I.T.U.

Faculty of Computer and Informatics

Computer Engineering

Lesson name: Analysis of Algorithms

Lesson Code: BLG 372E

Name Surname: Abdullah AYDEĞER

Number: 040090533

Instructor's Name: Zehra ÇATALTEPE

Assistant's Name: Ahmet Aycan ATAK

Due Date: 27.02.2012

Problem # 1

```
Create an empty DAYS list                  //For holding days
Create an array C[n] which shows di        //Only for holding # of students that class can contain
Create an array L[m] which shows ki        //Only for holding # of students which take the lecture
Add one day into DAYS                      //That should be Monday

For ( i ← 1; i <= m; i++ ) {               //For all lecture, loop will terminates
        Sum ← 0                            //Initialize variable for holding sum of quota
        IsAssigned ← FALSE                 //Initialize boolean variable for knowing lecture has
                                           // assigned any  classes or not
        DAYS ← DAYS.begin                  //while loop should starts in first day

        While ( DAYS & IsAssigned is FALSE ) {     //while all days are not tried and still
                                                   //lecture has not assigned any classes yet
                For ( j ← 1; j <= n & IsAssigned is FALSE; j++ ) {   //For all classes are tried
                                                   //and still lecture can not be assigned any of those
                        If ( C[j] is not used ) {  //Of course must see for which day the loop in
                                Sum += C[j].di     //If class is not used before, make it used
                                Make C[j] is 'used'   // and add its quota to sum
                        }
                        If ( Sum >= L[i].ki )      //If sum exceeds necessary number
                                IsAssigned ← TRUE  //Then lecture is assigned to those classes
                }

                If ( IsAssigned is FALSE )         //If still lecture has not assigned any classes
                        Make C[j]'s are not 'used' which were made 'used' just before //Make
                                                   //free back classes
                DAYS ← DAYS.next                   //Try for next day..
        } //end of while

        If ( IsAssigned is FALSE ) {               //If all days have been tried and still lecture is not assigned

                Add new day into DAYS              //New day are required
                Sum ← 0

                For ( j ← 0; j <= n & IsAssigned is FALSE; j++) {
                        Sum += C[j].di             //Classes will be used for the new day
                        Make C[j] is used
                        If ( Sum >= L[i].ki )
                                IsAssigned ← TRUE
                }
        } //end of If
} //end of for
```

Problem # 2

Expression of my algorithm:

In my algorithm;

- Firstly I take some memories.
- Give first day into to DAYS list.
- Let's terminates loop for # of lecture.
- In for loop, algorithm has while loop first.
- This while loop will terminate when lecture is ok(assigned) or DAYS list comes to the end.
- In this while loop, algorithm tries all classes in same day which are not used before for current lecture, and adds quotas to a 'sum' variable.
- If 'sum' variable exceeds from necessary number(# of students in the current lecture), these chosen classes are assigned to this lecture.
- Else, all classes are tried in this day and no good news are occurred, the taken classes are back to the free.
- And still lecture hasn't got any classes after trying all classes in all days, create a new day and use the classes(all of free for new day) for the current lecture.

Weak points of my algorithm:

Of course my algorithm is not perfect. Before the saying weak points, I must say that in my algorithm DAYS list is including(and also representing) sections, not real days. For an example, DAYS should be include Wednesday-Morning or Friday-Afternoon. Therefore as seeing my algorithm, you must be aware of this first. Later I can sort my weak points as follows;

- My algorithm works as first lecture in the L[m] array will take classes first, and the others as according to their sequence in the array. Therefore fragmentation can be maximum as possible. Since fragmentation is not important in algorithm. This causes some classes are free for some sections.
- Algorithm is a bit slow and has a high time complexity.
- Some students' final exams with different lecture can take place in same section.
- First classes in the C[n] are always full, and last classes are generally empty for all sections.
- All lectures want to first day section in first place and second in the second section as follows.

Problem # 3

Algorithm

```
Create an empty DAYS list                 //For determining current day
Create an array C[n] which shows di       //Only for holding # of students that class can contain
Create an array L[m] which shows ki       //Only for holding # of student which takes the lecture
Add one day into DAYS                     //That should be Monday

Calculate all permutations PMT of 1 to m ( # of lectures)

While ( All PMT is not tried ) {

    For ( All Lectures  L [ i ]  ) {      // i is changing between current PMT
                                          // elements respectively
        Sum ← 0                           //Variable for holding sum of quota
        IsAssigned ← FALSE                //Boolean variable for knowing lecture has
                                          // assigned any  classes or not
        DAYS ← DAYS.begin                 //while loop should starts in first day

        While (  DAYS  &  IsAssigned is FALSE ) {     //while all days are not tried and
                                                      still   //lecture has not assigned
                                                      any classes yet
            For (  j ← 1;   j <= n & IsAssigned is FALSE;   j++ ) {    //For all classes are
                                //tried  and still lecture can not be assigned any of those
                If ( C[j] is not used ) {   //Of course must see for which day the
                                            // loop in
                    sum += C[j].di      //If class is not used before, make it
                    Make C[j] is used       // used and add its' quota to sum
                }
                If ( sum >= L[i].ki )       //If sum exceeds necessary number
                    IsAssigned ← TRUE       //Then lecture is assigned to those
                                            // classes
            }

            If ( IsAssigned is FALSE )    //If still lecture has not assigned any classes
                Make C[j]'s are not used which were made 'used' just before
                    //Make free back classes
            DAYS ← DAYS.next      //Try for next day..
        } //end of while
```

```
            If ( IsAssigned is FALSE ) {   //If all days have been tried and still lecture is not
                                              assigned

                Add new day into DAYS        //New day are required
                Sum ← 0


                For (  j ← 0; j <= n & IsAssigned is FALSE; j++) {
                    sum += C[j].di           //Class will be used for the new day
                    Make C[j] is used
                    If ( sum >= L[i].ki )
                        IsAssigned ← TRUE
                }
            } //end of If
        } //end of for-L[ i]

        Calculate Frag [ i]

} //end of while-PMT
```

Determine minimum fragmentation value of Frag [ ] and use this PMT value for For loop(L[i]).

Weak Points

I develop my algorithm, that is used in the first problem, for this fragmentation problem. I try my lecture in all possible permutation respectively (that's why I used while loop for PMT). But in this situation, I've got more weak points. I can say weak points as:

- Algorithm can work very slowly because of the loops in the loops which are also in the loops.