# Time complexity Analysis

**Algorithm:**

//Maximum number of subjects a class can have

Maximum **M\_SUBJECT** 8

//Data structure to store details about subject

Structure **SUBJECT**

{

SUBNAME

CHRS

SUBTYPE

}

As it is declarative statement so it will take time

**T(n) = θ(1)**

//data structure to store number of lectures in a day

Structure **TIMETABLE**

{

T\_ID

LEC1

LEC2

LEC3

LEC4

TBREAK

LEC5

LEC6

LEC7

}

As it is declarative statement so it will take time

**T(n) = θ(1)**

//function to allocate slot  **COST**

**SLOT\_ALLOCATION (SUBJECT S[ ], TIMETABLE T[ ] , M\_SUBJECT)** θ( n )

1 Let **WEEKDAY** = 1 θ(1)

2 For i=1 to M\_SUBJECT θ( n )

3 If ( i=1 | | i =3 || i =5 || i =7 ) θ( n - 1)

4 if (S[ i ].SUBTYPE == "Lab" || S[ i ].SUBTYPE == "lab" || S[ i ].SUBTYPE == "l") θ( n - 1)

5 T[ WEEKDAY ].LEC1=S[ i ].SUBNAME θ( n - 1)

6 T[ WEEKDAY ].LEC2=S[ i ].SUBNAME θ( n - 1)

7 T[ WEEKDAY ].LEC3=S[ i ].SUBNAME θ( n - 1)

8 else if(S[ i ].SUBTYPE =="Theory" || S[ i ].SUBTYPE == "theory" || S[ i ].SUBTYPE == "th")

θ( n - 1)

9 if(S[ i ].CHRS=="1") θ( n - 1)

10 T[ WEEKDAY ].LEC1=S[ i ].SUBNAME θ( n - 1)

11 T[ WEEKDAY ].LEC2= “ – ” θ( n - 1)

12 T[ WEEKDAY ].LEC3= “ – ” θ( n - 1)

13 else if(S[ i ].CHRS =="2") θ( n - 1)

14 T[ WEEKDAY ].LEC1=S[ i ].SUBNAME θ( n - 1)

15 T[ WEEKDAY ].LEC2=S[ i ].SUBNAME θ( n - 1)

16 T[ WEEKDAY ].LEC3= “ – ” θ( n - 1)

17 else if(S[ i ].CHRS =="3") θ( n - 1)

18 T[ WEEKDAY ].LEC1=S[ i ].SUBNAME θ( n - 1)

19 T[ WEEKDAY ].LEC2=S[ i ].SUBNAME θ( n - 1)

20 T[ WEEKDAY ].LEC3=S[ i ].SUBNAME θ( n - 1)

21 else θ( n - 1)

22 T[ WEEKDAY ].LEC1= “ – ” θ( n - 1)

23 T[ WEEKDAY ].LEC2= “ – ” θ( n - 1)

24 T[ WEEKDAY ].LEC3= “ – ” θ( n - 1)

25 If ( i =2 || i =4 || i =6 || i =8) θ( n - 1)

26 if (S[ i ].SUBTYPE == "Lab" || S[ i ].SUBTYPE == "lab" || S[ i ].SUBTYPE == "l") θ( n - 1)

27 T[ WEEKDAY ].LEC4= “ – ” θ( n - 1)

28 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

29 T[ WEEKDAY ].LEC5=S[ i ].SUBNAME θ( n - 1)

30 T[ WEEKDAY ].LEC6=S[ i ].SUBNAME θ( n - 1)

31 T[ WEEKDAY ].LEC7=S[ i ].SUBNAME θ( n - 1)

32 else if((S[ i ].SUBTYPE == "Theory" || S[ i ].SUBTYPE == "theory" || S[ i ].SUBTYPE == "th") &&

(S[ i – 1 ].SUBTYPE == "Theory" || S[ i – 1 ].SUBTYPE == "theory" || S[ i – 1 ].SUBTYPE == "th") &&

(S[ i - 1 ].CHRS || S[ i – 1 ].CHRS == "2" || S[ i – 1 ].CHRS == "3")) θ( n - 1)

33 if (S[ i ].CHRS == "1") θ( n - 1)

34 T[ WEEKDAY ].LEC4=S[ i ].SUBNAME θ( n - 1)

35 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

36 T[ WEEKDAY ].LEC5= “ – ” θ( n - 1)

37 T[ WEEKDAY ].LEC6= “ – ” θ( n - 1)

38 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

39 else if (S[ i ].CHRS == "2") θ( n - 1)

40 T[ WEEKDAY ].LEC4=S[ i ].SUBNAME θ( n - 1)

41 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

42 T[ WEEKDAY ].LEC5= S[ i ].SUBNAME θ( n - 1)

43 T[ WEEKDAY ].LEC6= “ – ” θ( n - 1)

44 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

45 else if (S[ i ].CHRS == "3") θ( n - 1)

46 T[ WEEKDAY ].LEC4=S[ i ].SUBNAME θ( n - 1)

47 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

48 T[ WEEKDAY ].LEC5= S[ i ].SUBNAME θ( n - 1)

49 T[ WEEKDAY ].LEC6= S[ i ].SUBNAME θ( n - 1)

50 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

51 else if ((S[ i ].SUBTYPE == "Theory" || S[ i ].SUBTYPE == "theory" || S[ i ].SUBTYPE == "th") &&

(S[ i - 1 ].SUBTYPE == "Lab" || S[ i – 1 ].SUBTYPE == "lab" || S[ i – 1 ].SUBTYPE == "l")) θ( n - 1)

52 if (S[ i ].CHRS == "1") θ( n - 1)

53 T[ WEEKDAY ].LEC4=S[ i ].SUBNAME θ( n - 1)

54 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

55 T[ WEEKDAY ].LEC5= “ – ” θ( n - 1)

56 T[ WEEKDAY ].LEC6= “ – ” θ( n - 1)

57 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

58 else if (S[ i ].CHRS == "2") θ( n - 1)

59 T[ WEEKDAY ].LEC4=S[ i ].SUBNAME θ( n - 1)

60 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

61 T[ WEEKDAY ].LEC5= S[ i ].SUBNAME θ( n - 1)

62 T[ WEEKDAY ].LEC6= “ – ” θ( n - 1)

63 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

64 else if (S[ i ].CHRS == "3") θ( n - 1)

65 T[ WEEKDAY ].LEC4=S[ i ].SUBNAME θ( n - 1)

66 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

67 T[ WEEKDAY ].LEC5= S[ i ].SUBNAME θ( n - 1)

68 T[ WEEKDAY ].LEC6= S[ i ].SUBNAME θ( n - 1)

69 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

70 Else θ( n - 1)

71 T[ WEEKDAY ].LEC4=“ – ” θ( n - 1)

72 T[ WEEKDAY ].TBREAK = “BREAK” θ( n - 1)

73 T[ WEEKDAY ].LEC5= “ – ” θ( n - 1)

74 T[ WEEKDAY ].LEC6= “ – ” θ( n - 1)

75 T[ WEEKDAY ].LEC7= “ – ” θ( n - 1)

76 WEEKDAY = WEEKDAY+ 1 θ( n - 1)

As we ignore the smaller terms so the time complexity is as,

**T(n) = θ(n)**

As **SLOT\_ALLOCATION** is our main function and it uses SUBJECT and TIMETABLE data structure. Moreover, subject type is the prime part of subject and it decides the slot allocation of subject in the table which is crucial for the whole algorithm and our algorithm takes **θ(n).** Our algorithm is a linear time algorithm. Time complexity of our algorithm is efficient and it seems to be an optimize algorithm and its time complexity is less.

**Correctness of algorithm**

**Introduction:**

The algorithm takes teachers list, classroom, subjects, department name, number of lectures in a week along with their contact hours as input. The method followed is that the algorithm will check the contact hours of subject. It will check its subject type that either it is a theory or lab subject. Keeping in the view the contact hours, it will generate a timetable that will compensate all the subjects considering their contact hours in minimum days.

**Correctness of algorithm**

If zero activities are added to algorithm it will generate an optimal solution. After adding desired values in the required files, the algorithm will generate a timetable that will deal with subject contact hours and will create a timetable that will manage all subjects in minimum days.

If we generate a k timetable with input values that is the best solution, then the next one k+1 generated will also be the best one.

The time complexity of function; SLOT\_ALLOCATION is Ѳ(n) the declarative statements will take Ѳ(1) time. So, the overall time complexity calculated is Ѳ(n). This is a linear time complexity. It seems an efficient and good solution.

**Conclusion:**

Our algorithms complexity turns out to be **Ѳ(n).** The computational effort of algorithm also deals with the performance of algorithm.

The constraints and number of inputs as well as the structure of optimization algorithm are related to the performance. The algorithm will provide the optimal or best solution. We never rule the solution. We got some solutions at the end of algorithm and we expect them to be the best solution to our problem. Our algorithm has Ѳ(n) cost and we hope that it will give the best solution.