# images

**LOVELY PROFESSIONAL UNIVERSITY**

A Report on Operating Systems(OS) Project

Done by:

Name: Akashdeep Chattopadhyay

Regd No.: 11806436

Roll No.: B40

Section: K18GE

Email: [akashdeep721301@yahoo.com](mailto:akashdeep721301@yahoo.com)

Github: <https://github.com/Akashdeep-312>

Under the guidance of:

Name: Milanjit Kaur

Designation: Asst. Professor

Subject: CSE316 (Operating Systems)

Contents Page No

Code to Help…………..…………………………………………………………………3

Problem Description.…………………………………………………………………5

Proposed Algorithm…………………………………………………………………..5

Complexity of Algorithm…………..……………………………………………….7

Purpose of Use……………………...………………………………………………….7

Code Snippet……….……………………………………………………………………7

Boundary Condition.…………………………………………………………………7

Test Cases……...…………………………………………………………………………7

Code to Help

#include<stdio.h>

#include<unistd.h>

int scheduling\_process(int processing\_no, int processing\_no\_2)

{

printf("Process with %d burst time is under execution. \n", processing\_no);

sleep(processing\_no);

printf("The execution of the process with larger burst time is completed.\n");

printf("Process with %d burst time is under execution \n", processing\_no\_2);

sleep(processing\_no\_2);

printf("The execution of the process with smaller burst time is completed.\n");

printf("Now both the processes have completed executing.\n");

}

int main()

{

int array\_burst\_time[2], array\_arrival\_time[2], array\_final\_burst\_time[2], completion\_time;

char array\_process\_no[2] = {'a','b'};

printf("We have 2 process. ");

printf("Enter the values of arrival time: \n");

for(int i=0;i<2;i++)

{

scanf("%d",&array\_arrival\_time[i]);

}

for(int i=0;i<2;i++)

{

array\_burst\_time[i] = array\_arrival\_time[i] \* 2;

}

printf("The respective values of the arrival time and burst time is:\n");

printf("PROCESS BT AT \n");

for(int i=0;i<2;i++)

{

printf("%c %d %d \n",array\_process\_no[i], array\_burst\_time[i], array\_arrival\_time[i]);

}

printf("The value with the largest burst time will be selected: \n");

for(int i=0;i<2;i++)

{

int average\_waiting\_time, average\_turnaround\_time, turnaround\_time\_1, turnaround\_time\_2, waiting\_time\_1, waiting\_time\_2;

if(array\_burst\_time[i] > array\_burst\_time[i+1])

{

int process\_no;

array\_final\_burst\_time[i] = array\_burst\_time[i];

array\_final\_burst\_time[i+1] = array\_burst\_time[i+1];

completion\_time = array\_final\_burst\_time[i] + array\_final\_burst\_time[i+1];

printf("Larger burst time= %d. Execution time= %d \n",array\_final\_burst\_time[i], array\_final\_burst\_time[i]);

printf("Smaller burst time= %d. Execution time= %d \n",array\_final\_burst\_time[i+1], completion\_time);

turnaround\_time\_1 = array\_final\_burst\_time[i] - array\_arrival\_time[i];

turnaround\_time\_2 = completion\_time - array\_arrival\_time[i+1];

average\_turnaround\_time = (turnaround\_time\_1 + turnaround\_time\_2)/2;

printf("The average turnaround time= %d \n",average\_turnaround\_time);

waiting\_time\_1 = turnaround\_time\_1 - array\_burst\_time[i];

waiting\_time\_2 = turnaround\_time\_2 - array\_burst\_time[i+1];

average\_waiting\_time = (waiting\_time\_1 + waiting\_time\_2)/2;

printf("The average waiting time= %d \n",average\_waiting\_time);

process\_no = scheduling\_process(array\_final\_burst\_time[i], array\_final\_burst\_time[i+1]);

break;

}

else if(array\_burst\_time[i] < array\_burst\_time[i+1])

{

int process\_no;

array\_final\_burst\_time[i] = array\_burst\_time[i+1];

array\_final\_burst\_time[i+1] = array\_burst\_time[i];

completion\_time = array\_final\_burst\_time[i] + array\_final\_burst\_time[i+1];

printf("Larger burst time= %d. Execution time= %d \n",array\_final\_burst\_time[i], array\_final\_burst\_time[i]);

printf("Smaller burst time= %d. Execution time= %d \n",array\_final\_burst\_time[i+1], completion\_time);

turnaround\_time\_1 = array\_final\_burst\_time[i] - array\_arrival\_time[i];

turnaround\_time\_2 = completion\_time - array\_arrival\_time[i+1];

average\_turnaround\_time = (turnaround\_time\_1 + turnaround\_time\_2)/2;

printf("The average turnaround time= %d \n",average\_turnaround\_time);

waiting\_time\_1 = turnaround\_time\_1 - array\_burst\_time[i];

waiting\_time\_2 = turnaround\_time\_2 - array\_burst\_time[i+1];

average\_waiting\_time = (waiting\_time\_1 + waiting\_time\_2)/2;

printf("The average waiting time= %d \n",average\_waiting\_time);

process\_no = scheduling\_process(array\_final\_burst\_time[i], array\_final\_burst\_time[i+1]);

break;

}

else if(array\_burst\_time[i] = array\_burst\_time[i+1])

{

printf("There will be context switching among the processes since they are arriving at the same time.\n");

break;

}

}

return 0;

}

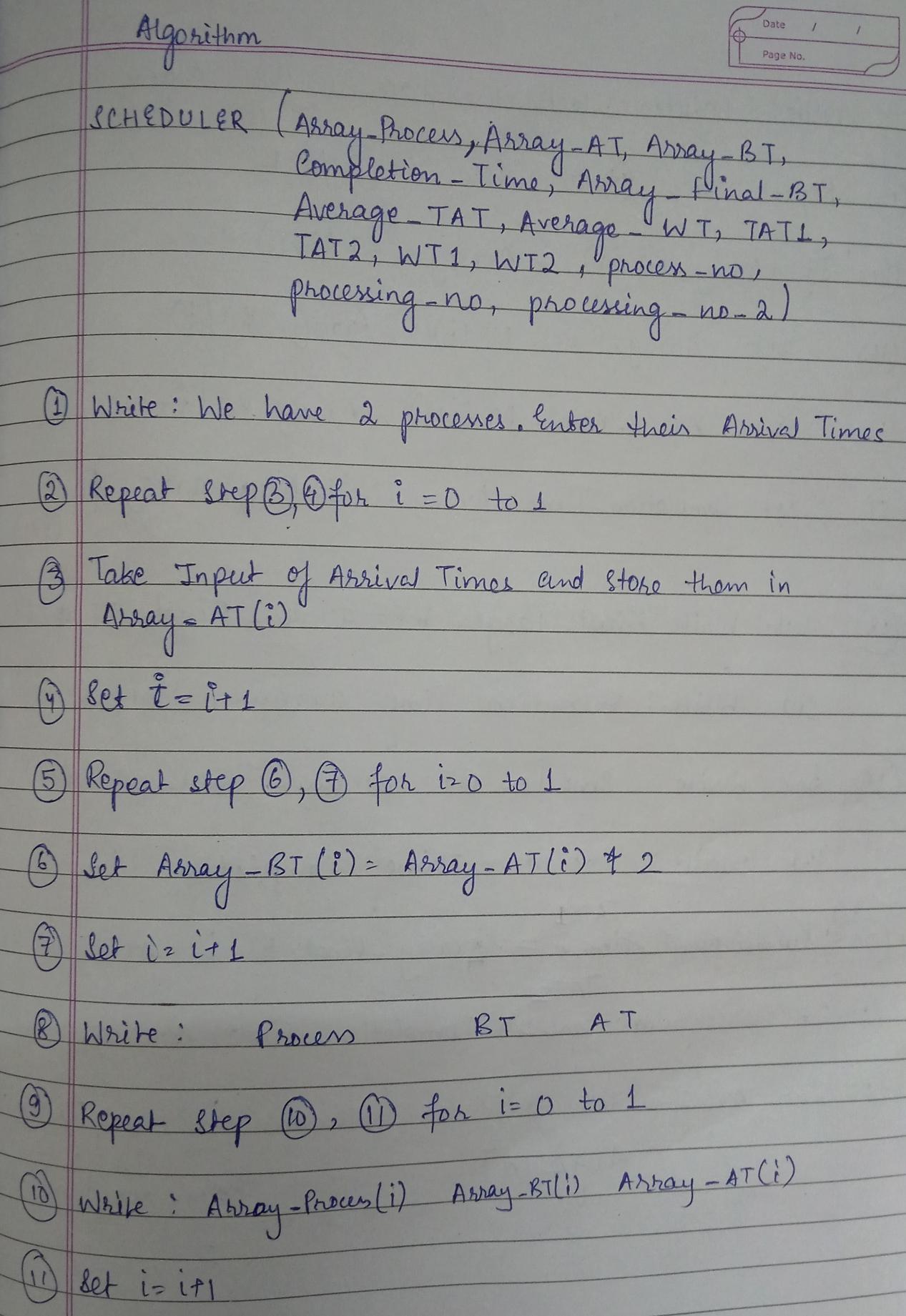
Problem Description

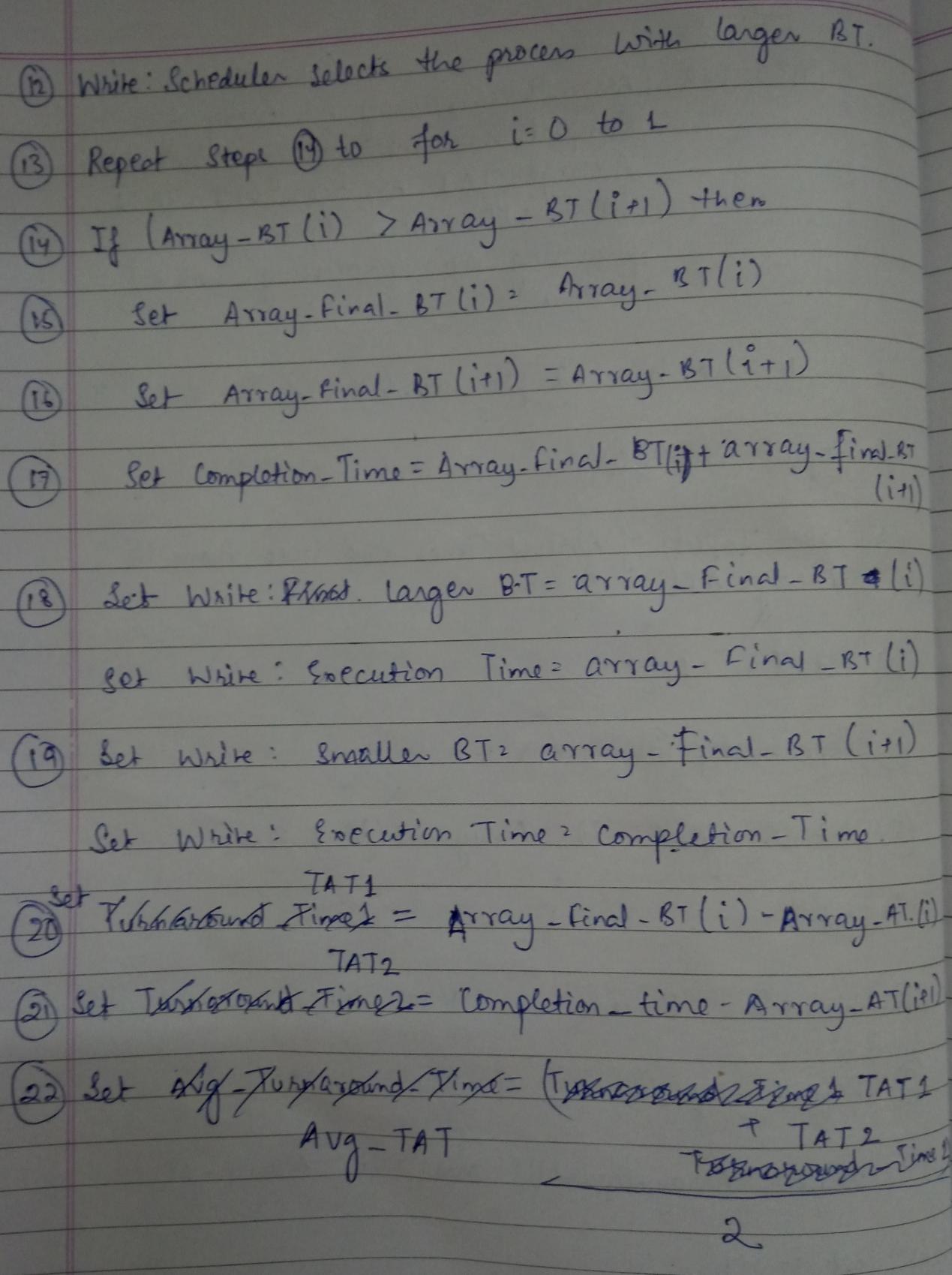
The problem assigned to me was to make a scheduler which selects the process which are having larger burst time. In view of the operating system, we see that the system CPU(s) selects the processes which has the smaller burst time, in order to complete them fast, but the result happens out to be unusual sometimes. The process which has the smaller burst time takes too much time to execute due to some internal problems, which in turn makes the processes which has the larger burst time wait.

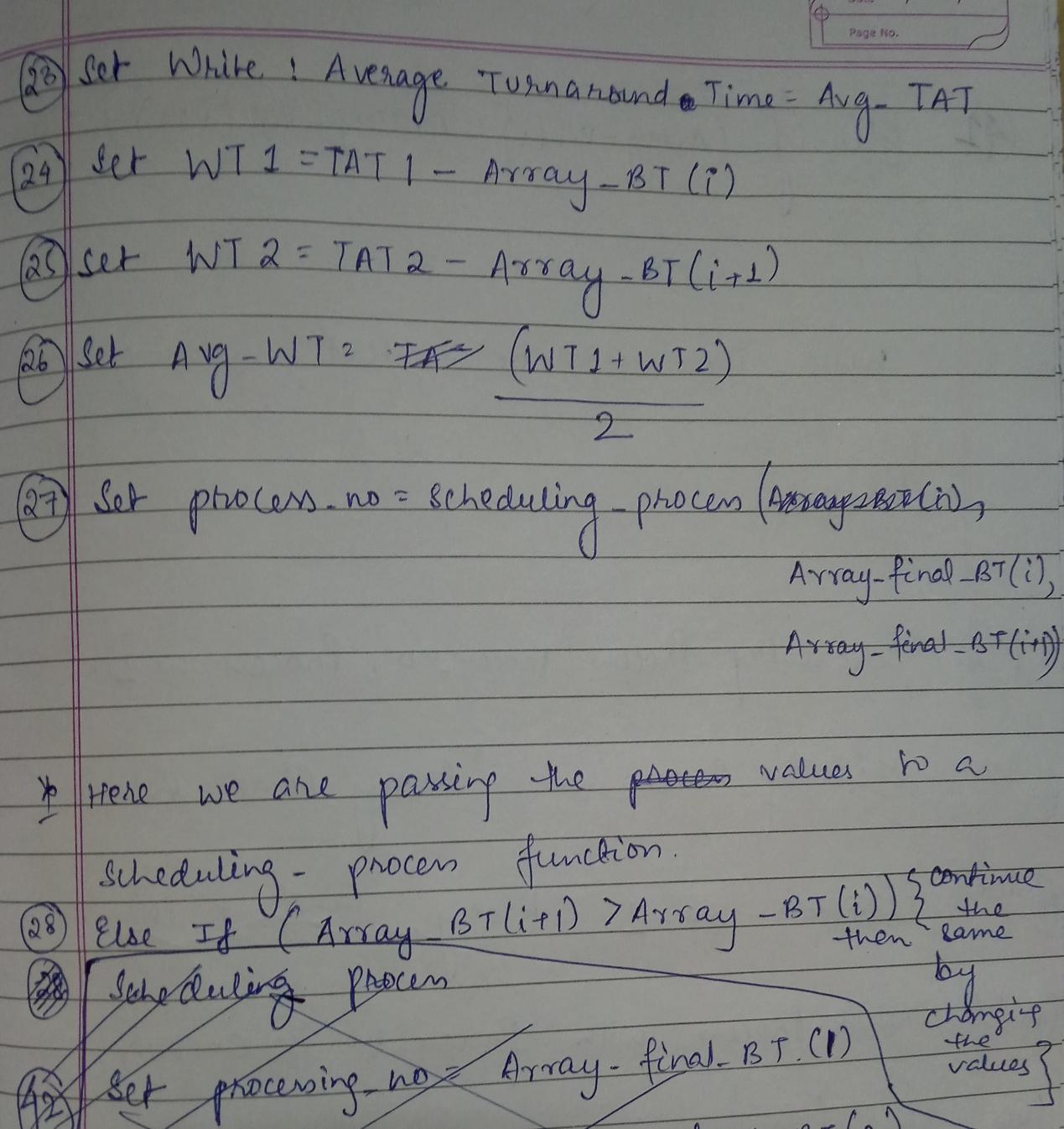
Proposed Algorithm

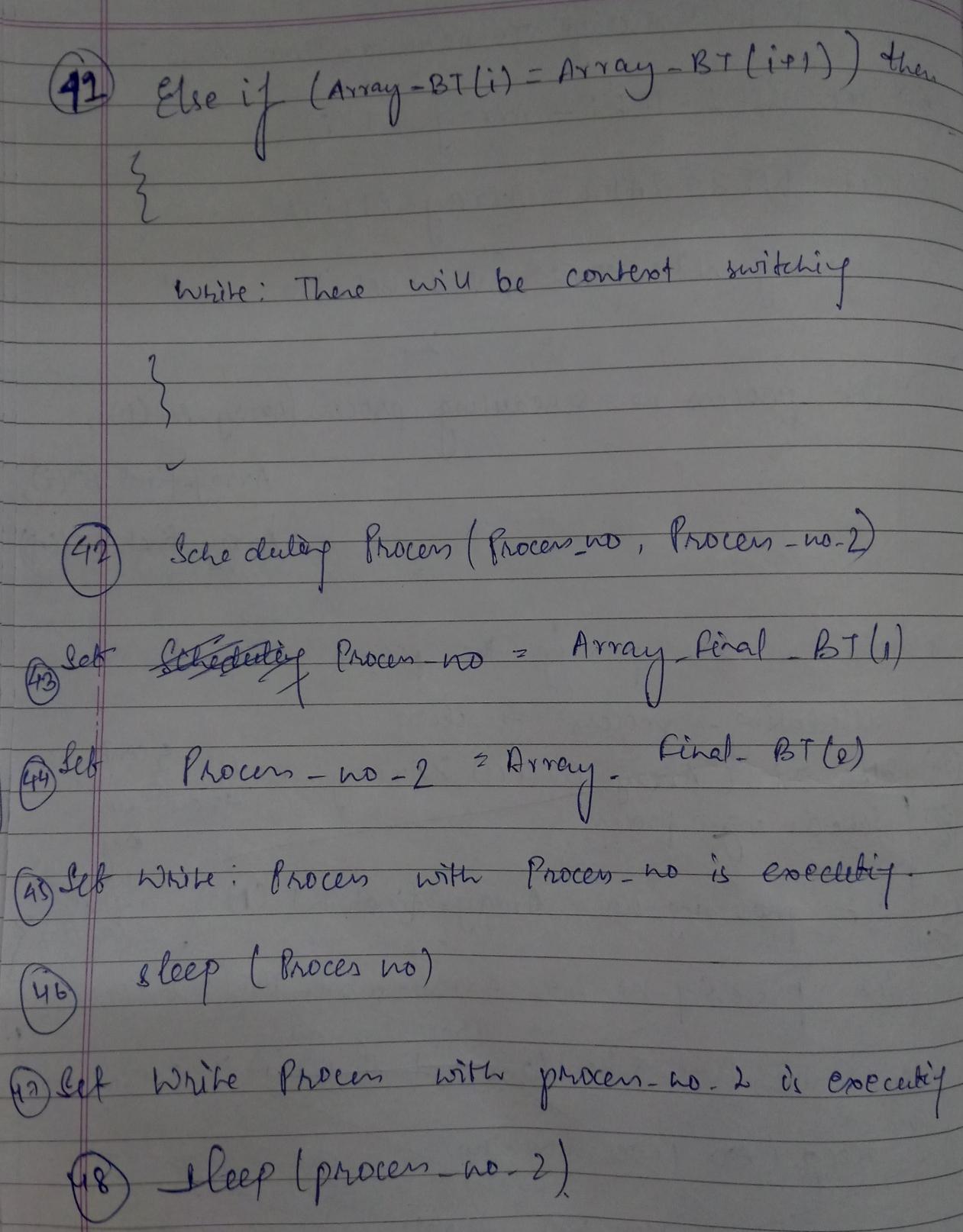
After sorting the issue of the problem, we came up with a solution in the form of algorithm, which describes as follows:

1. We have some functions that contains some statements, main function that describes all the arrays (in which the arrival time, burst time, completion time {of the processes} are going to be stored, arrays which will store the processes with larger burst time.
2. We have methods to take arrival time as input to produce burst time as output and calculate turnaround time and waiting time for our necessity.









Complexity of Algorithm

Each algorithm has some complexity (describes how difficult the algorithm is to solve). The algorithm proposed here has the complexity: **“n” (since we have “for” loops in the serial order where 1. In the first loop, we are giving “n” inputs; 2. In the second “for” loop, we are getting “n” outputs; 3. In the third “for” loop, we are comparing “n” values, calculating values for “n” values, passing “n” values to the functions mentioned in the code.)**

Purpose of Use

Schedulers that select processes which has larger burst time helps in executing the processes much faster. Thus, the efficiency of such schedulers are better.

Code Snippet

No extra algorithm applied to create this scheduler.

Boundary Conditions

The boundary conditions of the code is that we can’t have float values for the inputs.

Test Cases

A sample test case is used to describe the code. This is as follows:

