Instructions (for your *Gradescope* submissions)

In recognition of our campus *Academic Integrity* policy, on the top of the first page of your submission, write: "*I affirm that I did not give or receive any unauthorized help on this exam, and that all work submitted is my own. I will destroy all electronic and printed copies of this questionnaire after submitting my exam solutions in Gradescope. I will not store or distribute copies of this questionnaire.*" Write your name and today's date below the statement.
Note: *Gradescope* submissions that do not have this signed statement will not be graded.

CSCI 650

Algorithms and Computability

FINAL EXAM, Fall 2020

Due in *Gradescope* Dec 17(Thu), 11:59 PM Pacific

1. **Exact String Matching.**

   Given the alphabet $\Sigma = \{$`'A'`, `'B'`, `'C'`, `'D'`, `'E'`$\}$ and the following strings:

   - pattern, $P =$ "`ABCDABD`"
   - text, $T =$ "`ABCDEABCDABEABCDABCDABDE`"

   Answer the following questions. For full credit, when calculating the number of comparisons performed, your answer must be supported by the number of iterations (or alignments or equivalent) performed and the number of comparisons for each alignment. Use of a table (or a spreadsheet) or similar method is encouraged to make your answers more easily readable.

[5] (a) Determine the actual number of comparisons performed when using the *naive* string–matching algorithm.

[5] (b) Assuming perfect hashing, determine the actual number of comparisons performed when using the *Rabin–Karp (1987)* string matching algorithm.

(c) For the *finite automaton–based* string matching algorithm ...

[5] i. Derive the *transition table* using the COMPUTE-TRANSITION-FUNCTION() preprocessing step and draw the equivalent *transition diagram*.

[5] ii. Use the FINITE-AUTOMATON-MATCHER() algorithm to determine the *state transitions* while matching P with T. (Note: For your supporting table, write the current state under each character of T. Recall that each state corresponds to a character comparison.)

(d) For the *Knuth–Morris–Pratt (1977)* string matching algorithm ...

[5] i. Derive the `lps[]` table (*a.k.a.* the π table) using the COMPUTE-PREFIX-FUNCTION() preprocessing step.

[5] ii. Use the KMP-MATCHER() algorithm to determine the q values while matching P with T. Write the current q value under each character in T, starting a new row when consecutive "jumps" are performed using the `lps[]` table.

[5] (e) Identify three features of the *Boyer–Moore (1977)* string matching algorithm that make it an improvement over the other string matching algorithms we covered.

[10] 2. **Approximate String Matching.** Use the *edit distance* dynamic programming algorithm to determine the similarity between the following pairs of strings:

- $s_1 =$ `DECEMBER`
- $s_2 =$ `WINTER`

[10] 3. **Greedy Algorithms.** A woodcarver has N jobs (orders from customers) which she must schedule and complete. The woodcarver can only be working on one job each day. Each i^{th} job takes an integer $1 \leq T_i \leq 1000$ time in days for the woodcarver to finish the job. Each day of delay before starting to work for the i^{th} job costs the woodcarver $1 \leq C_i \leq 10000$ cents (like a guarantee she will get the job done in time; otherwise, she starts paying the customer back).

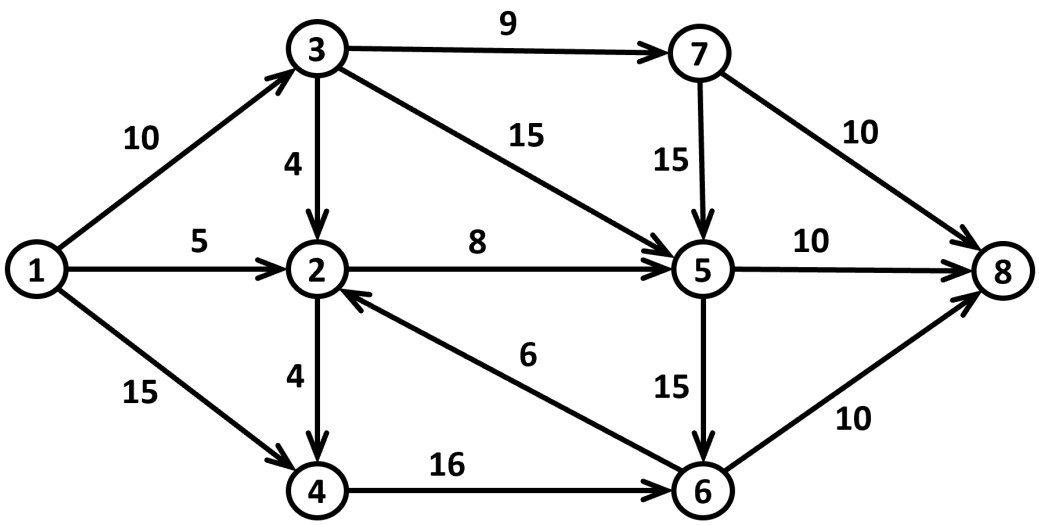
If the woodcarver is given the following jobs:

i	*Time*, T_i	*Delay cost*, C_i
1	3	4
2	1	1000
3	2	2
4	5	5

What is the sequence of jobs that minimizes the overall cost to the woodcarver?

Describe the algorithm you used to get the answer and explain why it is general enough to work on an unknown number of tasks that the woodcarver needs to schedule.

[30] 4. **Maximum Flow.** Use the *Ford–Fulkerson method with the Edmonds–Karp algorithm (1972)* to find a *maximum flow* for the following *network graph*:



For full credit, for each iteration of the algorithm you must

(a) draw the *flow network*;

(b) identify the *augmenting path*; and

(c) determine the *flow*.

Note that when applying BFS, prioritize vertices in *ascending* order.

[10] 5. **Video response and upload.**

Question: Of all the algorithms we covered and discussed in class this semester (except any that you or your teammate facilitated), which one did you find the most interesting and why?

Instructions:

(a) Think about your response to this question and organize your thoughts.

(b) Use your webcam or mobile device to record a video of yourself responding to the question. Start by introducing yourself and then repeating the question, then provide your response. *Your whole video response should not be more than a minute long.*

(c) Under *Weekly Content Folders* > *Finals Week* of our Blackboard course site, is an entry named **FINAL EXAM - Video Response Upload**. Click and start this *Blackboard Test* when you are ready to upload your video response.

(d) After you have uploaded and submitted your video response, make sure you click the **Save and Submit** button to complete your submission.

Instructions (after completing the exam)

This exam questionnaire is Ben Juliano's intellectual property. Portions of this questionnaire may appear in future exams. In recognition of our campus *Academic Integrity* policy, do not store or distribute copies of this document. After submitting your exam solutions in *Gradescope*, destroy any local electronic or printed copies of this document.