

Niloufar Babaahmadi
(610398103)

Project: Priority Scheduling

Introduction

Operating system (OS) is a collection or package of numerous soft-wares which act as an intermediary between the computer hardware and end user. It provides a means for the user to work with the computer. Operating system performs major functions like memory management, process scheduling and resource management. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). CPU Scheduling is the basis of multi programmed operating systems. By switching the CPU among the processes, the operating system can make the computer system productive.

CPU Scheduler

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler or CPU Scheduler. The scheduler selects a process from the processes in memory that are ready to execute and allocate the CPU to that process.

CPU SCHEDULING POLICIES

Generally, the policies of scheduling either pre-emptive or non-preemptive

Non-preemptive scheduling

Non-preemptive occurs in a multiprogramming system which means the short-term scheduler permits the process to run until it terminates or in some cases waiting for an event. In another word, the current process frees the CPU either by finishing or by changing to the state of waiting.

Preemptive scheduling

Preemptive rules push the process which is active currently for releasing the CPU on particular events for instance, a clock interrupts, input & output interrupts and a call of the system. The process that is executed currently needs to release the CPU involuntarily if a high-priority process puts into the ready queue.

CPU SCHEDULING ALGORITHMS

Scheduling stands for deciding which jobs run whenever there are more than runnable jobs. There are various objectives for competing these scheduling algorithms. These objectives are

the throughput and turnaround time as well as the response time. In another term of meaning the scheduling of CPU is the procedure of identifying which process in the queue will allocate firstly to the CPU. Many types of well-known scheduling algorithms will be described in the next subsections.

Priority scheduling for preemptive and non-preemptive types

In the priority scheduling algorithm, there is a classification of processes according to some system criteria depending on the processed type. The priority of the process is made by a group of measures and any process inserting the ready queue provides its importance as a priority. Only the priority number determines the allocating decision to the CPU for a process in a way that the high-value priority of the process will arrive at the CPU first or next. Two types of versions exist for this algorithm which is preemptive and non-preemptive. In the preemptive Type of this algorithm, the lowest priority process may be suffering from starvation in case of a process with big priority keep to coming to the ready queue.

Characteristics of Priority Scheduling

- A CPU algorithm that schedules processes based on priority.
- It used in Operating systems for performing batch processes.
- If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED basis.
- In priority scheduling, a number is assigned to each process that indicates its priority level.
- Lower the number, higher is the priority.
- In this type of scheduling algorithm, if a newer process arrives, that is having a higher priority than the currently running process, then the currently running process is preempted.

Advantages of priority scheduling

- Easy to use scheduling method
- Processes are executed on the basis of priority so high priority does not need to wait for long which saves time
- This method provides a good mechanism where the relative important of each process may be precisely defined.
- Suitable for applications with fluctuating time and resource requirements.

Disadvantages of priority scheduling

- If the system eventually crashes, all low priority processes get lost.
- If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinite time.
- This scheduling algorithm may leave some low priority processes waiting indefinitely.
- A process will be blocked when it is ready to run but has to wait for the CPU because some other process is running currently.

- If a new higher priority process keeps on coming in the ready queue, then the process which is in the waiting state may need to wait for a long duration of time.

Example1:

PROCESS	BURST TIME	PRIORITY
P1	21	3
P2	3	1
P3	6	4
P4	2	2

```

11, 2, 4, 2, 21
Total Waiting Time:15.0
Average Waiting Time:3.0
Total Turnaround Time:18.0
Average Turnaround Time:3.5
Menu:
1. priority Algorithm
2. quit
Enter : 1
Enter the no of processes:
4
Enter the burstTime and priority for process p0
21 3
Enter the burstTime and priority for process p1
3 1
Enter the burstTime and priority for process p2
6 4
Enter the burstTime and priority for process p3
2 2
11, 2, 4, 21
12, 1, 4, 21
Total Waiting Time:15.0
Average Waiting Time:3.0
Total Turnaround Time:18.0
Average Turnaround Time:3.5
Menu:
1. priority Algorithm
2. quit
select : 1

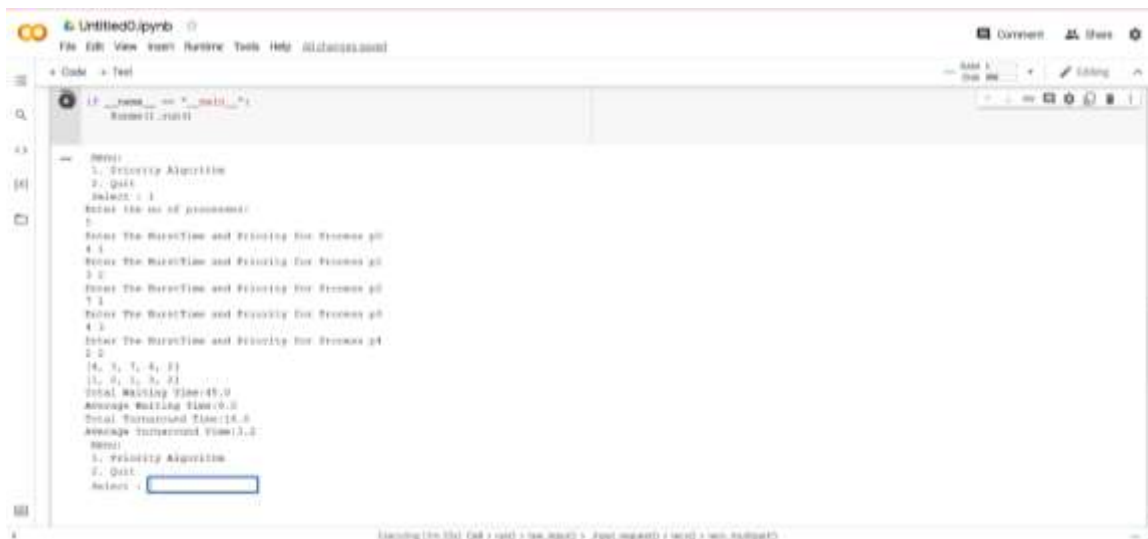
```

Example2:

Process	Priority	Burst time
P1	1	4
P2	2	3
P3	1	7
P4	3	4
P5	2	2

Implementation:

1. First, we will input the processes with their burst time and priority. (P_n arrives earlier than P_{n+1})
2. First process will schedule, which have the lowest arrival time, if two or more processes will have lowest arrival time, then whoever has higher priority will schedule first.
3. Now further processes will be schedule according to the arrival time and priority of the process.
4. Once all the processes have been arrived, we can schedule them based on their priority.



```
if __name__ == '__main__':
    Run()

def Run():
    menu()
    1. Priority Algorithm
    2. Quit
    select = 1
    while True:
        print the no of processes:
        5
        Enter The BurstTime and Priority for Process p0
        4 1
        Enter The BurstTime and Priority for Process p1
        3 2
        Enter The BurstTime and Priority for Process p2
        1 1
        Enter The BurstTime and Priority for Process p3
        4 1
        Enter The BurstTime and Priority for Process p4
        2 2
        [4, 1, 1, 4, 1]
        [1, 0, 1, 3, 2]
        Total Waiting Time:45.0
        Average Waiting Time:9.0
        Total Turnaround Time:18.0
        Average Turnaround Time:3.6
    menu()
    1. Priority Algorithm
    2. Quit
    select =
```

```

# Priority Algorithm
def Priority(self):
    Twt = 0.0
    Bst = 0.0
    Tat = 0.0
    w=0.0
    B = []
    P = []
    self.n=4
    self.n = int(input("Enter the no of processes:\n"))
    pMax=6
    Wt = [0]*self.n
    for i in range(int(self.n)):
        b, p = input("Enter The Range of BurstTime and Priority for Process p" + str(i)+"\n").split()
        b = int(b)
        if b > 1:
            b = self.Rnd(b)
            print("Chosen BurstTime:"+str(b))
        B.append(b)
        p = int(p)
        if p > 1:
            p = self.Rnd(p)
            print("Chosen Priority:"+str(p))
        P.append(p)
        if pMax<int(p):
            pMax=int(p)

    print(B)
    print(P)

    for j in range(pMax):
        for i in range(0,self.n):
            if P[i]==j:
                Wt[i]=w
                w=w+B[i]

    for i in range(0, int(self.n)):
        Twt = Twt + Wt[i]
    for i in range(0, int(self.n)):
        Bst = Bst + Wt[i]
    for i in range(0, int(self.n)):
        Tat = Wt[i]+int(B[i])

    print("Total Waiting Time:"+str(Twt))

```

Implementation of the Random Function:

1. First, we will take the Random range.
2. A list will get created in this range.
3. Next, we will shuffle this list and choose an index and output the random number.

```

def Rnd(self, n):

    ls = []

    for i in range(1, n+1):

        ls.append(i)

    random.shuffle(ls)

    return(ls[n-1])

```

Mean and Derivation:

```

for i in range(0, int(self.n)):
    Twt = Twt + Wt[i]
for i in range(0, int(self.n)):
    Bst = Bst + Wt[i]
for i in range(0, int(self.n)):
    Tat = Wt[i]+int(B[i])
    T.append(Tat)

mean_Twt = Twt/int(self.n)
mean_Tat = Tat/int(self.n)

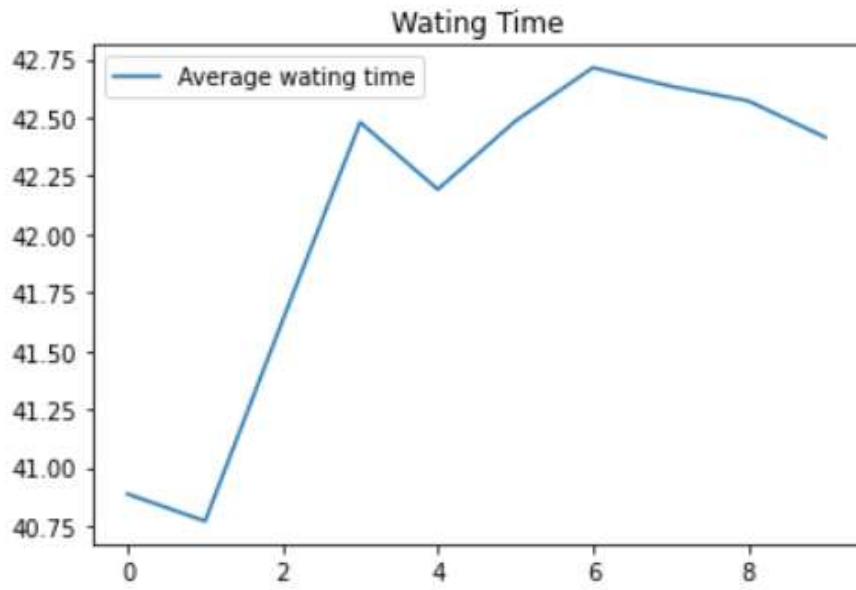
self.Twt_m_ls.append(mean_Twt)
self.Tat_m_ls.append(mean_Tat)

```

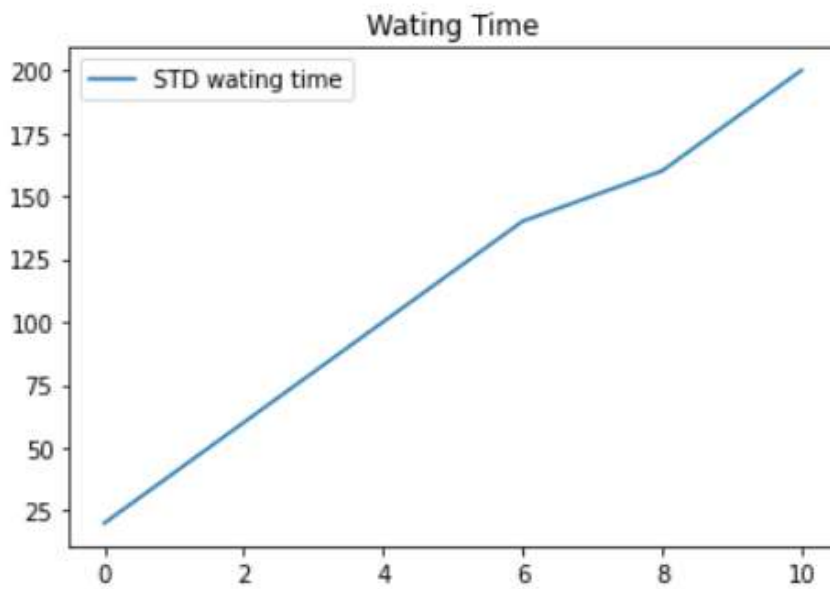
Plotting:

Each attribute was tested 200 times and the mean was calculated to get the best answers.

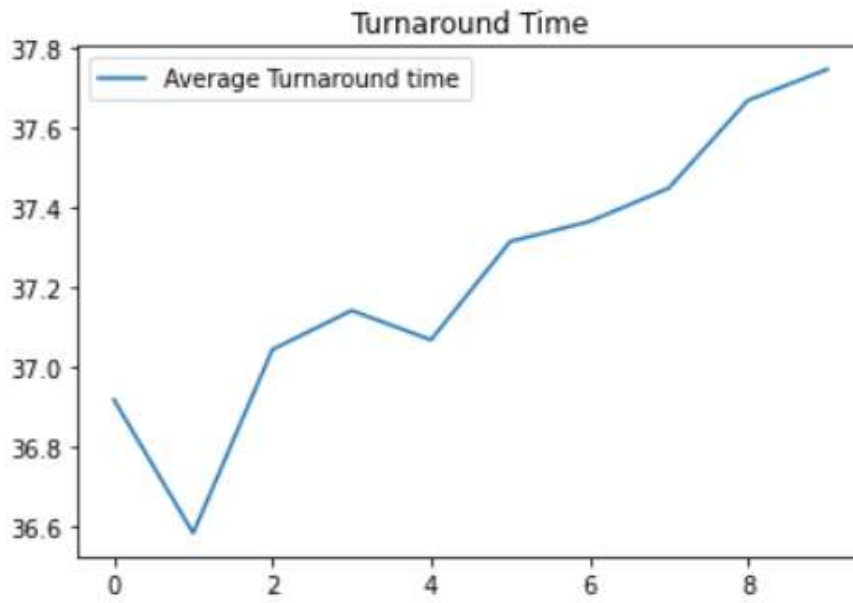
As you see; After approximately 35 processes, There is a sharp growing rate in the Average Waiting Time.



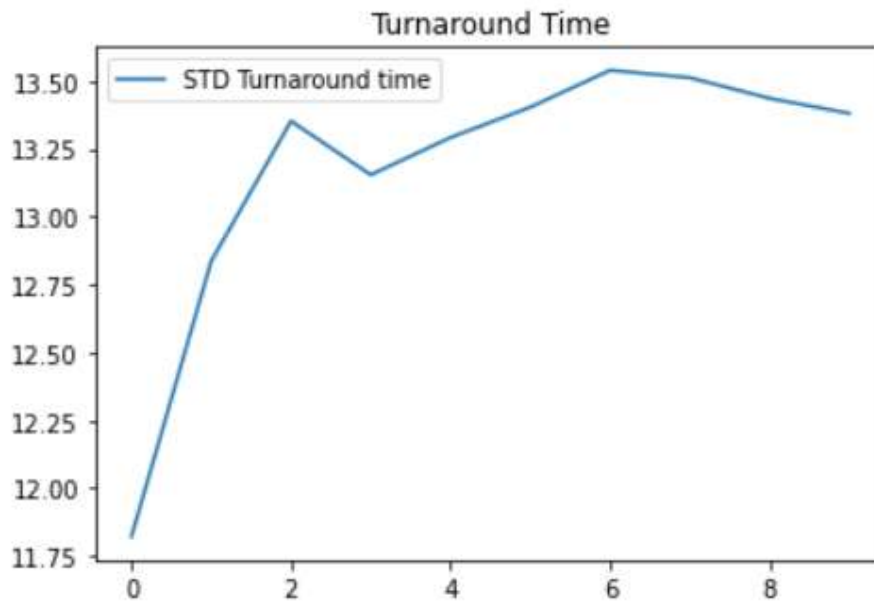
Standard Deviation's plot shows that It has an ascending chart.



There is a sharp growing rate in the Average Turnout Time.



Standard Deviation's plot shows that It has an acending chart.



Bar Graph:

In Priority Scheduling, we generally have a larger average Turnaround time than the average waiting time.

