CSE 331: Algorithm and Complexity Summer 2020

Programming Homework

Instructor: Xiangyu Guo **Deadline: Jul/31/2020**

Your Name: _____ Your Student ID: _____

Problems	1	2	3	Total
Max. Score	10	10	10	30
Your Score				

Submission format

1. You should submit **one** source file for each problem.
2. Your code need to read the input from the console and output directly to the console.
3. Your code shouldn't output anything other than the desired output. For example, do not output messages like "The output is:".
4. Allowed programming languages:
   - Python: use Python version ≥ 3. You can run "`python --version`" to check its version. Do not use any third-party packages other the standard library.
   - C++: use C++ 11 or later. Do not use any library files other than the STL.
   - Java: use Java version $\geq$ 1.8. You can run "`java -version`" to check its version. Do not use any third-party libraries.

Problem 1 (10 points).

You need to implement the Prim's algorithm for the minimum spanning tree problem. An $O(n^2 + m)$-time algorithm is sufficient to pass all test cases.

Input: You need to read the input from the console. In the first line of the input, we have two positive integers n and m. n is the number of vertices in the graph and m is the number of edges in the graph. The vertices are indexed from 1 to n. You can assume that $1 \leq n \leq 1000$ and $1 \leq m \leq 100000$. In the next m lines, each line contains 3 integers: u, v and w,with $1 \leq u < v \leq n$ and $1 \leq w \leq 10^6$. This indicates that there is an edge (u, v) of weight w. You can also assume that the graph is connected and there are no parallel edges.

Output: You need to output to the console. The first line of the output is an integer indicating the total weight of the minimum spanning tree. From line 2 to line n, you need to output the $n - 1$ edges in the minimum spanning tree. Each line contains 2 integers between 1 and n, indicating the two end-points of an edge.

Example: Your code should be able to (compile and) execute as follows:

Python	`python prim.py <input.txt >output.txt`
C++	`g++ -std=c++11 prim.cpp -o prim`
	`./prim <input.txt >output.txt`
Java	`javac prim.java`
	`java prim <input.txt >output.txt`

Below are examples for input and output:

input.txt	output.txt
9 14 1 2 5 1 8 12 2 3 8 2 8 11 3 4 13 3 6 4 3 9 2 4 5 9 4 6 14 5 6 10 6 7 3 7 8 1 7 9 6 8 9 7	42 1 2 2 3 3 6 3 9 4 5 5 6 6 7 7 8

Problem 2 (10 points).

You need to implement the *divide-and-conquer* algorithm for counting inversions. As in the Problem 1, you need to read from the standard input (i.e, the terminal) and output to the standard output (i.e, the screen).

Input format: The first line of the input contains one positive integers $n, 1 \leq n \leq 10^6$. The next n lines contain the n integers $A[1], A[2], \ldots, A[n]$; every integer is between 0 and 10^8.

Output format: Just output 1 line, which is total number of inversions.

Example: Your code should be able to compile and execute like in Problem 1. Below are examples for input and output. Output is 7 because there are 7 inversion pairs: $(7, 3), (7, 5), (20, 16), (20, 5), (20, 8), (16, 5), (16, 8)$.

input.txt	output.txt
6 7 3 20 16 5 8	7

Problem 3 (10 points).

You need to implement the *dynamic programming* algorithm for the longest common subsequence problem.

Input: As in previous problems, you need to read the input from the console. It contains two lines, each containing one string. You can assume each string only contains upper and lower case letters and numbers; the length of each string is at most 1000.

Output: You need to output to the console. The first line of the file is an integer indicating the length of the longest common subsequence between the two strings. The second line contains the longest common subsequence (which may not be unique).

Example: Your code should be able to compile and execute like in Problem 1. Below are examples for input and output.

input.txt	output.txt
bacdca adbcda	4 adca