Data Structure Assignment#4

German language and literature

2016130927 Park Jun Yeong

Ⅰ. Development environment: Microsoft Visual Studio 2013 Professional

Ⅱ. Explanation of the algorithm and the code

ⅰ. A complete overview of code:

1). Graph construction

At the first moment, this program accepts command-line arguments (input.txt and output.txt) and recognized the number of nodes and edges, from which this graph is connected (that is, whether this input graph has only one component is checked by implementing DFS algorithm) simultaneously. If there are any exceptions such as unconnected graph or unspecified argument text files, error message is printed out and program exits.

2). MST by Kruskal algorithm:

Edges of graph are sorted in increasing order by weight, implemented by min priority queue. After the disjoint sets are constructed by the number of nodes in graph, the smallest weighted edge is selected one by one and written in output text file in that order, if its two incident node do not belong to the same set at that moment and then a union set including them is updated: or that edge is just discarded so that a cycle would not be constructed in this spanning tree.

ⅱ. A complete explanation of the algorithm

Below are the explanations of what I intended to implement in this program.

1). Graph implementation

A structure graph is composed of two parts: the information of graph (i.e. number of nodes and edges) and a detailed list of header nodes, where each node indicates all of its adjacent nodes so that how these nodes are connected could be checked from any kind of incident nodes.

\*Since this input graph is regarded as undirected, the number of insertion is twice of the exact edges in input graph: a certain edge between A-B is regarded as A→B and B→A. This doubled insertion process is needed to confirm if there is no other component in this graph: in order to verify the number of components of this graph with DFS algorithm, the searching process is needed not only from [tail] dimension, but also [head] dimension.

2). Connectedness

Whether this input graph is connected or not is verified with DFS algorithm. If it is connected, all of nodes could be visited from one invocation of DFS. If not, DFS should be invoked several times in order to visit all of nodes. That is, the number of calling DFS is the exact number of components of that graph. That is, whether a certain graph is connected is verified by counting the number of components.

3). Kruskal algorithm

Before edges are sorted, redundant ones are eliminated so that not only doubly inserted edges (A→B and B→A: A→B) but also multiple edges (among several edges A→B, the smallest weighted edge is selected) for the sake of computational convenience.

From this new-made graph, which informs about how needed edges are constructed, edges are sorted in increasing order by weight through min-priority queue, where stored items are efficiently sorted and then popped.

\*even if the exact heap-size of this priority queue is same as the number of edges, it is limited to MAX\_SIZE, which is set irrelevant to the input number of edges for the sake of convenience: unintended error would occur if MAX\_SIZE is smaller so that some edition is needed if any.

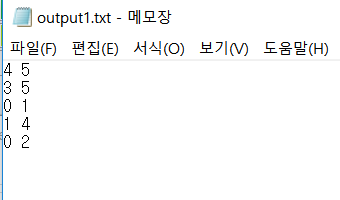
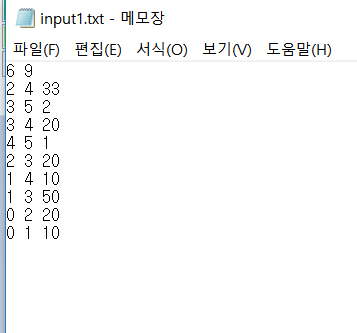
It is initialized that all of nodes are in each disjoint sets before edges are selected. In order to check whether a certain selected edge leads to a cycle in spanning tree or not, whether these two incident nodes of that edges belong to the same set is verified through Find and Union function: with the array Kruskal\_Parent, Kruskal\_Find function finds out a certain parent node, where value is identical to the index of Kruskal\_Parent and they are unified in the same set if they are in different sets at this moment. This Find-set and Union function is the main logic of Kruskal algorithm to select which kinds of edges should be inserted.

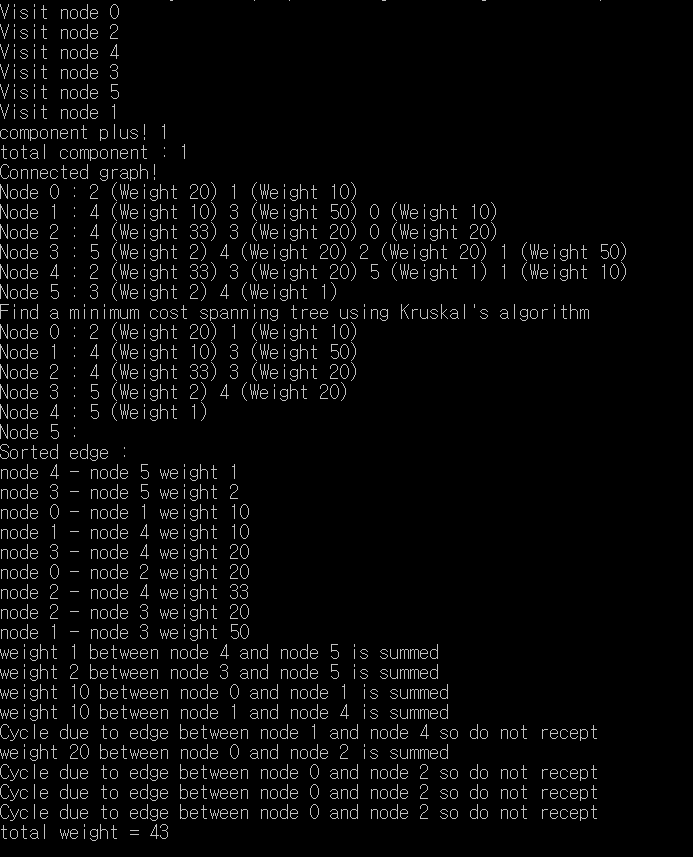
\*the size of Kruskal\_Parent is also set as MAX\_SIZE, which is irrelevant to the number of nodes in this case so that you may check whether it is smaller or not.

If selection occurs, that edge is also written in output text file so that the result of minimum spanning tree could be determined in this increasing order by weight of edges.

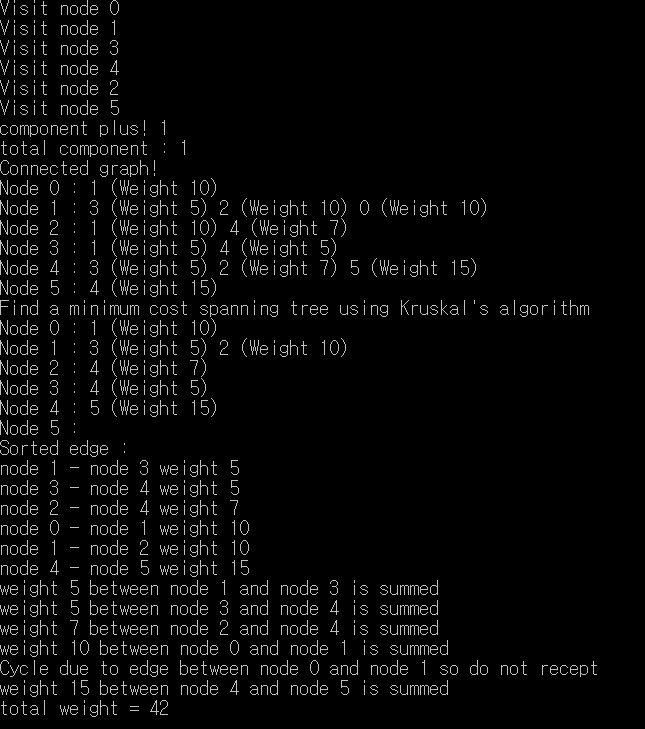
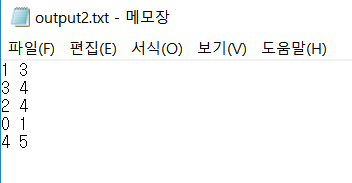
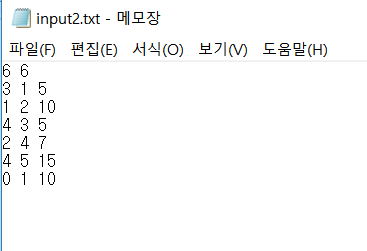
ⅲ. Result of program with screenshot

1). First test

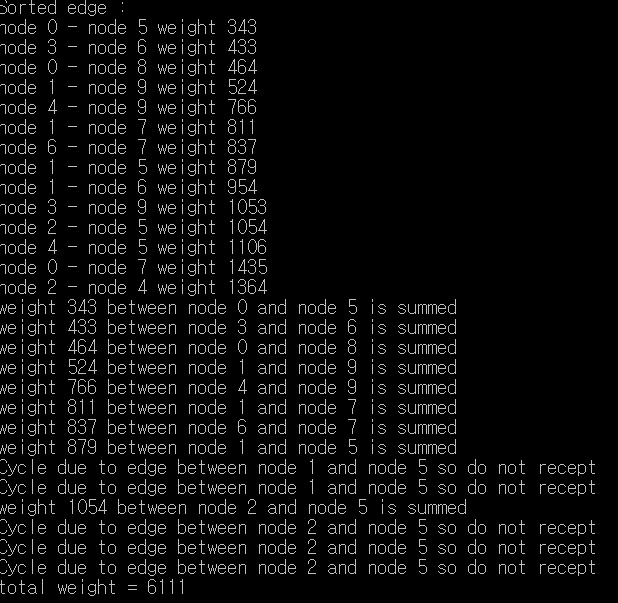
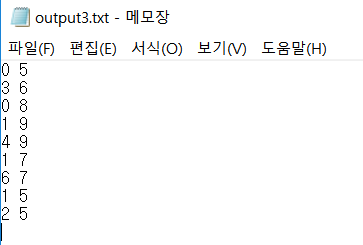
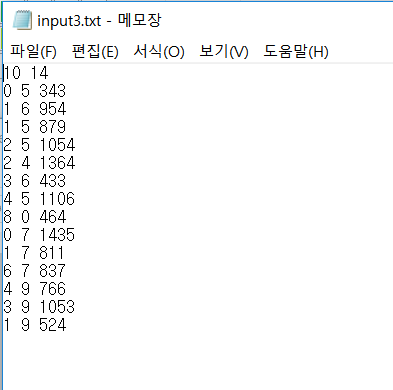




2). Second test



3). Third test



4). Exception cases

(1). Input.txt or Output.txt fopen failed



(2). Input graph is unconnected

