

**Assignment 4 Cover Page**

Due: Monday, April 5 at midnight

Please read “How to Prepare Assignment Reports” from the course web page.

Please use this page as the cover page for your assignment.

Due to the time constraints, you are required to submit solutions for only 2 of the 4 problems. However, since all 4 problems relate to material that may be on the final exam, you are strongly encouraged to attempt the other 2 problems as well (but please do not submit your solutions). A solution key will be posted on the website shortly after the due date.

Family Name:

Given Name:

Student #:

Email:

Section (M, N or P):

Family Name:

Given Name:

Student #:

Email:

Section (M, N or P):

Please indicate below the two problems you would like marked:

Problem	Marking Scheme	Score
1	20 marks	
2	20 = 10 + 10 marks	
3	20 = 10 + 10 marks	
4	20 = 10 + 10 marks	
Total	40 marks	

### Assignment 4

Due: Monday, April 5 at 12:00 noon

Please read “How to Prepare Assignment Reports” from the course web page. You are required to do only 2 of the 4 problems assigned. However, since all 4 problems relate to material that may be on the final exam, you are strongly encouraged to attempt the other 2 problems as well (but please do not submit your solutions). A solution key will be posted on the website shortly after the due date.

[20] 1. **BFS: partitioning**

A group of people is planning an expedition. For safety, they decide to split into two groups; it does not matter how many people are in each group. Not all of them are good friends, and one reason for splitting is to separate all pairs that are not friendly to each other. Your goal is to design an algorithm that, given the map of friendship relation between people in the group, would find a partition of people into two groups of friends, or say that it is impossible.

Assume that the input is an  $n \times n$  matrix  $M$ , where  $n$  is the number of people in the group and  $M(i, j) = 1$  if  $i$  and  $j$  are friends, and 0 otherwise. Also, if  $i$  is friendly to  $j$ , then  $j$  is friendly to  $i$ . Your output should be the sequence of numbers corresponding to people assigned to the first group. **Hint:** Consider a graph in which edges represent non-friendly relations.

[10] (a) Provide pseudo-code for an algorithm that solves this problem.

[5] (b) Provide a brief argument for why the algorithm finds a correct solution.

[5] (c) Provide tight bounds on best- and worst-case running times.

[20] 2. **Fixed edge on a shortest path**

Given a directed, weighted graph  $G$  with non-negative weights, vertices  $s, t$  and an edge  $(u, v)$  in  $G$ :

[10] (a) Describe (in English or pseudo-code) how to determine whether  $(u, v)$  occurs on *every* path of minimal cost from  $s$  to  $t$ .

[10] (b) Describe (in English or pseudo-code) how to determine whether  $(u, v)$  occurs on *some* path of minimal cost from  $s$  to  $t$ .

[20] 3. **Descending Kruskal's algorithm**

Another approach to the minimum spanning tree algorithms is to remove “heavy” edges from the graph until only the tree is left, rather than adding “light” edges to an originally empty graph.

[10] (a) Design a version of Kruskal's algorithm that would exploit this idea. Your algorithm should work by removing heavy edges from the original graph until only an MST is left. Provide pseudo-code for your algorithm.

[10] (b) Prove that your algorithm always returns an MST of  $G$ . The steps of your proof will be similar to that of Kruskal's algorithm.

[20] 4. **Greenhouses and plants**

A botanical garden is planning to build  $n$  greenhouses, each of which is intended to represent a different climate zone. They have a list of  $m$  kinds of plants that they would like to grow there. Not every kind of plant can grow in every greenhouse, though usually one kind can grow in more than one climate; so associated with every kind of plants  $i$  is a list  $C_i$  of greenhouses where it can grow. In addition to that, every greenhouse can host no more than 20 varieties of plants.

[10] (a) Given the  $n, m$  and  $C_i$  for  $1 \leq i \leq m$ , specify an algorithm to determine the maximal number of different varieties of plants that can be grown in these greenhouses (total over all greenhouses). Hint: design a flow network that represents this problem.

[10] (b) How can you produce a list of chosen varieties for each greenhouse given the maximum flow on your flow network? Is it uniquely determined by the original network?