University of Toronto
Scarborough Campus
December 17, 2022

CSC C73 Final Examination
Instructor: Vassos Hadzilacos

Aids allowed: One 8.5 × 11 ‘cheat sheet’ (may be written on both sides)

Duration: Three hours

READ THE INSTRUCTIONS CAREFULLY

- There should be 16 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 graded***.
- The last three pages are blank and can be used for rough work or for overflow. ***Do not detach them*** from the booklet. If you use them for overflow, you must ***clearly indicate this*** on the page(s) containing the question(s) whose answer(s) you are providing in the overflow pages.
- In your answers you may use any result discussed in this course or its prerequisites merely by naming or describing it.
- Good luck!

QUESTION 1. (30 marks)

For each of the statements below, indicate whether it is true or false by circling the appropriate response, and give a brief but convincing justification for your answer.

a. In the interval scheduling problem, if all intervals have the same length then the greedy algorithm that considers intervals in increasing start time and keeps each interval if it does not overlap with any of the previously kept ones computes a set of non-overlapping intervals of maximum size.

Circle the correct answer: **True** / **False**

Justification:

b. Divide-and-conquer algorithm A divides the problem into three subproblems, each one-half the size of the original problem, and requires linear time for the divide and combine steps. Divide-and-conquer algorithm B divides the problem into three subproblems, each one-third the size of the original problem, and requires quadratic ($O(n^2)$) time for the divide and combine steps. Algorithm A is asymptotically faster than Algorithm B.

Circle the correct answer: **True** / **False**

Justification:

c. For directed graphs in which every node has $O(\sqrt{n})$ outgoing edges, where n is the number of nodes of the graph, Johnson's all-pairs shortest paths algorithm is asymptotically faster than the Floyd-Warshall algorithm.

Circle the correct answer: **True** / **False**

Justification:

d. There is an algorithm that, given a directed graph with n nodes and m edges with arbitrary weights, can determine in $O(mn)$ time if the graph has a cycle of positive weight.

Circle the correct answer: **True** / **False**

Justification:

e. If we add one to the capacity of every edge of a flow network, then the set of minimum cuts does not change. (That is, (S, T) is a minimum cut under the original capacities if and only if it is a minimum cut under the increased capacities.)

Circle the correct answer: **True** / **False**

Justification:

f. If we double the capacity of every edge of a flow network, then the set of minimum cuts does not change. (That is, (S, T) is a minimum cut under the original capacities if and only if it is a minimum cut under the increased capacities.)

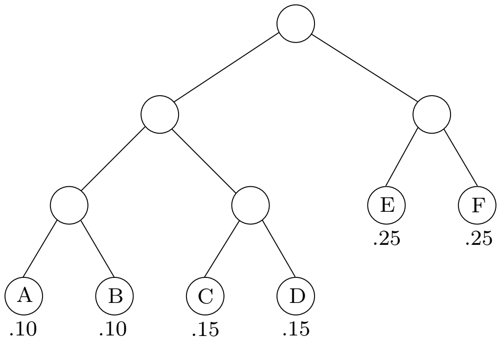
Circle the correct answer: **True** / **False**

Justification:

QUESTION 2. (15 marks)

Shown below is a full binary tree that represents an encoding of the symbols A, B, C, D, E, and F (shown as leaves of the tree), where each symbol appears with the frequency indicated below the corresponding leaf.

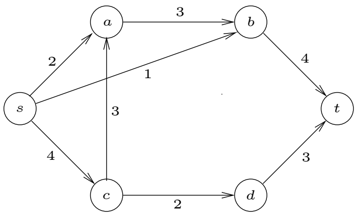
Is the encoding represented by this tree optimal? That is, does it minimize the number of bits required to encode a text over the six symbols A-F, where the symbols appear with the indicated frequencies? Justify your answer and show your work for possible partial credit.



Answer:

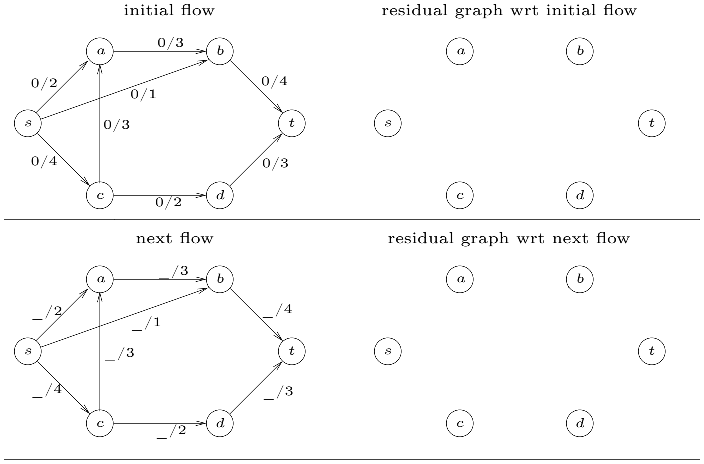
QUESTION 3. (20 marks)

Shown below is a flow network.



a. Describe an execution of the Ford-Fulkerson algorithm where, at each augmenting step, you choose a ***widest augmenting path*** (i.e., a path with the largest bottleneck). For each augmenting step: (a) show the residual graph with respect to the current flow; and (b) highlight the (***widest***) augmenting path along which you update the current flow; and (c) show the flow that results after the augmenting step.

Answer:


initial flow

residual graph wrt initial flow

next flow

residual graph wrt next flow

continue augmenting steps below and to the right, as needed

b. Give a minimum (S, T) cut of the given flow network.

Answer:

$S =$ __________ $T =$ __________

c. How can you verify that your answers to (a) and (b) are correct?

Answer:

QUESTION 4. (15 marks)

Give an algorithm that takes as input an array $A[1..n]$ of numbers that represent the incomes of n individuals and returns the average of these numbers, ***excluding the top 5% and bottom 5% incomes***. $A[1..n]$ is not sorted in any particular order. You may assume that n is a multiple of 20, and that all the numbers in A are distinct.

Your algorithm should run in $O(n)$ time (in the worst case). It should be based on an algorithm we learned in this course, which you should explicitly identify. Do not justify the correctness of your algorithm and do not analyze its running time. Describe it in clear pseudocode or point-form English; it should take no more than ten lines.

Algorithm:

QUESTION 5. (20 marks)

A butcher has n customers, each of whom has ordered one turkey. The butcher also has exactly n turkeys. The ith turkey weighs $T[i]$ kilos and the ith customer has specified that the ideal weight of the turkey they want is $C[i]$. An ***assignment*** (of turkeys to customers) is a permutation $p = p_1, p_2, \ldots, p_n$ of $1, 2, \ldots, n$, interpreted to mean that customer i is assigned turkey p_i. The ***disappointment*** of customer i in assignment p is $|C[i] - T[p_i]|$, i.e., the amount by which the specified ideal turkey weight differs from the weight of the turkey assigned to customer i. The butcher wants to find an assignment p that minimizes the maximum disappointment, i.e., minimizes $\max_{1 \leq i \leq n} |C[i] - T[p_i]|$.

a. Give a small example ($n \leq 3$) that shows that the following greedy strategy for assigning turkeys to customers does ***not*** find an assignment with the desired property: *Find a pair (i, j) such that $|C[i] - T[j]|$ is as small as possible; assign to customer i turkey j (i.e., set $p_i = j$); remove customer i and turkey j; repeat for the remaining customers and turkeys until there are no customers (or turkeys) left.*

Counterexample:

b. Give a polynomial-time ***greedy*** algorithm that takes as input the arrays $T[1..n]$ and $C[1..n]$ (each in arbitrary order), and returns an assignment with the desired property. State the running time of your algorithm. Do ***not*** justify its correctness.

Algorithm:

Running time:

QUESTION 6. (15 marks)

Shown below is an incomplete divide-and-conquer algorithm SEARCH(A, i, j) that takes as input an array $A[1..n]$ of ***distinct integers sorted in increasing order*** and two indices i, j such that $1 \le i \le j \le n$, and returns an index k between i and j such that $A[k] = k$, if such an index exists, or the integer 0 if no such index exists.

a. Complete the algorithm by filling in the missing spaces.

Algorithm:

```
SEARCH($A, i, j$)
if $i = j$ then
    if __________________ then return __________________
    else return 0
else
    $k := \lfloor (i + j)/2 \rfloor$
    if __________________ then return __________________
    else return __________________
```

b. Use the Master Theorem to analyze the running time of the above algorithm.

Running time analysis:

QUESTION 7. (30 marks)

Emon Lusk runs a single-person consulting service operating in Toronto, Canada and Melbourne, Australia. The consulting service has one-day projects lined up for each of the next $n \geq 2$ days in each of these locations, and for each project Emon knows the profit to be made from working on that project: For each i, $1 \leq i \leq n$, $T[i]$ is the profit for working on the day i project in Toronto and $M[i]$ is the profit for working on the day i project in Melbourne; both are non-negative numbers. To work on a project in either of these locations, Emon needs to be physically present at that location: working on these projects remotely is impossible. Moving from one location to the other takes one day during which Emon cannot work on any project.

A ***schedule*** is an array $S[1..n]$ such that $S[i]$ is $\mathtt{T}$, $\mathtt{M}$, or $\perp$ meaning, respectively, that on day i Emon is in Toronto, Melbourne, or traveling from one to the other. The ***value*** of a schedule S is the total profit that Emon makes by working on the project in the city where he is presently located according to schedule S (no profit on days when Emon is traveling). Emon can start on day 1 in either location, and wants to find the maximum possible value of a schedule.

For example, if $n = 6$, $T = [10, 3, 5, 3, 17, 30]$ and $M = [1, 5, 10, 20, 18, 10]$ the schedule of maximum value is $S = [\mathtt{T}, \perp, \mathtt{M}, \mathtt{M}, \perp, \mathtt{T}]$ and its value is $10 + 0 + 10 + 20 + 0 + 30 = 70$. The value Emon wants in this case is 70.

Describe a polynomial-time ***dynamic programming*** algorithm that, given the profit arrays $T[1..n]$ and $M[1..n]$, returns the maximum value of a schedule. (It is easy to retrofit the algorithm to also compute the corresponding schedule, but this is not required here.)

a. Define clearly the subproblems that your dynamic programming algorithm will solve.

Answer:

b. Give recursive formulas to compute the solutions to the subproblems in part (a). No justification of correctness is needed.

Answer:

[continued on the next page]

c. Describe your dynamic programming algorithm in pseudocode. Do ***not*** explain why your algorithm is correct.

<u>**Answer:**</u>

d. Analyze the running time of your algorithm.

<u>**Answer:**</u>

QUESTION 8. (35 marks)

The Department of Commuter and Metaphorical Sciences offers a one-year graduate program. A set of m students $S = \{1, 2, \ldots, m\}$ are registered in the program. A set of n courses $C = \{1, 2, \ldots, n\}$ are offered for the students of this program; C is partitioned into three subsets F, W, and R consisting, respectively, of the courses offered in the fall, winter, and summer semesters (each course is offered in exactly one semester). Each student i declares the set of courses $C_i \subseteq C$ that he or she is interested in taking. Each course j has an enrolment limit ℓ_j. No course has prerequisites.

An ***enrolment*** is a set of pairs $(i, j) \in S \times C$; such a pair indicates that student i is enrolled in course j.

a. ***By reduction to maximum flow*** describe an algorithm that returns an ***enrolment of maximum cardinality*** that satisfies the following constraints:

(1) each student is enrolled only in courses that he or she is interested in;
(2) each student is enrolled in at most five courses;
(3) each student is enrolled in at most three courses during the same semester;
(4) the number of students enrolled in course j does not exceed its enrolment limit ℓ_j.

The input to your algorithm consists of the following information:

(i) the set of students S;
(ii) the sets F, W, R into which the set C of courses is partitioned;
(iii) the set of courses C_i that student i is interested in taking, for each student $i \in S$;
(iv) the enrolment limit ℓ_j of each course $j \in C$;

Your description can include (but should not consist only of) a helpful diagram. Analyze the running time of your algorithm, but do ***not*** justify its correctness.

Description of the flow network:

[continued on the next page]

Algorithm that uses the flow network to find an optimal enrolment:

Running time analysis of the algorithm:

[part (b) on the next page]

b. The University charges students in this program by the course, but ***different courses may cost different amounts***. Specifically, each student taking course j is charged a fee f_j. In addition, each student i specifies the maximum amount b_i that they can afford for all the courses in which they are enrolled.

Define a 0-1 linear program (0-1 LP) to determine a enrolment that ***maximizes the total amount charged to the students*** subject to the constraints (1)-(4) in part (a) and the additional constraint:

(5) each student i is enrolled in courses whose total cost is at most b_i.

The constants of your 0-1 LP depend on (i)-(iv) in part (a) and on the following two additional pieces of information:

(v) the fee f_j for each course $j \in C$,
(vi) the budget b_i of each student $i \in S$.

Describe the variables and their meaning, the objective function, and the constraints of your 0-1 LP. For each linear constraint your specify, state to which of constraints (1)-(5) it refers. Also explain how to obtain the desired enrolment (i.e., set of pairs of the form $(i, j) \in S \times C$) from the solution to your linear program.

Zero-one linear program:

THE END

[Page for overflow or rough work — do not detach!]

[Page for overflow or rough work — do not detach!]

[Page for overflow or rough work — do not detach!]