with the highest unique bid determining the winner. Since multiple users can place bids at the same time, the system needs an efficient way to store and process them while ensuring fairness.

The document also explores different data structures and algorithms that can handle many bids effectively. It considers how to quickly check for the highest unique bid, resolve ties, and restart the auction if needed. Additionally, an analysis of time complexity is included to determine the best approach for managing large amounts of data while keeping the system fast and reliable. The focus is on ensuring that bids are processed correctly, minimizing delays, and maintaining a smooth user experience.

# 2. System Overview

The Unique Bid Blind Online Auction System is designed to manage online auctions where bidders submit blind bids. Each bid remains hidden until the auction closes, and the highest unique bid wins. If multiple bidders place the same highest bid, a tie occurs, and the auction is either restarted with a new minimum bid or resolved based on set rules.

The system must efficiently store and process many bids while ensuring fairness. It needs to handle thousands of bids, track each bidder’s submissions, and determine the winner quickly. To achieve this, the system relies on well-structured data storage and retrieval methods. The auction process follows these key steps:

1. Bidders submit bids until the auction closes.
2. Bids are checked to ensure they meet the minimum bid requirement and are in whole Canadian dollars.
3. Once bidding ends, the system identifies the highest unique bid.
4. If there is a tie, the auction is either restarted or resolved based on the auction rules.
5. The winner is announced, and the auction concludes.

By using optimized data structures and efficient processing methods, the system ensures a fair and reliable bidding experience, even when handling many participants.

# 3. Design Considerations

The Unique Bid Blind Online Auction System must be efficient, scalable, and fair while handling many bids.

**Scalability**

The system should process thousands of bids quickly using optimized data structures to prevent delays.

**Bid Validation**

Bids must be in whole Canadian dollars, meet the minimum bid, and allow multiple entries per bidder while avoiding duplicates.

**Fairness and Tie Resolution**

If multiple bidders submit the highest bid, the system must either restart the auction with a higher minimum bid or award the item to the first bidder.

**Efficient Bid Processing**

The system should quickly track bids, check for duplicates, and determine the highest unique bid as soon as the auction ends.

**Data Storage**

A combination of hash maps and priority queues will allow fast bid retrieval and winner determination.

With these considerations, the system will be reliable, scalable, and fair for all users.

# 4. Part A: Complexity

## 4.1 Time Complexity Analysis

For large values of n, choosing an algorithm with the lowest time complexity ensures efficiency. Below are the complexities given and their impact:

* **O(n³)** – Grows quickly and becomes inefficient for large n.
* **Ω(n³)** – Only provides a lower bound, meaning the actual performance could be worse.
* **O(n² log n)** – More efficient than O(n³), commonly used in sorting algorithms.
* **Θ(n² log n)** – Defines both the upper and lower bound, making it a predictable choice.
* **O(2ⁿ)** – Exponential growth, highly inefficient for large n.

**Best Choice**

The most efficient option for large n is Θ(n² log n) because it provides a well-defined performance bound and is significantly faster than cubic and exponential time complexities. Using this complexity ensures that the system remains scalable and performs well even with many bids.

# 5. Part B: Data Structures

## 5.1 Proposing a Solution for Storing and Processing Bids Efficiently

To handle bids efficiently, the system needs a structure that supports fast storage, quick identification of the highest unique bid, and scalability for large volumes of bids.

**Proposed Data Structures**

1. **HashMap (Dictionary in Python, HashMap in Java)**
   * **Key:** Bid amount
   * **Value:** List of bidders who placed that bid
   * This allows constant-time lookup and insertion to track bid frequency.
2. **Priority Queue (Min-Heap/Max-Heap)**
   * Used to retrieve the highest bid efficiently.
   * Extracting the max/min bid takes logarithmic time, making it faster than sorting all bids.
3. **LinkedHashMap (Ordered Dictionary in Python)**
   * Maintains bid order, helping determine the first bidder in case of a tie.

**Processing Bids Efficiently**

1. Each bid is stored in HashMap to track its frequency.
2. It is also added to the **Priority Queue** for fast access to the highest bid.
3. When the auction closes, the system scans the **HashMap** to find the highest unique bid.
4. If a tie occurs, the **LinkedHashMap** identifies which bidder placed it first.

This method ensures fast, fair, and scalable bid processing.

## 5.2 Changes Needed for Handling Tie Situations

If multiple bidders place the highest bid, the system must resolve the tie fairly and efficiently. Depending on the auction rules, two possible solutions can be implemented:

**1. Restarting the Auction**

* If a tie occurs, the auction restarts with a new minimum bid set to the highest tied bid plus one.
* This ensures a unique highest bid in the next round but may extend the auction duration.

**2. Awarding the Bid to the First Bidder**

* Instead of restarting, the system awards the item to the bidder who placed the highest bid first.
* To track bid order, a **LinkedHashMap** (or timestamp-based tracking) is used, maintaining both bid values and entry order.

**Implementation Adjustments**

* If the restart method is used, the system must reset bids and notify users.
* If the first-bidder method is used, bids must be stored with timestamps to determine priority.

Either method ensures fairness while maintaining efficiency in processing bids.

## 5.3 Solution Implementation

The auction system must efficiently process bids, track their frequency, and determine the winner. Below is an overview of how the solution is implemented using the proposed data structures.

**1. Storing and Processing Bids**

* When a bid is placed, it is stored in a **HashMap**, where the key is the bid amount, and the value is a list of bidders who placed that bid.
* The bid is also added to a **Priority Queue** to allow quick access to the highest bid.
* LinkedHashMap or timestamp-based tracking ensures the order of bids is maintained.

**2. Determining the Winner**

* Once the auction closes, the system scans the **HashMap** to find the highest bid with only one bidder.
* If there is a tie, the system either restarts the auction with a new minimum bid or awards the item to the first bidder using the **LinkedHashMap**.

**3. Handling Ties and Restarting Auctions**

* If a tie is detected and the auction restarts, all bids are reset, and the minimum bid is updated.
* If the first-bidder method is used, timestamps or insertion order determine the winner.

This approach ensures that the auction runs efficiently while maintaining fairness and scalability.

# 6. Tie Resolution Logic

If multiple bidders place the highest bid, the system resolves the tie using one of two methods:

**1. Restarting the Auction**

* The auction restarts with a new minimum bid set to the highest tied bid plus one.
* All previous bids are cleared, and bidders must submit new bids.
* This ensures a unique highest bid in the next round but may extend the auction duration.

**2. Awarding the First Bidder**

* Instead of restarting, the item is awarded to the first bidder who placed the highest bid.
* A **LinkedHashMap** or timestamp-based tracking is used to maintain bid order.
* This method ensures a faster resolution without requiring a new auction round.

After the auction closes, the system identifies the highest bid and applies the selected tie-resolution method, ensuring a fair and efficient process.

# 7. Performance Considerations

The system must handle a high volume of bids efficiently while ensuring quick processing and fairness. Several factors affect performance, including data structure choices and algorithm efficiency.

**1. Bid Storage and Retrieval**

* A **HashMap** allows fast O(1) insertion and lookup of bid values.
* A **Priority Queue** enables quick access to the highest bid in O(log n) time.

**2. Tie Resolution Efficiency**

* Restarting the auction requires clearing and resetting bids, which can slow down performance.
* Awarding the first bidder is faster, as the system only needs to check timestamps.

**3. Scalability**

* The system must handle thousands of bids without lag.
* Using optimized data structures prevents performance slowdowns as the number of bidders increases.

By ensuring efficient data handling and minimizing redundant operations, the system remains fast and scalable, even in high-demand auctions.

# 8. Conclusion

The Unique Bid Blind Online Auction System is designed to efficiently process and store bids while ensuring fairness in winner selection. By using optimized data structures like HashMaps, Priority Queues, and LinkedHashMaps, the system can handle many bids while maintaining fast performance.

The auction’s tie-resolution logic ensures fairness by either restarting with a higher minimum bid or awarding the item to the first highest bidder. Performance considerations focus on scalability and minimizing delays, ensuring that the system remains responsive even with thousands of participants.

With this approach, the auction runs smoothly, processes bids efficiently, and provides a fair experience for all users.

A flowchart of a bid

AI-generated content may be incorrect.

This flowchart illustrates the process of handling bids in the Unique Bid Blind Online Auction System. It shows how bids are validated, stored, and checked for the highest unique bid. If a tie occurs at the highest bid, the auction restarts with a new minimum bid. Otherwise, the highest unique bid is declared the winner.