**Mars Missions Management**

**Course:** Data Structures and Algorithms Analysis

**Course code:** CIE 205

**Phase (1)**:

Each mission and rover list you chose The DS you chose for each list Your justification of all your choices with the complexity of the most frequent or major operation for each list.

* Chosen list.
* DS for each list.
* Justification for chosen list.
* Complexity of main operations.

**Rovers:**

1. **Available rovers:**

* Available rovers of each type are stored in separate linked queues
* When one is needed for a mission

-We check if queue is empty.

-If not, we dequeue directly first one available.

**Main operations:**

-Insertion (Enqueue)

**-**Deletion (Dequeue)

**Complexity: O (1**)

1. **Busy in a mission**

* Busy rovers are stored in priority queue (priority is according to mission duration)

**Main operations:**

-Insertion (to start mission)

-Swap (if mission is canceled, it swapped to be of highest priority and get deleted)

-Deletion (if mission is canceled or mission ended)

* After it finishes its mission=> we check total number of missions it did (One comparison)

-If it passes the allowed number of missions before checkup, it is enqueued in checkup list and directly starts it checkup duration.

-If not, it is enqueued in Availability queue of its own type

**Main Operations**:

* -Insertion and deletion

***Best*: Complexity O(logn)**

***Worst*: Olog(n)**

1. **Busy in Checkup**

* Rovers in checkup are stored in priority queue (Priority is not an equation, it is according to the end time of duration).
* Rovers that end checkup are dequeued first

**Main operations:**

-Insert (To start checkup)

-Deletion

**Complexity O (1)**

**Decision of priority of assignment of missions of same type:**

1. **Emergency missions**

We implemented priority queue using BST

Complexity O (Tree height) and worst case is O(n)

After implementing it we noticed that it is better to implement it using heaps in order to have complexity O (log n) which is better.

1. **Mountainous and polar missions**

Implemented using linked queues since missions that are formulated first are the ones assigned first.

**Once mission is formulated, and reached its turn to be assigned (Begins rover searching):**

1. **Mountainous**

* Check available Mountainous Rovers queue
* if empty->check available emergency
* if empty ->enter wait list priority queue

**Main operations:**

-Searching (First element in each queue)

-Deletion ( if available dequeue)

-Insert (enter waiting list)

**Complexity O (1)**

1. **Polar**

* Check available polar Rovers queue

if empty -> enter wait, if not dequeue

**Main operation:**

-Searching

-Deletion

-Insertion

**(3) Emergency**

* Check available emergency rovers’ queue

if empty -> check available mountainous

if empty -> check available polar

if empty -> enter wait

**Main operations:**

**-**Searching

-Deletion

-Insertion

BST best cas(Tree height)

Best case O(log n)

WorstO (n)

**Linked list O(n) , O(1)**