Programming Assignment #1

TOTAL POINTS 1

1. Download the following text file:

SCC.txt

The file contains the edges of a directed graph. Vertices are labeled as positive integers from 1 to 875714. Every row indicates an edge, the vertex label in first column is the tail and the vertex label in second column is the head (recall the graph is directed, and the edges are directed from the first column vertex to the second column vertex). So for example, the 11^{th} row looks liks : "2 47646". This just means that the vertex with label 2 has an outgoing edge to the vertex with label 47646

Your task is to code up the algorithm from the video lectures for computing strongly connected components (SCCs), and to run this algorithm on the given graph.

Output Format: You should output the sizes of the 5 largest SCCs in the given graph, in decreasing order of sizes, separated by commas (avoid any spaces). So if your algorithm computes the sizes of the five largest SCCs to be 500, 400, 300, 200 and 100, then your answer should be "500,400,300,200,100" (without the quotes). If your algorithm finds less than 5 SCCs, then write 0 for the remaining terms. Thus, if your algorithm computes only 3 SCCs whose sizes are 400, 300, and 100, then your answer should be "400,300,100,0,0" (without the quotes). (Note also that your answer should not have any spaces in it.)

WARNING: This is the most challenging programming assignment of the course. Because of the size of the graph you may have to manage memory carefully. The best way to do this depends on your programming language and environment, and we strongly suggest that you exchange tips for doing this on the discussion forums.

Enter answer here

I, **Veinstin Furtado**, understand that submitting another's work as my own can result in zero credit for this assignment. Repeated violations of the Coursera Honor Code may result in removal from this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit

Programming Assignment #2

TOTAL POINTS 1

1. In this programming problem you'll code up Dijkstra's shortest-path algorithm. 1

   Download the following text file:

   dijkstraData.txt

   The file contains an adjacency list representation of an undirected weighted graph with 200 vertices labeled 1 to 200. Each row consists of the node tuples that are adjacent to that particular vertex along with the length of that edge. For example, the 6th row has 6 as the first entry indicating that this row corresponds to the vertex labeled 6. The next entry of this row "141,8200" indicates that there is an edge between vertex 6 and vertex 141 that has length 8200. The rest of the pairs of this row indicate the other vertices adjacent to vertex 6 and the lengths of the corresponding edges.

   Your task is to run Dijkstra's shortest-path algorithm on this graph, using 1 (the first vertex) as the source vertex, and to compute the shortest-path distances between 1 and every other vertex of the graph. If there is no path between a vertex v and vertex 1, we'll define the shortest-path distance between 1 and v to be 1000000.

   You should report the shortest-path distances to the following ten vertices, in order: 7,37,59,82,99,115,133,165,188,197. You should encode the distances as a comma-separated string of integers. So if you find that all ten of these vertices except 115 are at distance 1000 away from vertex 1 and 115 is 2000 distance away, then your answer should be 1000,1000,1000,1000,1000,2000,1000,1000,1000,1000. Remember the order of reporting DOES MATTER, and the string should be in the same order in which the above ten vertices are given. The string should not contain any spaces. Please type your answer in the space provided.

   IMPLEMENTATION NOTES: This graph is small enough that the straightforward $O(mn)$ time implementation of Dijkstra's algorithm should work fine. OPTIONAL: For those of you seeking an additional challenge, try implementing the heap-based version. Note this requires a heap that supports deletions, and you'll probably need to maintain some kind of mapping between vertices and their positions in the heap.

   Enter answer here

Programming Assignment #3

TOTAL POINTS 1

1. Download the following text file: 1 point

Median.txt

The goal of this problem is to implement the "Median Maintenance" algorithm (covered in the Week 3 lecture on heap applications). The text file contains a list of the integers from 1 to 10000 in unsorted order; you should treat this as a stream of numbers, arriving one by one. Letting x_i denote the ith number of the file, the kth median m_k is defined as the median of the numbers $x_1, \ldots, x_k$. (So, if k is odd, then m_k is $((k+1)/2)$th smallest number among $x_1, \ldots, x_k$; if k is even, then m_k is the $(k/2)$th smallest number among $x_1, \ldots, x_k$.)

In the box below you should type the sum of these 10000 medians, modulo 10000 (i.e., only the last 4 digits). That is, you should compute $(m_1 + m_2 + m_3 + \cdots + m_{10000}) \bmod 10000$.

OPTIONAL EXERCISE: Compare the performance achieved by heap-based and search-tree-based implementations of the algorithm.

Enter answer here

I, **Veinstin Furtado**, understand that submitting another's work as my own can result in zero credit for this assignment. Repeated violations of the Coursera Honor Code may result in removal from this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit

Programming Assignment #4

TOTAL POINTS 1

1. Download the following text file: 1 point

algo1-programming_prob-2sum.txt

The goal of this problem is to implement a variant of the 2-SUM algorithm covered in this week's lectures.

The file contains 1 million integers, both positive and negative (there might be some repetitions!).This is your array of integers, with the i^{th} row of the file specifying the i^{th} entry of the array.

Your task is to compute the number of target values t in the interval [-10000,10000] (inclusive) such that there are *distinct* numbers x, y in the input file that satisfy $x + y = t$. (NOTE: ensuring distinctness requires a one-line addition to the algorithm from lecture.)

Write your numeric answer (an integer between 0 and 20001) in the space provided.

OPTIONAL CHALLENGE: If this problem is too easy for you, try implementing your own hash table for it. For example, you could compare performance under the chaining and open addressing approaches to resolving collisions.

Enter answer here

I, **Veinstin Furtado**, understand that submitting another's work as my own can result in zero credit for this assignment. Repeated violations of the Coursera Honor Code may result in removal from this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit