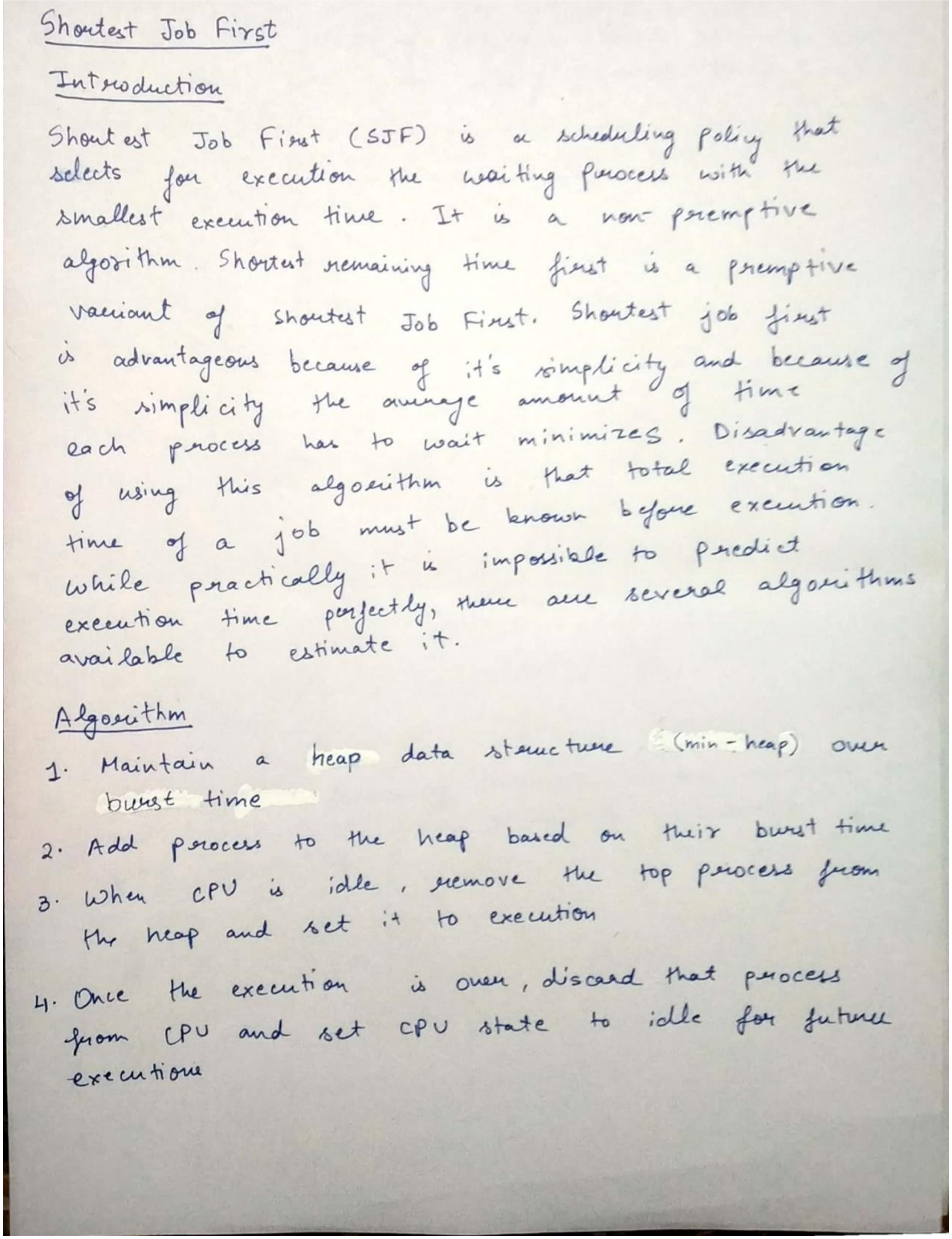
Experiment – 2



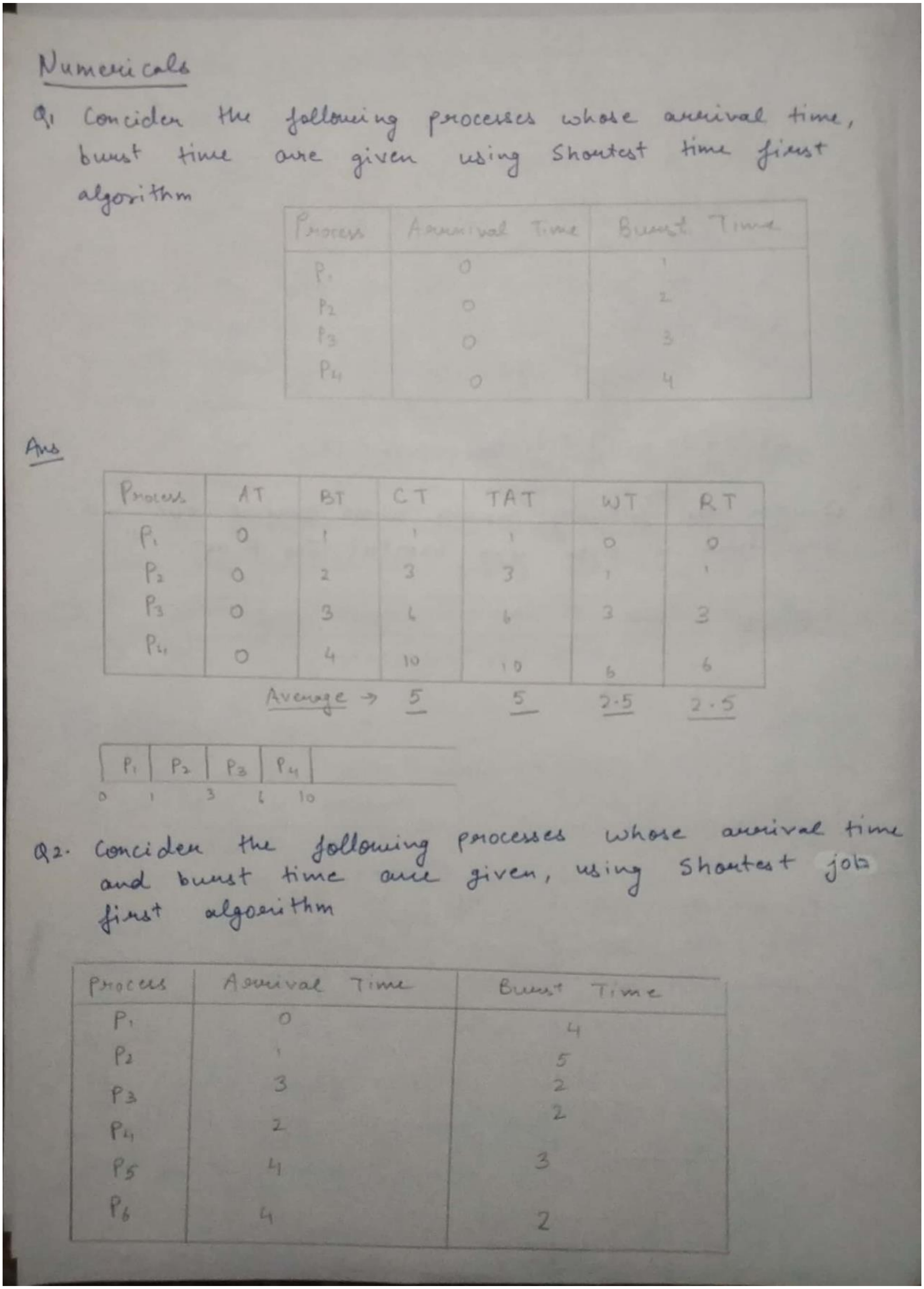
Shortest Job First

Introduction

Shortest Job First (SJF) is a scheduling policy that selects for execution the waiting process with the smallest execution time. It is a non premptive algorithm. Shortest remaining time first is a premptive variant of shortest Job First. Shortest job first is advantageous because of it's simplicity and because of it's simplicity the average amount of time each process has to wait minimizes. Disadvantage of using this algorithm is that total execution time of a job must be known before execution. while practically it is impossible to predict execution time perfectly, there are several algorithms available to estimate it.

Algorithm

1. Maintain a heap data structure (min-heap) over burst time
2. Add process to the heap based on their burst time
3. When CPU is idle, remove the top process from the heap and set it to execution
4. Once the execution is over, discard that process from CPU and set CPU state to idle for future execution



Numericals

Q1 Concider the following processes whose arrival time, burst time are given using Shortest time first algorithm

Process	Arrival Time	Burst Time
P1	0	1
P2	0	2
P3	0	3
P4	0	4

Ans

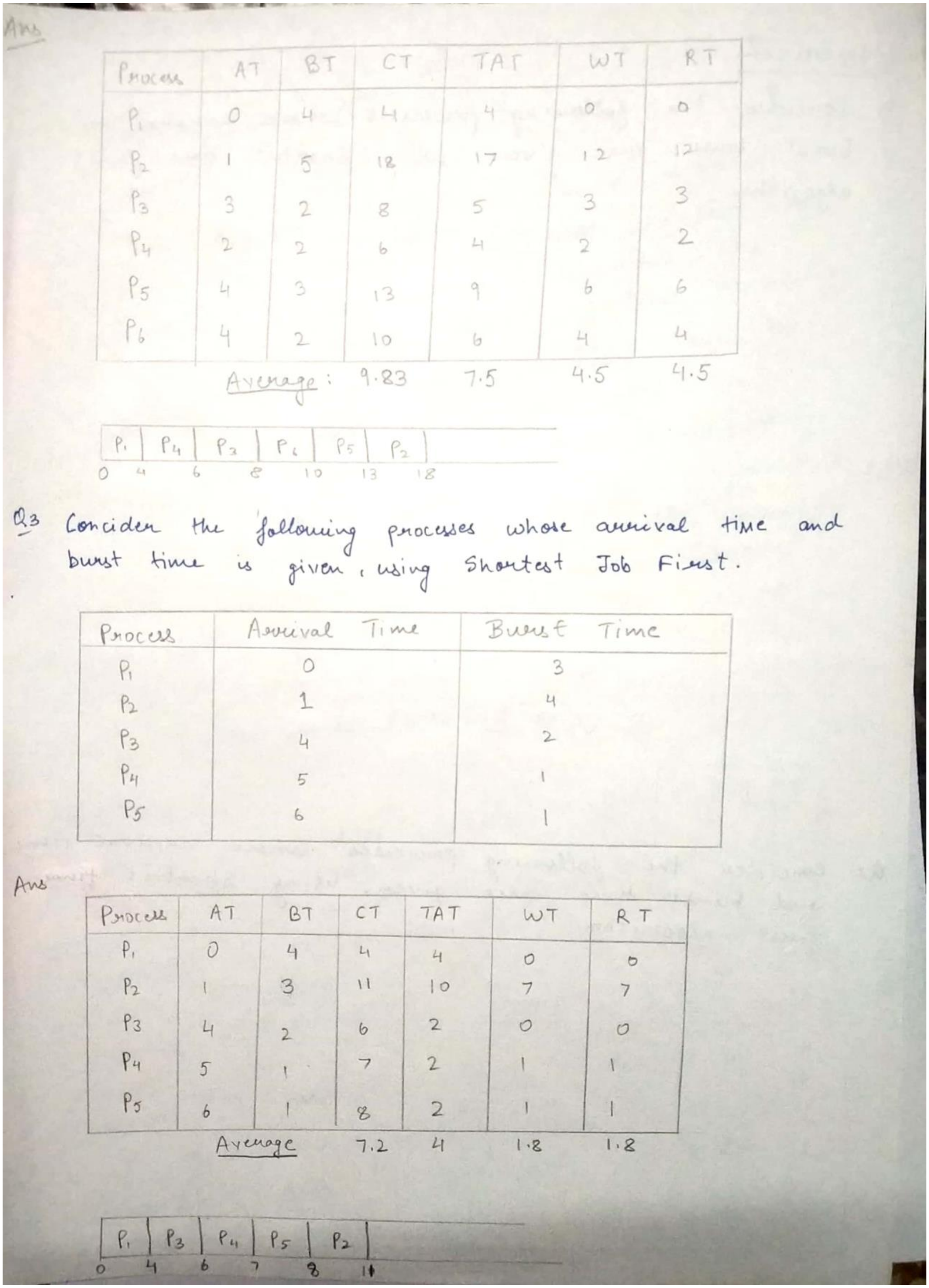
Process	AT	BT	CT	TAT	WT	RT
P1	0	1	1	1	0	0
P2	0	2	3	3	1	1
P3	0	3	6	6	3	3
P4	0	4	10	10	6	6
		Average →	5	5	2.5	2.5

P1	P2	P3	P4

0 1 3 6 10

Q2. Concider the following processes whose arrival time and burst time are given, using Shortest job first algorithm

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	3	2
P4	2	2
P5	4	3
P6	4	2



Ans

Process	AT	BT	CT	TAT	WT	RT
P₁	0	4	4	4	0	0
P₂	1	5	18	17	12	12
P₃	3	2	8	5	3	3
P₄	2	2	6	4	2	2
P₅	4	3	13	9	6	6
P₆	4	2	10	6	4	4

Average : 9.83 7.5 4.5 4.5

P₁	P₄	P₃	P₆	P₅	P₂

0 4 6 8 10 13 18

Q3 Concider the following processes whose arrival time and burst time is given, using Shortest Job First.

Process	Arrival Time	Burst Time
P₁	0	3
P₂	1	4
P₃	4	2
P₄	5	1
P₅	6	1

Ans

Process	AT	BT	CT	TAT	WT	RT
P₁	0	4	4	4	0	0
P₂	1	3	11	10	7	7
P₃	4	2	6	2	0	0
P₄	5	1	7	2	1	1
P₅	6	1	8	2	1	1

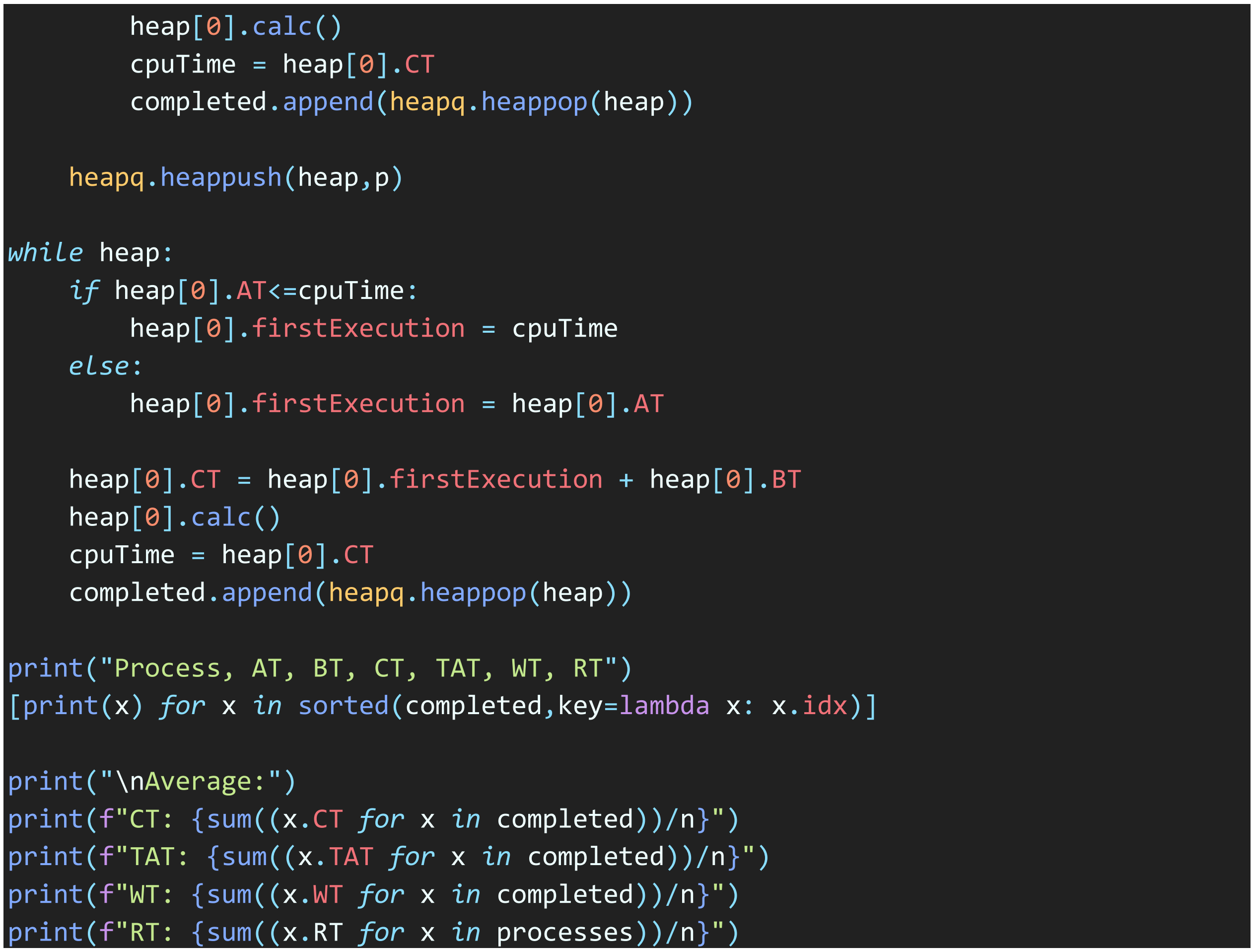
Average 7.2 4 1.8 1.8

P₁	P₃	P₄	P₅	P₂

0 4 6 7 8 11

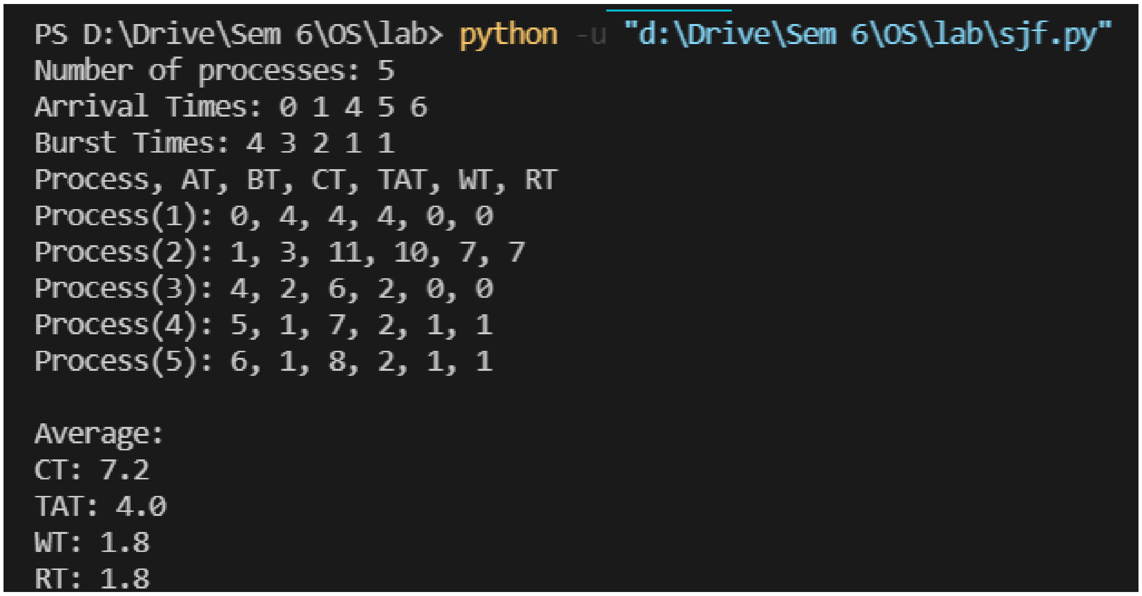
Code

```
import heapq

class Process:
    def __init__(self, idx, AT, BT,) -> None:
        self.idx = idx
        self.AT = AT
        self.BT = BT
        self.CT = None
        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):
        return self.BT<other.BT

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

processes = sorted([Process(x+1,arrivalTime[x],burstTime[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

        if heap[0].AT<=cpuTime:
            heap[0].firstExecution = cpuTime
        else:
            heap[0].firstExecution = heap[0].AT

        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)

while heap:
    if heap[0].AT<=cpuTime:
        heap[0].firstExecution = cpuTime
    else:
        heap[0].firstExecution = heap[0].AT

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

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



```
PS D:\Drive\Sem 6\OS\lab> python -u "d:\Drive\Sem 6\OS\lab\sjf.py"
Number of processes: 5
Arrival Times: 0 1 4 5 6
Burst Times: 4 3 2 1 1
Process, AT, BT, CT, TAT, WT, RT
Process(1): 0, 4, 4, 4, 0, 0
Process(2): 1, 3, 11, 10, 7, 7
Process(3): 4, 2, 6, 2, 0, 0
Process(4): 5, 1, 7, 2, 1, 1
Process(5): 6, 1, 8, 2, 1, 1

Average:
CT: 7.2
TAT: 4.0
WT: 1.8
RT: 1.8
```