Time Comparison Algorithm for University Examination Scheduling

Mohamed Shajahan H
College of Engineering and Information Technology
University of Business and Technology
Jeddah, Saudi Arabia

H Mohamed Zakir
Preparatory year Deanship
King Faisal University
Saudi Arabia

ABSTRACT

Every educational institution needs to create either lecture or examination time tables for their students. Almost all institution prefers to perform this task with the help of software. The difficult and complex part involved during the development of suchtype of software's are comparing times. Programmer should come across a situation to find logic for time comparison and if it is not performed properly, it may leads to an improper output. For example, one student may get schedule for two different coursesin the same time or in other scenario, same examination hall will be allocated for two different courses in the same time. Both the situation leads to a mess.

This paper discusses about the time comparison techniques for such software's. There are various methods available in modern day programming languages for comparing two timings. But in some cases, we have to performthe comparison up to four or more timings. The reason for this paper is, the same situation was faced by me and browsed through different articles and blogs for a possible solutions. But most blogs remains unanswered or answered with some complex solutions. So, it was an inevitable situation for me to find a new solution for the problem. This paper proposes an algorithm for comparing up to four timings.

In this paper, we discussed about the multiple instances where the time comparison is required while developing such software's and the various methods available inprogramming environmentfor addressing the same. The implementationdifficulties for the specific requirement and the outputs acquired through the above methods are also discussed. Also, the comparison of outputs from the Dotnet methodsand our algorithm is also addressed.

Keywords: Course or Examination Scheduling, Timetabling, Time Comparison, Timeslots, University

1- INTRODUCTION

In every educational institution, always there is a need to prepare course or exam schedules for their learners. In a small to medium size institutions with less number of learners, it is easy to prepare the schedulemanually. But, it is a difficultand tedious task when comes to a large scale educational institutions. Moreover, there are more possibilities of error in manual scheduling, which results into the disaster. So, there comes a need for software which should capable of preparing schedules in a single click.

When developing scheduling software's, the programmers may require comparingtimes in their logics. There may be a situation for performing a comparison between two timings or more. For example, if we consider two times ***TIME-A*** (Start Time) and ***TIME-B*** (End Time), it requires the following comparisons.

a. ***TIME-A*** is greater than ***TIME-B***
b. ***TIME-B*** is lesser than ***TIME-A***
c. ***TIME-X***(User Time) is greater than ***TIME-A*** and ***TIME-B***
d. ***TIME-X***(User Time) is lesser than ***TIME-A***and ***TIME-B***
e. ***TIME-X***((User Time) falls between ***TIME-A***and ***TIME-B***

There are several functions and methods available in modern day languages to compare two timings, like ***DateTime.Compare*** method in visual studio.NET, which can be used to compare two **DateTime** instances. But, when there are cases, where more than two time comparisons are required, then it is inevitable to find a new logic or algorithms.

Even though there are multiple methodsand functions available to compare timings, we would like to explainour own time comparison algorithm in order to perform the task efficiently and quickly. This algorithm has been already tested in Dotnetenvironment. Also, this can be customized based on other IDE's or database applications.

2- SCENARIOS IN TIMETABLING

For instance, if we develop a project which schedules examinations for an educational institution, the programmer may come across many situations to compare times in his program. For better understanding, we will proceed with an example of examination scheduling.

Now, consider the case, there are five labs in an institution (as shown in *table1*)***2037, 2039,2013,2001,2002*** which can be used for scheduling the exams. These labs will be used for all departments' examination within the institution. Under this condition, the programmer needs to check the below scenarios in his program.

Scenario1

a. All labs should be utilized for ***COURSE- A*** if all the labs are free on ***DATE-A***
b. No labs should be allocated for ***COURSE-B*** on ***DATE-A,*** *since* the labs are already occupied by***COURSE- A on DATE-A***

Scenario2

a. If only three labs(***2037,2039,2013***) are utilized for ***COURSE-A*** on ***DATE-A,*** then, the remaining two labs (***2001, 2002***) should be available for ***COURSE-B*** on ***DATE-A.***

Scenario3

a. If all labs are utilized for ***COURSE-A*** in the time range ***TIME-A to TIME-B*** on ***DATE-A,*** then no labs should be allocated for ***COURSE-B*** in the time range ***TIME-A to TIME-B*** on ***DATE-A***
b. But all labs should be available for ***COURSE-B OR COURSE-C***from ***TIME-C*** onwards on ***DATE-A***

Scenario4

a. If only three labs(***2037,2039,2013***) are utilized for ***COURSE-A*** in the time range ***TIME-A to TIME-B*** on ***DATE-A,*** then the remaining two labs (***2001, 2002***) should be available for ***COURSE-BOR COURSE-C*** in the time range ***TIME-A to TIME-B*** on ***DATE-A***

Scenario5

a. All labs should be available for all courses, if the schedules are generated in different dates.
b. ***Example1:*** All labs should be available for ***COURSE-A***under any time rangeon***DATE-A***
c. ***Example2:*** All labs should be available for ***COURSE-B*** under any time rangeon***DATE-B***
d. ***Example3:*** All labs should be available for ***COURSE-C*** under any time rangeon***DATE-C***

From the *table1*, all labs are already allocated for the ***COURSE-BLUE*** on ***DATE-01/28/2016***in the time range ***08.00:00 to 09:20:00AM***

Table 1 Labs Allocation

Department	Course	Date	Day	From_Time	To_Time	LabsUsed
English	Blue	01/28/2016	Thursday	08:00:00	09:20:00	2037
English	Blue	01/28/2016	Thursday	08:00:00	09:20:00	2039
English	Blue	01/28/2016	Thursday	08:00:00	09:20:00	2013
English	Blue	01/28/2016	Thursday	08:00:00	09:20:00	2001
English	Blue	01/28/2016	Thursday	08:00:00	09:20:00	2002

In our application, we used a relational database and particularly we have taken the table "***LabsAllocation***" from it as shown in the *table1*for addressing the above scenarios. This table has seven attributes as below with 5 tuples.

- Department
- Course
- Date
- Day
- From_Time
- To_Time
- Labs Used

3- EXISTING METHODS

In this section, we will focus the methodswith outputs available for time comparison in Dotnet environment.

3.1- Method1: DateTime.Compare(DateTime, DateTime)

This method compares two instances of DateTime and returns an integer valuewhich indicates whether the first instance is ***earlier than, the same as, or later*** than the second instance.

Syntax (in c#)

Public static int **Compare**
(
DateTime t1, t2
)
t1- the first object to compare &***t2***- the second object to compare.

Table 2 Output of DateTime.Compare method

Value Type	*Condition*
Less than zero	t1 is earlier than t2
Zero	t1 is the same as t2
Greater than zero	t1 is later than t2

The above method allows comparing two DateTime instances and the requirement for comparing more than two DateTime instances cannot be satisfied by the method.

3.2- Method2: Timespan.Compare(TimeSpan, TimeSpan)

This method compares two TimeSpan values and returns an integer value which indicates whether the first value is ***shorter than, equal to, or longer*** than the second value.

Syntax (in c#)

Public static int Compare
(
TimeSpan t1, t2
)
t1- the first time interval to compare &***t2***- the second time interval to compare.

Table 3 Output of Timespan. Compare method

Value	*Description*
-1	t1 is shorter than t2
0	t1 is equal to t2
1	t1 is longer than t2

The above method also allows comparing two TimeSpan instances and the requirement for comparing more than two TimeSpan instances cannot be satisfied by the method.

3.3- Using Operators for time comparisons

Generally, while comparison, it is a normal practice to use operators like ***LessThan, LessThanOrEqual GreaterThan, GreaterThanOrEqual***etc. in SQL queries which may not produce the desired outputs all the times. In technical web blogs, we can be find more queries related to using operators for time comparisons which remains unanswered or answered with complex logics. It may be a difficult task for a basic programmer to understand the logics behind the posted solutions or implementing the solutions in his programs. For example, the *table4* is one of the solutions given in the blogs for selecting all dates between two dates.

Table 4 Selecting dates between two dates using SQL

```
Select * from table_name where col_Date between '2011/02/25' AND DATEADD (s,-1,DATEADD(d,1,'2011/02/27'))
```

In *table4*, the logic is to add a day to the current endDate, it will be *2011-02-28 00:00:00*, then subtract one second to make the endDate*2011-02-27 23:59:59*, so that all dates between the given intervals can be retrieved.

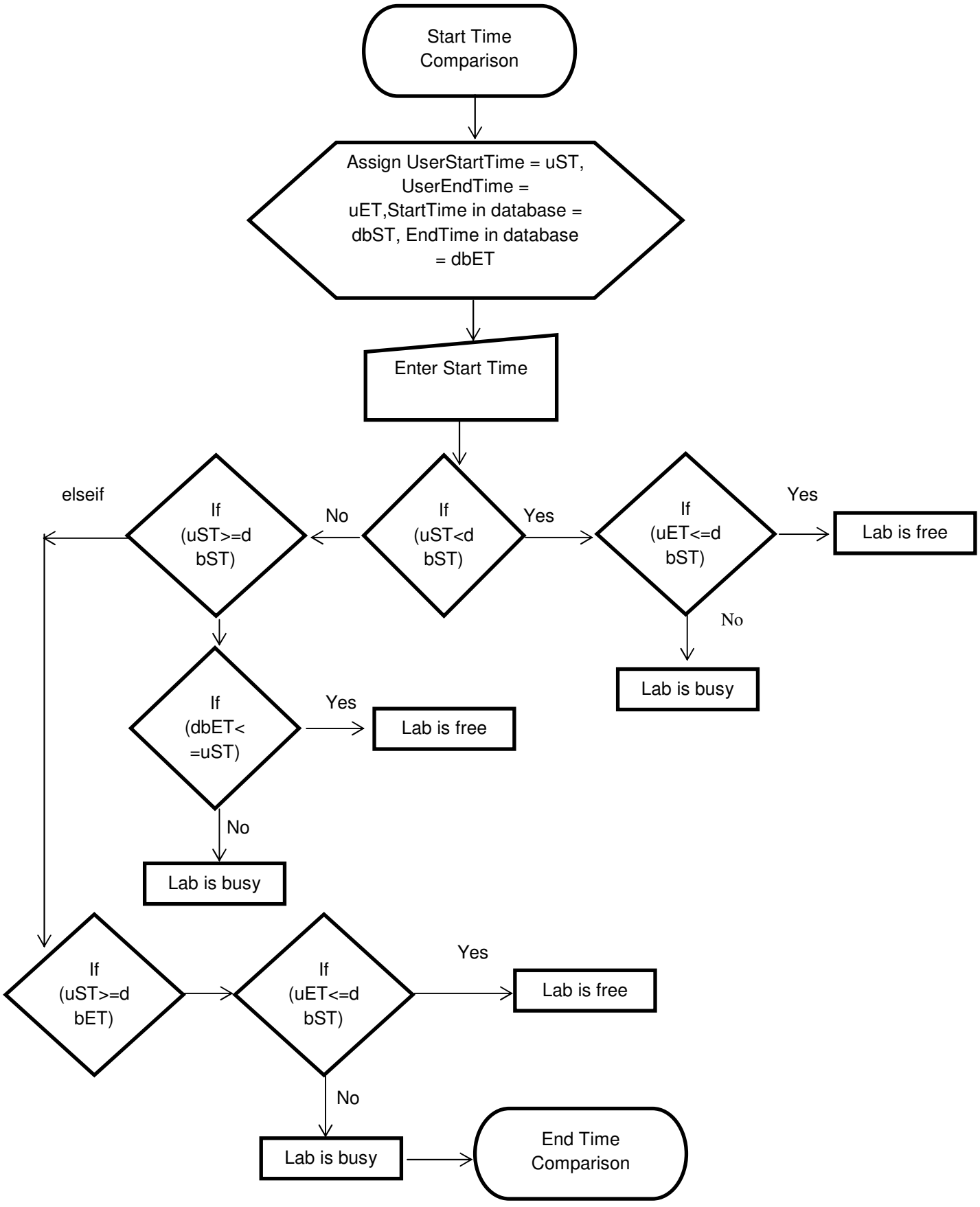
4- PROPOSED ALGORITHM

In the above section, we discussed about the different methods and its limitations for comparing two timings. In this section, we will brief the comparisons techniques of our proposed algorithm. Using this algorithm, we can compare up to four different timings as described in t*able1*.

Table 5 Time Value Descriptions

Serial No	Value Type	Descriptions
1	tsUserStartTime	This is the START-TIME from the user in which he needs to start the exam
2	tsUserEndTime	This is the END-TIME from the user in which he needs to end the exam
3	tsDBStartTime	This is the existing START-TIME available in database(in table Labs Allocation) which has been already allocated for another course
4	tsDBEndTime	This is the existing END-TIME available in database(in table Labs Allocation) which has been already allocated for another course

This algorithm can handle any number of labs and time slots if it is executed through a *for* loop statement. The below section shows the flowchart for the proposed algorithm.


Start Time Comparison
Assign UserStartTime = uST, UserEndTime = uET,StartTime in database = dbST, EndTime in database = dbET
Enter Start Time
If (uST<d bST)
No
elseif
If (uST>=d bST)
Yes
If (uET<=d bST)
Yes
Lab is free
No
Lab is busy
If (dbET< =uST)
Yes
Lab is free
No
Lab is busy
If (uST>=d bET)
If (uET<=d bST)
Yes
Lab is free
No
Lab is busy
End Time Comparison

```
if (tsUserStartTime < tsDBStartTime)
{
	if (tsUserEndTime <= tsDBStartTime)
	{
		//Lab is free
		  //Change Lab status to AVAILABLE

	}
	else if (tsUserEndTime > tsDBStartTime)
	{
	    //Lab is busy
		//Change Lab status to NOT AVAILABLE
	}
}
else if(tsUserStartTime >= tsDBStartTime)
{
	if(tsDBEndTime <= tsUserStartTime)
	{
		//Lab is free
		//Change Lab status to AVAILABLE

	}
	else if(tsDBEndTime > tsUserStartTime)
	{
		//Lab is busy
		//Change Lab status to NOT AVAILABLE
	}
}
else if(tsUserStartTime >= tsDBEndTime)
{
	if(tsUserEndTime <= tsDBStartTime)
	{
		//Lab is free
		//Change Lab status to AVAILABLE

	}
	else if(tsUserEndTime >= tsDBStartTime)
	{
		//Lab is busy
		//Change Lab status to NOT AVAILABLE

	}
}
```

5- RESULTS AND DISCUSSION

In this section, we will consider some scenarios discussed in section 3 and compare the outputs by applying the methods available in Dotnet environment and our time algorithm.

From the *table1*, we can understand that all labs are already allocated for the ***COURSE-BLUE*** on ***DATE-01/28/2016***in the time range ***08.00:00 to 09:20:00AM.***

Now, the user wants to create schedule for another ***COURSE-GREEN***on same ***DATE-01/28/2016***in the time range ***08.15:00 to 09:15:00AM***

Now assign the variables as below

Table 6 Start Time and End Time from user

Variable Name	Value
tsDBStartTime	08.00:00
tsDBEndTime	09:20:00
tsUserStartTime	08.15:00
tsUserEndTime	09:15:00

Since, the labs are already occupied, the software should not allow the schedule generation and stop the process by displaying the message "***Labs are busy. Choose another time***" to the user.

If we apply the ***Timespan.Compare(TimeSpan, TimeSpan)***method for this scenario, wehave to write the code as below

Timespan.Compare(tsUserStartTime, tsDBStartTime**) = *Timespan.Compare(***08.15:00, 08.00:00**)**

Timespan.Compare(tsUserEndTime, tsDBEndTime**) = *Timespan.Compare(***09.15:00, 09.20:00**)**

The first comparison will return an integer value 1, because the tsUserStartTime is longer than tsDBStartTime and the second comparison will return an integer value -1, because the tsUserEndTime is shorter than tsDBEndTime. Since, we can pass only two parameters to this ***Timespan.Compare***method, we have to perform the comparison twice and with the returned values (1 & -1) around the neck logics should be written to achieve the required output.

Now, apply the time comparison algorithm for this scenario. As it is mentioned earlier, pass the required time slots (as in *table6*) as a parameter for the labs (***2037, 2039, 2013, 2001, 2002***) through a*for* loop to this algorithm.

```
If (08.15:00<08.00:00)  ──────────►  Condition False
{
        If (09:15:00<=08.00:00)
        {       }
        Else if (09:15:00 > 08.00:00)
        {       }
}
Else if (08.15:00 >= 08.00:00)  ──────────►  Condition True
{
        If (09:20:00 <= 08.15:00)
        {       }  ──────────►  Condition Fails
        Else if (09:20:00 > 08.15:00)
        {
                Labs are busy. Choose another time  ──────────►  Condition True
        }
}
Else if (08.15:00 >= 09:20:00)
{
        If (09:15:00 <= 08.00:00)
        {       }
        Else if (09:15:00 >= 08.00:00)
        {       }
}
```

The above algorithm will check the availability of each lab for the time slots given by the user.

6- CONCLUSIONS

Thus, in this paper we have discussed about the time comparison part, the core of any type of scheduling and proposed a method for comparison. All the scenarios discussed above in section3and other scenarios if any can be satisfied with the help of this algorithm. There are methods available in programming languages to compare two DateTime or time span values. But, through this algorithm we can perform comparison up to four timings for any number of labs and time slots.

7- REFERENCES

[1] Liam T.G.Merlot, Natashia Boland, Barry D. Hughes, Peter J. Stuckey,"A Hybrid Algorithm for the Examination Timetabling Problem", 2740,pp 207-231

[2] Ben Paechter, R.C. Rankin, Andrew Cumming, Terence C. Fogarty,"Timetabling the classes of an entire university with an evolutionary algorithm",1498,pp 865-874

[3] N Balakrishnan,"Examination scheduling: A computerized application", 19.2003

[4] E. K. Burke, J. P. Newall, R. F. Weare,"A memetic algorithm for university exam timetabling", 1153,pp 241-250

[5] E. Burke, K. Jackson, J.H. Kingston, R. Weare3, "Automated University Timetabling: The State of the Art", 40(9)

[6] Philipp Kostuch,"The University Course Timetabling Problem with a Three-Phase Approach", 3616, pp 109-125.