

Mohammed Nizamuddin

2503B05144

Lab Task: Round Robin CPU Scheduling (Time Quantum = 3 ms)

A system receives the following processes NOT in arrival order:

Proces s	Arrival Time (AT)	Burst Time (BT)
P1	4	7
P2	0	5
P3	6	3
P4	2	9
P5	1	4

Student Tasks

1. Do NOT reorder by arrival time.
Simulate exactly as given.
2. Apply Round Robin Scheduling with Time Quantum = 3 ms.
3. Draw the complete Gantt Chart.
4. Compute for each process:
 - o Completion Time (CT)
 - o Turnaround Time (TAT = CT - AT)
 - o Waiting Time (WT = TAT - BT)
5. Compute:
 - o Average TAT
 - o Average WT
6. Write a C program that performs RR scheduling.
7. Compare your manual result with the program output.

CODE:

```
#include <bits/stdc++.h>
using namespace std;

struct GSeg { int pid; int start; int end; };

int main() {
    int n = 5;
    vector<int> AT = {4, 0, 6, 2, 1};
    vector<int> BT = {7, 5, 3, 9, 4};
    int quantum = 3;

    vector<int> rem = BT;
    vector<int> CT(n, 0);
    vector<int> added(n, 0);

    deque<int> q;
    int time = 0;
    int completed = 0;

    for (int i = 0; i < n; ++i) {
        if (AT[i] <= time && !added[i]) {
            q.push_back(i);
            added[i] = 1;
        }
    }

    if (q.empty()) {
        int nextAt = INT_MAX;
        for (int i = 0; i < n; ++i) if (!added[i]) nextAt = min(nextAt, AT[i]);
        time = nextAt;
        for (int i = 0; i < n; ++i) {
            if (AT[i] <= time && !added[i]) {
                q.push_back(i);
                added[i] = 1;
            }
        }
    }

    vector<GSeg> gantt;

    while (completed < n) {
        if (q.empty()) {
            int nextAt = INT_MAX;
            for (int i = 0; i < n; ++i) if (!added[i]) nextAt = min(nextAt, AT[i]);
            time = nextAt;
        }
```

```

        for (int i = 0; i < n; ++i) {
            if (AT[i] <= time && !added[i]) {
                q.push_back(i);
                added[i] = 1;
            }
        }
        continue;
    }

    int idx = q.front(); q.pop_front();
    int exec = min(rem[idx], quantum);
    int start = time;
    int end = time + exec;

    rem[idx] -= exec;
    time = end;

    gantt.push_back({idx+1, start, end});

    for (int i = 0; i < n; ++i) {
        if (AT[i] <= time && !added[i]) {
            q.push_back(i);
            added[i] = 1;
        }
    }

    if (rem[idx] == 0) {
        CT[idx] = time;
        completed++;
    } else {
        q.push_back(idx);
    }
}

cout << "Gantt chart segments:\n";
for (auto &s : gantt) cout << "| P" << s.pid << " ";
cout << "|\n";
for (auto &s : gantt) cout << s.start << " ";
cout << gantt.back().end << "\n\n";

cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
    << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\n";

double sumTAT = 0.0, sumWT = 0.0;
for (int i = 0; i < n; ++i) {
    int tat = CT[i] - AT[i];
    int wt = tat - BT[i];
    sumTAT += tat;
}

```

```

        sumWT += wt;
        cout << "P" << (i+1) << setw(5) << ""
            << setw(6) << AT[i] << setw(6) << BT[i]
            << setw(6) << CT[i] << setw(6) << tat << setw(6) << wt << "\n";
    }

    cout << fixed << setprecision(2);
    cout << "\nAverage TAT = " << (sumTAT / n) << " ms\n";
    cout << "Average WT = " << (sumWT / n) << " ms\n";

    return 0;
}

```

Screenshot of an online judge interface showing the execution of the provided C++ code. The code implements a Round-Robin scheduling algorithm for five processes (P1-P5) with a quantum of 3 units. The Gantt chart shows the execution segments and the resulting Turnaround Time (TAT) and Waiting Time (WT) for each process.

Gantt Chart Segments:

P2	P4	P5	P2	P1	P3	P4	P5	P1	P4	P1	
0	3	6	9	11	14	17	20	21	24	27	28

Process Data:

Proc	AT	BT	CT	TAT	WT
P1	4	7	28	24	17
P2	0	5	11	11	6
P3	6	3	17	11	8
P4	2	9	27	25	16
P5	1	4	21	20	16

Average Metrics:

- Average TAT = 18.20 ms
- Average WT = 12.60 ms

Screenshot of an online judge interface showing the execution of an alternative version of the C++ code. This version uses a queue (q) instead of a stack for scheduling. The Gantt chart and process data are identical to the first execution, indicating equivalent performance.

Gantt Chart Segments:

P2	P4	P5	P2	P1	P3	P4	P5	P1	P4	P1	
0	3	6	9	11	14	17	20	21	24	27	28

Process Data:

Proc	AT	BT	CT	TAT	WT
P1	4	7	28	24	17
P2	0	5	11	11	6
P3	6	3	17	11	8
P4	2	9	27	25	16
P5	1	4	21	20	16

Average Metrics:

- Average TAT = 18.20 ms
- Average WT = 12.60 ms

leetcode.com/playground/new/empty

Explore Problems Contest Discuss Interview Store

Run Code Untitled

```
50 }
51 }
52 continue;
53 }
54
55 int idx = q.front(); q.pop_front();
56 int exec = min(rem[idx], quantum);
57 int start = time;
58 int end = time + exec;
59
60 rem[idx] -= exec;
61 time = end;
62
63 gantt.push_back({idx+1, start, end});
64
65 for (int i = 0; i < n; ++i) {
66     if (AT[i] <= time && !added[i]) {
67         q.push_back(i);
68         added[i] = 1;
69     }
70 }
71
72 if (rem[idx] == 0) {
73     CT[idx] = time;
74     completed++;
75 }
76
77 cout << "Gantt chart segments:\n";
78 for (auto &s : gantt) cout << "| P" << s.pid << " ";
79 cout << "|\\n";
80 for (auto &s : gantt) cout << s.start << " ";
81 cout << gantt.back().end << "\\n\\n";
82
83 cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
84 << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\\n";
85
86 double sumTAT = 0.0, sumWT = 0.0;
87 for (int i = 0; i < n; ++i) {
88     int tat = CT[i] - AT[i];
89     int wt = tat - BT[i];
90     sumTAT += tat;
91     sumWT += wt;
92     cout << "P" << (i+1) << setw(5) << ""
93     << setw(6) << AT[i] << setw(6) << BT[i]
94     << endl;
95 }
96
97 cout << fixed << setprecision(2);
98 cout << "\\nAverage TAT = " << (sumTAT / n) << " ms\\n";
99 cout << "Average WT = " << (sumWT / n) << " ms\\n";
100
101 return 0;
102 }
```

Output: Finished

Finished in 0 ms

Gantt chart segments:

P2 P4 P5 P2 P1 P3 P4 P5 P1 P4 P1
0 3 6 9 11 14 17 20 21 24 27 28

Proc AT BT CT TAT WT

P1 4 7 28 24 17
P2 0 5 11 11 6
P3 6 3 17 11 8
P4 2 9 27 25 16
P5 1 4 21 20 16

Average TAT = 18.20 ms
Average WT = 12.60 ms

leetcode.com/playground/new/empty

Explore Problems Contest Discuss Interview Store

Run Code Untitled

```
73 }
74 }
75 completed++;
76 } else {
77     q.push_back(idx);
78 }
79
80 cout << "Gantt chart segments:\\n";
81 for (auto &s : gantt) cout << "| P" << s.pid << " ";
82 cout << "|\\n";
83 for (auto &s : gantt) cout << s.start << " ";
84 cout << gantt.back().end << "\\n\\n";
85
86 cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
87 << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\\n";
88
89 double sumTAT = 0.0, sumWT = 0.0;
90 for (int i = 0; i < n; ++i) {
91     int tat = CT[i] - AT[i];
92     int wt = tat - BT[i];
93     sumTAT += tat;
94     sumWT += wt;
95     cout << "P" << (i+1) << setw(5) << ""
96     << setw(6) << AT[i] << setw(6) << BT[i]
97     << endl;
98 }
99
100 cout << fixed << setprecision(2);
101 cout << "\\nAverage TAT = " << (sumTAT / n) << " ms\\n";
102 cout << "Average WT = " << (sumWT / n) << " ms\\n";
103
104 return 0;
105 }
```

Output: Finished

Finished in 0 ms

Gantt chart segments:

P2 P4 P5 P2 P1 P3 P4 P5 P1 P4 P1
0 3 6 9 11 14 17 20 21 24 27 28

Proc AT BT CT TAT WT

P1 4 7 28 24 17
P2 0 5 11 11 6
P3 6 3 17 11 8
P4 2 9 27 25 16
P5 1 4 21 20 16

Average TAT = 18.20 ms
Average WT = 12.60 ms

leetcode.com/playground/new/empty

Explore Problems Contest Discuss Interview Store

Run Code Untitled

```
82 cout << "|\\n";
83 for (auto &s : gantt) cout << s.start << " ";
84 cout << gantt.back().end << "\\n\\n";
85
86 cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
87 << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\\n";
88
89 double sumTAT = 0.0, sumWT = 0.0;
90 for (int i = 0; i < n; ++i) {
91     int tat = CT[i] - AT[i];
92     int wt = tat - BT[i];
93     sumTAT += tat;
94     sumWT += wt;
95     cout << "P" << (i+1) << setw(5) << ""
96     << setw(6) << AT[i] << setw(6) << BT[i]
97     << setw(6) << CT[i] << setw(6) << tat << setw(6) << wt << "\\n";
98 }
99
100 cout << fixed << setprecision(2);
101 cout << "\\nAverage TAT = " << (sumTAT / n) << " ms\\n";
102 cout << "Average WT = " << (sumWT / n) << " ms\\n";
103
104 return 0;
105 }
```

Output: Finished

Finished in 0 ms

Gantt chart segments:

P2 P4 P5 P2 P1 P3 P4 P5 P1 P4 P1
0 3 6 9 11 14 17 20 21 24 27 28

Proc AT BT CT TAT WT

P1 4 7 28 24 17
P2 0 5 11 11 6
P3 6 3 17 11 8
P4 2 9 27 25 16
P5 1 4 21 20 16

Average TAT = 18.20 ms
Average WT = 12.60 ms