Job Scheduling Using C language

Submission Report

Submitted To: - Cherry Khosla

Name: Surendra Jajra
Roll no: RK21SBB61
Registration no: 12115452

CODE

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

#define MAX_JOBS 100

// function to calculate priority
float priority(int waiting_time, int estimated_run_time) {
    return 1 + (float)waiting_time / estimated_run_time;
}

int main() {
    int num_jobs;
    int estimated_run_time[MAX_JOBS];
    int waiting_time[MAX_JOBS] = {0};
    int run_time[MAX_JOBS];
    int job_index[MAX_JOBS];
    bool completed[MAX_JOBS] = {false};
    int time = 0;
    int gantt_chart[MAX_JOBS];
    int num_completed = 0;
    float avg_waiting_time;

    // input the number of jobs
    printf("Enter the number of jobs: ");
    scanf("%d", &num_jobs);

    // input the details of each job and add it to the list
    for (int i = 0; i < num_jobs; i++) {
        printf("Enter details of job %d:\n", i+1);
        printf("  Estimated run time: ");
        scanf("%d", &estimated_run_time[i]);
        job_index[i] = i;
```

```
        run_time[i] = estimated_run_time[i];
    }

    // process jobs until all are done
    printf("\nGantt chart:\n");
    printf("  ");
    while (num_completed < num_jobs) {
        // find the job with the highest priority that is not completed
        int highest_priority_job = -1;
        float highest_priority = -1.0;
        for (int i = 0; i < num_jobs; i++) {
            if (!completed[i]) {
                float p = priority(waiting_time[i], estimated_run_time[i]);
                if (highest_priority_job == -1 || p > highest_priority) {
                    highest_priority_job = i;
                    highest_priority = p;
                }
            }
        }

        // update waiting time and add job to gantt chart
        waiting_time[highest_priority_job] = time -
run_time[highest_priority_job];
        gantt_chart[time] = job_index[highest_priority_job];

        // update time and remaining run time
        time += run_time[highest_priority_job];
        run_time[highest_priority_job] = 0;

        // mark job as completed and increment counter
        completed[highest_priority_job] = true;
        num_completed++;

        // update waiting time of other jobs
        for (int i = 0; i < num_jobs; i++) {
```

```
            if (!completed[i] && i != highest_priority_job) {
                waiting_time[i] += run_time[highest_priority_job];
            }
        }

        // display priority of each process after each unit of time
        printf("%.1f ", priority(waiting_time[highest_priority_job],
estimated_run_time[highest_priority_job]));
    }

    // calculate average waiting time
    int total_waiting_time = 0;
    for (int i = 0; i < num_jobs; i++) {
        total_waiting_time += waiting_time[i];
    }
    avg_waiting_time = (float)total_waiting_time / num_jobs;

    // output results
    printf("\nWaiting times: ");
    for (int i = 0; i < num_jobs; i++) {
        printf("%d ", waiting_time[i]);
    }
    printf("\nAverage waiting time: %.2f\n", avg_waiting_time);

    return 0;
}
```

- **Methodology**

The problem involves scheduling a set of jobs with varying estimated run times and finding the order in which they should be processed to minimize the average waiting time of the jobs. The solution approach is based on the Priority Scheduling algorithm.

The Priority Scheduling algorithm involves assigning priorities to jobs based on some criteria, such as the estimated run time or the priority value assigned to the job. Jobs with higher priority values are executed before those with lower priority values.

In this case, the priority of each job is calculated as the ratio of the waiting time to the estimated run time plus 1. The waiting time is the time for which the job has been waiting since it was added to the queue. The estimated run time is the time the job is expected to take to complete based on some estimate.

The algorithm involves processing jobs one by one based on their priority values. The job with the highest priority value is processed first, and its remaining run time is updated. The waiting time of the other jobs is updated based on the run time of the completed job. The process is repeated until all the jobs are completed.

The waiting time of each job is the difference between the completion time of the job and its estimated run time. The average waiting time is the sum of the waiting times of all the jobs divided by the number of jobs.

The methodology adopted involves:

- Input the number of jobs and the details of each job, including the estimated run time.
- Calculate the priority of each jobs based on the waiting time and estimated run time.
- Process jobs based on priority values until all jobs are connected.
- Update waiting time of other jobs and calculate average waiting time.
- Output the Gantt chart , waiting times and average waiting time.

Overall, the approach is to prioritize jobs based on some criteria and process them in the order of their priorities. This helps minimize the waiting time of the jobs and improves the efficiency of the scheduling process.

➢ Output example:

```
Enter the number of jobs: 4
Enter details of job 1:
  Estimated run time: 2
Enter details of job 2:
  Estimated run time: 3
Enter details of job 3:
  Estimated run time: 1
Enter details of job 4:
  Estimated run time: 4

Gantt chart:
  1.0 2.0 2.5 3.5 4.0 4.0 4.0 4.0
Priority of each process:
Job 1: 1.0 1.5 0.0 0.0 0.0 0.0 0.0 0.0
Job 2: 1.5 1.67 1.5 1.0 0.0 0.0 0.0 0.0
Job 3: 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Job 4: 1.25 1.0 1.0 1.0 1.0 1.0 1.0 1.0
Waiting times: 2 5 0 9
Average waiting time: 4.00
```

```
Enter the number of jobs: 3
Enter details of job 1:
  Estimated run time: 4
Enter details of job 2:
  Estimated run time: 2
Enter details of job 3:
  Estimated run time: 6

Gantt chart:
  1.2 1.6 2.2 2.8 3.0 3.5 3.83 4.67 5.17 5.67 6.0 6.0
Priority of each process:
Job 1: 1.2 1.6 2.2 2.8 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
Job 2: 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Job 3: 1.33 1.4 1.33 1.2 1.0 0.83 0.67 0.5 0.33 0.17 0.0 0.0
Waiting times: 4 0 15
Average waiting time: 6.33
```

```
Enter the number of jobs: 2
Enter details of job 1:
  Estimated run time: 6
Enter details of job 2:
  Estimated run time: 3

Gantt chart:
  1.17 1.67 2.0 2.33 2.67 3.0 3.0 3.0 3.0 3.0 3.0 3.0
Priority of each process:
Job 1: 1.17 1.67 2.0 2.33 2.67 3.0 3.0 3.0 3.0 3.0 3.0 3.0
Job 2: 1.0 1.5 1.0 0.67 0.5 0.4 0.33 0.29 0.25 0.22 0.2 0.18
Waiting times: 9 2
Average waiting time: 5.50
```

```
Enter the number of jobs: 4
Enter details of job 1:
  Estimated run time: 6
Enter details of job 2:
  Estimated run time: 10
Enter details of job 3:
  Estimated run time: 8
Enter details of job 4:
  Estimated run time: 7

Gantt chart:
  1.0 2.2 3.4 4.9 1.8 2.9 3.9 1.4 2.7 4.0 4.9 4.9 4.9
Waiting times: 4 20 12 11
Average waiting time: 11.75
```