Final Assessment

Instructions: Please read the following carefully.

- By submitting this final assessment, you have agreed to the Academic Integrity Declaration on edX. Be sure to read the wording carefully.
- Each of the questions is to be submitted to Crowdmark as a separate pdf. Although scanned documents are acceptable, please keep in mind that only legible writing will be marked (subject to the markers' discretion) and that Crowdmark forces a limit of 25MB per pdf file, which might be exceeded by high resolution scans.
- Be extra careful to upload each answer to the correct question, and make sure that you do not accidentally replace an answer with an answer to another question. As Crowdmark does not keep a repository of earlier submissions, your work will be lost. Before the deadline, double-check your last submission for answers submitted to the correct questions.
- The assessment is designed to be a take-home version of a standard final exam. If you know and understand the course content and have been keeping up all term, you should be able to finish it in the time it takes to do a regular, in-person exam. (However, it is up to you how much time to spend, and whether to complete it in one sitting or over several days.)
- In questions that specify the use of a particular paradigm, you are expected to come up with a new algorithm using that paradigm. It is not sufficient to implement a class example as a helper function and declare that the paradigm has been used.
- The assessment is intended to test your understanding of how to apply various paradigms to problems, as well as whether and when paradigms are appropriate for particular problems. Please keep in mind, while completing the assessment, that instructions to use a particular paradigm for a particular problem does not necessarily imply that a good algorithm is possible using that paradigm.

- Treat this assessment as an in-person exam in the sense that you are not to use illegal aids, including AI. **If cheating is strongly suspected, an oral exam (in person or on-line on MS Teams (camera on) to test your understanding of and competency with course material will be arranged and your final assessment mark may be based on some or all of the oral exam).**

- The number of marks for a question does not necessarily indicate its difficulty.

- The order of questions does not indicate the difficulty of questions, nor their relative importance.

- If you do not understand the wording of a question, you may post a private question to the discussion forum. Do not make any public posts about any of the questions. Course personnel may answer clarifying questions only.

- You are responsible for regularly checking the Course Updates, which will list any corrections or clarifications that are needed. Be sure to read it before you ask a question and before you submit your answers.

- **Start early; we cannot guarantee accommodation for problems caused by leaving the final assessment to the last minute, including last-minute illness. Submit early and often to Crowdmark as evidence that you are working on it and so there will be something to mark if your computer crashes, you lose files, etc. You can replace earlier submitted work on Crowdmark until the assessment deadline.** There is no make-up final assessment. Students with a valid, verified reason for not completing on time will have an oral final exam, in person or on-line on MS Teams (camera on).

- To further emphasize how important it is for your success in the course to start early and because it is very difficult to fairly deal with many, last minute, panicked questions, expect that there may be no responses to discussions after Wednesday, August 7, 4:00PM (24 hours before due time until deadline).

Problem definitions

Union Set: To describe the size of an instance of the following problem, we use n for the size of A and d for the maximum size of any set in A.

UNION SET

Input: A finite set $\mathcal{A}$ of finite sets of numbers
Output: A subset $\mathcal{B}$ of $\mathcal{A}$ such that the union of all the sets in $\mathcal{B}$ is equal to the union $\mathcal{U}$ of all the sets in A and the size of B is as small as possible.

For example, if $A = \{\{0,1\},\{2,3\},\{0\},\{1\},\{2\},\{0,3\},\{0,2\},\{1,2\}\}$, $B = \{\{0,1\},\{2,3\}\}$ is a solution to the problem because $U = \{0,1,2,3\}$ and the union of the sets in B is also $\{0,1,2,3\}$. Here $n = 8$ and $d = 2$.

W1. [*6 marks*] All subquestions are based on the following fact: "The worst-case running time of Algorithm A is in $\Theta(n^3 + m^2 + mn^2)$." For each of the following statements below, specify whether the statement is true, false, or there is not enough information to determine whether the statement is true or false.

For full marks, you must also provide a brief justification of your answer, using a sentence or two. If your answer is not enough information, give a situation in which it could be true and another in which it could be false. For example, for the statement "Algorithm A determines all the vertices in an input graph on n vertices and m edges", you would specify not enough information, give an example of an algorithm A with the given running time that solves the problem, and give another example of an algorithm A with the given running time that does not solve the problem. To choose true (respectively, false) for a statement about A, make sure that it is true (respectively, false) for all possible choices of A.

(a) *[1 mark]* The worst-case running time of A is in $\Omega(n^2)$.

(b) *[1 mark]* The worst-case running time of A is in $\Omega(n^4)$.

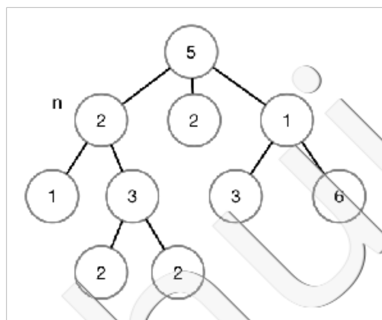
(c) *[1 mark]* The **best-case** running time of A is in $\Omega(n^3 + m^2 + mn^2)$.

(d) *[1 mark]* The **average-case** running time of A is in $O(n^3 + m^2 + mn^2)$.

(e) *[1 mark]* If $n \in \Theta(m)$, then the worst-case running time of Algorithm A is in $\Theta(n^3)$.

(f) *[1 mark]* If $n \in \Omega(m^2)$, then the worst-case running time of Algorithm A is in $O(m^6)$.

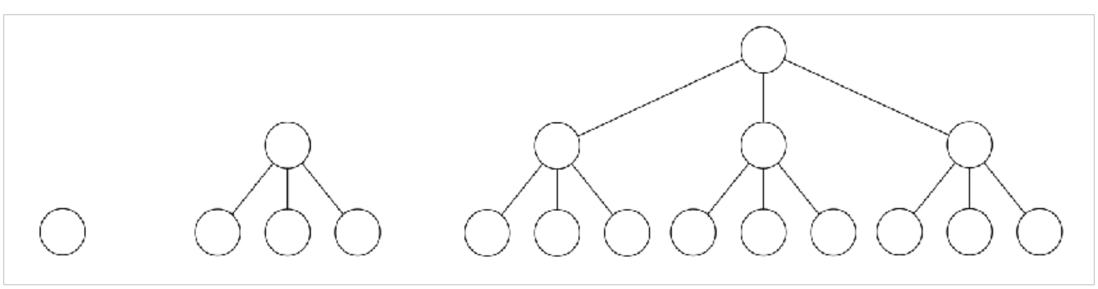
W2. *[10 marks]* In this question, you will write and analyze a divide-and-conquer algorithm Total that consumes a tree and the ID of a node in the tree and determines the sum of the weights of all nodes in the subtree rooted at the input node. In the example tree T illustrated below, Total(T, n) should produce 10, since $10 = 2 + 1 + 3 + 2 + 2$.



(a) *[4 marks]* Write a pseudocode function Total that solves the problem. You may wish to number your lines to use in parts (b) and (c).

(b) *[2 marks]* Briefly explain why your function is a divide-and-conquer algorithm by referring to specific line numbers.

(c) *[4 marks]* In a perfect ternary tree, each internal node has three children and all the leaves are at the same level (that is, the length of the path from the root to each leaf is the same). The illustration below depicts the three smallest nonempty perfect ternary trees.



Write the recurrence relation for the running time $T(n)$ of your algorithm when the input is a perfect ternary tree on n nodes and the ID of the root of that tree. You can assume that the value of n is such that there exists a perfect ternary tree on n nodes. For full marks, justify your answer by referring to specific lines in your algorithm.

W3. *[7 marks]*

(a) *[1 mark]* What type of problem is Union Set?

(b) *[2 marks]* By adding a bound, give the input and output for a decision problem related to Union Set.

(c) *[2 marks]* An algorithm for Union Set forms a solution B as follows: considering each set in A one by one, randomly choose whether or not to put the set in B. The probability of adding a particular set is $1/2$, so the probability of omitting it is also $1/2$. Is this a Las Vegas algorithm, a Monte Carlo algorithm, neither, or both? Briefly justify your answer.

(d) *[1 mark]* Is there ever a reason that one might choose to use a randomized algorithm for a problem in P ? Briefly justify your answer.

(e) *[1 mark]* What algorithm paradigm might you use on a problem that does not satisfy the optimal substructure property? Briefly justify your answer.

W4. *[8 marks]* Consider the following algorithm for Union Set.

```
Find_Sets(𝒜)
INPUT:    A set 𝒜 of sets of numbers
OUTPUT:   A subset of 𝒜
1  ℬ ← ∅
2  while there exists some x ∈ 𝒰 not contained in the union of the sets in ℬ
3      Choose a set S in 𝒜 that contains x and such that no larger set in 𝒜 contains x
4      Add S to ℬ
5  return ℬ
```

(a) *[2 marks]* What is the algorithm paradigm used? Briefly justify your answer by indicating a feature of the paradigm used in the algorithm. For full marks, your justification must refer to specifics of the algorithm.

(b) *[3 marks]* Give an example of an input on which the algorithm produces the optimal solution, for any order in which elements are considered in line 2. Your example should contain at least four sets and at least four elements. Explain both how the algorithm obtains the output and why the output is an optimal solution.

(c) *[3 marks]* Give an example of an input on which the algorithm does not produce the optimal solution, for any order in which elements are considered in line 2. Your example should contain at least four sets and at least four elements. Explain both how the algorithm obtains the output and why the output is not an optimal solution.

W5. *[9 marks]* We wish to use dynamic programming to solve determine the value of $M(m, n, m, n)$, where:

$$M(i,j,k,\ell) = \max\{M(i,j,k,\ell-1)+ijk, M(i,j,k-1,\ell)+i+j, M(i,j-1,k,\ell)+10, M(i-1,j,k,\ell)+j+k\}$$

For the base cases, $M(i,j,k,\ell) = 0$ for all values such that $i+j+k+\ell \leq 3$.

(a) *[3 marks]* Describe the type of table needed for the calculation as well as its specific size. Briefly justify your answer.

(b) *[2 marks]* State and briefly justify a possible evaluation order.

(c) *[1 mark]* How can the answer be extracted from the table?

(d) *[3 marks]* State and briefly justify the worst-case running time of the algorithm in Θ notation as a function of m and n.

W6. *[6 marks]* All subquestions are based on the following facts: "Problem A is in P, problem B is in NP, and problem C is NP-complete."

For each of the following statements, specify whether the statement is true for **all**, **none**, or **some** of the problems that could be A, B, and/or C, or if it is not yet known.

For full marks, you must also provide a brief justification of your answer, using a sentence or two. If your answer is some, give an example for which it is true and another example for which it is false.

For example, for the statement "In this course, we solved problem A using the divide-and- conquer paradigm", you would specify some, provide an example such as Closest Pair of Points for which it is true, and an example such as Minimum Spanning Tree for which it is false. Notice that both examples are in P , since the facts cannot be false.

(a) *[1 mark]* A is reducible to B.

(b) *[1 mark]* B is reducible to C.

(c) *[1 mark]* C is reducible to A.

(d) *[1 mark]* It is possible to solve B in time $O(n^3)$.

(e) *[1 mark]* It is possible to solve C using a fixed-parameter algorithm.

(f) *[1 mark]* There exists a polynomial-time verification algorithm for C.

W7. *[5 marks]* Suppose you wished to use backtracking to solve the problem Union Set.

(a) *[1 mark]* Specify what should be stored at each node of the search tree.

(b) *[2 mark]* Explain how to form the children of a node so that each feasible solution is explored exactly one time. (You will justify your choice in the next subquestion.)

(c) *[2 mark]* Explain why your answer to part (b) results in each feasible solution being explored exactly one time.

W8. *[8 marks]* You have a new electric car. It can go at most L kilometers on a charge. $G = (V, E)$ is an undirected graph with vertices representing towns and edges representing roads between some pairs of towns. Each edge has a positive weight corresponding to the length, in kilometers, of the road. You can recharge your car in any town but you cannot recharge your car between towns.

(a) Give a linear time algorithm to decide if you can get from every town to every other town with your car.

(b) Give an $O(|E| \log |V|)$ time algorithm to find the smallest value L^* such that if you can go L^* km on a charge then you can get from every town to every other town. Justify correctness of your algorithm.