International Journal of Computer Applications (0975 – 8887)
Volume 145 – No.11, July 2016

Design of Task Scheduling System for Conference Management Application

Anish Narkhede
B.E.(Computer)
Department of Computer Engineering
Savitribai Phule Pune University
Pune, MH, India.

ABSTRACT

Task scheduling systems play an important role in managing processes in an efficient way. They assess the time duration taken up by a particular process, and accordingly allot the processes slots so that they can complete their execution. Managing tasks in this way allows efficient use of the system resources and allows faster and optimized performance. Using similar techniques, tasks (other than system processes) can also be managed efficiently by filtering them using 'time/duration' as a primary component. The System proposed in this paper aims at creating a system which helps to manage Conferences into three slots viz. Shortest Conferences first, Longest Conferences First, & First Come First Serve, which would be done by obtaining Conference details from the user. Details would include title of the Conference and its duration. The slots will then be filled depending on duration of a particular conference.

General Terms

Task Scheduling, Conference Management, Insertion Sort

Keywords

First Come First Servce Approach, Insertion Sort, Data Structures

1. INTRODUCTION

With the increasing rate at which conferences are being organized, may it be tech conferences, trade shows, exhibits, conventions and seminars, or business meetings, it is of utmost importance to manage these events in an organized way so as to ensure their smooth functioning. With time being the key factor that comes into play during such events, it is important to manage them accordingly. The System proposed in this paper would work on the principle of 'First Come First Approach' which commonly used in the field on System programming and Operating Systems for Job Scheduling of system processes. The system would allow the user to organize the conferences based on 3 criteria:

- Schedule Conferences in Ascending Order (Time)
- Schedule Conferences in Descending Order (Time)
- Schedule the Conferences in an FCFS Manner

2. WORKING

The system would initially require data from the user. Details regarding the Conference, which would include the title/name of the conference and also the duration would need to be specified by the user. Also the date on which the conference is to be organized will also have to be specified. The conferences will then be pushed into a queue, where based on the duration of the conference, they will then be allotted a slot in either of the three Sessions. Every time a new conference arrives in the queue, the queue management will first check for the duration of the conference, and then accordingly, it will be allotted to either of the three Sessions. Insertion Sort algorithm will be used in order to populate the queue, as this sorting algorithm is efficient with sorting small data sets. The sorting algorithm will keep sorting the data in such a way that, even upon insertion of new data, the output of data displayed will remain to be sorted at all times.

3. LITERATURE REVIEW

3.1 Introduction

Scheduling, in computing refers to the process by which work that is specified is assigned to resources that would complete this work. [1] The work refers to threads, processes, data flows, which are then delegated to hardware resources such as processors. Schedulers are implemented so that they keep the system resources busy in the most efficient way possible. A scheduler aims at maximizing throughput, minimizing response time or latency.

3.2 First Come First Serve Approach

First Come, First Serve, also known as the First In, First Out (FIFO) method, is the simplest Scheduling Algorithm. FIFO simply queues the processes/tasks in a Queue, commonly referred to as the Task Queue. Implementation of this methodology is of relevant to the System proposed in this paper, as in First Come First Serve, the context of the operation switches only upon termination of the previous task. and no re-organizing of the Task Queue needs to be done. Thus the scheduling overhead is minimum. Also concepts such as wait time, turn-around time, response time are not relevant with regard to organizing Conferences, thus using FIFO approach is viable. Also lack of prioritization means that as long as every conference is being allotted to a Slot, no starvation of tasks would occur.

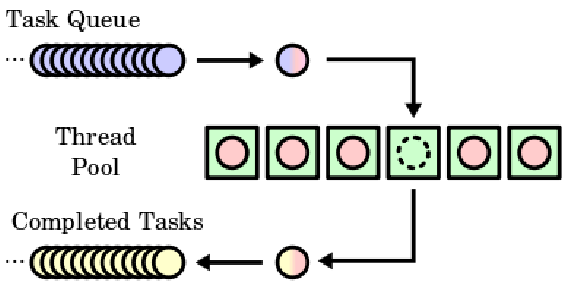


Figure 1: First Come First Serve Queue[4]

3.3 Queue Data Structure

A Queue is a data structure where we add data to the back, and remove data from the front. It is like 'waiting-in-line', the first one to be added, will be the first one to be removed. As a result, this Data Structure is also called as the First In First Out (FIFO) Data Structure.

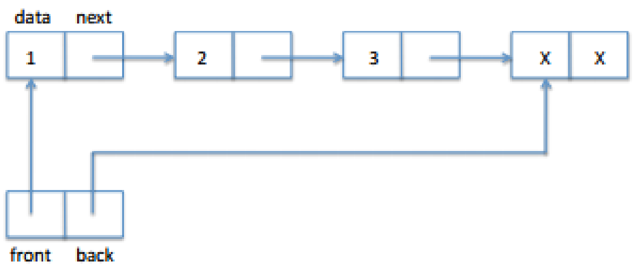


Figure 2: Queue Implementation

[2]A queue is usually implemented using a *struct* with a *front* field and a *back* field. While using arrays, we deal with an index, which is always one greater than the length of the array. Queues use a similar strategy, wherein the back field points to one element past the end of the queue.

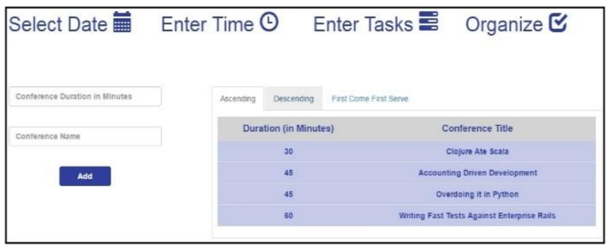
```
struct queue{
        list front;
        list back;
};
```

The queue data structure has two primary operations, which are used to manage the content of the Queue. These methods are – ***Enqueue*** and ***De-queue.*** The Enqueue method is responsible for adding data elements into the Queue from the back, while the Dequeue does the exact opposite, i.e. removes data elements from the front of the Queue.

4 METHODOLOGY/APPROACH

4.1 GUI

The UI of the application is designed using Web Technologies such as HTML, CSS and JavaScript. Besides, the application also employs Bootstrap (CSS front-end framework), which is an intuitive and powerful framework for front-end development. Also, the application is built using AngularJS MVC client-side framework, which allows efficient management of the system.



Duration (in Minutes)	Conference Title
30	Clojure Ate Scala
45	Accounting Driven Development
45	Overdoing it in Python
60	Writing Fast Tests Against Enterprise Rails

Figure 3: UI of the System used for taking user input

4.2 Insertion Sort Algorithm

The conference queue will be implemented using an array. In order to sort the conferences according to the 3 slots, sorting is key. These conferences will be sorted using Insertion sort algorithm.

Input: A sequence of *n* numbers ($a_1, a_2, \ldots . a_n$)

Output: A reordering of the *n* numbers such that $a_1 <= a_2 <, \ldots . <= a_n$.

[3] Insertion sort is an efficient algorithm to sort small data sets. It works the way people sort a hand while playing cards. Starting with an empty hand, and the cards face down, we pick up one card. This card is inserted into the correct position in the left hand. In order to find the correct position, we compare the card with one card before it, and one card after it. This process continues till a certain number of cards are in the player's hand.

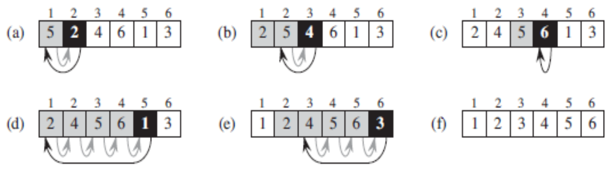


Figure 4: Operation of Insertion Sort on array

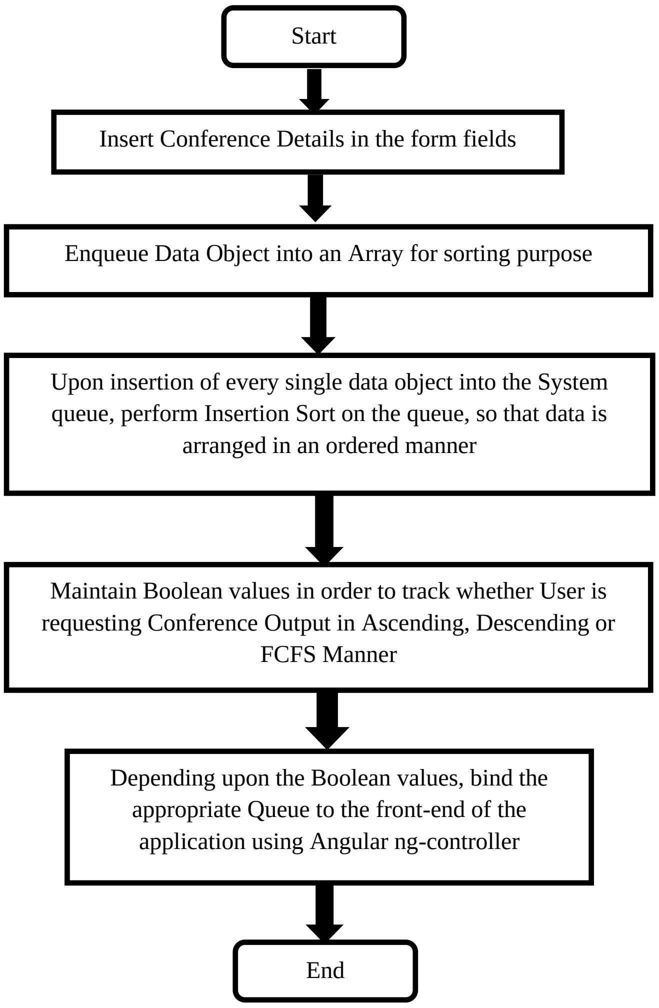
Pseudocode:

```
INSERTION-SORT(A)
    1  for j D 2 to A.length
    2  key = A[j]
    3  i = j - 1
    4  while i > 0 and A[i] > key
    5          A[i+1] = A[i]
    6          i = i - 1
    7  A[i+1] = key
```

The UI allows the user to provide details regarding the conference. After this detail has been obtained, the detail is encapsulated in an object and then pushed into an array. This array is then sorted using insertion sort, so as to provide sorting, right at the time of insertion, thus allowing the user to see the sequence of the conference as and when he/she adds it to the System.

,

5 SYSTEM DESIGN


Start
Insert Conference Details in the form fields
Enqueue Data Object into an Array for sorting purpose
Upon insertion of every single data object into the System queue, perform Insertion Sort on the queue, so that data is arranged in an ordered manner
Maintain Boolean values in order to track whether User is requesting Conference Output in Ascending, Descending or FCFS Manner
Depending upon the Boolean values, bind the appropriate Queue to the front-end of the application using Angular ng-controller
End

5.1 Data Insertion

The GUI of the application uses a form to collect data from the user. This data is then inserted into the System array, as an Object. This detail maintained in the queue is further used in order to perform Insertion Sort operation. Sorting is typically done in-place, by iterating up the array, growing the sorted list behind it. At each array-position, it checks the value there against the largest value in the sorted list (which happens to be next to it, in the previous array-position checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position.

```
if($scope.eventObj.name!=""&&$scope.eventObj.time!=""){
   $scope.events.push($scope.eventObj);
$scope.fcfsEvents.push($scope.eventObj);
}
```

The $scope.events array is used in order to maintain the conference information, while the $scope.fcfsEvents is an array used to maintain Conference information for the FCFS part of the application.

```
$scope.fcfsEvents = [
     {
        time : '',
        name : ''
     }
  ];
$scope.eventObj =  {time:"",name:""};
```

After the user adds data to the System, it is then encapsulated into an $scope.eventObj object, and is then pushed into the System queue.

5.2 Scheduling (Ascending)

Shortest Job First (SJF) is job scheduling principle which is most commonly used in System Programming & Operating System related tasks (System Processes). The principle is based on the simple logic in which the job which consumes the shortest amount of time is to be scheduled first. It is advantageous because of its simplicity and also reduces the average amount of time that each process has to wait. While all this is in regards with the System processes, this phenomenon can easily be applied to the Conference Management System, which would allow the user to sort the Conferences in such a way that, the Conference which takes up the minimum amount of time will be scheduled first. This requires the system to simply apply the insertion sort algorithm on the System Queue (array) which would then sort the data based on the duration of all the Conferences.

This Module uses the following array as a queue in order to sort the Conferences.

```
 $scope.events = [
    {
      time : '',
      name : ''
    }
  ];
```

The array consists of an Object that is modelled in such a way that allows the Insertion Sort algorithm to arrange all the data objects in an Ascending order.

```
 $scope.add_event = function(){


            if($scope.eventObj.name!=""&&
            $scope.eventObj.time!=""){
            $scope.events.push($scope.eventObj);
 if($scope.asc == true){
 for(var j=0;j<$scope.events.length;j++){
            var key = $scope.events[j];
            var i = j - 1;
 while(i > -1 && $scope.events[i].time > key.time){
   $scope.events[i+1]=$scope.events[i];
                        i = i -1;
 }
            $scope.events[i+1] = key;
 }}}
```

Boolean Function (Ascending)

```
 $scope.ascActivate = function(){
    $scope.desc = false;
    $scope.asc = true;
    $scope.sort();
 }
```

The System maintains Boolean value in order to keep a track of whether the user wants to sort the Conferences in an Ascending or a Descending order. Change in this value is triggered by a click event, upon which the Boolean is changed to *true*. The sorting algorithm picks up the *time* attribute of the *eventsObj* object. This value is then compared repeatedly with the *key* value in the array in order to re-arrange the array in a way that sorts the entire array.

5.1 Scheduling (Descending)

This Module adopts the exactly opposite functionality to that of the previous module. After checking the Boolean value that indicates the preference of sorting provided by the user, the System checks if the *$scope.desc* is true, upon which the Insertion Sort algorithm re-sorts the array, in case any additional data has been appended to the Queue. After sorting is done, the array is then reversed using the standard .reverse() method used on JavaScript arrays. The system then binds this data to the table on the front-end.

```
if($scope.desc == true){
for(var j=0;j<$scope.events.length;j++){
        var key = $scope.events[j];
        var i = j - 1;
while(i>-1 && $scope.events[i].time > key.time){
  $scope.events[i+1] = $scope.events[i];
                        i = i -1;
               }
        $scope.events[i+1] = key;
        }
$scope.events.reverse();}
```

Boolean Function (Descending)

```
 $scope.descActivate = function(){
    $scope.asc = false;
    $scope.desc = true;
    $scope.reverseSort();
 }
```

6 CONCLUSION

The proposed system allows the end user to manage/schedule conference according to his needs. It will allow the user to completely re-order or re-organize the entire schedule if he wishes to do so, by re-arranging the schedule upon selection of a different tab. Insertion of Conference data is sorted as and when the user inputs the information, thus allowing the user to observe how the schedule will be laid out, which is possible due to the implementation of Insertion Sort.

7 FUTURE SCOPE

Currently, the system employs simple sort algorithms which allow easy and quick sorting of the data that the system receives. Insertion sort works well, and in an efficient way on small data sets. However, if the amount of Conference data that the system handles is to be increased, the system needs to be scaled up in way that would allow quicker sorting of data. This would in turn require implementation of advanced sorting algorithms which would provide a consistent throughput. The data that is received, can also be backed up in database, that would allow the users to access old Conference Tracks if necessary.

8 REFERENCES

[1] https://en.wikipedia.org/wiki/Scheduling_(computing)#First_in.2C_first_out

[2] https://www.cs.cmu.edu/~wlovas/15122-r11/lectures/09-queues.pdf

[3] Introduction to Algorithms by Charles E. Leiserson, Clifford Stein, Ronald Rivest, and Thomas H. Cormen

[4] https://en.wikipedia.org/wiki/File:Thread_pool.svg

9 AUTHOR PROFILE

Anish Narkhede received his B.E. degree in Computer Engineering from Savitribai Phule Pune University (formerly known as University of Pune), Maharashtra, in 2015.