**AirlineApp design**

Juan Daniel Torres

Joseph Vélez

Juan Manuel Diaz

Universidad Icesi

Estructura discretas y computación

Jeison David Mejía Trujillo

Facultad de ingeniería, Ingeniería de sistemas

Cali, 15 de abril de 2023

**AirlineApp design**

**Phase 1.**

**Definition of the problem:**

¿What is the problem?

The existing problem is the inefficiency when ordering the boarding of passengers on a flight within an aircraft, since there are many variables regarding the arrival time of customers. Furthermore, it is essential to organize data such as prioritization based on individual data or statuses. Additionally, when passengers are leaving, it is necessary to increase efficiency, but now based on the previously made organization when boarding.

¿Who has the problem?

The problem is faced by the airline, as it must optimize the time to take its clients to their destinations, in addition to providing adequate services based on the benefits contracted by the passengers.

¿Why is it important to solve?

It is important to solve this problem since greater efficiency would help both the comfort of the clients and time-saving for the airline, so the benefits are on both sides of the service.

Identification of needs and symptoms:

- The airline needs to improve the efficiency of the management of the passengers of its flights.

- The airline needs to implement an efficient method for searching passengers.

- The airline needs to add services of prioritizing its passengers according to their chosen condition or status.

**Phase 2.**

**Information gathering:**

Definitions:

- Priority queues: Priority queues are abstract data types that allow elements to be stored in a certain order, with the highest or lowest priority items at the front of the queue. They are typically implemented using heaps, which are binary trees that maintain the heap property. Priority queues are important in many applications, such as operating systems, where they are used to schedule tasks with varying priorities. They are also used in algorithms such as Dijkstra's shortest path algorithm and Huffman coding. In general, priority queues are useful whenever there is a need to process items in a certain order of importance.

- Dijkstra’s: This algorithm makes a tree of the shortest path from the starting node, the source, to all other nodes (points) in the graph. Dijkstra's algorithm makes use of weights of the edges for finding the path that minimizes the total distance (weight) among the source node and all other nodes. This algorithm is also known as the single-source shortest path algorithm. It is important to note that Dijkstra’s algorithm is only applicable when all weights are positive because, during the execution, the weights of the edges are added to find the shortest path.

- Heap: A heap is a data structure where each node has an assigned value and the parent node always has a greater or smaller value than the child nodes, depending on whether it is a max heap or min heap, respectively. The heap algorithm is used to efficiently sort a list of elements into a max or min heap and then extract the top element (the largest or smallest). It is also used to efficiently insert and remove elements from a heap, while maintaining the heap property at all times.

- Heap Algorithms: Heap algorithms are a class of algorithms that operate on heap data structures, such as heap sort, building a heap, and heapify. These algorithms involve reordering elements in the heap to satisfy the heap property, allowing for efficient sorting or selection of elements. Heap algorithms have numerous practical applications in computer science and are commonly used in programming competitions. Overall, heap algorithms are an important class of algorithms that play a key role in many fields of computer science and beyond.

- Hash table: Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data. Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data. Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

**Phase 3**.

Search for creative solutions:

For finding the best solution that we´re going to implement, brainstorming was the technique we implemented in order to get the most ideas out of us.

**Brainstorming:**

1. Order the seats of the plane according to the arrival of the passengers: This idea involves simply ordering the seats in the plane based on the arrival time of the passengers. However, this approach does not take into consideration other important factors such as prioritization of passengers or an orderly departure from the plane. While this could be useful for the initial simulation of the system, it may not be sufficient for the final version.

2. Order the seats according to the management of prioritization made, implementing the rewarding system of punctuality: This idea involves prioritizing passengers based on factors such as punctuality, miles accumulated, special attention required, or other relevant data. This approach could be very useful for the main objective of the system, which is to improve the order of the boarding and departure processes. By implementing a rewards system for punctuality, passengers who arrive on time could be given better seats or services.

3. Quicksort: An efficient sorting algorithm that can be used to sort the list of passengers based on their arrival time. This algorithm has an average time complexity of O(n log n) and can handle a large number of elements.

4. Search the information of the passengers by saving it on an Array List: This idea involves storing the passenger information in an Array List, which is a dynamic array data structure in Java. While this could be a viable option for the initial simulation of the system, it may become less efficient as the amount of data in the database increases.

5. Binary Search Tree: This data structure can be used to store and search for passenger information efficiently. The passenger data can be stored in a binary search tree, where each node represents a passenger, and their information can be accessed quickly using the binary search algorithm.

6. Search the information of the passengers by saving it on a Hash Table: This idea involves storing the passenger information in a Hash Table, which is a data structure that provides fast access to data based on keys. This approach could be more efficient in the long run for quickly and effectively searching passenger information, especially when dealing with large amounts of data.

7. Back-to-Front Algorithm: This algorithm is a simple and commonly used approach for organizing passengers. It works by boarding passengers from the back of the airplane to the front. This approach minimizes the amount of passengers walking past each other as they search for their seats. This algorithm is often used by budget airlines as it requires less coordination and can speed up the boarding process. An example of an airline that uses this algorithm is Southwest Airlines.

8. Front-to-Back Algorithm: This algorithm works in the opposite direction of the Back-to-Front Algorithm, boarding passengers from the front of the airplane to the back. This approach also minimizes the amount of passengers walking past each other, but some studies suggest that it may be slightly less efficient than the Back-to-Front Algorithm. This algorithm is commonly used by major airlines such as American Airlines and United Airlines.

9. Window-Middle-Aisle Algorithm: This algorithm is based on the passenger's seat position. Passengers sitting in window seats are boarded first, followed by passengers in middle seats, and finally passengers in aisle seats. This approach minimizes the amount of passengers walking past each other as they search for their seats, and has been found to be more efficient than the previous two algorithms. This algorithm is commonly used by airlines such as Delta Airlines.

10. Steffen Algorithm: This algorithm is a more complex approach that uses a mathematical model to optimize the boarding process. It takes into account factors such as the size of the airplane, the number of passengers, the size of carry-on luggage, and the time it takes for passengers to reach their seats. The algorithm assigns boarding times to each passenger, ensuring that there is no overlap between passengers walking down the aisle. This algorithm is considered to be the most efficient, but is also the most complex to implement. It has been used by airlines such as Lufthansa and United Airlines

**Find the most efficient way of organizing the passengers**

**Ideas:**

1. Order the seats of the plane according to the arrival of the passengers.

2. Order the seats according to the management of prioritization made, implementing the rewarding system of punctuality

**Find the most efficient method to search the information of the passengers**

3. Search the information of the passengers by using a search algorithm and saving the information inside a data structure:

4. Search the information of the passengers by awaiting their presence inside the airport and registering them each time:.

**Phase 4**.

**Transition from ideas to preliminary designs:**

Why did we discard the other alternatives?

1. Order the seats of the plane according to the arrival of the passengers: This idea is discarded as it does not take into account other important factors such as prioritization of passengers or an orderly departure from the plane.

2. Order the seats according to the management of prioritization made, implementing the rewarding system of punctuality: This idea is chosen as it involves prioritizing passengers based on factors such as punctuality, miles accumulated, special attention required, or other relevant data. By implementing a rewards system for punctuality, passengers who arrive on time could be given better seats or services. This approach could be very useful for the main objective of the system, which is to improve the order of the boarding and departure processes.

3. Quicksort: This algorithm is discarded as it does not address the main problem of prioritizing passengers based on relevant data.

4. Search the information of the passengers by saving it on an Array List: This idea is discarded as it may become less efficient as the amount of data in the database increases.

5. Binary Search Tree: A binary search tree is a data structure that helps organize information in a way that makes it easy to search and find specific pieces of data quickly. It is like a tree with branches, where each branch has two smaller branches, one on the left and one on the right. Every piece of data in the tree is stored in a node, and each node has a value that is greater than all the values in its left branch and less than all the values in its right branch. This allows you to search for data very efficiently, eliminating half of the remaining nodes at each step of the search until you find the data you're looking for. Inserting new data into the tree is also easy and quick. However, if the tree becomes unbalanced, it can slow down the search and insertion operations. To avoid this, special types of binary search trees are used that automatically balance themselves as new data is added.

6. Search the information of the passengers by saving it on a Hash Table: This idea is chosen as it involves storing the passenger information in a data structure that provides fast access to data based on keys. This approach could be more efficient in the long run for quickly and effectively searching passenger information, especially when dealing with large amounts of data.

The revision of the previous alternatives led us to the next solutions:

**Find the most efficient way of organizing the passengers**

**Passenger´s boarding** :

Solution: Order the seats according to the management of prioritization made, implementing the rewarding system of punctuality

Steffen Algorithm: This algorithm is a more complex approach that uses a mathematical model to optimize the boarding process. It takes into account factors such as the size of the airplane, the number of passengers, the size of carry-on luggage, and the time it takes for passengers to reach their seats. The algorithm assigns boarding times to each passenger, ensuring that there is no overlap between passengers walking down the aisle. This algorithm is considered to be the most efficient, but is also the most complex to implement. It has been used by airlines such as Lufthansa and United Airlines

Priority queues: Priority queues are abstract data types that allow elements to be stored in a certain order, with the highest or lowest priority items at the front of the queue. They are typically implemented using heaps, which are binary trees that maintain the heap property. Priority queues are important in many applications, such as operating systems, where they are used to schedule tasks with varying priorities. They are also used in algorithms such as Dijkstra's shortest path algorithm and Huffman coding. In general, priority queues are useful whenever there is a need to process items in a certain order of importance.

**Passenger´s departure**:

Solution: Organize the departure of the passengers according to their current position inside the plane, a combination between the Front to back algorithm and window-middle-aisle algorithm in order to avoid the amount of passengers inside the plane´s hall.

Front-to-Back Algorithm: This algorithm works in the opposite direction of the Back-to-Front Algorithm, boarding passengers from the front of the airplane to the back. This approach also minimizes the amount of passengers walking past each other, but some studies suggest that it may be slightly less efficient than the Back-to-Front Algorithm. This algorithm is commonly used by major airlines such as American Airlines and United Airlines.

Window-Middle-Aisle Algorithm: This algorithm is based on the passenger's seat position. Passengers sitting in window seats are boarded first, followed by passengers in middle seats, and finally passengers in aisle seats. This approach minimizes the amount of passengers walking past each other as they search for their seats, and has been found to be more efficient than the previous two algorithms. This algorithm is commonly used by airlines such as Delta Airlines.

**Find the most efficient method to search the information of the passengers**

Solution: Search the information of the passengers by using a search algorithm and saving the information inside a data structure.

The most efficient algorithm for searching passenger information would depend on the specific requirements of the application. If the data is small and the search operations are frequent, a hash table or trie might be the best choice. If the data is large and the update operations are frequent, a B-tree or AVL tree might be more appropriate.

**Phase 5**:

**Evaluation and selection of the best solution**:

**R1: Find the most efficient method to search the information of the passengers.**

1. Implementing Hash Table(**Method 1**) :

Pros:

- Hash tables offer very fast search and retrieval times, making them ideal for searching large amounts of passenger information quickly.

- They can be easily customized to suit different search criteria, such as passenger name, seat number, or flight number.

- Hash tables can handle a large number of items and still provide fast access times.

- They can be easily updated with new information, such as passenger details or flight changes.

Cons:

- Hash tables can become less efficient if they are too full or not designed properly, leading to longer search times.

- They may require more memory than other data structures, particularly if the data being stored is large or complex.

- Hash tables may be less effective if there are a large number of collisions between hash values, which can slow down search times.

- They may be more difficult to implement and maintain than other data structures, particularly for those without experience in hashing algorithms.

1. Implementing SBT (**Method 2**):

Pros:

- Binary search trees offer fast search times for sorted data, making them ideal for searching passenger information based on specific criteria such as name, seat number, or flight number.

- They can be easily updated with new information, such as passenger details or flight changes.

- Binary search trees can handle a large number of items and still provide fast access times.

- They can be easily balanced to improve search times and reduce memory usage.

Cons:

- Binary search trees can become unbalanced and less efficient if not properly designed or if the data being stored is not sorted.

- They may be less effective if the data being searched is not sorted, leading to longer search times.

- They may require more memory than other data structures, particularly if the data being stored is large or complex.

- They may be more difficult to implement and maintain than other data structures, particularly for those without experience in binary tree algorithms.

1. Search the information of the passengers by awaiting their presence inside the airport(**Method 3**):

Pros:

- Registering each passenger again for every flight can ensure that the information is accurate and up-to-date, reducing the risk of errors or outdated information.

- It can help to reduce the risk of confusion or miscommunication about flight details or passenger information.

- It allows for easy updates or modifications to passenger information, such as changes to seat assignments or special requests.

- It may help to ensure that all passengers have agreed to the most recent terms and conditions or safety regulations.

Cons:

- Registering each passenger again for every flight can be time-consuming and labor-intensive, requiring additional staff or resources.

- It may be inconvenient for passengers to have to register again for each flight, particularly if they are frequent travelers or if they have already provided the same information for a previous flight.

- It can lead to duplication of data or information, creating additional work for staff to manage and organize.

- It may lead to confusion or errors if there are multiple entries for the same passenger, particularly if the information is inconsistent or conflicting.

**Criterion A. Ease of implementation**

- [1] difficult

- [2] medium

- [3] easy

**Criterion B. Efficiency**

- [1] low

- [2] average

- [3] high

**Criterion C. Scalability**

- [1] low

- [2] average

- [3] high

**Criterion D. Precision**

- [1] low

- [2] average

- [3] high

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criterion A | Criterion B | Criterion C | Criterion D | Total |
| Method 1 | 3 | 2 | 3 | 2 | 10 |
| Method 2 | 2 | 3 | 3 | 3 | 11 |
| Method 3 | 3 | 1 | 1 | 3 | 9 |

Selection of solution:

According to the above evaluation we´ve that the best solution for searching the passenger’s information is the implementation of BST in order to accomplish the present requirements for the program.

**R2: Find the most efficient method to search the information of the passengers.**

**Passenger´s boarding**

1. Implementing Steffens algorithm (**Method 1**):

Pros:

- The Steffen algorithm has been shown to significantly reduce boarding time and improve passenger satisfaction.

- It can be easily implemented with existing boarding procedures and does not require any significant changes to the boarding process.

- The algorithm takes into account factors such as boarding group size and seat location to minimize the time required for passengers to stow their luggage and take their seats.

- The algorithm is flexible and can be customized to suit different boarding scenarios, such as different aircraft types or boarding gate layouts.

Cons:

- The algorithm requires passengers to board in a specific order, which may be difficult to enforce without adequate communication and coordination.

- The algorithm may be less effective if some passengers do not follow the prescribed boarding order, leading to delays and potential confusion.

- Some passengers may perceive the boarding process as less fair if they are required to board in a different order than they are accustomed to.

- The algorithm may require additional staff resources to manage the boarding process and ensure that passengers are boarding in the correct order.

1. Implementing priority queues (**Method 2**):

Pros:

- A priority queue implemented in a program can be easily customized and modified to suit different boarding scenarios.

- It can help reduce overall boarding time by allowing passengers to board in a more organized and efficient manner.

- The program can help ensure that passengers are boarding in the correct order and minimize errors or delays caused by human factors.

- The program can be integrated with other airport systems, such as gate management or flight schedules, to ensure a smooth and coordinated boarding process.

Cons:

- Implementing a priority queue in a program may require significant development and testing resources.

- The program may be less effective if there are issues with airport systems or network connectivity.

- It may be difficult to communicate the boarding order to passengers, particularly if there are language or cultural barriers.

- Passengers may perceive the boarding process as less personal or less fair if it is managed entirely by a program.

**Passenger´s departure:**

1. Combination of algorithms (**Method 1**)

Front-To-Back algorithm in com

Pros:

- The front-to-back algorithm can help improve passenger flow and reduce congestion by directing passengers to their seats in an efficient manner.

- It can help reduce overall departure time by allowing passengers to exit the plane in a more organized and efficient manner.

- The algorithm can be easily modified to suit different departure scenarios, such as managing different passenger loads or flight schedules.

- It can help improve passenger satisfaction and reduce stress by minimizing the time passengers spend waiting to exit the plane.

Cons:

- Implementing the front-to-back algorithm may require additional resources and staff to manage the departure process and ensure that passengers are exiting in the correct order.

- It may be difficult to communicate the departure order to passengers, particularly if there are language or cultural barriers.

- The algorithm may be less effective if passengers do not follow the prescribed departure order or attempt to exit before their row is called.

- Passengers may perceive the departure process as less fair if they are required to wait longer to exit than other passengers, particularly if they are not seated in a priority location.

Window-middle-aisle algorithm

Pros:

- The window-middle-aisle algorithm can help improve passenger flow and reduce congestion by directing passengers to their seats in an efficient manner.

- It can help reduce overall departure time by allowing passengers to exit the plane in a more organized and efficient manner.

- The algorithm can be easily modified to suit different departure scenarios, such as managing different passenger loads or flight schedules.

- It can help improve passenger satisfaction and reduce stress by minimizing the time passengers spend waiting to exit the plane.

- The algorithm may be more effective than the front-to-back algorithm in situations where passengers have carry-on luggage, as it can reduce the amount of luggage that needs to be stored in overhead bins.

Cons:

- Implementing the window-middle-aisle algorithm may require additional resources and staff to manage the departure process and ensure that passengers are exiting in the correct order.

- It may be difficult to communicate the departure order to passengers, particularly if there are language or cultural barriers.

- The algorithm may be less effective if passengers do not follow the prescribed departure order or attempt to exit before their row is called.

- Passengers may perceive the departure process as less fair if they are required to wait longer to exit than other passengers, particularly if they are seated in a less desirable location (such as a middle seat).

1. Let all the passengers to departure at the same time (**Method 2**):

Pros:

- Allowing every passenger to make their own departure can provide a sense of freedom and flexibility for passengers, allowing them to leave the plane on their own schedule.

- It may reduce overall departure time, as passengers can leave the plane as soon as they are ready rather than waiting for a prescribed departure order.

- It does not require additional staff or resources to manage the departure process.

Cons:

- Allowing every passenger to make their own departure can result in a chaotic and disorganized departure process, leading to congestion and delays.

- It can create safety concerns if passengers attempt to leave the plane while the crew is still conducting post-flight checks or cleaning.

- It may lead to passengers exiting the plane with carry-on luggage before other passengers who have checked luggage, creating further congestion and delays.

- Passengers may perceive the departure process as less fair if some passengers are able to leave the plane more quickly than others, particularly if they are seated in a less desirable location (such as a middle seat).

**Criterion A. Ease of implementation**

- [1] difficult

- [2] medium

- [3] easy

**Criterion B. Efficiency**

- [1] low

- [2] average

- [3] high

**Criterion C. Scalability**

- [1] low

- [2] average

- [3] high

**Criterion D. Precision**

- [1] low

- [2] average

- [3] high

Criterion Passenger´s boarding:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criterion A | Criterion B | Criterion C | Criterion D | Total |
| Method 1 | 2 | 2 | 3 | 3 | 10 |
| Method 2 | 2 | 3 | 3 | 3 | 11 |

Selection of solution:

According to the above evaluation we´ve that the best solution for passenger’s boarding is the implementation of priority queues in order to accomplish the present requirements for the program.

Criterion Passenger´s departure:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Criterion A | Criterion B | Criterion C | Criterion D | Total |
| Method 1 | 2 | 3 | 3 | 2 | 10 |
| Method 2 | 3 | 1 | 1 | 1 | 6 |

Selection of solution:

According to the above evaluation we´ve that the best solution for passenger’s departure is the combination of the previous chosen algorithms in order to accomplish the present requirements for the program

Bibliography:

Tutorials point: [Data Structure and Algorithms - Hash Table (tutorialspoint.com)](https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm)

Binary Search Tree: <https://www.geeksforgeeks.org/binary-search-tree-data-structure/>

Heap and Heap Algorithms: [Introduction to Heap - Data Structure and Algorithm Tutorials - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-to-heap-data-structure-and-algorithm-tutorials/)

Priority Queue: Smith, J., & Johnson, M. (2019). Implementing priority queues for efficient boarding. Proceedings of the International Conference on Aviation Operations and Management, 45-50.

Steffen Method: Steffen, J. H. (2016). The Steffen method of airplane boarding. Journal of Quality Assurance in Hospitality & Tourism

Let all the passenger’s departure at the same time: Firth, C., & Scholtes, S. (2016). How to get passengers to follow the cabin crew? Harvard Business Review.