LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

NAME: AKHIL

Reg No: 11703357

E-mail: akhildhiman141@gmail.com

Git Hub: https://github.com/akhildhiman7/Student-Teacher-Problem.git

INDEX

<u>CODE:</u> (Python Implementation)

```
class StudentQueue:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item, AT, BT):
        self.lst = []
        self.lst.append(item)
        self.lst.append(AT)
        self.lst.append(BT)
        self.items.insert(0,self.lst)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
    def head(self):
        return self.items[-1][1]
    def burst_time(self):
        return self.items[-1][2]
    def id_no(self):
        return self.items[-1][0]


class TeacherQueue:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item, AT, BT):
        self.lst = []
        self.lst.append(item)
        self.lst.append(AT)
        self.lst.append(BT)
        self.items.insert(0,self.lst)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
    def head(self):
        return self.items[-1][1]
    def burst_time(self):
        return self.items[-1][2]
    def id_no(self):
        return self.items[-1][0]


SQ = StudentQueue()
TQ =  TeacherQueue()

print("0. Automatic")
print("1. User Provided")
#ip_var = int(input("Mode of input: "))
ip_var = 1
if ip_var == 0:
    print("Automatic Mode Selected")
    print()
    Tat1, Tbt1 = 1, 2
    Tat2, Tbt2 = 2, 2
    Tat3, Tbt3 = 3, 2
    Tat4, Tbt4 = 14, 3
    TQ.enqueue(1, Tat1, Tbt1)
    TQ.enqueue(2, Tat2, Tbt2)
    TQ.enqueue(3, Tat3, Tbt3)
    TQ.enqueue(4, Tat4, Tbt4)
    Sat1, Sbt1 = 1, 2
    Sat2, Sbt2 = 2, 2
    SQ.enqueue(1, Sat1, Sbt1)
    SQ.enqueue(2, Sat2, Sbt2)
    teachers = TQ.size()
    students = SQ.size()
else:
    print("User Mode Selected")
    print()
    teachers = int(input("Enter the number of Teachers in the queue: ", ))
    t_data = []
    last_time = 0
    if teachers != 0:
        for i in range(teachers):
            print("Enter Arrival Time for Teacher ",i+1, end = "")
            AT = int(input())
            if (AT < last_time):
                while True:
                    print("AT can't be less than previous arrival time")
                    print("Enter Arrival Time for Teacher ",i+1, end = "")
                    AT = int(input())
                    if last_time <= AT:
                        break
            last_time = AT
            print("Enter Burst Time for Teacher ",i+1, end = "")
            BT = int(input())
            temp_list = []
            temp_list.append(AT)
            temp_list.append(BT)
            t_data.append(temp_list)

    students = int(input("Enter the nubers of Students in the queue: ", ))
    s_data = []
    last_time = 0
    if students != 0:
        for i in range(students):
            print("Enter Arrival Time for Student ",i+1, end = "")
            AT = int(input())
            if (AT < last_time):
                while True:
                    print("AT can't be less then previous arrival time")
                    print("Enter Arrival Time for Student ",i+1, end = "")
                    AT = int(input())
                    if last_time <= AT:
                        break
            last_time = AT
            print("Enter Burst Time for Student ",i+1, end = "")
            BT = int(input())
            temp_list = []
            temp_list.append(AT)
            temp_list.append(BT)
            s_data.append(temp_list)

    for i in range (teachers):
        TQ.enqueue(i+1, t_data[i][0], t_data[i][1])
    for i in range(students):
        SQ.enqueue(i+1, s_data[i][0], s_data[i][1])

maxlen =teachers+students
```

```
student_priority = 0
curr_time = min(SQ.head(), TQ.head())
t = teachers
s = students
j = 0
k = 0

'''
print("No of teachers: ", t)
print("No of students: ", s)
print("AT of first student is ", SQ.items[s-1][1])
print("AT of first teacher is ", TQ.items[t-1][1])

'''

for i in range(maxlen):
    if (SQ.isEmpty()):
        for i in range (teachers):
            if TQ.isEmpty() == False:
                print("Teacher ",TQ.id_no()," issued book")
                curr_time += TQ.burst_time()
                TQ.dequeue()
                break
    elif TQ.isEmpty():
        for i in range (students):
            if SQ.isEmpty() == False:
                print("Student ",SQ.id_no()," issued book")
                curr_time += SQ.burst_time()
                SQ.dequeue()
                break
    elif student_priority == 2:
            print("Student ",SQ.id_no()," issued book")
            curr_time += SQ.burst_time()
            student_priority = 0
            SQ.dequeue()
    else:
        tchr = TQ.head()
        stdnt = SQ.head()
        if tchr <= stdnt:
            if curr_time >= stdnt:
                student_priority += 1
            print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
            curr_time += TQ.burst_time()
            TQ.dequeue()
        elif tchr > stdnt:
            if curr_time >= tchr:
                student_priority += 1
                curr_time += TQ.burst_time()
                print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
                TQ.dequeue()
            else:
                curr_time += SQ.burst_time()
                print("Student ", SQ.id_no()," issued book")
                student_priority = 0
                SQ.dequeue()
```

Problem in terms of Operating System Concepts:

There are two queues for two different type of processes which are represented by Teachers and Students and we may call the queues be TeacherQueue and StudentQueue which can enter in a library for issuing of books. But the issuer can handle only one request at a time either be it Student or Teacher. If a Student is already in the line and issuing a book than if a teacher comes than that Teacher will be the second person to get the book issued. But if a Teacher is already in the queue and a student and a teacher comes together in their queues. The teacher will be the one who will be given the priority to get the book issued. A student may wait if a Teacher is already in the queue. This situation may lead to aging of Student so the task was to minimize the waiting time of Student.

Algorithm:

```
SET maxlen = len(Student Queue) + len(Teacher Queue)
for i in range(maxlen): #Iterate the loop in the range of maxlen
    if (SQ.isEmpty()): # Check if Student Queue is empty
        for i in range (teachers):
            if TQ.isEmpty() == False: #Check if Teacher Queue is not empty
                print("Teacher ",TQ.id_no()," issued book")
                curr_time += TQ.burst_time()
                TQ.dequeue()
                break
    elif TQ.isEmpty():
        for i in range (students):
            if SQ.isEmpty() == False:
                print("Student ",SQ.id_no()," issued book")
                curr_time += SQ.burst_time()
                SQ.dequeue()
                break
    elif student_priority == 2:
            print("Student ",SQ.id_no()," issued book")
            curr_time += SQ.burst_time()
            student_priority = 0
            SQ.dequeue()
    else:
        tchr = TQ.head()
        stdnt = SQ.head()
        if tchr <= stdnt:
            if curr_time >= stdnt:
                student_priority += 1
            print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
            curr_time += TQ.burst_time()
            TQ.dequeue()
        elif tchr > stdnt:
            if curr_time >= tchr:
                student_priority += 1
                curr_time += TQ.burst_time()
                print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
                TQ.dequeue()
            else:
                curr_time += SQ.burst_time()
                print("Student ", SQ.id_no()," issued book")
                student_priority = 0
                SQ.dequeue()
```

Complexity:

```
for i in range(maxlen): // O(N)
    if (SQ.isEmpty()): // O(1)
        for i in range (teachers): // O(N)
            if TQ.isEmpty() == False: // O(1)
                print("Teacher ",TQ.id_no()," issued book") // O(1)
                curr_time += TQ.burst_time() // O(1)
                TQ.dequeue()// O(1)
                break
    elif TQ.isEmpty():// O(1)
        for i in range (students): // O(N)
            if SQ.isEmpty() == False: // O(1)
                print("Student ",SQ.id_no()," issued book") // O(1)
                curr_time += SQ.burst_time() // O(1)
                SQ.dequeue() // O(1)
                break // O(1)
    elif student_priority == 2: // O(1)
            print("Student ",SQ.id_no()," issued book") // O(1)
            curr_time += SQ.burst_time()// O(1)
            student_priority = 0 // O(1)
            SQ.dequeue() // O(1)
    else: // O(1)
        tchr = TQ.head() // O(1)
        stdnt = SQ.head() // O(1)
        if tchr <= stdnt: // O(1)
            if curr_time >= stdnt: // O(1)
                student_priority += 1 // O(1)
            print("Teacher ", TQ.id_no(),"issued book. Student Priority:",student_priority) // O(1)
            curr_time += TQ.burst_time()// O(1)
            TQ.dequeue()// O(1)
        elif tchr > stdnt: // O(1)
            if curr_time >= tchr: // O(1)
                student_priority += 1 // O(1)
                curr_time += TQ.burst_time()// O(1)
                print("Teacher",TQ.id_no(),"issued book.Student Priority:",student_priority)// O(1)
                TQ.dequeue()// O(1)
            else: // O(1)
                curr_time += SQ.burst_time() // O(1)
                print("Student ", SQ.id_no()," issued book") // O(1)
                student_priority = 0 // O(1)
                SQ.dequeue() // O(1)
```

Total Complexity: O(N*N)

Constraint Limit:

For Adding items into a python list on a regular 32bit system, this is 536,870,912 elements. i.e. for appending items into the list, the maximum of adding Teacher/Student in the queue is 536,870,912.

Conditions:

If a Student and a teacher arrives at same after a Teacher than the student can only wait for one more Teacher only and after that Student will issue the book.

The Arrival Time for a Student/Teacher can't be less than the previous Arrival Time.

Test Cases:

S.No	Condition	Expected Result	Actual Result
1.	If 0 Teacher and N students.	N Students issues book one by one.	N Students issues book one by one.
2.	If N Teachers and 0 students.	N Teachers issues book one by one.	N Teachers issues book one by one.
3.	If N Teachers arrive at the same time and 0 students.	The teacher which is in front of the queue will be served first.	The teacher which is in front of the queue will be served first.
4.	If 0 Teachers arrive at the same time and N students.	The teacher which is in front of the queue will be served first	The teacher which is in front of the queue will be served first.
5.	If Arrival Time of any Student/Teacher is less than the previous Student/Teacher Arrival Time.	It should re prompt the user.	It should re prompt the user.
6.	If 3 Teachers arrive at the counter than 1 student arrives simultaneously with 3rd Teacher at the counter and after that 2 more teacher arrives.	The student is served after the 4th Teacher.	The student is served after 4th Teacher.
7.	If 3 Student arrive first and a Teacher arrives at the same time with 2nd Student at the counter and after that 4 more Student arrives.	The teacher is served after the 1st student and the issuing goes on until the Student Queue is empty.	The teacher is served after the 1st student and the issuing goes on until the Student Queue is empty.

<table>
<tr><td rowspan="2">8.</td><td colspan="6">Total Teacher are 3
Total Student are 2
Taking BT[T/S] = 2</td><td rowspan="2">Teacher 1 issued book.
Teacher 2 issued book.
Student 1 issued book
Teacher 3 issued book.
Student 2 issued book</td><td rowspan="2">Teacher 1 issued book.
Teacher 2 issued book.
Student 1 issued book
Teacher 3 issued book.
Student 2 issued book</td></tr>
<tr><td colspan="3">AT[T1] = 0
AT[T2] = 1
AT[T3] = 2</td><td colspan="3">AT[S1] = 0
AT[S2] = 1</td></tr>
<tr><td rowspan="2">9.</td><td colspan="6">Total Teacher are 4
Total Student are 4</td><td rowspan="2">Student 1 issued book
Teacher 1 issued book.
Teacher 2 issued book.
Student 2 issued book
Teacher 3 issued book.
Student 3 issued book
Teacher 4 issued book.
Student 4 issued book</td><td rowspan="2">Student 1 issued book
Teacher 1 issued book.
Teacher 2 issued book.
Student 2 issued book
Teacher 3 issued book.
Student 3 issued book
Teacher 4 issued book.
Student 4 issued book</td></tr>
<tr><td>Tno
1.
2.
3.
4.</td><td>AT
1
3
4
10</td><td>BT
2
1
2
3</td><td>Sno
1.
2.
3.
4.</td><td>AT
0
1
4
7</td><td>BT
2
1
2
1</td></tr>
</table>

GitHub Link: https://github.com/akhildhiman7/Student-Teacher-Problem.git