Midterm 02

COSC 311: Algorithms, Fall 2022

Instructions. This exam is closed book and closed notes. You may not consult any outside resources while taking the exam. You have 50 minutes to complete the exam.

Affirmation. I attest that that work presented here is mine and mine alone. I have not consulted any disallowed resources while taking this exam.

Name: Will

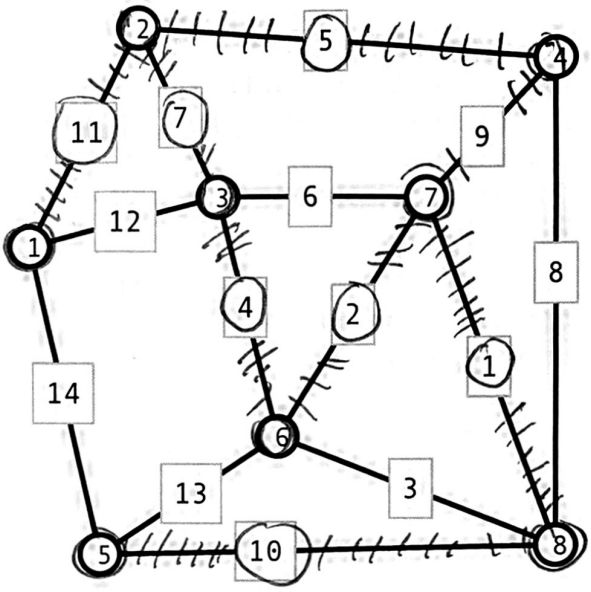
Signature: _____

Problem	Score
1	
2	
3	
4	

Totals:

0	1	2	3

Problem 1. Consider the following weighted graph.



(a) Use Prim's algorithm to find a minimum spanning tree for the graph depicted above starting from vertex 1. Please indicate on that figure which edges are included in the MST. In what order are the edges added to the MST by Prim's algorithm?

(1,2), (2,4), (2,3), (3,6), (6,7), (7,8), (8,5)

(b) In what order does Kruskal's algorithm add the edges to the MST?

(7,8), (6,7), (3,6), (2,4), (2,3), (5,8), (1,2)

Problem 2. The change making problem is defined as follows: you are given denominations of coins and an amount of money that must be represented using coins of the given denominations. The goal is to make change in this way using the fewest coins possible. For example, US coins have denominations $1, 5, 10, 25$ (pennies, nickels, dimes, and quarters). The value of 42 cents can be represented using one quarter, one dime, one nickel, and two pennies, for a total of five coins. 42 cannot be represented with fewer US coins. With US currency, the following greedy strategy always gives a representation using the fewest possible coins: start with the highest valued coin (quarter), and use it as many times as possible (until the remaining sum is less than 25). Then move on to successively smaller coin denominations (dimes, nickels, pennies) until you reach the desired amount.

(a) A (now defunct) currency used coins with denominations $1, 5, 10, 20, 25$. Give an example of an amount n for which the greedy strategy does *not* yield the minimum number of coins needed to represent n. (Be sure to show both the result of the greedy procedure as well as the minimum number of coins needed.)

? 40: greedy gives 25 + 10 + 5
(3 coins)

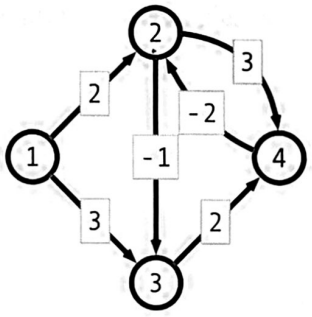
optimal is 2 × 20
(2 coins)

Using coins from part (a)

(b) For the denominations of part (a), let $C(n)$ denote the minimum number of coins needed to represent n. Write a recurrence relation for $C(n)$. To simplify your expression, it may help to use the convention that $C(n) = \infty$ for $n < 0$.

$$C(n) = 1 + \min\big(C(n-1), C(n-5), C(n-10), C(n-20), C(n-25)\big)$$

Problem 3. Consider the following graph.



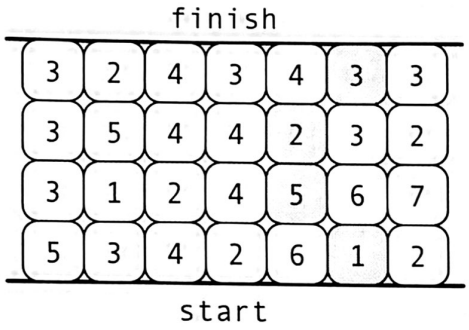
(a) Run the Bellman-Ford algorithm on the graph above in order to find the length of the shortest path containing at most 3 hops from vertex 1 to every other vertex in the graph. Be sure to include the table you fill out while simulating the algorithm.

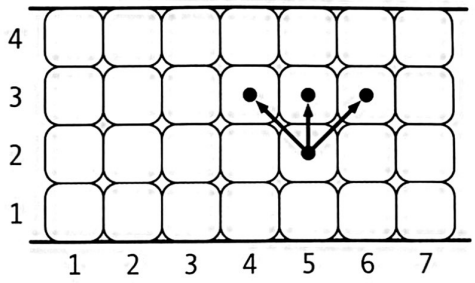
	1	2	3	4
4	0	1	1	3
3	0	2	1	3
2	0	2	1	5
1	0	2	3	·
0	0	·	·	·

(b) Suppose you run one extra iteration of the Bellman-Ford procedure on the graph above. How could you use the output of the extra iteration to determine whether or not the graph contains a negative weight cycle?

Distance to node 2 got updated. If no neg-weight cycle, then distances are not updated after n-1 iterations.

Problem 4. Consider the following game. The board consists of an $n \times m$ (n rows, m columns) rectangular grid of tiles, where each tile has an associated value indicating a danger level. From a given tile, you may move to any one of the three tiles (diagonally) above the tile, as indicated on the figure on the right below.





You may start at any tile in the bottom row, and your goal is to find a path from the bottom row to the top that minimizes the total danger—i.e., the *sum* of the danger levels of the tiles you cross. For example, the path indicated has a total danger of 11.

(a) For a given position (i, j) (i.e., row i, column j), let $\text{danger}(i, j)$ denote the danger of tile (i, j). Let $\text{best}(i, j)$ denote the danger of the least dangerous path from (i, j) to the top row. Write a recurrence relation for $\text{best}(i, j)$. To simplify the expression, you may assume that $\text{best}(i, j) = \infty$ when $j < 0$ or $j > m$.

best(i,j) = danger(i,j) +
min(best(i+1, j-1),
best(i+1, j),
best(i+1, j+1))

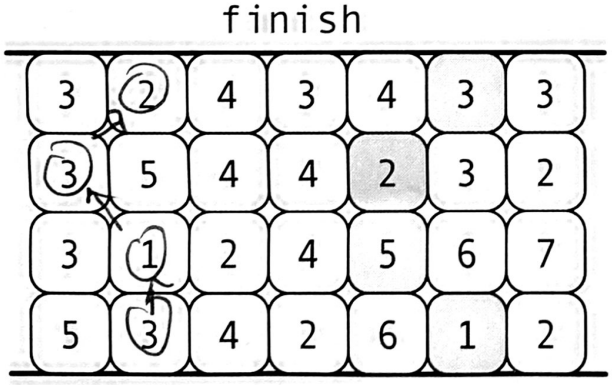
best(n+1, j) = 0 for all j.

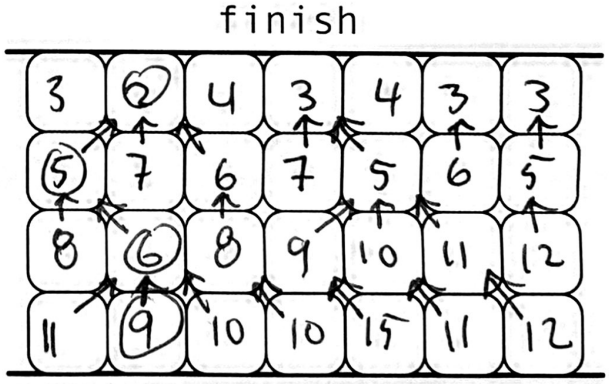
(b) Apply your recurrence relation from part (a) to write an efficient dynamic programming procedure that determines the total danger of the least dangerous path from the bottom to the top row. The running time of your algorithm should be $O(nm)$ when the game has n rows and m columns.

Fill out ~~from to~~ best from top to bottom

```
for j = 1 to m
    best(n, j) = danger(n, j)
for i = n-1 to 1
    for j = 1 to m
        best(i,j) = danger(i,j) + min(best(i+1, j-1),
                                      best(i+1, j),
                                      best(i+1, j+1))
~~best overall~~ min
return ~~max~~ best(1,j)
```

(c) Apply your procedure from part (b) to find the least dangerous path for the example below. You may find the blank table helpful for your bookkeeping.





min = 9