Due 14th July 2022 at 4pm Sydney time

In this assignment we apply maximum flow. There are *four problems* each worth 20 marks, for a total of 80 marks.

Your solutions must be typed, machine readable PDF files. *All submissions will be checked for plagiarism!*

For each question requiring you to design an algorithm, you *must* justify the correctness of your algorithm. If a time bound is specified in the question, you also *must* argue that your algorithm meets this time bound.

You may only use maximum flow algorithms presented in this course, namely Ford-Fulkerson and Edmonds-Karp.

Partial credit will be awarded for progress towards a solution.

Question 1

There is an $n \times n$ grid of squares. Each square is either *special*, or has a positive integer *cost* assigned to it. No square on the border of the grid is special.

A set of squares S is said to be *good* if it does not contain any special squares and, starting from any special square, you cannot reach a square on the border of the grid by performing up, down, left and right moves without entering a cell belonging to S.

1.1 **[5 marks]** In the following diagram, squares are labelled 'X' if they are special, otherwise they are labelled by their cost. Identify a good set of squares with minimum total cost for this particular grid.

5	3	4	9
4	X	3	6
1	9	X	4
1	2	3	5

1.2 **[15 marks]** Design an algorithm which receives an arbitrary $n \times n$ grid, runs in time polynomial in n and determines a good set of squares with minimum total cost.

Question 2

There are k people living in a city, whose n suburbs and m roads can be represented by an unweighted directed graph.

Every person is either slow or fast. Every night,

- Each slow person can stay in the same suburb, or move along a road to an adjacent suburb.
- Each fast person can stay in the same suburb, or move to any other suburb in the entire city.

Over the last d days, you know how many people were located in each suburb. That is, for each day i and suburb j (where $1 \leq i \leq d$ and $1 \leq j \leq n$), you know $p_{i,j}$, the number of people located in suburb j on day i. You are guaranteed that $\sum_{j=1}^{n} p_{i,j} = k$ for all i.

2.1 **[10 marks]** Design an algorithm which runs in $O(d(n+m))$ time and determines whether there could possibly be at least one slow person.

2.2 **[10 marks]** Design an algorithm which runs in time polynomial in n, m, and d, and determines the minimum possible number of fast people.

Question 3

There n boys and n girls at a party. Whenever a song starts, they will form exactly n pairs to dance. No boy will dance with the same girl twice.

Some pairs of boys and girls like each other, and all other pairs of boys and girls dislike each other. Every boy will dance with at most k girls that he dislikes, and each girl will dance with at most k boys that she dislikes, where $k < n$.

As the DJ, your job is to determine the maximum number of songs you can play, such that it is possible for pairs to be formed for all songs according to the above requirements.

Design an algorithm which achieves this and runs in time polynomial in n:

3.1 **[10 marks]** for $k = 0$.

3.2 **[10 marks]** for arbitrary k.

You may choose to skip 3.1, in which case we will mark your submission for 3.2 as if it was submitted for 3.1 also.

Question 4

You are given a flow network G with n vertices, including a source s and sink t, and m directed edges with integer capacities.

Your friend will ask you several queries of the form: "If an edge were to be added to G, going from vertex a to vertex b with capacity c, what would the maximum flow of the modified network be?" Note that each query considers adding an edge to the original flow network, so the modified network has $m + 1$ edges.

Before asking any queries, your friend gives you some time to prepare, which you can spend on precomputation.

Design an algorithm which performs any required precomputation and then answers each query in constant time, subject to the following restrictions:

4.1 **[5 marks]** $O(nm^2)$ precomputation; $b = t$ and $c = 1$ in all queries.

4.2 **[6 marks]** $O(nm^2)$ precomputation; $c = 1$ in all queries.

4.3 **[7 marks]** $O(n^2m^2)$ precomputation; $b = t$ in all queries.

4.4 **[2 marks]** $O(n^2m^2)$ precomputation; queries unrestricted.