Timothee Guerin
260447866

Comp 251 - Assignment 5

Question 1

We can use a bipartite graph with two group: the time and the classes then using the FordFulkerson algorithm we can get the mapping of classes to their time.

```
/**
*       Question 1
*/
ComputeExamSchedule(D,R,C) {
        //Create graph and flow
        Graph G=createGraph(D,R,C);
        int maxFlow=FordFulkerson(G);
        //recover graph G' with weights that indicate flow.

        for all c in class
                list<nodes> incoming= new list;
                Fill incoming with all nodes that have an edge from time t to class c with flow of 1;
                for all node x in incoming
                        schedule temporary=new schedule (class c at time t);
                        add temporary to calendar;

        return calendar;


        return calendar;//list that will give day, time and room associated to each class
}
createGraph(D,R,C)
        G = new Graph();
        Node start,finish = new nodes in G;
        Set Times, Classes, Students;

        //Add all classes
        for all class c in C
                Node n = new node c in G;
                Add node to a set Y;
                add edge from n to finish with weight 1;

        //Add the time nodes(3 per days)
        for all day d in D
                Node n1,n2,n3 = create new Node in G.
                Add these node to set Times;
                create edge from start to n1, n2, n3 with weight |R|;

        //Map all time to all students
        for each node t in Times
                for(each node c in class)
                        create edge from t to s with weight 1;

        return G;

schedule{
        string class;
        string time;
        string room;
        Constructor that takes four strings as arguments and sets them;
}
```

Timothee Guerin
260447866

Question 2:

We can expend the graph by adding a third group: student. All student are linked to all time but only to the classes they are taking. Then this will prevent student to have twice the same time as well as having multiple time for classes(weight is maximum 1)

```
/**
*       Question 2
*/
ComputeExamSchedule(D,R,C) {
        //Create graph and flow
        Graph G=createGraph(D,R,C);
        int maxFlow=FordFulkerson(G);
        //recover graph G' with weights that indicate flow.

        /*At index i of following array, we will have the number of rooms
         * we have already assigned at timeslot i. The loop below will
         * give room 0 to the first course that we found that was
         * associated with the time slot and then increment that position
         * of the list like that the second course associated with this
         * time slot will get room 1 and so on.
         */
        list<integer> roomindex = new list of size |3D|;
        list<schedule> calendar = new list of size |C|;
        fill roomindex with 0s;
        for all s in student
                list<nodes> outcoming= new list;
                list<nodes> incoming= new list;
                Fill incoming with all nodes that have an edge to a student s with flow of 1;
                Fill outcoming with all nodes that have an edge from a student s with flow of 1;
                for all node x in incoming
                        schedule temporary=new schedule (class associated to first node in outgoing, time
associated to node x, room #roomindex[x]);
                        add temporary to calendar;

        return calendar;


        return calendar;//list that will give day, time and room associated to each class
}
createGraph(D,R,C)
        G = new Graph();
        Node start,finish = new nodes in G;
        Set Times, Classes, Students;

        //Add all classes
        for all class c in C
                Node n = new node c in G;
                Add node to a set Y;
                add edge from n to finish with weight 1;

        //Add the time nodes(3 per days)
        for all day d in D
                Node n1,n2,n3 = create new Node in G.
                Add these node to set Times;
                create edge from start to n1, n2, n3 with weight |R|;

        //Map all student to the courses he is taking
        for each s in student
                Node s = create new node in G;
                add s node to set Students;
                for each class c in student s )
                        create egde from s to c with weight 1;

        //Map all time to all students
        for each node t in Times
                for(each node s in Students)
                        create edge from t to s with weight 1;

        return G;

schedule{
        string class;
        string time;
        string room;
        Constructor that takes four strings as arguments and sets them;
}
```

Timothee Guerin
260447866

Question 3

Question 3.a

Variable:

- n: Number of courses
- m: Number of days
- r: Number of rooms

Edges:

1. $3m$ edges between start and time
2. $3m * s$ edges between time and students
3. $k * s$ edges between students and class(k number of class per student)
4. n edges between class and finish

Then we have $3m + 3m * s + k * s + n$ edges

Question 3.b

Algorithm

1) Create graph
    1. Add n node(Courses): $O(n)$
    2. Add 3m node(Days): $O(3m)$
    3. Add s node(Students): $O(s)$
    4. Add $3m * s$ edges between time and students: $O(3m * s)$
    5. Add $k * s$ edges between students and class: $O(s * k)$
2) Find
    1. Loop for all students: $O(s)$
        a. Loop for all incoming nodes: O(k)

Then we have $n + 3m + s + 3m * s + s * k + s * k = 3m(1 + s) + n + s(1 + 2k)$

$$O(3m * s + n + s)$$

Question 4

Lower bound:

$$\min = \frac{n}{3 * r}$$

Upper bound:

$$\max = \frac{n}{3}$$

Range

$$range = \max - min$$
$$range = \frac{n(3r - 1)}{3r}$$

+We can do a loop

```
int days = min;
while(notfound)
      D = init with days;
      ComputeExamSchedule(D,R,C)
      days++;
```

Complexity:

Timothee Guerin
260447866

1. FFalgorithm: $n * edges = n * (3m + 3m * s + k * s + n)$

$$O(n * (3m + 3m * s + s + n))$$

2. ComputeExamSchedule:

$$O(n * (3m + 3m * s + s + n) + 3m * s + n + s)$$

3. Algorithm :

$$O(\frac{n(3r-1)}{3r} * n * (3m + 3m * s + s + n) + 3m * s + n + s))$$

$$O(n * n^2)$$

$$O(n^3)$$