**UIT2512---Operating Systems Practices Lab**

# Implementation of Priority CPU Scheduling Algorithm (Non Preemptive & Preemptive) in Python

# Name: Vasundhara.B

# Roll no: 3122 21 5002 119

# AIM:

# To implement the Priority CPU Scheduling Algorithm in Python, demonstrating the allocation of CPU time to processes based on their priority in both non-preemptive and preemptive modes.

# Description:

# The Priority CPU Scheduling Algorithm assigns CPU time to processes based on their priority values. In the non-preemptive version, once a process starts executing, it will continue until it completes, regardless of whether a higher-priority process arrives later. In contrast, the preemptive version may preempt the currently executing process if a higher-priority process becomes available.

# CODE (non\_preemptive):

class PrioritySchedulingProcess:

    def \_\_init\_\_(self, process\_name, burst\_time, waiting\_time, turnaround\_time, completion\_time, priority):

        self.process\_name = process\_name

        self.burst\_time = burst\_time

        self.waiting\_time = waiting\_time

        self.turnaround\_time = turnaround\_time

        self.completion\_time = completion\_time

        self.priority = priority

def main():

    number\_of\_process = int(input("Enter the total number of Processes: "))

    processes = []

    ASCII\_number = 65  # ASCII 'A'

    for i in range(number\_of\_process):

        process\_name = chr(ASCII\_number)

        print(f"\nEnter the details of process {process\_name}:")

        burst\_time = int(input("Enter the burst time: "))

        priority = int(input("Enter the priority: "))

        process = PrioritySchedulingProcess(process\_name, burst\_time, 0, 0, 0, priority)

        processes.append(process)

        ASCII\_number += 1

    # Sort processes based on priority (higher priority first)

    processes.sort(key=lambda x: x.priority, reverse=True)

    total\_waiting\_time = 0

    total\_turnaround\_time = 0

    processes[0].waiting\_time = 0

    processes[0].completion\_time = processes[0].burst\_time

    for i in range(1, number\_of\_process):

        processes[i].waiting\_time = 0

        for j in range(i):

            processes[i].waiting\_time += processes[j].burst\_time

        processes[i].completion\_time = processes[i - 1].completion\_time + processes[i].waiting\_time

        total\_waiting\_time += processes[i].waiting\_time

        total\_turnaround\_time += processes[i].completion\_time

    average\_waiting\_time = total\_waiting\_time / number\_of\_process

    average\_turnaround\_time = total\_turnaround\_time / number\_of\_process

    print("GANTT CHART\n")

    gc="|"

    for process in processes:

        gc+=(process.process\_name+("\_"\*process.completion\_time)+"|")

    print(gc)

    print()

    print("\n\nProcess Name\tBurst Time\tCompletion Time\tWaiting Time\tTurnaround Time")

    print("-------------------------------------------------------------------------")

    for process in processes:

        process.turnaround\_time = process.completion\_time

        print(f"{process.process\_name}\t\t{process.burst\_time}\t\t{process.completion\_time}\t\t{process.waiting\_time}\t\t{process.turnaround\_time}")

        print("-------------------------------------------------------------------------")

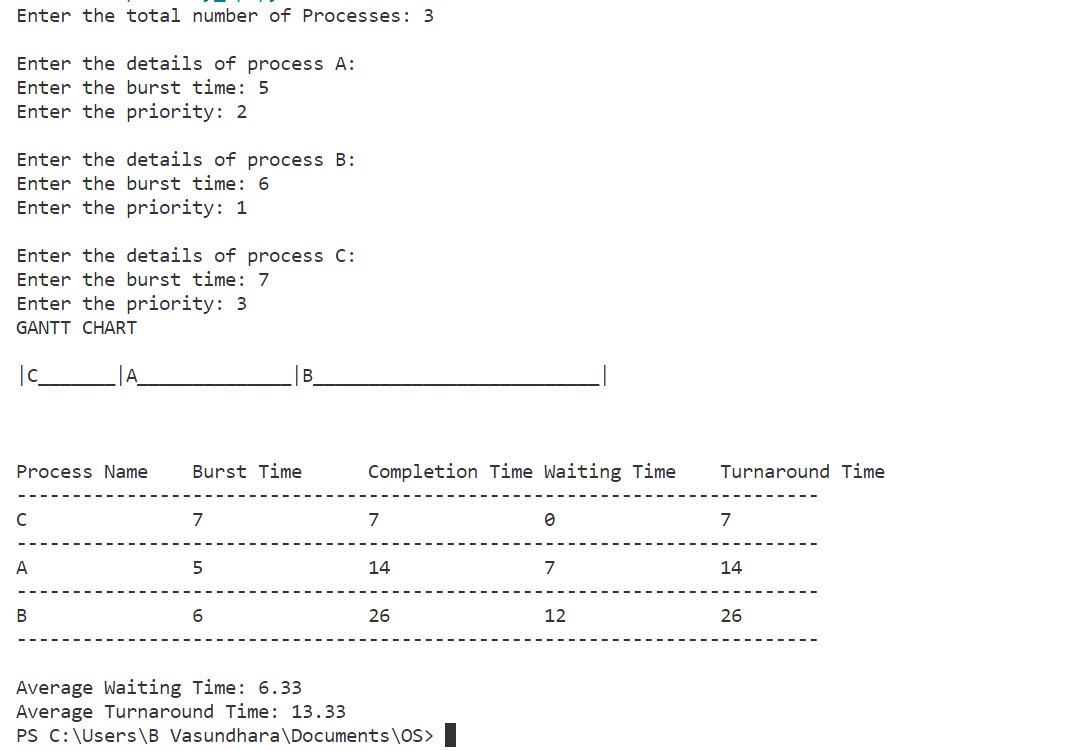
    print(f"\nAverage Waiting Time: {average\_waiting\_time:.2f}")

    print(f"Average Turnaround Time: {average\_turnaround\_time:.2f}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**OUTPUT:**



# CODE (preemptive):

class Process:

    def \_\_init\_\_(self, no, at, bt, rt, ct, wt, tat, pri, temp):

        self.no = no

        self.at = at

        self.bt = bt

        self.rt = rt

        self.ct = ct

        self.wt = wt

        self.tat = tat

        self.pri = pri

        self.temp = temp

def read(i):

    print("\nProcess No:", i)

    no = i

    at = int(input("Enter Arrival Time: "))

    bt = int(input("Enter Burst Time: "))

    rt = bt

    pri = int(input("Enter Priority: "))

    temp = pri

    return Process(no, at, bt, rt, 0, 0, 0, pri, temp)

def main():

    print("<--Highest Priority First Scheduling Algorithm (Preemptive)-->\n")

    n = int(input("Enter Number of Processes: "))

    processes = [read(i + 1) for i in range(n)]

    remaining = n

    execution\_order = []  # Track the order of execution

    gantt\_chart = "|"  # Initialize the Gantt chart

    for i in range(n - 1):

        for j in range(n - i - 1):

            if processes[j].at > processes[j + 1].at:

                processes[j], processes[j + 1] = processes[j + 1], processes[j]

    max\_val = processes[0].temp

    max\_index = 0

    for j in range(n):

        if processes[j].at <= processes[0].at:

            if processes[j].temp > max\_val:

                max\_val = processes[j].temp

                max\_index = j

    i = max\_index

    c = processes[i].ct = processes[i].at + 1

    processes[i].rt -= 1

    execution\_order.append(processes[i].no)

    gantt\_chart += str(processes[i].no) + "\_" \* (processes[i].ct - 1) + "|"

    if processes[i].rt == 0:

        processes[i].temp = float("-inf")

        remaining -= 1

    while remaining > 0:

        max\_val = processes[0].temp

        max\_index = 0

        for j in range(n):

            if processes[j].at <= c:

                if processes[j].temp > max\_val:

                    max\_val = processes[j].temp

                    max\_index = j

        i = max\_index

        processes[i].ct = c = c + 1

        processes[i].rt -= 1

        execution\_order.append(processes[i].no)

        gantt\_chart += str(processes[i].no) + "\_"  + "|"

        if processes[i].rt == 0:

            processes[i].temp = float("-inf")

            remaining -= 1

    print("\nProcessNo\tAT\tBT\tPri\tCT\tTAT\tWT")

    avgtat = 0

    avgwt = 0

    for i in range(n):

        processes[i].tat = processes[i].ct - processes[i].at

        avgtat += processes[i].tat

        processes[i].wt = processes[i].tat - processes[i].bt

        avgwt += processes[i].wt

        print(f"P{processes[i].no}\t\t{processes[i].at}\t{processes[i].bt}\t{processes[i].pri}\t"

              f"{processes[i].ct}\t{processes[i].tat}\t{processes[i].wt}")

    avgtat /= n

    avgwt /= n

    print(f"\nAverage TurnAroundTime={avgtat}\nAverage WaitingTime={avgwt}")

    # Print the order of execution in the Gantt chart format

    print("\nGANTT CHART\n")

    print(gantt\_chart)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**OUTPUT:**

