University of London International Programmes
Computing coursework assignments 2014-15

CO3352 Operations research and combinatorial Optimisation
Coursework assignment 1

This coursework assignment is designed to help you enrich your learning experience and to encourage self-study and creativity. Assumptions may be added if necessary in coursework answers to simplify programming tasks or help understand issues. You should, however, attempt the exercises in the subject guide before doing the coursework. Otherwise, you may find the tasks in the coursework assignment difficult and the experience less rewarding. You should read the assignments or questions carefully and pay particular attention to the Submission Requirements on page 3. Note: program source codes may be checked using plagiarism-indicating programs. The available marks are given in square brackets.

1. Let $E = (a, b, c, d)$ and $\boldsymbol{I} = (\emptyset, (a), (b), (c), (d), (a, b), (a, c), (a, d), (b, c), (b, d), (c, d), (a, b, c))$. Explain why $(E, \boldsymbol{I})$ is not a matroid. [10%]

2. Consider the incidence matrix A of graph G below.

A:

	1	2	3	4	5	6	7
a	1	0	0	1	0	0	1
b	1	1	1	0	0	0	0
c	0	1	0	1	1	1	0
d	0	0	1	0	1	1	0

   (a) Draw the corresponding graph $G(V, E)$ with labels $V = (a, b, c, d)$, and $E = (1, 2, 3, 4, 5, 6, 7)$. [10%]
   (b) Write all linearly independent subsets $\boldsymbol{I}$ of E. [10%]

3. Given a graph G with a weight function w on its edges, the Kruskal's algorithm outlined below aims to find a minimum-weight spanning tree S.

---

i) Initialize S: $S \leftarrow \emptyset$.
ii) Choose edge e_i of a minimal weight.
iii) If $S \cup e_i$ contains no cycle, then set $S \leftarrow S \cup e_i$, else remove e_i from consideration, and repeat previous steps from ii).
iv) The greedy algorithm concludes, returning a minimum-weight spanning tree S.

   (a) Demonstrate how the greedy algorithm can be recognised as a matroid optimisation technique. [20%]
   (b) (Programming task) Implement the Kruskal's algorithm in Java (or MATLAB). [50%]

[END OF COURSEWORK ASSIGNMENT 1]

University of London International Programmes
Computing coursework assignments 2014-15

CO3352 Operations research and combinatorial Optimisation
Coursework assignment 2

This coursework assignment is designed to help you enrich your learning experience and to encourage self-study and creativity. Assumptions may be added if necessary in coursework answers to simplify programming tasks or help understand issues. You should, however, attempt the exercises in the subject guide before doing the coursework. Otherwise, you may find the tasks in the coursework assignment difficult and the experience less rewarding. You should read the assignments or questions carefully and pay particular attention to the Submission Requirements on page 3. Note: program source codes may be checked using plagiarism-indicating programs. The available marks are given in square brackets.

FirstJob is a job agency that helps people find their professional jobs. Essentially three types of data are maintained in the FirstJob: *Jobs* currently advertised, *Candidates* available for certain jobs and *Salaries* for specific candidates at certain jobs. In mathematical terms, we have $Jobs = (j_1, j_2, \cdots, j_n)$, $Candidates = (c_1, c_2, \cdots, c_m)$, $Salary = (s_{11}, s_{12}, \cdots, s_{1n}; s_{21}, s_{22}, \cdots, s_{2n}; \cdots; s_{m1}, s_{m2}, \cdots, s_{mn})$, where m and n are positive integers.
Typically the advertised annual salary is a range of figures such as 15K–50K pounds to attract candidates with various skills and experiences. Hence, two candidates may expect different salaries from an employer. To simplify the model, we assume the following (you may add more assumptions if necessary): (1) It is more important for an employer to find a skilful candidate than pay a lower salary. (2) It is important for a candidate to get a highly paid job. (3) According to skills and experience, FirstJob assigns an agreeable salary value to each candidate for each interested job. (4) FirstJob aims to match each candidate with only one available job, and to match as many candidates as possible. (5) FirstJob charges each successful candidate a one-off agency fees of 2% of the candidate's first annual salary. The table below shows an example of four jobs (see the first row) and five candidates (see the first column). The integers represent the expected salary figures in thousand pounds, and each empty entry represents a *null* relationship between a candidate and job.

	Teacher	Programmer	Secretary	Technician
Anna		15	20	15
Bob	10	25		15
Carol	20	20	10	15
David		50		
Elena	15	20		10

1. Formulate an OR problem for the instance above and write a problem statement for the manager of FirstJob. What is to be minimised or maximised? [10%]
2. Demonstrate how the partition-matroid approach in Example 141 (page 58, the Subject Guide) can be applied to solve the instance of the OR problem in the above example. [20%]
3. (Programming task) Develop a program in Java (or MATLAB) to fulfil the following tasks:
   (a) Display a Candidates-Jobs bipartite graph given a Candidates-Jobs matrix. [20%]
   (b) Display two partition matroids $M(Jobs)$ and $M(Candidates)$ given a list of Candidates and Jobs. [20%]
   (c) Implement Algorithm 139 (Binet-Cauchy Intersection) (page 56, the Subject Guide). (Hint: Attempt first Exercise 143, page 58, the Subject Guide.) [30%]

[END OF COURSEWORK ASSIGNMENT 2]

Submission requirements

These requirements apply to both courseworks.

1. Your coursework submission must include a report Document and the program Code.
   The Document (preferable in .pdf format) should include the following sections:
   (a) Algorithms (in flow-chart)
   (b) Design (in block diagram or class-diagram in UML)
   (c) Demonstration (in 5 best screen-shots)
   (d) Discussion (including answers to any questions/problems in the Assignment, your experience in attempt of the coursework, and full bibliography)

   The program code should include the
   (a) Java source codes .java
   (b) executable version .class.
2. Execution of your programs:
   [Penalty] A ZERO mark may be awarded if
   - your program(s) cannot be run from the coursework directory by a simple command '*java menu*' (this means that you should name your main class 'menu', or adopt the `menu.java` that can be found in the Appendix on page 4);
   - your source code(s) does not compile and you give no information on your program execution environment;
   - your program(s) does not do what you claim it should do;
   - your program(s) crashes within the first *three* interactive execution steps;
   - your program(s) works for the first time of execution only;
   - there is no comment in your source code.
3. You should monitor and report the time you have spent for each part of the coursework, and leave a note to the marker if you need to raise any issue at the beginning of your coursework as follows:

Total Number of Hours Spent	
Hours Spent for Algorithm Design	
Hours Spent for Programming	
Hours Spent for Writing Report	
Hours Spent for Testing	
Note for the marker (if any):	

4. Show *all* your work. Any use of others' work should be declared at the point of use and referred to the *Bibliography* section at the end of your coursework.

Appendix

```
import java.lang.*;
import java.io.*;
// Modify the display content to suit your purposes...
class menu {
private static final String TITLE =
"\n2910325 Data Compression coursework\n"+
"   by firstname-FAMILYNAME_SRN\n\n"+
"\t********************\n"+
"\t1. Declaration: Sorry but part of the program was copied
from the Internet! \n" +
"\t2. Question 2 \n"+
"\t3. Question 3 \n"+
"\t4. no attempt \n"+
"\t0. Exit \n"+
"\t********************\n"+
"Please input a single digit (0-4):\n";
menu() {
int selected=-1;
while (selected!=0) {
System.out.println(TITLE);
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
// selected = Integer.parseInt(in.readLine());
try {
                        selected = Integer.parseInt(in.readLine());

                            switch(selected) {
                                 case 1:  q1();
                                   break;
                                 case 2:  q2();
                                   break;
                                 case 3:  q3();
                                   break;
                                 case 4:  q4();
                                   break;}      }
   catch(Exception ex) {}  } // end while
               System.out.println("Bye!");
}
// Modify the types of the methods to suit your purposes...
private void q1() {
        System.out.println("in q1");
}
private void q2() {
System.out.println("in q2");
}
private int q3() {
System.out.println("in q3");
return 1;
}
private boolean q4() {
System.out.println("in q4");
return true;
}
   public static void main(String[] args) {
new menu();
   }
}
```

[END OF SUBMISSION REQUIREMENTS]