**Documentation of Priority Queues**

**1. Data Structure Used**

For managing job queue we are using custom priority queue. It is designed on the python list.

**1.1 Priority Queue Class Characteristics**

* **Initialization:** When priority queue instances are created than empty list self.queue is created to store the jobs.
* **is\_empty method:** It checks that queue is empty or not by verifying the length of self.queue if it is zero. It will help to find if jobs are waiting or not.
* **enqueue method:** Job is added to queue by this method. First jobs are appended to self.queue list. Then sort the list in descending order based on the priority of job. Highest priority job is always at the front of the list.
* **dequeue method:** It is used to remove the jobs and give jobs with highest priority from the queue. Also it check if the list is not empty than pop the first element from the queue.

**2. Main procedure**

**2.1 Job Class**

It is used to represent a single job in the system. Every jobs has the attributes that are given below

* **name:** A string will identify the jobs that are “A”, “B”.
* **priority:** The highest priority is 10 and the lowest is 1.
* **time:** Time between 1 and 50 that show how time is needed to complete the job.
* **waiting\_time:** It shows the time spent by the job waiting in the queue before execution.

**3. simulate\_jobs Function**

It is important function for simulating the job.

* **input\_jobs:** It is a list of strings every string represents a job in the format “Job Name Job Priority Time Required to Complete”.
* **Simulate\_random\_arrivals:** A boolean flag that will determines that new jobs will be added randomly while simulation.

Functions perform the following steps.

* **Initialization:** It will create instances of the PriorityQueue class and populates with the jobs that are in input\_jobs list.
* **Simulation Loop:** It will enter the looping process until the jobs are available in the priority queue and when job is completed it is added into the result list.
* **Return Result:** This function will return the result list that will contains the jobs order they are completed.

**4. Efficiency Analysis**

Let here *n*  is representing the number of jobs that are executed by the program.

**4.1 Time Complexity**

* **Initial Queue Population:** Time taken for parsing the jobs and then adding them into the priority queue takes *O*(*n*) time. The reason is that we have to perform iteration for the *n* inputs of the job once. So job object will be created for each and then add it to queue.
* **enqueue Operation:** It is used for sorting the queue every time new job is added into the queue. Sorting operation time complexity is *O*(*mlogm*). *m* is number of jobs in the queue.
* If jobs are added one by one then worst case will occur and time needed for performing simulation operation will be *O*(*n2logn).*
* **dequeue Operation:** It has time complexity *O(*1) because its role is just to remove the first element from the list.
* **Simulation Operation:** Main simulation loop is iterating through all jobs once. Inner loop that simulates the job execution will take time *O(*t) that is the most time taken to complete the jobs. As all jobs time is proportional to *n*. So the overall time complexity of the simulation is *O(n).*

Overall time complexity is more in the enqueue method due to sorting operations.

So the worst case time complexity is *O*(*n2logn).*

**5. Space Complexity**

**Job Storage:** As all jobs are stored in priority queue by the program. So for the worst case maximum number jobs in the queue in given time are *n.* So memory needed to store the jobs is *O*(*n).*

**Result List:** It is used to store of all jobs that are completed. As it contains *n* jobs so space needed for result list is *O*(*n).*