PROBLEM STATEMENT:

Considering the arrival time and burst time requirement of the process the scheduler schedules the processes by interrupting the processor after every 6 units of time and does consider the completion of the process in this iteration. The scheduler than checks for the number of process waiting for the processor and allots the processor to the process but interrupting the processor every 10 unit of time and considers the completion of the processes in this iteration. The scheduler checks the number of processes waiting in the queue for the processor after the second iteration and gives the processor to the process which needs more time to complete than the other processes to go in the terminated state.

**The inputs for the number of requirements, arrival time and burst time should be provided by the user.**

Consider the following units for reference. Process Arrival time Burst time

|  |  |  |
| --- | --- | --- |
| P1 | 0 | 20 |
| P2 | 5 | 36 |
| P3 | 13 | 19 |
| P4 | 26 | 42 |
|  |  |  |

ANALYSIS OF THE PROBLEM STATEMENT (DESCRIPTION):

* There are the 3 parts in the given problem statement.
* Firstly it’s clearly mentioned that the arrival and burst time are two parameters to be considered while scheduling the processes.
* As mentioned earlier there are three parts/constrains in the statement are as follows:
* First: The scheduler schedules the processes by interrupting the processor after every 6 units of time.

Then the scheduler than checks for the number of process waiting for the processor and allots the processor to the process

* Second : The scheduler then checks for the number of process waiting for the processor and allots the processor to the process but interrupting the processor every 10 unit of time
* Third: Then the scheduler gives the processor to the process which needs more time i.e. (burst time) to complete than the other processes to go in the terminated state.

<https://github.com/Nandasagarre/sagar>

ALGORITHM

* The code was chosen to be written in python.
* The explination for the code flow and the algorithm is shown in fig 1.0
* The working the of the code on an given example is shown in fig 1.2

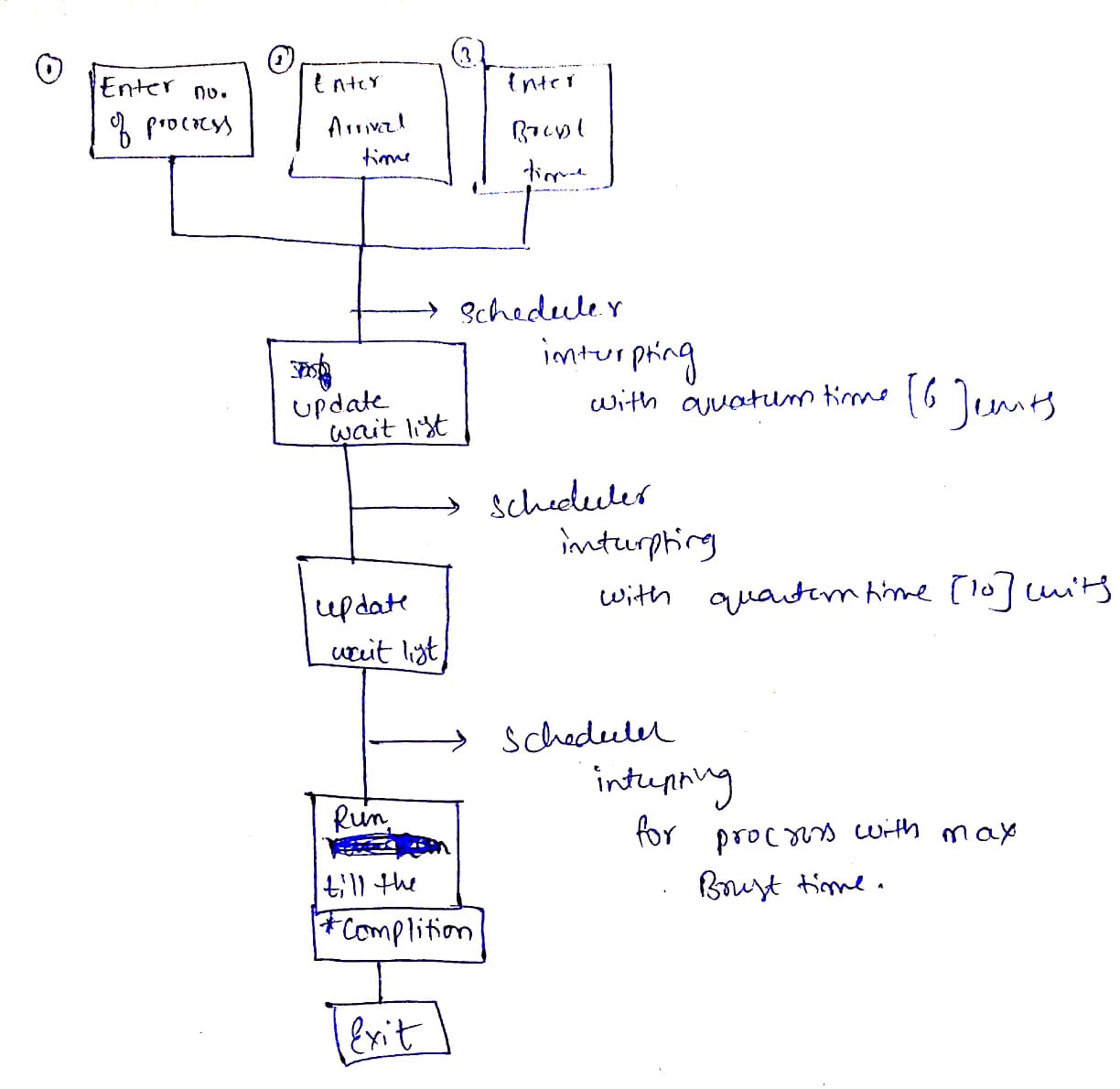


Fig 1.0 explains the code flow and the algorithm

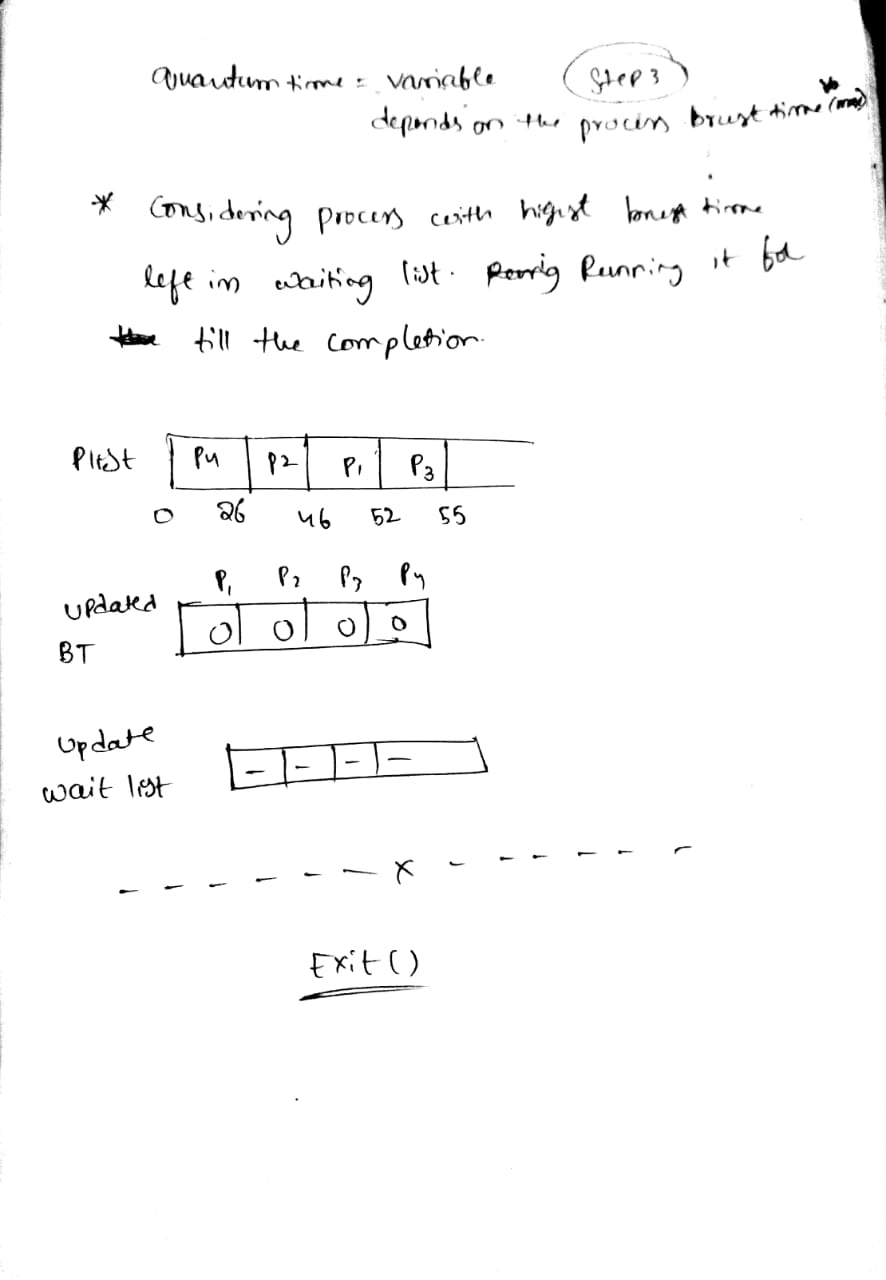
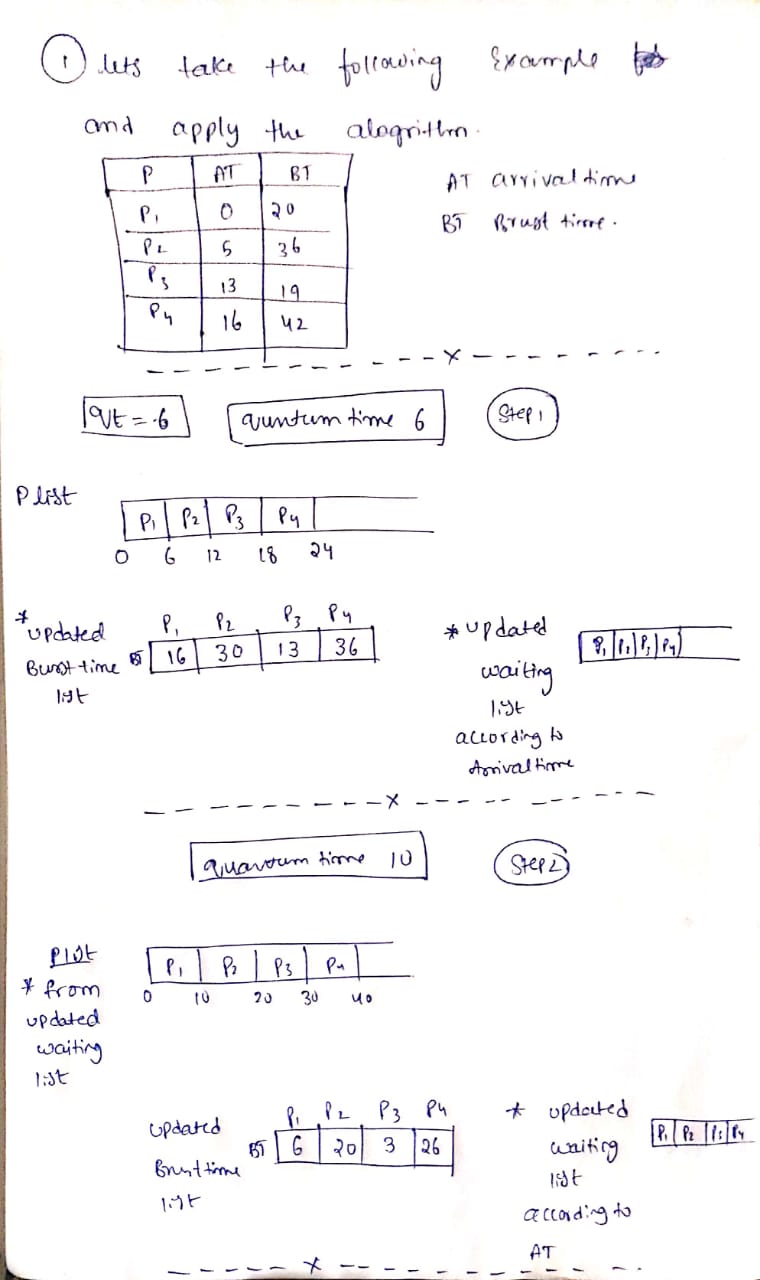


Fig 1.2 explains how the algo works on an given example