

University of Toronto
Scarborough Campus
December 17, 2007

CSC C73 Final Examination

Aids allowed: One 8.5 × 11 handwritten, non-photocopied ‘cheat sheet’

Duration: Three hours

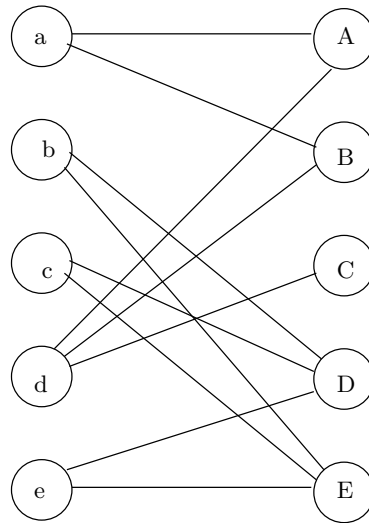
- There should be 9 pages in this exam booklet, including this cover page.
- Answer all questions.
- Put all answers in this booklet, in the spaces provided.
- For rough work, use the backs of the pages; *these will not be marked.*
- Good luck!

Family Name _____ Given Name _____
Student Number _____

Problem	Marks Rec'ved	Marks Worth
1.		5
2.		15
3.		15
4.		20
5.		20
6.		30
7.		30
8.		15
TOTAL		150

QUESTION 1. (5 marks)

Does the following bipartite graph have a perfect matching? Justify your answer.



ANSWER:

QUESTION 2. (15 marks)

Prove that if every symbol has frequency (strictly) less than $1/3$, then Huffman's algorithm cannot produce a codeword of length 1.

PROOF:

QUESTION 3. (15 marks)

Demonstrate the polynomial multiplication algorithm that uses the FFT by using it to multiply the polynomials $x^2 + 1$ and $x + 1$: Use the FFT to evaluate each polynomial at an appropriate set of points, and show the resulting vectors; multiply the vectors component-wise; and use the inverse FFT to obtain the coefficients of the final result. Show your work.

ANSWER:

QUESTION 4. (20 marks)

For each $k \in \mathbb{N}$, the **Hadamard matrix** H_k is a $2^k \times 2^k$ matrix defined recursively as follows:

- $H_0 = [1]$.
- For $k > 0$, $H_k = \left[\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array} \right]$

Let \vec{v} be a column vector of length $n = 2^k$. Describe an algorithm that computes the product $H_k \vec{v}$ in $O(n \log n)$ arithmetic operations (additions, subtractions, multiplications, or divisions). Explain why your algorithm achieves the stated complexity.

ANSWER:

QUESTION 5. (20 marks)

Recall the chain matrix multiplication problem: We are given a sequence of matrices A_1, A_2, \dots, A_n , of dimensions $m_0 \times m_1, m_1 \times m_2, \dots, m_{n-1} \times m_n$, respectively. We wish to compute the product $A_1 \times A_2 \times \dots \times A_n$. The cost of computing the product $A_i \times A_{i+1}$ (which is a matrix of dimension $m_{i-1} \times m_{i+1}$) is $m_{i-1} \cdot m_i \cdot m_{i+1}$. By associativity of matrix multiplication, we can compute this product in many different orders. The problem is to find an optimal order in which to multiply the matrices, i.e., an order with the minimum total cost.

In the course we saw a dynamic programming algorithm for this problem. Consider now the following greedy algorithm:

Let $i, 1 \leq i < n$, be such that the product $m_{i-1} \cdot m_i \cdot m_{i+1}$ is as small as possible. Multiply $A_i \times A_{i+1}$ first (i.e., we first perform the cheapest possible multiplication.) We are now left with a sequence of $n - 1$ matrices (where A_i and A_{i+1} are replaced by their product). Repeat the same procedure, until we are left with a single matrix.

Does this greedy algorithm always find an optimal chain matrix multiplication sequence? If so, prove it. If not, give a counterexample.

ANSWER:

QUESTION 6. (30 marks)

Let a and b be strings. A **common substring** of a and b is a string c that is a substring of both a and b . A **longest common substring** of a and b is a common substring of a and b that is at least as long as any other common substring of a and b . (Note that this is not the same as the longest common **subsequence**: a substring is a **contiguous** subsequence of characters in a string.)

Describe an algorithm that takes as input two strings a and b , and returns the length of a longest common substring of a and b . Explain why your algorithm is correct, and analyse its time complexity. For full marks, your algorithm should run in $O(mn)$ time, where m and n are the lengths of a and b .

ANSWER:

QUESTION 7. (30 marks)

We are given a flow graph $G = (V, E)$ with integer capacities and $f : E \rightarrow \mathbb{N}$, an integral **maximum** flow in G . Suppose that the capacity of a single edge $e \in E$ is increased by 1 resulting in a new flow graph G' . (Thus, G' is identical to G except that the capacity of e is increased by 1.)

Describe an algorithm which, given G , f and e , returns a maximum flow of G' . Your algorithm should run in $O(n + m)$ time, where n is the number of vertices and m is the number of edges of G . (Thus, running the Ford-Fulkerson algorithm from scratch with the new capacities will not do.)

Explain why your algorithm is correct, and analyse its time complexity.

ANSWER:

QUESTION 8. (15 marks)

Greedy Productions is seeking *actors* and *investors* for their new movie. The set of available actors is A , partitioned into the set of male actors M and the set of female actors F . The set of potential investors is I .

Each actor $a \in A$ requires a salary of s_a dollars to be cast in the movie. Each potential investor $i \in I$ has a set of favourite actors $A_i \subseteq A$, and will invest p_i dollars in the production of the movie, but only if *all* of his/her favourite actors are cast. Greedy Productions wants to maximise its profit, which is the total amount invested minus the actors' salaries. The movie requires at least 5 male and at least 4 female actors; however, any number of additional roles can be written into the script, as long as doing so advances the company's objective.

Express the above optimisation problem as a 0-1 integer program. Justify your answer.

ANSWER:

THE END