

# ONLINE AUCTION SYSTEM

## AIM:

To design and implement an Online Auction System using Data Structures where bidders can place bids on items, manage bid history, undo the last bid, and efficiently organize auction data using Linked List, Queue, Stack, Tree, Graph, and Hash Map.

## ALGORITHM:

### Step 1: Start

---

### Step 2: Initialize Data Structures

1. Create a **Linked List** to store bidder names
  2. Create a **Queue** to manage bid processing order
  3. Create a **Stack** to support undo of last bid
  4. Create a **Binary Search Tree** to store auction items
  5. Create a **Hash Map** to store bid ID and bid details
  6. Create a **Graph (Adjacency List)** to represent bidder relationships
  7. Initialize bid counter to generate unique bid IDs
- 

### Step 3: Display Menu

Display the following options to the user:

1. Add Bidder
2. Add Item
3. Place Bid
4. Process Bid
5. Undo Last Bid
6. Display Items

- 
7. Exit
- 

#### **Step 4: Add Bidder**

1. Read bidder name from the user
  2. Insert bidder into the linked list
  3. Create an empty adjacency list entry in the graph
  4. Display confirmation message
- 

#### **Step 5: Add Item**

1. Read item name from the user
  2. Insert item into the binary search tree
  3. Display confirmation message
- 

#### **Step 6: Place Bid**

1. Read bidder name and bid amount
  2. Verify bidder exists in the linked list
  3. Generate a unique bid ID
  4. Create a new bid object
  5. Insert bid into the queue
  6. Push bid onto the stack
  7. Store bid ID and bid object in the hash map
  8. Display bid ID
- 

#### **Step 7: Process Bid**

1. Check if bid queue is empty
  2. Remove bid from the front of the queue
  3. Display bid details (ID, bidder name, amount)
-

#### **Step 8: Undo Last Bid**

1. Check if undo stack is empty
  2. Pop last bid from the stack
  3. Remove corresponding entry from the hash map
  4. Display undo confirmation
- 

#### **Step 9: Display Auction Items**

1. Perform inorder traversal of the binary search tree
  2. Display all auction items in sorted order
- 

#### **Step 10: Repeat**

Repeat Steps 3 to 9 until the user selects **Exit**

---

#### **Step 11: Stop**

### **PROGRAM:**

```
import java.util.*;  
  
class Bid {  
    int bidId;  
    String bidderName;  
    int amount;  
  
    Bid(int bidId, String bidderName, int amount) {  
        this.bidId = bidId;  
        this.bidderName = bidderName;  
        this.amount = amount;  
    }  
}
```

```
class ItemNode {  
    String itemName;  
    ItemNode left, right;  
  
    ItemNode(String itemName) {  
        this.itemName = itemName;  
    }  
}  
  
class ItemTree {  
    ItemNode root;  
  
    ItemNode insert(ItemNode root, String item) {  
        if (root == null) return new ItemNode(item);  
        if (item.compareTo(root.itemName) < 0)  
            root.left = insert(root.left, item);  
        else  
            root.right = insert(root.right, item);  
        return root;  
    }  
  
    void inorder(ItemNode root) {  
        if (root != null) {  
            inorder(root.left);  
            System.out.print(root.itemName + " ");  
            inorder(root.right);  
        }  
    }  
}
```

```
}

public class OnlineAuctionSystem {

    static LinkedList<String> bidders = new LinkedList<>();

    static Queue<Bid> bidQueue = new LinkedList<>();

    static Stack<Bid> undoStack = new Stack<>();

    static HashMap<Integer, Bid> bidMap = new HashMap<>();

    static HashMap<String, List<String>> bidderGraph = new HashMap<>();

    static ItemTree itemTree = new ItemTree();

    static int bidCounter = 1;

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int choice;

        do {

            System.out.println("\n--- ONLINE AUCTION SYSTEM ---");

            System.out.println("1. Add Bidder");

            System.out.println("2. Add Item");

            System.out.println("3. Place Bid");

            System.out.println("4. Process Bid");

            System.out.println("5. Undo Last Bid");

            System.out.println("6. Display Items");

            System.out.println("7. Exit");

            System.out.print("Enter choice: ");

            choice = sc.nextInt();

            sc.nextLine();
        }
    }
}
```

```
switch (choice) {  
    case 1 -> addBidder(sc);  
    case 2 -> addItem(sc);  
    case 3 -> placeBid(sc);  
    case 4 -> processBid();  
    case 5 -> undoBid();  
    case 6 -> displayItems();  
}  
}  
}  
  
sc.close();  
}  
  
static void addBidder(Scanner sc) {  
    System.out.print("Enter bidder name: ");  
    String name = sc.nextLine();  
    bidders.add(name);  
    bidderGraph.put(name, new ArrayList<>());  
    System.out.println("Bidder added successfully");  
}  
  
static void addItem(Scanner sc) {  
    System.out.print("Enter item name: ");  
    String item = sc.nextLine();  
    itemTree.root = itemTree.insert(itemTree.root, item);  
    System.out.println("Item added to auction");  
}
```

```
static void placeBid(Scanner sc) {  
    System.out.print("Enter bidder name: ");  
    String name = sc.nextLine();  
  
    if (!bidders.contains(name)) {  
        System.out.println("Bidder not found");  
        return;  
    }  
  
    System.out.print("Enter bid amount: ");  
    int amount = sc.nextInt();  
  
    Bid bid = new Bid(bidCounter++, name, amount);  
    bidQueue.add(bid);  
    undoStack.push(bid);  
    bidMap.put(bid.bidId, bid);  
  
    System.out.println("Bid placed with ID: " + bid.bidId);  
}  
  
static void processBid() {  
    if (bidQueue.isEmpty()) {  
        System.out.println("No bids to process");  
        return;  
    }  
  
    Bid bid = bidQueue.poll();  
    System.out.println("Processing Bid → ID: " + bid.bidId +
```

```

        ", Bidder: " + bid.bidderName +
        ", Amount: " + bid.amount);
    }

static void undoBid() {
    if (undoStack.isEmpty()) {
        System.out.println("No bid to undo");
        return;
    }

    Bid bid = undoStack.pop();
    bidMap.remove(bid.bidId);
    System.out.println("Last bid undone → ID: " + bid.bidId);
}

static void displayItems() {
    System.out.print("Auction Items (Inorder): ");
    itemTree.inorder(itemTree.root);
    System.out.println();
}

```

## **OUTPUT:**

--- ONLINE AUCTION SYSTEM ---

1. Add Bidder
2. Add Item
3. Place Bid
4. Process Bid

5. Undo Last Bid

6. Display Items

7. Exit

Enter choice: 1

Enter bidder name: Ravi

Bidder added successfully

Enter choice: 2

Enter item name: Laptop

Item added to auction

Enter choice: 3

Enter bidder name: Ravi

Enter bid amount: 50000

Bid placed with ID: 1

Enter choice: 4

Processing Bid → ID: 1, Bidder: Ravi, Amount: 50000