Data Structure

Mini Project

Netaji Subhash University of Technology

Session – 2024-25

Made By :

Abhigyan Kumar Roy: 2024UCS1591

Aditya Kumar : 2024UCS1594

Vipin Gupta: 2024UCS1607

SUBMITTED TO:

MAHIMA MAAM

PROJECT TOPIC:

RAILWAY RESERVATION SYSTEM

TOPICS:

- **Register User**
- **Login User**
- **Add Trains As Admin**
- **Show Trains**
- **Book Tickets**
- **Cancel Tickets**

Data Structures Used:

- **Linked Lists**
- **Vectors**
- **Priority queue**
- **Unordered Maps**

CODE

- **Register User**

```
unordered_map<string,string> data;
void registerUser(){
    string username,pass;

    cout << " Enter Username: ";
    cin >> username;

    if(data.find(username)!= data.end()){
        //this means we have found an entry with a same username
        cout<<"Username already Registered!!"<<"\n "<< "Kindly Login";
        return;
    }
    cout<<"Enter a Password ";
    cin>> pass;
    data[username]=pass;// storing data in maps so to check double id is
not created
    cout<<"Registration Done"<<"\n";
}
```

Time Complexity:

- **O(1) on average (insertion into unordered_map)**

Space Complexity:

- **O(U) where U is the number of users.**

- **LOGIN USER**

```
bool loginUser(string &username) {
    string pass;
    int attempts = 3;

    while (attempts > 0) {
        cout << "Username: ";
        cin >> username;
        cout << "Password: ";
        cin >> pass;

        // Check hardcoded admin
        if (username == "admin" && pass == "admin123") {
            cout << "Admin Login Successful\n";
            return true;
        }

        // Normal user login
        if (data.find(username) != data.end() && data[username] == pass)
{
            cout << "Login Successful\n";
            return true;
        } else {
            attempts--;
            cout << "Wrong Credentials. Attempts Left: " << attempts <<
"\n";
        }
    }

    cout << "Attempts Over! Try again later\n";
    return false;
}
```

Time Complexity:

- **O(1) average (lookup in unordered_map)**
- **Worst case O(U) if there's a hash collision (very rare)**

Space Complexity:

- **O(1) (temporary strings only)**

Waiting List and Train Structure

```
class waitingList{
public :
    waitingNode* head;
    waitingNode*tail;
    waitingList(){
        head=tail=nullptr;
    }
    void addPassengers(string username){
        waitingNode* temp=new waitingNode(username);
        if(!head){
            head=tail=temp;
        }else {
            tail->next=temp;
            tail=temp;
        }
    }
    string removePassengers(){
        if(!head) return "";
        string name=head-> username;
        waitingNode *temp=head;
        head = head->next;
        delete temp;
        if (!head) tail = nullptr;
        return name;
    }

    bool isEmpty() {
        return head == nullptr;
    }

};
struct train {
    int trainNo;
    string source;
    string destination;
    string departureTime;
    vector<int> bookedSeats;
    priority_queue<int, vector<int>, greater<int>>
av_seats;
    waitingList waitingList;

    train(int no, string src, string dest, string
deTime, int totseats) {
        trainNo = no;
        source = src;
        destination = dest;
        departureTime = deTime;
        for (int i = 1; i <= totseats; ++i) {
            av_seats.push(i);
        }
    }
};
```

ADD TRAINS

```cpp
vector<train> trainList;

void addTrain() {
    int no, totseats;
    string src, dest, time;
    cout << "Enter Train Number, Source, Destination, Departure Time, Total Seats: ";
    cin >> no >> src >> dest >> time >> totseats;
    trainList.push_back(train(no, src, dest, time, totseats));
    cout << "Train Added Successfully!" << endl;
}
```

Time Complexity:

- **O(S log S) where S is the number of seats (to push into priority_queue)**

Space Complexity:

- **O(S) for storing available seats in priority_queue**
- **O(1) for other train data per train**
- **So overall, O(T * S) where T is number of trains**

Book Tickets

```
void bookTicket(string username) {
    int trainNo;
    cout << "Enter Train Number to Book: ";
    cin >> trainNo;
    for (auto& t : trainList) {
        if (username == "") {
            cout << "Please login first!" << endl;
            break;
        }
        if (t.trainNo == trainNo) {
            if (!t.av_seats.empty()) {
                int seat = t.av_seats.top();
                t.av_seats.pop();
                t.bookedSeats.push_back(seat);
                cout << "Ticket Booked! Seat Number: "
<< seat << endl;
            } else {
                t.waitingList.addPassengers(username);
                cout << "Train Full. Added to Waiting
List." << endl;
            }
            return;
        }
    }
    cout << "Train Not Found." << endl;
}
```

- **Time Complexity:**
  - **O(T) to find the train**
  - **O(log S) to get the smallest available seat from priority_queue**
  - **O(1) to insert into bookedSeats (amortized vector push_back)**
- **Space Complexity:**
  - **O(1) additional (but modifies internal train data structures)**

SHOW TRAINS

```
void showTrains() {
    if (trainList.empty()) {
        cout << "No Trains Available!\n";
        return;
    }
    sort(trainList.begin(), trainList.end(), [](train
&a, train &b) {
        return a.trainNo < b.trainNo;
    });
    cout << "\nAvailable Trains:\n";
    for (auto& t : trainList) {
        cout << "Train No: " << t.trainNo
             << " | From: " << t.source
             << " | To: " << t.destination
             << " | Departure: " << t.departureTime
             << " | Seats Available: " <<
t.av_seats.size()
             << " | Booked Seats: " <<
t.bookedSeats.size() << "\n";
    }
}
```

- **Time Complexity:**
  - **O(T log T) due to sorting trains by train number**
- **Space Complexity:**
  - **O(1) (just iterating and printing)**

CANCEL TICKETS

```
void cancelTicket(string username, int trainNo, int
seatNo) {
    for (auto& t : trainList) {
        if (t.trainNo == trainNo) {
            auto it = find(t.bookedSeats.begin(),
t.bookedSeats.end(), seatNo);
            if (it != t.bookedSeats.end()) {
                t.bookedSeats.erase(it);
                t.av_seats.push(seatNo);
                cout << "Ticket Cancelled. Seat " <<
seatNo << " is now free." << endl;
                if (!t.waitingList.isEmpty()) {
                    string nextUser =
t.waitingList.removePassengers();
                    int newSeat = t.av_seats.top();
                    t.av_seats.pop();
                    t.bookedSeats.push_back(newSeat);
                    cout << "Waiting List Cleared for
User: " << nextUser << " | Seat: " << newSeat << endl;
                }
                return;
            } else {
                cout << "Seat not found in booking
list." << endl;
                return;
            }
        }
    }
    cout << "Train Not Found." << endl;
}
```

- **Time Complexity:**
  - **O(T) to find train**
  - **O(S) to find seat in bookedSeats (since it's a vector)**
  - **O(log S) to push seat back into priority_queue**
  - **If waiting list is used: O(1) to remove from waitingList**
- **Space Complexity:**
  - **No extra space except for temporary variables**

Menu

```
int main() {
    int choice;
    string username = "";

    while (true) {
        cout << "\n     Railway Booking System    \n";
        cout << "1. Register\n2. Login\n3. Add Train
(Admin)\n4. Show Trains\n5. Book Ticket\n6. Cancel
Ticket\n7. Exit\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                if (loginUser(username)) {

                }
                break;
            case 3:
                if (username == "admin") {
                    addTrain();
                    username = ""; // Log out admin
after adding train
                    cout << "Admin logged out
automatically.\n";
                } else {
                    cout << "Only admin can add
trains!\n";
                }
                break;

            case 4:
                showTrains();
                break;
            case 5:
                if (username == "") {
                    cout << "You must be logged in to
book tickets!\n";
                } else {
                    bookTicket(username);
                }
                break;
            case 6:
                if (username == "") {
                    cout << "You must be logged in to
cancel tickets!\n";
                } else {
                    int trainNo, seatNo;
                    cout << "Enter Train Number and Seat
Number to Cancel: ";
                    cin >> trainNo >> seatNo;
                    cancelTicket(username, trainNo,
seatNo);
                }
                break;
            case 7:
                return 0;
            default:
                cout << "Invalid Choice!" << endl;
        }
    }
}
```

FULL CODE

```
#include <bits/stdc++.h>
using namespace std ;

// Railway Booking Project

// Task :1
// create user Credentials
unordered_map<string,string> data;
void registerUser(){
    string username,pass;

    cout << " Enter Username: ";
    cin >> username;

    if(data.find(username)!= data.end()){
        //this means we have found an entry with a same
username
        cout<<"Username already Registered!!"<<"\n "<<
"Kindly Login";
        return;
    }
    cout<<"Enter a Password ";
    cin>> pass;
    data[username]=pass;// storing data in maps so to
check double id is not created
    cout<<"Registration Done"<<"\n";
}
bool loginUser(string &username) {
    string pass;
    int attempts = 3;

    while (attempts > 0) {
        cout << "Username: ";
        cin >> username;
        cout << "Password: ";
        cin >> pass;
        // Check hardcoded admin
        if (username == "admin" && pass == "admin123") {
            cout << "Admin Login Successful\n";
            return true;
        }

        // Normal user login
        if (data.find(username) != data.end() &&
data[username] == pass) {
            cout << "Login Successful\n";
            return true;
        } else {
            attempts--;
            cout << "Wrong Credentials. Attempts Left: "
<< attempts << "\n";
        }
    }

    cout << "Attempts Over! Try again later\n";
    return false;
}

struct waitingNode{
    string username;
    waitingNode* next;
    waitingNode(string name):
username(name),next(nullptr){}
};
class waitingList{
public :
    waitingNode* head;
    waitingNode*tail;
    waitingList(){
        head=tail=nullptr;
    }
    void addPassengers(string username){
        waitingNode* temp=new waitingNode(username);
        if(!head){
            head=tail=temp;
        }else {
            tail->next=temp;
            tail=temp;
        }
    }
    string removePassengers(){
        if(!head) return "";
        string name=head-> username;
        waitingNode *temp=head;
        head = head->next;
        delete temp;
        if (!head) tail = nullptr;
        return name;
    }

    bool isEmpty() {
        return head == nullptr;
    }

};
struct train {
    int trainNo;
    string source;
    string destination;
    string departureTime;
    vector<int> bookedSeats;
    priority_queue<int, vector<int>, greater<int>>
av_seats;
    waitingList waitingList;

    train(int no, string src, string dest, string
deTime, int totseats) {
        trainNo = no;
        source = src;
        destination = dest;
        departureTime = deTime;
        for (int i = 1; i <= totseats; ++i) {
            av_seats.push(i);
        }
    }
};


vector<train> trainList;

void addTrain() {
    int no, totseats;
    string src, dest, time;
    cout << "Enter Train Number, Source, Destination,
Departure Time, Total Seats: ";
    cin >> no >> src >> dest >> time >> totseats;
    trainList.push_back(train(no, src, dest, time,
totseats));
    cout << "Train Added Successfully!" << endl;
}

void bookTicket(string username) {
    int trainNo;
    cout << "Enter Train Number to Book: ";
    cin >> trainNo;
    for (auto& t : trainList) {
        if (username == "") {
            cout << "Please login first!" << endl;
            break;
        }
        if (t.trainNo == trainNo) {
            if (!t.av_seats.empty()) {
                int seat = t.av_seats.top();
                t.av_seats.pop();
                t.bookedSeats.push_back(seat);
                cout << "Ticket Booked! Seat Number: "
<< seat << endl;
            } else {
                t.waitingList.addPassengers(username);
                cout << "Train Full. Added to Waiting
List." << endl;
            }
            return;
        }
    }
    cout << "Train Not Found." << endl;
}
void showTrains() {
    if (trainList.empty()) {
        cout << "No Trains Available!\n";
        return;
    }
    sort(trainList.begin(), trainList.end(), [](train
&a, train &b) {
        return a.trainNo < b.trainNo;
    });
    cout << "\nAvailable Trains:\n";
    for (auto& t : trainList) {
        cout << "Train No: " << t.trainNo
             << " | From: " << t.source
             << " | To: " << t.destination
             << " | Departure: " << t.departureTime
             << " | Seats Available: " <<
t.av_seats.size()
             << " | Booked Seats: " <<
t.bookedSeats.size() << "\n";
    }
}


void cancelTicket(string username, int trainNo, int
seatNo) {
    for (auto& t : trainList) {
        if (t.trainNo == trainNo) {
            auto it = find(t.bookedSeats.begin(),
t.bookedSeats.end(), seatNo);
            if (it != t.bookedSeats.end()) {
                t.bookedSeats.erase(it);
                t.av_seats.push(seatNo);
                cout << "Ticket Cancelled. Seat " <<
seatNo << " is now free." << endl;
                if (!t.waitingList.isEmpty()) {
                    string nextUser =
t.waitingList.removePassengers();
                    int newSeat = t.av_seats.top();
                    t.av_seats.pop();
                    t.bookedSeats.push_back(newSeat);
                    cout << "Waiting List Cleared for
User: " << nextUser << " | Seat: " << newSeat << endl;
                }
                return;
            } else {
                cout << "Seat not found in booking
list." << endl;
                return;
            }
        }
    }
    cout << "Train Not Found." << endl;
}

int main() {
    int choice;
    string username = "";

    while (true) {
        cout << "\n     Railway Booking System    \n";
        cout << "1. Register\n2. Login\n3. Add Train
(Admin)\n4. Show Trains\n5. Book Ticket\n6. Cancel
Ticket\n7. Exit\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                if (loginUser(username)) {

                }
                break;
            case 3:
                if (username == "admin") {
                    addTrain();
                    username = ""; // Log out admin
after adding train
                    cout << "Admin logged out
automatically.\n";
                } else {
                    cout << "Only admin can add
trains!\n";
                }
                break;

            case 4:
                showTrains();
                break;
            case 5:
                if (username == "") {
                    cout << "You must be logged in to
book tickets!\n";
                } else {
                    bookTicket(username);
                }
                break;
            case 6:
                if (username == "") {
                    cout << "You must be logged in to
cancel tickets!\n";
                } else {
                    int trainNo, seatNo;
                    cout << "Enter Train Number and Seat
Number to Cancel: ";
```

```
                    cin >> trainNo >> seatNo;
                    cancelTicket(username, trainNo, 
seatNo);
                }
                break;
            case 7:
                return 0;
            default:
                cout << "Invalid Choice!" << endl;
        }
    }
}
```