CMPT 306 Algorithms
Final Examination
Fall 2017

100 Total Points

This is a take-home exam. It is expected that you clearly and fully answer all questions, except the bonus question, and that your work is well organized and neatly done. Be sure to show all of your work to support your answers. **Full** credit will **only** be given for **fully supported** answers. You **CANNOT** only write the final answer. **Partial** credit will be given for **incomplete** or **incorrect** answers. You may not discuss any portion of this exam with anyone (both in and outside of this class); all work is to be completed by you and only you. If you have any questions about this exam, please contact me via email `jliang@westminstercollege.edu` . Please do not contact me using Canvas as I do not check that regularly.

This exam is due by 4:00 PM on Wednesday, December 13, 2017. No exams will be accepted after this date and time.

A dropbox labeled **Final Exam** will be posted on the Canvas site for this class.

By signing this I attest that I have neither given nor received aid on this test.

Name:

What to submit:

1. A copy of this exam for question 1-5.
2. To the dropbox, submit a zip file consisting of python code for question 7-9.

[1] (10 points) Please write the pseudocode of these sorting algorithms:

1. Bubble Sort
2. Quick Sort
3. Merge Sort
4. Heap Sort

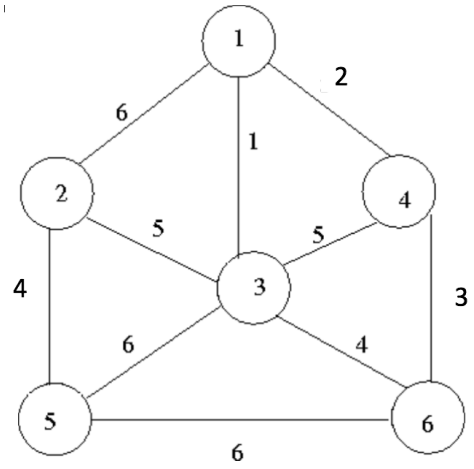
[2] (10 points) Please describe the similarity and difference of these searching algorithms:

1. BFS
2. DFS
3. UCS
4. Best-Fit
5. A Star

[3] (10 points) Use heapsort to sort list [2,9,7,6,5,8]. Please use a max heap. Please show all necessary steps such as heapify, push, or pop. You will not get any credits if you only write the final result.

[4] (10 points) How many bits are needed to encode string "barry andy barbara" using Huffman code? For example, there are five bits in code "01110". You need to calculate the frequency of each character in this string before you construct the Huffman tree. For example, the frequency of 'a' is 5/18. Do not forget that space ' ' is also one character. Please show all necessary steps. The code is not unique. However, the length of your code is unique. You will not get any credits if you only write the final result.

[5] (20 points) Please use Prim's Algorithm and Kruskal's Algorithm to find a minimum spanning tree from this graph:



[6] (20 points, 10 points for the cost, 10 points for the path) A social network consisting of sixteen **gossipers** is shown in Figure 1. Each circle represents a **gossiper** and each directed edge represents a path to another gossiper. Two gossipers are said to be *adjacent* if there is an edge connecting them. Associated with each edge is a time delay which is the amount of time in minutes for a **rumor** to be sent over that edge. Each gossiper transmits its rumor using the *broadcast approach*, meaning that it may simultaneously transmit a rumor over all its edges it is connected to. For example, gosspier 1 may simultaneously transmit a message to gossipers $2, 3, 4$, and 5. (The delay is due to when adjacent gossipers check their Twitter feed!)

Assume that gossiper 1 wishes to broadcast a rumor to every other gossiper in the network. Gossiper 1 will simultaneously transmit the rumor to each of its adjacent gossipers. In turn, when an adjacent gossiper receives the rumor, that gossiper will simultaneously transmit (i.e. forward) the rumor along all of its outgoing edges.

Write a python code to answer this question:

If a rumor originates at gossiper 1, how quickly does gossiper 16 receive the rumor, and what was the path of gossipers the rumor took?

Please refer lab-11 using Dijkstra's algorithm to find the shortest path and cost. The output of your code should be like this:

The shortest cost from 1 to 11 is: 15
The shortest path from 1 to 11 is: ['1', '3', '7', '11']

Please name your code as shortestpath.py or shortestpath.ipynb. Please download the network.txt from https://cmpt306.github.io/exam/final/network.txt

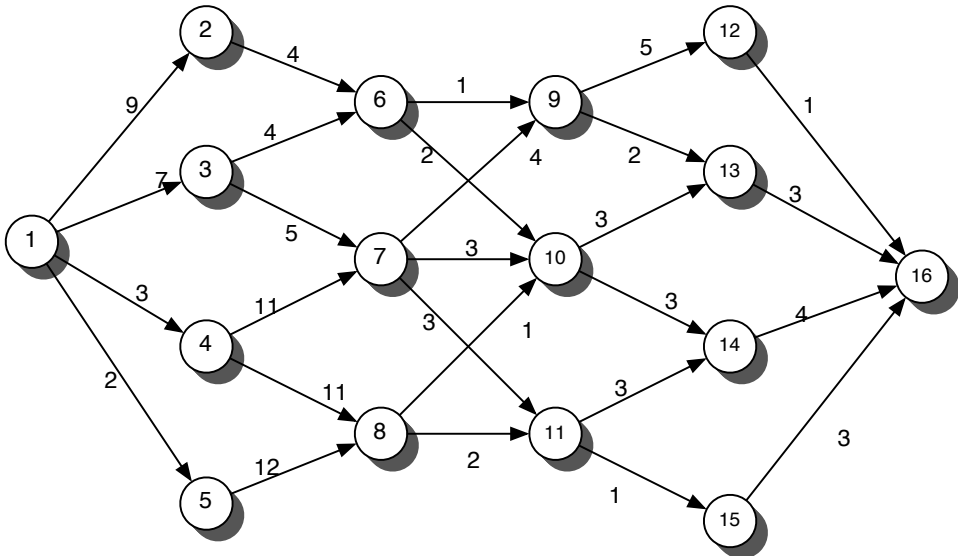


Figure 1: Social network with gossipers.

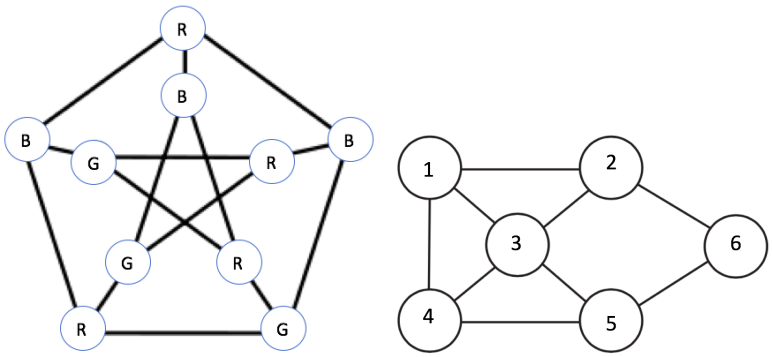
[**7**] (20 points, 10 points for the max value, 10 points for the path) Several coins are placed in cells of an n by m board, no more than one coin per cell. A robot, located in the upper left cell of the board, needs to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it always picks up that coin. Write a python code to find the maximum value of coins the robot can collect and a path it needs to follow to do this. Please name your code as robot.py or robot.ipynb. The output of your code should be like:

The max values of coins is: 1
The path is: ['Down', 'Right', 'Down']

Hints: This is a dynamic programming problem. You need to create a 2-dimension array initialized by 0 at each position to update the value of coins. You also need to create another 2-dimension array initialized by empty list [ ] at each position to add the direction of either 'Down' or 'Right'.

			1 cent		
	1 cent			2 cents	
			1 cent		
			1 cent		1 cent
		2 cents			1 cent
3 cents			1 cent		

[**8**] **Bonus Question. This question is not required.** (10 points) The 3-color graph problem is that you have a graph G consisting of V vertices and E edges and you must assign one of three colors for each vertex so that no two adjacent vertices have the same color. For example, this following left graph is colored by only three colors, red, green, and blue, under the 3-color condition. Please write



a python code to assign 3-color 'R','G', 'B' for a given graph. The output of your code should be like: '1': 'R', '3': 'G', '2': 'B', '5': 'R', '4': 'B', '6': 'G'. Please name your code as color.py or color.ipynb. Please download files for these two graphs from https://cmpt306.github.io/exam/final/graph1.txt and https://cmpt306.github.io/exam/final/graph2.txt.

Hints: This is a back-tracking problem. You can assign color to vertices one by one based on the index of vertex. For example, in the right graph, you start to assign one color for vertex 1. Then

you try to assign one color for vertex 2 until you successfully assign one color for last vertex 6. If you cannot find a solution, then backtrack to previous vertex. In order to assigning a color to one vertex, you need to check for safety by considering colors of adjacent vertices.