University of Moratuwa, Sri Lanka
Faculty of Engineering
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BSc Engineering, Intake 2019, Semester 7
Jul 2023 – Nov 2023

CS4522 Advanced Algorithms

Assignment 2 (worth 7.5%; due 23rd Oct 2023)

Note the following important points before answering this assignment:
Use standard notations. All other notations and any assumptions must be clearly explained. For ease of grading, some questions may not be considered during grading; but students should answer all questions.

What to submit: Submit by uploading to the LMS a PDF document containing the answers to the questions. Make sure the size of the uploaded file is less than 5MB. You must include in your PDF document:

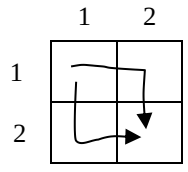
- the names and registration numbers of any students with whom you worked together on this assignment (As already informed, discussions with other students is allowed; but plagiarism, copying, giving or receiving aid are strictly not permitted. Working together cannot extend to writing the answers).
- the list of other resources (books, websites,…) you used in preparing your answers.

---

1. [20 marks] Given a matrix of size $m \times n$ where the rows are numbered 1,2,…,m and the columns are numbered 1,2,…,n , let us consider the problem of counting all the possible paths from the top-left cell (1,1) to the bottom-right cell (m, n) with the constraint that from each cell you can only move either to right (→) or down (↓).

Examples are shown in Fig. Q1(a), Q1(b) and Q1(c).

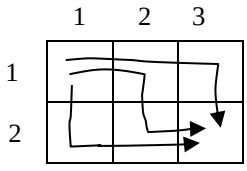
2x2 matrix



There are 2 paths from (1,1) to (2,2)

Fig. Q1(a)

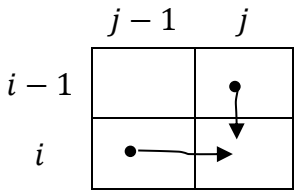
2x3 matrix



There are 3 paths from (1,1) to (2,3)

Fig. Q1(b)

Generic $m \times n$ Matrix



Paths to (i, j)

Fig. Q1(c)

(a) Develop a basic *recursive algorithm* to count the paths from the top-left cell (1,1) to the bottom-right cell (m, n), assuming m and n are given as inputs. (Do not worry about the efficiency of this basic algorithm. You are expected to improve its efficiency later).

Hint: You may use the Fig. Q1(c) to develop your solution: the number of paths to the cell (i, j) can be computed using the number of paths to its left neighbor cell $(i, j-1)$ and top neighbor cell $(i-1, j)$.

[6 marks]

(b) What is the time-complexity of your algorithm in (a) above?

[2 marks]

(c) Develop a more time-efficient algorithm than the algorithm in (a) above, using *dynamic programming* or *memoization* concepts. Present your idea clearly.

[10 marks]

(d) What is the time-complexity of your algorithm in (c) above?
Explain the extra cost you have to pay for this improved time-complexity, compared to the algorithm in (a).

[2 marks]

2. [25 marks] Consider the unweighted undirected graph $G = (V, E)$ where $V = \{1,2,3,4,5\}$ and whose adjacency matrix M, the square M^2 and the cube M^3 are as follows.

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad M^2 = \begin{bmatrix} 2 & 1 & 2 & 2 & 1 \\ 1 & 4 & 2 & 2 & 3 \\ 2 & 2 & 3 & 2 & 2 \\ 2 & 2 & 2 & 3 & 2 \\ 1 & 3 & 2 & 2 & 4 \end{bmatrix} \quad M^3 = \begin{bmatrix} 2 & 7 & 4 & 4 & 7 \\ 7 & 8 & 9 & 9 & 9 \\ 4 & 9 & 6 & 7 & 9 \\ 4 & 9 & 7 & 6 & 9 \\ 7 & 9 & 9 & 9 & 8 \end{bmatrix}$$

(a) What is the chromatic number of G? Show how you obtained it.

[4 marks]

(b) Can this graph G have a face coloring? If yes, show a face coloring with a minimum number of colors. If no, explain why.

[4 marks]

(c) What do the matrices M^2 and M^3 represent with respect to the graph G? Specifically, what do the (i, j)-th element in M^2 and M^3 represent?

[4 marks]

(d) Compute the number of triangles in the graph G *using any of the matrices* M, M^2 and M^3. Show your work. (Using any other method is not acceptable).

[5 marks]

(e) Compute the global clustering coefficient for G. Show your work.

[4 marks]

(f) Compute the local clustering coefficient C_2 of vertex 2 in G. Show your work.

[4 marks]

3. [20 marks] Suppose you are given two matrices A=(a_{ij}) and B=(b_{ij}) each of size $n \times n$ and you have to perform matrix addition to compute a new matrix C=(c_{ij}) as $c_{ij} = a_{ij} + b_{ij}$ for all $i, j = 1,2,\ldots,n$.

(a) Give a multi-threaded algorithm to compute C in parallel using parallel loops (i.e., *parallel for*). Parallelize as much as possible.

[7 marks]

(b) Give a multi-threaded algorithm to compute C in parallel using nested parallelism (i.e., *spawn*) with a divide-and-conquer strategy. State any assumptions and explain briefly how it works. Parallelize as much as possible.

[7 marks]

(c) Determine the *work*, *span* and *parallelism* for the parallel algorithm in (b) above.

[6 marks]

4. [15 marks] Suppose you are given an array A[1...*n*] of n numbers and you want to compute the product, P, of the elements in A; i.e., P = A[1] ×A[2] × ... ×A[*n*].

(a) Assuming you have *m* processors (*m* << *n*), give a parallel algorithm to compute P. How many operations and how many steps will there be in your algorithm?

[8 marks]

(b) Assuming you have an unlimited number of processors, give an efficient parallel algorithm to compute P. How many operations and how many steps will there be in your algorithm?

[7 marks]

5. [20 marks] Answer the following.

(a) Fig. Q5.1 shows a flow network. Edge labels indicate capacity of each edge.

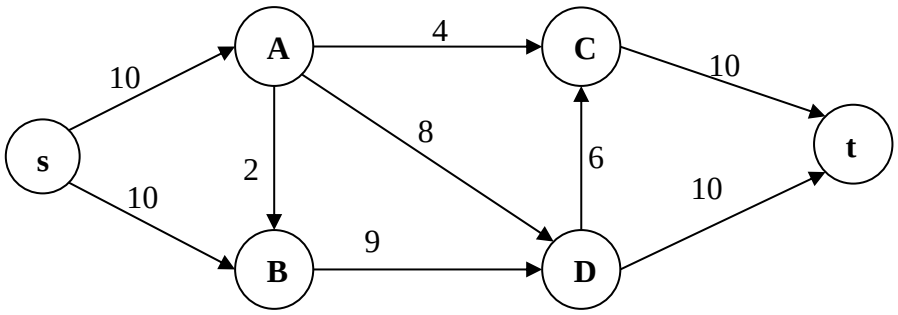


Fig. Q5.1

(i) For the flow network in Fig Q5.1, find the maximum flow from the source (**s**) to sink (**t**) using the Ford-Fulkerson method. Show all the steps.

[8 marks]

(ii) Show all *minimum s-t* cuts in the flow network in Fig Q5.1.

[2 marks]

s(b) A *bipartite graph* is a graph whose vertex set can be divided into two disjoint and independent sets, U and V, and every edge connects a vertex in U and a vertex in V. Fig. Q5.2 shows an example.

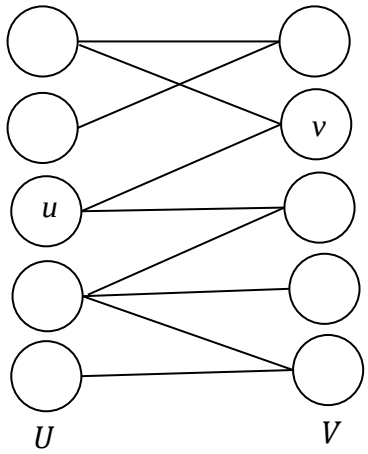


Fig. Q5.2

Such a graph can represent a scenario like a set of workers (U) and a set of tasks (V), where an edge from a u in U to a v in V means worker u can perform the task v. A worker may be able to perform many tasks and a task may be performed by many workers. In the graph in Fig. Q5.2, worker u can perform 2 tasks and the task v can be performed by 2 workers. A *matching* is *a set of edges chosen in such a way that no two edges share a vertex*. In our example, a matching gives an assignment of workers to tasks such that no worker (task) is assigned to more than 1 task (worker).

We want a *maximum bipartite matching* – a matching of maximum size (maximum number of edges), i.e., get as many tasks assigned as possible.

Show how this problem can be solved by converting it to a maximum flow problem. Describe all steps clearly.

[6 marks]

(c) Compute the transitive closure of the graph shown in Fig Q5.3 and show it in a matrix representation.

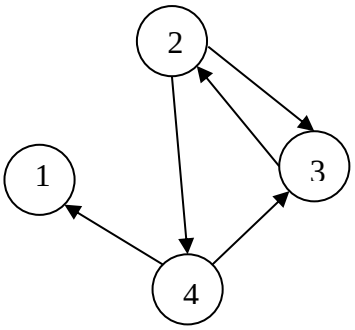


Fig. Q5.3

[4 marks]