Experiment – 4

Priority Scheduling

Introduction

Priority Scheduling is a scheduling process based on process priority. In this algorithm, the scheduler selects the tasks to work as per the priority. Max priority scheduler selects processes with higher priority first, whereas Min Priority scheduler selects processes with lower priority first. Processes with equal priorities are carried out. Process priority can be decided based on memory requirements, resource requirement, process urgency etc.

Priority Scheduling is of two types-

1. Preemptive Priority Scheduling

   In this type of priority scheduling the scheduler keeps looking for higher priority process even when a process is being executed, and if found then the currently executing process is stopped and swapped with the higher priority process.

2. Non-preemptive Priority

   In this type of priority scheduling the schedules process the higher priority process till it is completed irrespective of whether a higher priority process is waiting to be executed. Once the resources are allocated the high priority process holds on to it till termination, irrespective of any other higher priority process.

Algorithm

Preemptive Priority Scheduling

1. Maintain a heap on process priority to maintain process execution order
2. Update the heap every second based on process priority every second
3. If at any moment a process of higher priority is present in waiting heap, preempt the currently executing process in the CPU, and replace it with the higher priority task.
4. If a process is complete, discard it and set the CPU state to idle for future executions

Non-preemptive Priority Scheduling

1. Maintain a heap on process priority to maintain process execution order
2. Add new tasks to the heap when they arrive
3. If the CPU is idle, pick the highest priority task from heap for execution
4. Once the CPU is done with executing a process, discard it and set the CPU state to idle for future executions.

Numericals

Q1. Consider the following processes whose arrival time, burst time and priority is given, using Precemptive priority Scheduling and Non-Premptive priority scheduling

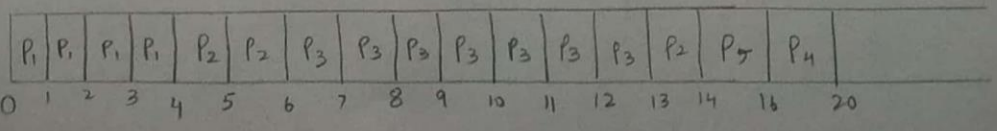
Process	Arrival Time	Burst Time	Priority
P_1	0	4	1
P_2	0	3	2
P_3	6	7	1
P_4	11	4	3
P_5	12	2	2

Ans Premptive Priority Scheduling

Process	AT	BT	Priority	CT	TAT	WT	RT
P_1	0	4	1	4	4	0	0
P_2	0	3	2	14	14	11	4
P_3	6	7	1	13	7	0	0
P_4	11	4	3	20	9	5	5
P_5	12	2	2	16	4	2	2
Average :					13.4	7.6	3.6	2.2

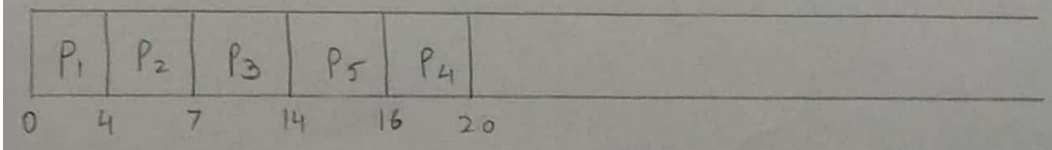
Processes

- P_1 (1) - ~~4~~ 0
- P_2 (2) - ~~3~~ ~~2~~ 0
- P_3 (1) - ~~7~~ ~~2~~ ~~1~~ 0
- P_4 (3) - ~~4~~ 0
- P_5 (2) - ~~2~~ 0



Non- Preemptive priority scheduling

Process	AT	BT	Priority	CT	TAT	WT	RT
P_1	0	4	1	4	4	0	0
P_2	0	3	2	7	7	4	4
P_3	6	7	1	14	8	1	1
P_4	11	4	3	20	9	5	5
P_5	12	2	2	16	4	2	2
			Average	12.2	6.4	2.4	2.4



Q2. Concider the following process whose arrival time, burst time and priority is given, using preemptive priority scheduling and non-preemptive priority scheduling

Process	Arrival time	Burst Time	Priority
P_1	0	6	3
P_2	4	1	1
P_3	5	2	2
P_4	7	3	4
P_5	8	4	1

Ans. Preemptive priority scheduling

Process	AT	BT	priority	CT	TAT	WT	RT
P_1	0	6	3	13	13	7	0
P_2	4	1	1	5	1	0	0
P_3	5	2	2	7	2	0	0
P_4	7	3	4	16	9	6	6
P_5	8	4	1	12	4	0	0
			Average	10.6	5.8	2.6	1.2

Processes

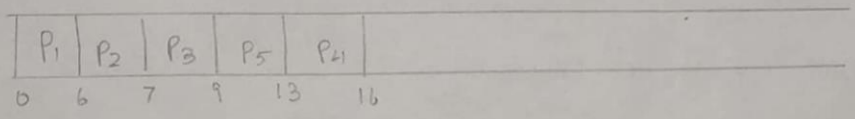
- P_1 (3) - ~~0~~ ~~1~~ 10
- P_2 (1) - X 0
- P_3 (2) - X 0
- P_4 (4) - 3
- P_5 (1) - X 0

P_1	P_2	P_3	P_1	P_5	P_1	P_4

0 4 5 7 8 12 13 16

Non- preemptive priority scheduling

Process	AT	BT	priority	CT	TAT	WT	RT
P_1	0	6	3	6	6	0	0
P_2	4	1	1	7	3	2	2
P_3	5	2	2	9	4	2	2
P_4	7	3	4	16	9	6	6
P_5	8	4	1	13	5	1	1
				Average	10.2	5.4	2.2	2.2



Q3. Consider the following processes whose arrival time, burst time and priority is given using preemptive and non-preemptive priority scheduling

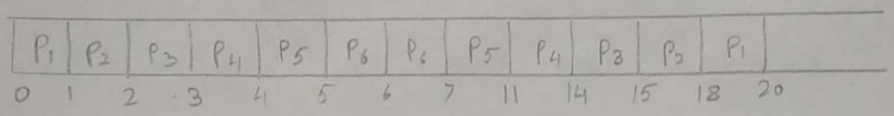
Process	Arrival time	Burst time	priority
P_1	0	3	6
P_2	1	4	5
P_3	2	2	4
P_4	3	4	3
P_5	4	5	2
P_6	5	2	1

Ans Preemptive priority scheduling

Process	AT	BT	Priority	CT	TAT	WT	RT
P_1	0	3	6	20	20	17	0
P_2	1	4	5	18	17	13	0
P_3	2	2	4	15	13	11	0
P_4	3	4	3	14	11	7	0
P_5	4	5	2	11	7	2	0
P_6	5	2	1	7	2	0	0
				Average	14.17	11.67	8.34	0

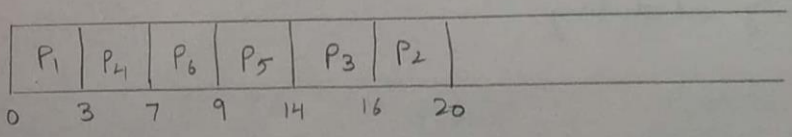
Processes

[0] · P_1 (6) - 32
[1] · P_2 (5) - 43
[2] · P_3 (4) - 21
[3] · P_4 (3) - 43
[4] · P_5 (2) - 510
[5] · P_6 (1) - 210



Non - preemptive priority scheduling

Process	AT	BT	priority	CT	TAT	WT	RT
· P_1	0	3	6	3	3	0	0
· P_2	1	4	5	20	19	15	15
· P_3	2	2	4	16	14	12	12
· P_4	3	4	3	7	4	0	0
· P_5	4	5	2	14	10	5	5
· P_6	5	2	1	9	4	2	2
			Average	11.5	9.0	5.67	5.67



Code (Pre-emptive Priority Scheduling)

```
import heapq

class Process:
    def __init__(self, idx, AT, BT, priority) -> None:
        self.idx = idx
        self.AT = AT
        self.BT = BT
        self.CT = None
        self.priority = priority
        self.firstExecution = None

        # record remaining time as algorithm is premptive,
        # hence any process can be stopped in between
        self.remaining = BT

    def calc(self) -> None:
        self.TAT = self.CT - self.AT
        self.WT = self.TAT - self.BT
        self.RT = self.firstExecution - self.AT

    def __lt__(self,other) -> bool:
        if self.priority==other.priority:
            return self.AT<other.AT
        return self.priority<other.priority

    def __repr__(self) -> str:
        return f"
Process({self.idx}): {self.AT}, {self.BT}, {self.priority}, {self.CT}, {self.TAT}
, {self.WT}, {self.RT}"

n = int(input("Number of processes: "))
arrivalTime = list(map(int, input("Arrival Times: ").split()))
burstTime = list(map(int, input("Burst Times: ").split()))
priority = list(map(int, input("Priorities: ").split()))

processes = sorted([Process(x+
1,arrivalTime[x],burstTime[x],priority[x]) for x in range(len(arrivalTime))],key
=lambda x:x.AT)

# as deletion in list is not allowed
# while iterating using for loop, hence
# keep a pointer to rember till which idx process are arrived
# pointer would be linear as process are already sorted using AT
```

```
processID = 0

heap = []
completed = []
cpuTime = processes[0].AT

while len(completed)<n:
    # Processes arriving
    for p in range(processID,n):
        p = processes[p]
        if(p.AT<=cpuTime):
            heapq.heappush(heap,p)
            processID+=1
        else:
            break

    cpuTime+=1

    # Arrival done for that second
    # solving cpu processing for past second
    if heap:
        # topmost process gets processed for 1 sec,
        # based on old and new process that arrived in the
        # past second
        heap[0].remaining-=1

        # if the process is encountered first time
        if heap[0].firstExecution == None:
            # -1 as it was processded in the previous second of cpuTime
            heap[0].firstExecution = cpuTime-1

        # process complete
        if heap[0].remaining==0:
            # no -1 as the process was completed after
            #  the previous cpuTime second hence -> now
            heap[0].CT = cpuTime
            heap[0].calc()
            completed.append(heapq.heappop(heap))

# outputs
print("Process, AT, BT, Priority, CT, TAT, WT, RT")
[print(x) for x in sorted(completed,key=lambda x: x.idx)]

print("\nDeekshant Wadhwa- 01296303118")
print("\nAverage:")
```

```
print(f"CT: {sum((x.CT for x in completed))/n}")
print(f"TAT: {sum((x.TAT for x in completed))/n}")
print(f"WT: {sum((x.WT for x in completed))/n}")
print(f"RT: {sum((x.RT for x in processes))/n}")
```

Output (Pre-emptive Priority Scheduling)

```
PS D:\Drive\Sem 6\OS\lab> python -u "d:\Drive\Sem 6\OS\lab\priority-prempt.py"
Number of processes: 6
Arrival Times: 0 1 2 3 4 5
Burst Times: 3 4 2 4 5 2
Priorities: 6 5 4 3 2 1
Process, AT, BT, Priority, CT, TAT, WT, RT
Process(1): 0, 3, 6, 20, 20, 17, 0
Process(2): 1, 4, 5, 18, 17, 13, 0
Process(3): 2, 2, 4, 15, 13, 11, 0
Process(4): 3, 4, 3, 14, 11, 7, 0
Process(5): 4, 5, 2, 11, 7, 2, 0
Process(6): 5, 2, 1, 7, 2, 0, 0

Deekshant Wadhwa- 01296303118

Average:
CT: 14.166666666666666
TAT: 11.666666666666666
WT: 8.333333333333334
RT: 0.0
PS D:\Drive\Sem 6\OS\lab>
```

Code (Non-Pre-emptive Priority Scheduling)

```
import heapq

class Process:
    def __init__(self, idx, AT, BT, priority) -> None:
        self.idx = idx
        self.AT = AT
        self.BT = BT
        self.CT = None
        self.priority = priority
        self.firstExecution = None

    def calc(self):
        self.TAT = self.CT - self.AT
        self.WT = self.TAT - self.BT
        self.RT = self.firstExecution - self.AT

    def __lt__(self,other):
        if self.priority==other.priority:
            return self.AT<other.AT
        return self.priority<other.priority


    def __repr__(self) -> str:
        return f"
Process({self.idx}): {self.AT}, {self.BT}, {self.priority}, {self.CT}, {self.TAT}
, {self.WT}, {self.RT}"

n = int(input("Number of processes: "))
arrivalTime = list(map(int, input("Arrival Times: ").split()))
burstTime = list(map(int, input("Burst Times: ").split()))
priority = list(map(int, input("Priorities: ").split()))

processes = sorted([Process(x+
1,arrivalTime[x],burstTime[x],priority[x]) for x in range(len(arrivalTime))],key
=lambda x:x.AT)
heap = []
completed = []
cpuTime = processes[0].AT

for p in processes:
    while heap:
        if cpuTime >= p.AT:
            break
```

```
        # for cases when cpu sits idle
        # ex if there is a gap between AT
        # of 2 consecutive processes
        if heap[0].AT>cpuTime:
            heap[0].firstExecution = heap[0].AT
        else:
            heap[0].firstExecution = cpuTime

        heap[0].CT = heap[0].firstExecution + heap[0].BT
        heap[0].calc()
        cpuTime = heap[0].CT
        completed.append(heapq.heappop(heap))

    heapq.heappush(heap, p)

# After no new process are incomming
while heap:
    if heap[0].AT>cpuTime:
        heap[0].firstExecution = heap[0].AT
    else:
        heap[0].firstExecution = cpuTime

    heap[0].CT = heap[0].firstExecution + heap[0].BT
    heap[0].calc()
    cpuTime = heap[0].CT
    completed.append(heapq.heappop(heap))

print("Process, AT, BT, Priority, CT, TAT, WT, RT")
[print(x) for x in sorted(completed,key=lambda x: x.idx)]

print("\nDeekshant Wadhwa- 01296303118")
print("\nAverage:")
print(f"CT: {sum((x.CT for x in completed))/n}")
print(f"TAT: {sum((x.TAT for x in completed))/n}")
print(f"WT: {sum((x.WT for x in completed))/n}")
print(f"RT: {sum((x.RT for x in processes))/n}")
```

Output (Non-Pre-emptive Priority Scheduling)

```
PS D:\Drive\Sem 6\OS\lab> python -u "d:\Drive\Sem 6\OS\lab\priority-non.py"
Number of processes: 6
Arrival Times: 0 1 2 3 4 5
Burst Times: 3 4 2 4 5 2
Priorities: 6 5 4 3 2 1
Process, AT, BT, Priority, CT, TAT, WT, RT
Process(1): 0, 3, 6, 3, 3, 0, 0
Process(2): 1, 4, 5, 20, 19, 15, 15
Process(3): 2, 2, 4, 16, 14, 12, 12
Process(4): 3, 4, 3, 7, 4, 0, 0
Process(5): 4, 5, 2, 14, 10, 5, 5
Process(6): 5, 2, 1, 9, 4, 2, 2

Deekshant Wadhwa- 0129633118

Average:
CT: 11.5
TAT: 9.0
WT: 5.666666666666667
RT: 5.666666666666667
PS D:\Drive\Sem 6\OS\lab>
```