

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Description:

Scheduling is a major part of any operating system and the algorithm to schedule processes can vary. The shortest job first is a way to let the process with the minimum burst time execute first from the pool of processes available at any given time which are ready for execution. With this algorithm, small processes don't have to wait for long processes that came before them for execution. In this assignment, a modified non-preemptive shortest job first algorithm is implemented. The modification is to give waiting processes a priority based on how much time they have spent waiting and their burst time. This particular modification helps us overcome a problem called starvation which may occur when processes with long burst times may never be executed or get executed at the last after waiting for so long.

Algorithm:

1. Calculate priority using the formula $p = 1 + (waiting_{time}/Burst_{time})$ for all processes.
2. Select the process with maximum priority and minimum arrival time. If more than one process has equal priorities select the one with minimum burst time and minimum arrival time then if more than two processes have equal burst times then select the one with minimum process number.
3. After executing the selected process in step 2, repeat the steps using the processes excluding the processes which are already executed until all the process are executed.

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Complexity:

```
while (1){                                     this loop: n+1
    found = 0;
    for(i=0;i<n;i++){                         nested loop: n
        if(process[i][2] == 0){
            priority[i] = -100000000.0;
            if(found != 2) {
                found = 1;
            }
            if(global_time - process[i][1] >= 0){
                found = 2;
                process[i][3] = global_time - process[i][1];
                // priority[i] = ((float)process[i][3]/(float)process[i][0]) - (float)(process[i][0]);
                priority[i] = ((float)process[i][3]/(float)process[i][0]);
                if(priority[i] == priority[loc]) {
                    if(i != loc){
                        temp1 = priority[loc];
                        priority[i] = -1 * (float)(process[i][0]);
                        priority[loc] = -1 * (float)(process[loc][0]);
                    }
                }
                if(priority[i] == temp1){
                    priority[i] = -1 * (float)(process[i][0]);
                }
                if(priority[i] > priority[loc]){
```

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

```
        loc = i;
    }
}
}
}
if(found == 1){
    global_time -= process[loc][0];
    if(global_time == process[0][1]){
        global_time = process[1][1];
    } else {
        global_time = process[0][1];
    }
    for (i=0;i<n;i++){
        if(process[i][2] == 0){
            if(process[i][1] < global_time){
                global_time = process[i][1];
            }
        }
    }
} else if(found == 0){
    break;
} else if(found == 2) {
    printf("p%d\t%10d\t%10d\t%10d\t%10d\n", loc + 1,
process[loc][0],process[loc][1],process[loc][3],process[loc][0]+process[loc][3]);
    global_time += process[loc][0];
    process[loc][2] = 1;
    priority[loc] = -100000000.0;
}
```

nested loop: n

Prateek Rathod

11814621

emailawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

```
}
```

Overall Complexity: $(n+1)(n+n) = 2n^2+2n$

$O(n) = n^2$

$W(n) = n^2$

Constraints:

This program doesn't have many constraints some of the possible constraints are:

1. The arrival time and burst time should not be negative.
2. The burst time and arrival time should be scaled down in case they are really big numbers which in the ideal case will be false.

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Test Cases:

Test Case 1:

3

5 2

4 2

2 2

An easy one to test the process with minimum non zero time can also be chosen.

Output:

Pord	bt	at	wt	tat
P3	2	2	0	2
P2	4	2	2	6
P1	5	2	6	11

Average Waiting Time: 2.667

Average Turnaround Time: 6.333

Test Case 2:

5

5 0

3 1

2 1

2 4

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

8 0

To test the algorithm doesn't use burst time instead of the priority value. The output of this test case confirms that the algorithm prioritizes an older process that has a higher burst time than a newer available process.

Output:

Pord	bt	at	wt	tat
P1	5	0	0	5
P3	2	1	4	6
P2	3	1	6	9
P4	2	4	6	8
P5	8	0	12	20

Average Waiting Time: 5.600

Average Turnaround Time: 9.600

Test Case 3:

1

5 1

To check if the program doesn't fail with single process.

Output:

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Pord	bt	at	wt	tat
P1	5	1	0	5

Average Waiting Time: 0.000

Average Turnaround Time: 5.000

Test Case 4:

3

5 0

5 0

5 0

To check the logic to preserve the order in case of collision.

Output:

Pord	bt	at	wt	tat
P1	5	0	0	5
P2	5	0	5	10
P3	5	0	10	15

Average Waiting Time: 5.333

Average Turnaround Time: 10.000

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Test Case 5:

4

599 234

21435 5435

755465 2322

34321 233

Big numbers.

Output:

Pord	bt	at	wt	tat
P4	34321	233	0	34321
P1	599	234	34320	34919
P2	21435	5435	29718	51153
P3	755465	2322	54266	809731

Average Waiting Time: 29576.000

Average Turnaround Time: 232531.000

Github Link:

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

This repository has a total of 9 commits. Test cases can be saved in a file called test.txt and a script can then compile the code and store all the formatted output in a file.

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Complete Program:

```
#include<stdio.h>
#include<stdlib.h>

int main(){
    // burst, arrival, flag, waiting
    int num_test, mode;
    printf("Enter number of batches: ");
    scanf("%d", &num_test);
    for(int j=0;j<num_test;j++)
    {system("clear");
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int process[n][4], global_time, loc=0, i, found=0, temp1=0, temp2=0;
    float priority[n];
    printf("Enter burst time and arrival time for each process (in same order). Format: int int\n");
    for(i=0; i<n;i++){
        printf("process %d: ", i+1);
        here:
        scanf("%d %d", &process[i][0], &process[i][1]);
        process[i][2] = 0;
        if(process[i][0] < 0){
            printf("\nnegative values not allowed\n");
            goto here;
        }
    }
```

Prateek Rathod

11814621

emailawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

```
if (process[i][1] < 0) {
    printf("\nnegative values not allowed\n");
    goto here;
}
if (i == 0) {
    global_time = process[i][1];
} else
{
    if (process[i][1] < global_time) {
        global_time = process[i][1];
    }
}

}

system("clear");
printf("\nTest Case %d\n\n%-10s\t%-10s\t%-10s\t%-10s\t%-10s\n",
j+1, "pord", "bt", "at", "wt", "tat");
while (1) {
    found = 0;
    for (i = 0; i < n; i++) {
        if (process[i][2] == 0) {
            priority[i] = -100000000.0;
            if (found != 2) {
                found = 1;
            }
            if (global_time - process[i][1] >= 0) {
                found = 2;
            }
        }
    }
}
```

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

```
    process[i][3] = global_time - process[i][1];
    // priority[i] = ((float)process[i][3]/(float)process[i][0]) - (float)(process[i][0]);
    priority[i] = ((float)process[i][3]/(float)process[i][0]);
    if (priority[i] == priority[loc]) {
        if (i != loc) {
            temp1 = priority[loc];
            priority[i] = -1 * (float)(process[i][0]);
            priority[loc] = -1 * (float)(process[loc][0]);
        }
    }
    if (priority[i] == temp1) {
        priority[i] = -1 * (float)(process[i][0]);
    }
    if (priority[i] > priority[loc]) {
        loc = i;
    }
}

if (found == 1) {
    global_time -= process[loc][0];
    if (global_time == process[0][1]) {
        global_time = process[1][1];
    } else {
        global_time = process[0][1];
    }
}
for (i=0; i<n; i++) {
    if (process[i][2] == 0) {
```

Prateek Rathod

11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

```
        if(process[i][1] < global_time){
            global_time = process[i][1];
        }
    }
}
} else if(found == 0){
    break;
} else if(found == 2) {
    printf("p%d\t%d\t%d\t%d\t%d\n", loc + 1,
process[loc][0],process[loc][1],process[loc][3],process[loc][0]+process[loc][3]);
    global_time += process[loc][0];
    process[loc][2] = 1;
    priority[loc] = -100000000.0;
}

}

for (i=0;i<n;i++){
    temp1 += process[i][3];
    temp2 += process[i][0]+process[i][3];
}

printf("\nAverage Waiting Time: %.3f\nAverage Turnaround Time: %.3f\n", (float)temp1/(float)n,
(float)temp2/(float)n);

printf("\n-----\n\n");
}

return 0;
}
```

Prateek Rathod

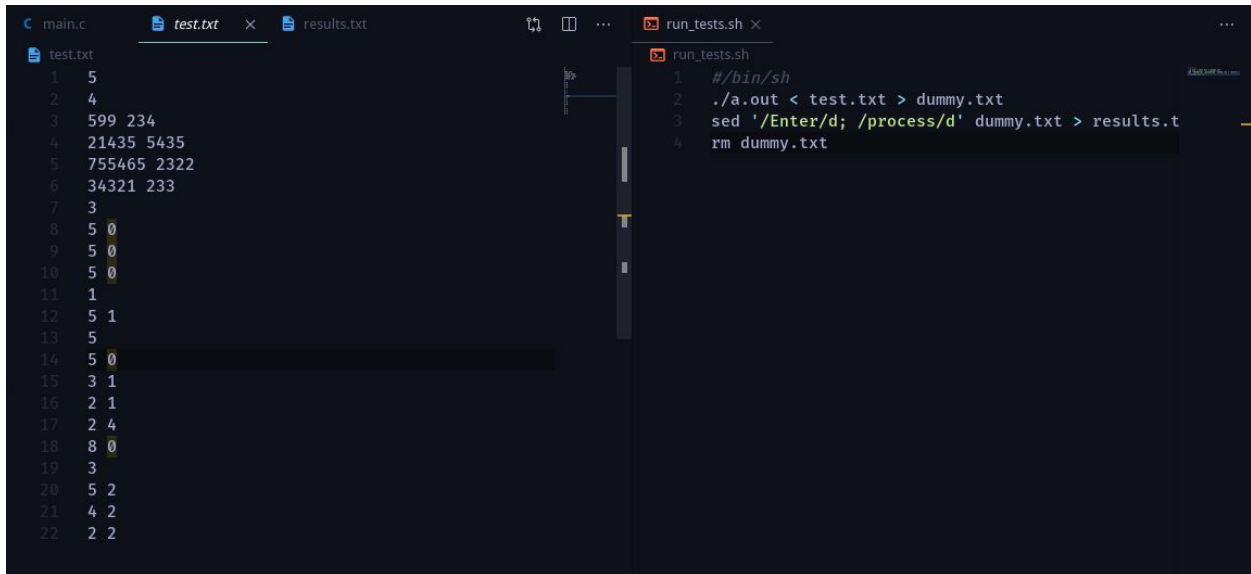
11814621

emaillawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Tests.txt and Bash Script



The image shows a code editor with two files open: `test.txt` and `run_tests.sh`. The `test.txt` file contains 22 lines of test data, including numbers and pairs of numbers. The `run_tests.sh` file contains a bash script that reads from `test.txt`, processes the data, and writes the results to `results.txt`.

```
main.c test.txt x results.txt run_tests.sh x
test.txt
1 5
2 4
3 599 234
4 21435 5435
5 755465 2322
6 34321 233
7 3
8 5 0
9 5 0
10 5 0
11 1
12 5 1
13 5
14 5 0
15 3 1
16 2 1
17 2 4
18 8 0
19 3
20 5 2
21 4 2
22 2 2
run_tests.sh
1 #/bin/sh
2 ./a.out < test.txt > dummy.txt
3 sed '/Enter/d; /process/d' dummy.txt > results.t
4 rm dummy.txt
```

Prateek Rathod

11814621

emailawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

Results.txt

```
results.txt
1  Test Case 1
2
3  pord      bt      at      wt      tat
4  p4      34321    233      0      34321
5  p1      599      234     34320    34919
6  p2     21435    5435    29718    51153
7  p3     755465   2322    54266    809731
8
9  Average Waiting Time: 29576.000
10 Average Turnaround Time: 232531.000
11
12 -----
13
14 Test Case 2
15
16 pord      bt      at      wt      tat
17 p1      5      0      0      5
18 p2      5      0      5     10
19 p3      5      0     10     15
20
21 Average Waiting Time: 5.333
22 Average Turnaround Time: 10.000
23
24 -----
25
26 Test Case 3
27
28 pord      bt      at      wt      tat
29 p1      5      1      0      5
30
31 Average Waiting Time: 0.000
32 Average Turnaround Time: 5.000
33
```

Prateek Rathod

11814621

emailawrathod@gmail.com

<https://github.com/lawRathod/Modified-Non-Preemptive-SJFS>

Code: 2

```
33
34 -----
35
36 Test Case 4
37
38 pord      bt      at      wt      tat
39 p1         5       0       0       5
40 p3         2       1       4       6
41 p2         3       1       6       9
42 p4         2       4       6       8
43 p5         8       0      12      20
44
45 Average Waiting Time: 5.600
46 Average Turnaround Time: 9.600
47
48 -----
49
50 Test Case 5
51
52 pord      bt      at      wt      tat
53 p3         2       2       0       2
54 p2         4       2       2       6
55 p1         5       2       6      11
56
57 Average Waiting Time: 2.667
58 Average Turnaround Time: 6.333
59
60 -----
61
62
```