CS 3310: Design and Analysis of Algorithms
Final Exam
(5/11/2023)

Name : ______________________

Last 4 digits of your Student ID #: ______________________

Read these instructions before proceeding.

- Closed book. Closed notes. You can use calculator.
- Length: **50 minutes**
- *Important Notes:*
  - *During the exam, all students need to join the Zoom meeting*
  - *No questions will be answered during the exam about the exam questions. Write down your assumptions and answer the best that you can.*
  - *Just in case you have trouble of submitting your exam here @Canvas, alternative way is to submit your completed exam to Prof. Young by emailing*
    gsyoung@cpp.edu
- Two ways to submit your exam:
  - ➢ Print out the exam paper. Write your answers on the exam paper. Scan your completed exam papers or take photos of them. Then turn in **one PDF file** here @ Canvas.
  - ➢ Read the exam from the computer screen and answer questions on your own white papers (number your answers). Scan your exam answers or take photos of them. Then turn in **one PDF file** here @Canvas.

Q.#1 (18)	Q.#2 (22)	Q.#3 (20)	Q.#4 (20)	Q.#5 (20)	Total (100)

1. (18 pts)

Decide ***"Easy"*** or ***"Hard"*** for each of the following problems that we discussed in the class. **No justifications are required**. (3 pts each)

(a) ***Longest Path*** Problem

(b) ***Shortest Path*** Problem

(c) ***0/1 Knapsack*** Problem

(d) ***Satisfiability (SAT)*** Problem

(e) ***Shortest-length-First Greedy algorithm for Optimal Storage on Tapes*** Problem

(f) ***Select algorithm using MM as pivot for Selection*** Problem (as in project 2 of your programming assignment).

2. (22 pts)

Part I: (10 pts) Let A = {20, 6, 5, 1} be a finite set of coin types. Each type in A is available in unlimited quantity. The coin changing problem is to make up an exact amount C using a minimum total number of coins. C is an integer > 0. A greedy solution to the problem will make change by using the coin types in the order of 20, 6, 5, 1. When a coin type is being considered, as many coins of this type as possible will be given. Give a counterexample to show that this greedy algorithm doesn't necessarily generate solutions that use the minimum total number of coins.

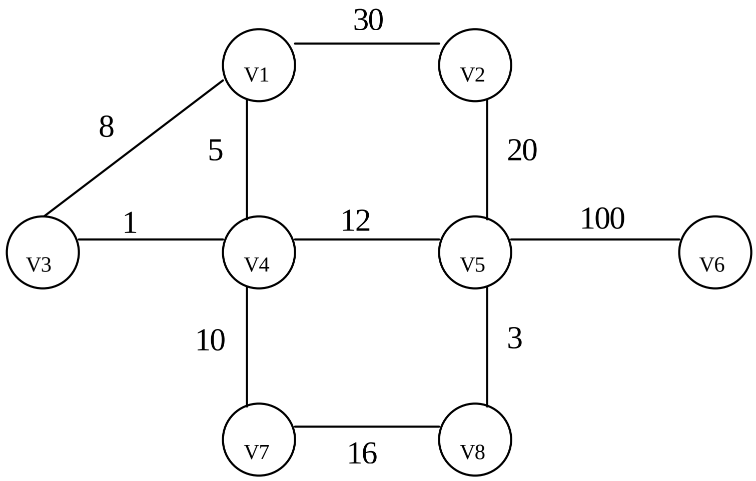
Part II: Decide ***True*** or ***False*** for each of the followings. You MUST briefly justify your answer. (4 pts each)

(e) (4 pts) (4 pts) It is known that PARTITION is a NP-complete problem. Assume that an $O(n^{2023})$ deterministic algorithm has been found for the PARTITION problem, then P = NP.

(f) (4 pts) If A is in NP, proving that A is solvable in polynomial time is sufficient to show that P = NP.

(g) (4 pts) If a decision problem A is NP-complete, proving that A is reducible to B, in polynomial time, is sufficient to show that B is NP-hard.

3. (20 points)



Assume that the algorithm starts from v1. Use **Kruskal's Algorithm** to find a minimum spanning tree and the cost of the minimum spanning tree for the above graph. ***Show all steps.***

4. (20 pts)

Given M_1 = [3×10], M_2 = [10×2], and M_3 = [2×5]. Use the dynamic programming algorithm (covered in the class) to compute the C(i,j)'s for all i < j and give the solution (i.e. the multiplication order) for this instance of **Chained Matrix Multiplication**..

(Hint: C(i, j), where i ≤ j, represents the minimum cost of computing $M_i \times M_{i+1} \times \ldots \times M_j$*. In this question, you need to find C(1,3), and then give the solution, the optimal multiplication order. You start from initializing all C(i,j)'s of size 1, and they are C(1,1), C(2,2), and C(3,3). Next is finding all C(i,j)'s of size 2, and so on)*

5. (20 pts)

Consider the **0-1-2-3 Knapsack Problem** obtained by replacing the 0/1 constraint of the 0/1 knapsack problem by x_i = 0 or 1 or 2 or 3 instead (i.e., we assume three copies of the i^{th} object are available, for all i). Description of the **0-1-2-3 Knapsack Problem** is given as follows.

0-1-2-3 Knapsack Problem:

Instance: n objects, $p_1, p_2, \ldots, p_n$ profits, knapsack of capacity M, $w_1, w_2, \ldots, w_n$ weights

Question: To find $x_1, x_2, \ldots, x_n$ s.t. $\sum p_i x_i$ is maximized subject to $\sum w_i x_i \le M$ where $x_i = 0$ or 1 or 2 or 3.

For the dynamic programming functional equation to solve the **0-1-2-3 Knapsack Problem**, we define $f_i(X)$ = max profit generated after considering the first i objects (or values of $x_1, x_2, \ldots, x_i$) subject to capacity X.

(a) (3 pts)

$$f_0(X) = \boxed{}\ ,\ \forall\ 0 \le X \le M$$

(b) (12 pts)

We assume that $f_i(X) = -\infty$, $\forall\ 0 \le i < n$ *and* $X < 0$

$$f_i(X) = \max\left\{\boxed{}\right\}$$

$$\forall\ 1 \le i \le n \text{ and } 0 \le X \le M$$

(c) (5 pts) Analyze the complexity of your algorithm in part (c)? **Justify your answer.**