OS PEDAGOGY

Team Members:

1. Prajwal Kulkarni, 1KS19CS070
2. Preethi KP, 1KS19CS071
3. Shreesha S, 1KS19CS088
4. Akshay R, 1KS19CS115

Algorithm:

Shortest Remaining Time First (SRTF)

Class:

4 'B', CSE

JUNE 3

KSIT, Dept. of CSE

Vaneeta Ma'am

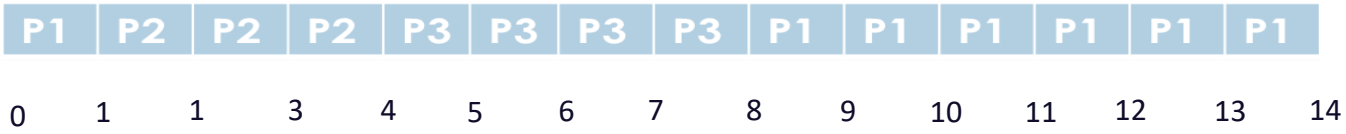
Shortest Remaining Time First:

- Shortest remaining time, also known as shortest remaining time first (SRTF), is a scheduling method that is a preemptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, the process will either run until it completes or get preempted if a new process is added that requires a smaller amount of time.
- Shortest remaining time is advantageous because short processes are handled very quickly. The system also requires very little overhead since it only makes a decision when a process completes or a new process is added, and when a new process is added the algorithm only needs to compare the currently executing process with the new process, ignoring all other processes currently waiting to execute.
- **Advantages**: SRTF algorithm makes the processing of the jobs faster than SJN algorithm, given its overhead charges are not counted.
- **Disadvantages:** In SRTF, the context switching is done a lot more times than in SJN due to more consumption of the CPU's valuable time for processing. The consumed time of CPU then adds up to its processing time and which then diminishes the advantage of fast processing of this algorithm.

Example: Consider the following 3 process:

Process queue	Arrival time	Burst time
P1	0	7
P2	1	3
P3	3	4

The Gantt chart for SRTF will be:



Explanation:

- **At the 0th unit of the CPU, there is only one process that is P1, so P1 gets executed for the 1 time unit.**
- **At the 1st unit of the CPU, Process P2 arrives. Now, the P1 needs 6 more units more to be executed, and the P2 needs only 3 units. So, P2 is executed first by preempting P1.**
- **At the 3rd unit of time, the process P3 arrives, and the burst time of P3 is 4 units which is more than the completion time of P2 that is 1 unit, so P2 continues its execution.**
- **Now after the completion of P2, the burst time of P3 is 4 units that mean it needs only 4 units for completion while P1 needs 6 units for completion.**
- **So, this algorithm picks P3 above P1 due to the reason that the completion time of P3 is less than that of P1.**

- **P3 gets completed at time unit 8, there are no new process arrived.**
- **So again, P1 is sent for execution, and it gets completed at the 14th unit.**
- **As Arrival Time and Burst time for three processes P1, P2, P3 are given in the above diagram. Let us calculate Turnaround time, completion time, and waiting time**

Calculation:

Process	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT
P1	0	7	14	14	7
P2	1	3	5	4	1
P3	3	4	8	5	1

In the Above table
CT-completion time
TAT - Turnaround time
WT-Waiting time
Average waiting time is calculated by adding the waiting time of all processes and then dividing them by no. of processes.
Average waiting time = waiting for time of all processes/ number of processes
Average waiting time = 7+1+1 = 9/3 = 3ms
Average turnaround time = 14+4+5 = 23/3 = 7.67ms

Our work:

Tech stack:

- HTML
- CSS
- Bootstrap
- JavaScript

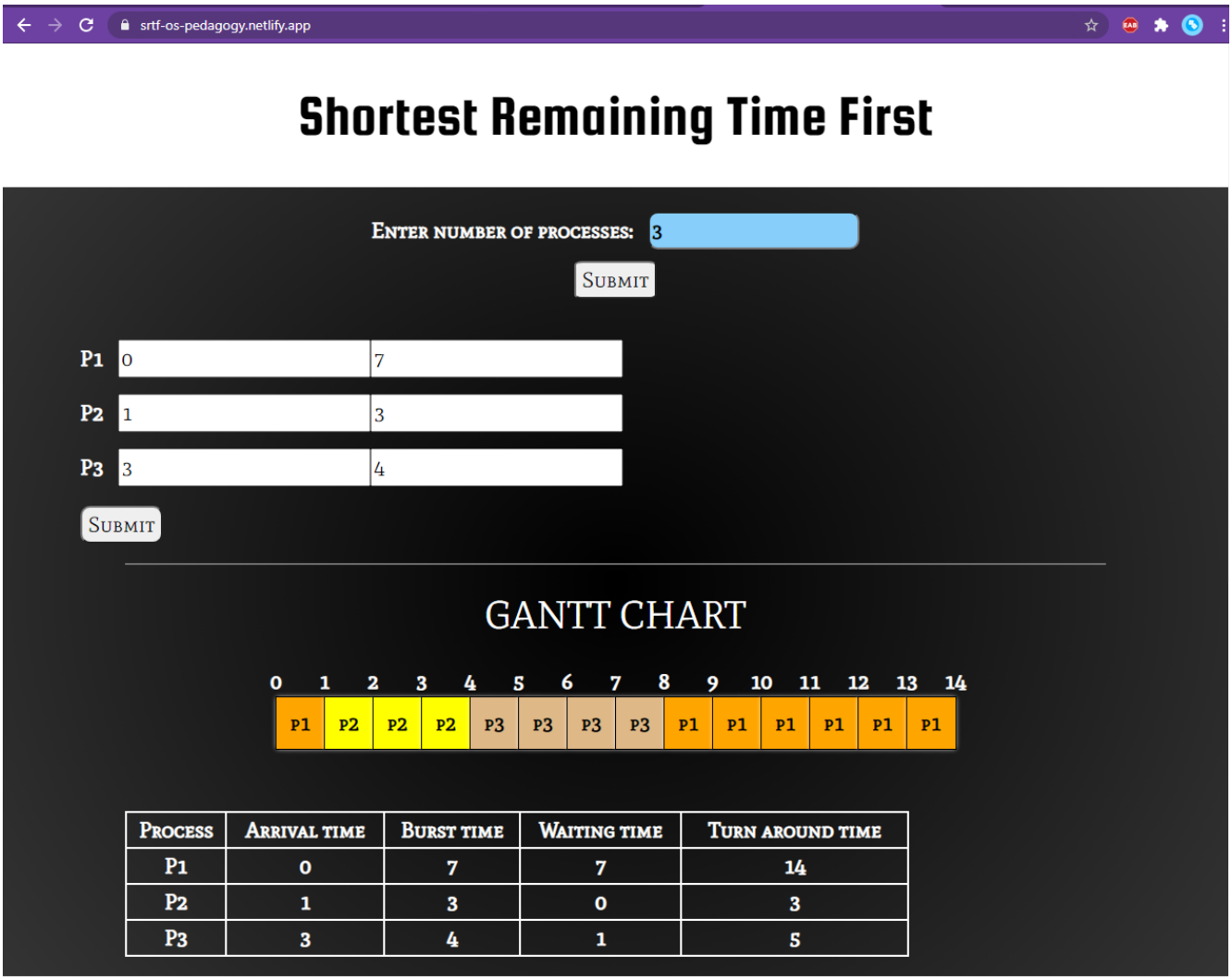
Website Link:

http://srtf-os-pedagogy.netlify.app/

CODE SNIPPET:

```
function strf(){
   let tt=0, t=0, gantValues = 0 ;
   for(i=0; i<a.length; i++)
      tt += a[i].bt;
   while (t!=tt){
      for(i=0; i<a.length; i++){
         if(t==a[i].at){
            b.push(a[i]);
            b.sort(function(x, y){return x.rt-y.rt});
         }
      }
      t++;
      if(b.length > 0){
         document.querySelector("#gantt").innerHTML += `<span class="gChart" style =
"background-color:${colors[b[0].pid-1]}; color:black; box-shadow: 0 0 5px
white;">p${b[0].pid}</span>`;
         document.querySelector("#gantvalues").innerHTML += `<span class="gChart
gChartValues">${gantValues++}</span>`;
         b[0].rt -= 1;
         if(b[0].rt==0){
            b[0].ct=t;
            b.shift();
         }
      }
   } document.querySelector("#gantvalues").innerHTML += `<span class="gChart
gChartValues">${gantValues}</span>`;
}
```

Snapshot of the Website:



Different input values: