DAA PROJECT

SJF SCHEDULING WITH PRIORITY

Using Heap Sort

- SHRESHT BHATIA (169105183)

SJF SCHEDULING WITH PRIORITY
Using Heap Sort

Problem Statement

Develop an algorithm for CPU scheduler for a SJF scheduling in which each job has an priority and the job with the highest priority gets scheduled first using heap sort.

Why This Topic?

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold(in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair. Heap Sort is efficient for CPU scheduling due to the worst case upper bound is O(nlogn) and helps improve the efficiency of CPU scheduling.

Objective & Scope

This is an online tool to demonstrate the working of a SJF CPU scheduler when each job has a priority and the highest priority gets scheduled first.
The two main features of this project are the following:

1. Implementation of Heap Sort on a normal Priority CPU scheduler. This algorithm can be used to acquire the processes in increasing order of priority.
2. The users can choose any number of processes and schedule them.

The SJF Priority scheduler code on first schedules the processes according to increasing order of priority and if priority is equal then on basis of their arrival time. The way it is scheduled is better than the normal way of scheduling which uses merge sort instead of heap sort as although the worst case upper bound of both the sorting techniques is O(nlogn), the merge sort actually needs O(n) extra space as compared to Heap Sort. So Heap Sort is more useful for real time(or time bound) processes as compared to merge sort.

Methodology

A user enters the burst times and the priorities of the processes. The user has the option to increase or decrease the number of processes by clicking the “+” or the “-“ button under the table. When the burst times and priorities are filled and the calculate button is clicked, the main_prio_sort function is called in which Heap Sort is implemented on the priorities. Condition for processes with same priorities is checked and such processes are sorted on basis of their burst time (sjf). After that the order of execution of the processes is found and the draw function is called which creates a table to print output on screen. This further enhanced by calling a animate function which creates a curtain effect which gives a appearance of the execution of processes at each moment in the time cycle while the time clock is updated. The step function is used to generate this timer while running from 0 to the sum of all the burst times of all the processes. This table is displayed on the GUI along with the time taken to schedule all the jobs.

Heap Sort-
Since, a Binary Heap is a Complete Binary Tree, it can be easily represented as array and array based representation is space efficient. If the parent node is stored at index I, the left child can be calculated by 2 * I + 1 and right child by 2 * I + 2.

1. Initially on receiving an unsorted list, the first step in heap sort is to create a Heap data structure(Max-Heap).
2. Once heap is built, the largest item is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of heap by 1.
3. Then we again make heap using the remaining elements by heapify-ing the root of tree.
4. Again pick the first element of the heap and put it into the array.
5. We keep on doing the same repeatedly until we have the complete sorted list in our array.
6. It continues while size of heap is greater than 1.

Time complexity of heapify is O(Logn). Time complexity of create And BuildHeap() is O(n) and overall time complexity of Heap Sort is O(nLogn).

Hardware & Software Used

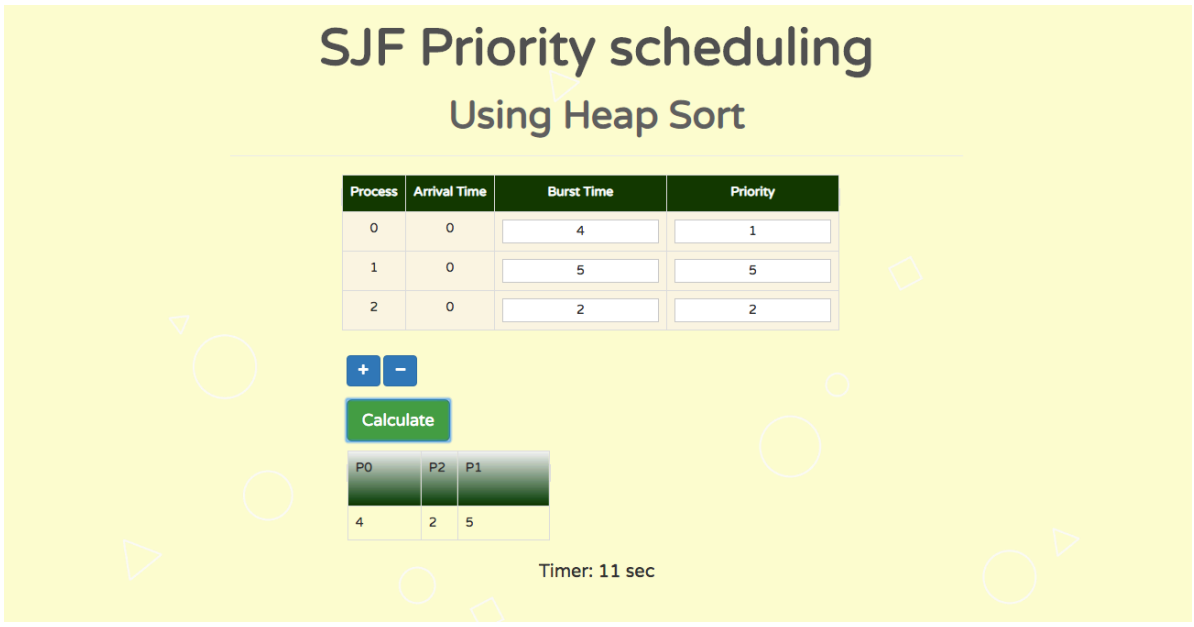
- HARDWARE
    - MacBook Air
        Configuration:
            Processor: 1.6 GHz Intel Core i5
            Ram: 4.00 GB
- SOFTWARE
    - Operating system software:
        - macOS High Sierra v.10.13.6
    - Project specific software:
        - Sublime Text Editor
        - Google Chrome

Testing

For testing we chose 10 different Source-Destination combination with Multiple paths between them to check whether our implementation of the Dijkstra Algorithm correctly chose the path with the least weight.

Sr.no	Number of processes	Burst times	Priorities	Execution Order
1	2	P0-2 P1-5	P0-2 P1-1	P1 | P0
2	2	P0-2 P1-5	P0-1 P1-2	P0 | P1
3	3	P0-2 P1-5 P2-3	P0-2 P1-2 P2-2	P0 | P2 | P1
4	3	P0-2 P1-2 P2-3	P0-2 P1-2 P2-3	P0 | P1 | P2
5	3	P0-2 P1-2 P2-2	P0-2 P1-5 P2-3	P0 | P2 | P1

Developed GUI



Contribution

The Path Calculator GUI has info about 50 places in Jaipur. Therefore other than important cities, the user can also get to know the shortest distance between two relatively lesser known places. It saves user time in travelling between places in the city.

- In Emergency conditions routing to the shortest path is absolutely critical and can Be the reason of ones survival.
- This site can help in exploring Jaipur with less money being spend on travelling.