Data Structures and Algorithms for Engineers

Programming Assignment 5: Connected Items

Task: You are required to develop a solution for the connected items problem and to produce a technical report of the solution.

Instructions

1) Download the code and sample data for this assignment from this location.
   Unzip this file to a folder on the VM (or own environment). Organize the code and the sample input into the following locations:
   /home/dsa/connecteditems/code/src/
   /home/dsa/connecteditems/code/bin/
   /home/dsa/connecteditems/sample_inputs/

2) Each line in an input file will contain coordinates of two locations on a grid, and a flag which indicates whether these locations have a direct connection or not. For example: The entry (2284, 510, 2284, 515, 1) indicates that the location x1=2284, y1=510 is connected to the location x2=2284, y2=515. The flag is 1, which means there is a direct connection. You can ignore any other values of the flag that you encounter in the input file.
3) Your goal is to group all locations that are connected to each other directly, or indirectly. There could be multiple groups in an input file.
4) In the output file, you will need to print all the connections, with one direct connection on each line. For example, for the input file sample_01.log, the result is in file: sample_01.log.res.txt. A few lines form the result file are shown below:
   (x1, y1, x2, y2, group number)
   (151, 1464, 151, 1465, 1)
   (151, 1465, 151, 1469, 1)
   ....
   (2232, 677, 2232, 681, 2)
   (2232, 681, 2232, 686, 2)

   This means that location 151, 1464 is connected to 151, 1465 and is in group 1. Location 151, 1465 is connected to 151, 1469 and is also in group 1. This particular input file has 4 groups. This is evident in the result file where the group number has the values: 1,2,3,4. For example, we have shown some other lines from the result file. These lines show that location 2232, 677 is connected to 2232, 681 and is in group 2. Location 2232, 681 is connected to 2232, 686 and is also in group 2.
5) There will be no deviations from the given format in the input files.
6) The groups are assigned based on size of the connected group. Group having the largest number of connected locations is given group 1, next is group 2 and so on.
7) You will need to remember following aspects while generating the output file:
   a) All locations must be printed in increasing order of group sizes. i.e., all points belonging to group 1 will be printed first and then group 2 and so on.
   b) Locations with smaller X, Y coordinates shall be printed before locations having larger X, Y coordinates. For example, in sample_01.log.res.txt (151, 1464, 151, 1465, 1) appears before (151, 1465, 151, 1469, 1)
   c) For groups with the same size:
      i) If there are two groups with the same size, group having the smallest X coordinate shall be printed first.
      ii) If two groups have the same size and the same smallest X coordinate value, the group with smallest Y coordinate will be printed first.

iii) It is not possible to have two groups with the same size and same value of smallest X and Y coordinate. If you get such groups, it means your code has a bug.

A. How to write your code:

Your code needs to be implemented in the file ConnectedItems.cpp within the following function:

ConnectedItems::getConnectedItems(**char*** inputFilePath, **char*** outputFilePath)

You can create other data structures, helper functions, and classes within the files ConnectedItems.h and ConnectedItems.cpp. You must NOT create any other source files. You will zip these two files ONLY and submit them on gradescope AND canvas. We will make a call only to the above function to grade your work.

You cannot use std::template packages. However, you can reuse code, or classes that YOU have written. You may also reuse codes that have been given in class. However, you must document and cite the sources for such codes. **Failure to cite** such codes will lead to stiff **penalties**. Furthermore, you must inform the TAs of the codes you have reused from class-based examples.

B. How to write a report:

Submit a two-page(max) report describing your solution and justifying your decisions. You may use a graphic/diagram to give a high-level architecture for the solution. In the results section, the report should describe solution timing and memory information. Based on your decisions and solution choice, you should give a critical analysis of the timing information and memory information that have been obtained by your solution. Finally, you should consult literature and identify (with citations) two potential real-world applications of the connected items solution.

Structure of report: The report should be of publication quality and in PDF format. The report should have the following sections: Title, Abstract, Introduction and Problem Definition, Methods, Results and Discussion, Recommendations and Conclusion, and References. It is recommended that you use the two-column format.

Formatting: 10-pt Serif or Sans Serif font for main text, single spaced, and justified. 11-pt, Serif or Sans Serif font, bold face for title. 10-pt, Serif or Sans Serif font, boldface, italicized for section headers.

Submissions:

1. Files ConnectedItems.h and ConnectedItems.cpp must be zipped into a single file “src_files.zip”.
2. Submit zip file of code on Gradescope and Canvas.
3. Submit pdf report on Canvas.
4. Any number of submissions are allowed. **Your last submission will be used for grading.**

Clues to implementation:

The first step to solve this would be to read the input file and store it in a format where locations having direct connections can be accessed by your algorithm. This assignment can be solved in an optimal manner if you learn about BitVector and using it to represent sparse matrices.

The second step is to identify all locations that are connected to each other through some other locations. This can be done using the BFS or the DFS algorithm that identifies connected nodes in a graph.

Grading method:

1) If your code runs and generates an output file in the correct format for each input file, you get submission points.
2) If your method generates correct results for each test file, you get points for correctness.
3) We will measure the points for memory consumption using the following values:
    a. submitted_memory: Your memory in Bytes.
    b. our_memory: The best memory value from all students i.e., the lowest memory consumed among all student submissions.
    c. Memory score will be based on the ratio: (our_memory / submitted_memory) * max score for memory. The maximum points you can get here is “max score for memory”.
4) We will measure the points for run time using the following values:
    a. submitted_time: We will measure your run time in milliseconds.
    b. our_time: The best time value from all students i.e., the lowest time consumed among all student submissions.
    c. Your run-time score will be based on the ratio: (our_time / submitted_time) * max score for runtime. The maximum points you can get here is “max score for runtime”.
5) The maximum score for submission, correctness, memory, and runtime is based on a percentage of total points. See the table below for the distribution.
6) If your report follows the recommended form and structures, provides clear details and justifies the decisions made while creating your solution, and provides a critical analysis of timing and memory information considering the design decisions, you will get maximum marks.

Table 1: Code Grading Rubric

Item	**Criteria**	**Points Awarded**
Output: Does the code run? Are the output files generated with the filenames and formats as specified in the problem?	Points obtained	
	Max Points	***9***
Correctness: Is the right output generated?	Points obtained	
	Max Points	***27***
Timing	Time taken to run (ms)	
	Comparison with other students (divide by min of others)	
	Max. Timing score	***27***
Memory	Memory consumed (Bytes)	
	Comparison with other students (divide by min of others)	
	Max. Memory score	***27***
	Total points obtained	
	Total for Task	**/90**

Further details on the report grading rubric. You obtain full marks by meeting the criteria provided below.

Table 2: Report Grading Rubric

Item	**Criteria**	**Score**
Title (1 pt)	Succinct and informative	
Abstract (3 pts)	Describes the problem, the main aim, the method(s), the results, and conclusion.	
Introduction and Problem Definition (5 pts)	Establishes the context and highlights some background supporting literature. Clearly outlines the problem to be solved.	
Methods-I (2 pts)	Well-labeled, well-described, and informative diagram summarizing the solution.	
Methods-2 (6 pts)	Clearly outlines the solution approach and provides reasoned rationale for the solution choice.	
Results and Discussion (8 pts)	Presents the results clearly and discusses the results (especially, timing and memory usage information).	
Conclusion & Recommendations (4)	Presents an informative summary; identifies and cites potential real-world applications; and provides recommendations on potential improvements.	
References (1 pt)	At least 5 references. References are relevant and well cited.	
	Total:	**/30**