IDE： Dev C++

Platform: x64 windows

Team members: 10427124 柯秉寬 04106034 何政霖

Mission One：

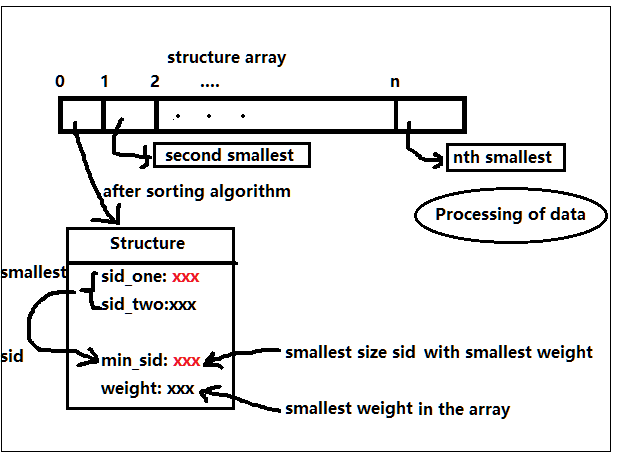
1. **Preprocessing of data**

Purpose:

Find all the nodes and edges in the maximum connected component. Sorting them by weight and the size of sid string

Details:

Put two endpoint and their weight into a structure. And then push this structure into a structure array. Repeat this action until the structure array contains all the nodes and edges. Sorting this structure array by weight and the size of sid string, so that the structure which has the minimum weight and shortest sid string will be located to the first position in the array. The structure behind the first structure decrease progressively.



1. **Algorithm Design**

Purpose:

1. One algorithm to sorting the structure array by weight and the size of sid string. First make the smallest weight to the first position in the array, and then, sorting the structures that are have the smallest weight by theirs the size of sid string. After this, we will make the structure which have both the smallest weight and the smallest size of sid string be located to the first position in the array. And the structure behind the fisrt structure decrease progressively.

Details :

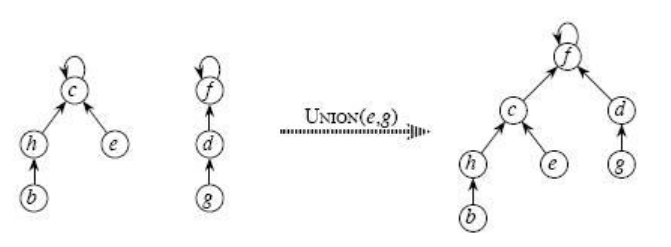
(1)Using quick sort algorithm sorting the structure array by weight

(2)And then using quick sort algorithm sorting the structure array’s subarray which has the same weight with different sid by the size of sid string

2、One algorithm to determine whether the edge to be added into the minimum spanning tree will case a loop. If will, determine the second structure in the structure array. Repeat this action until find a edge won’t cause a loop and put the edge into the minimum spanning tree.

Details:

Making all the linked sid with the same ancestor.( *i.e. all the sid in the same spanning tree has the same ancestor(sid)*). When we want to add a new edge to the minimum spanning tree,(*i.e. Merge two spanning tree into one*) if the two sid at the both end of the edage with the same ancestor, we cancel add this edge because it will cause a loop. Only two spanning tree with different ancestor can be merge into one.



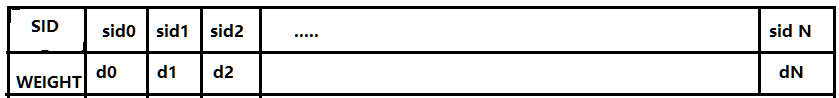
Mission Two:

Purpose:

Using Dijkstra’s Algorithm to find the shortest distance from the given sid to any sid it can reach.

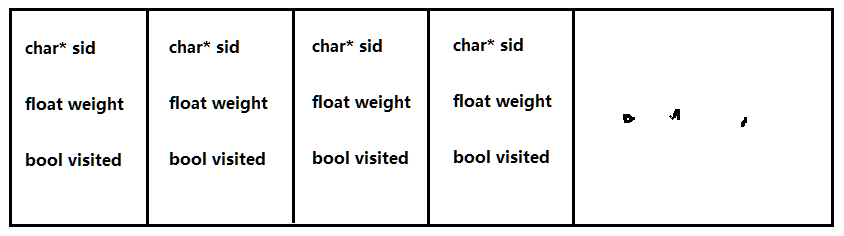
Details:

First of all, create a WEIGHT TABLE contains all the sid and the corresponding distance from the given sid to the sid. At the beginning, we initialize all the weight value to -1.



After this, we create another dynamic array pointer which contains the sid and the weight it correspond to we have recorded.(i.e. the field WEIGHT is not equal to -1)

Second, start from the given sid, record the distance from those sid which connected to the given sid to the given sid. Once record one sid, update the WEIGHT field in the table and add it into dynamic array, if the sid have already existed in the dynamic array, just update its weight equal to the table.



We use this dynamic array for sorting. Because the Dijkstra’s algorithm need to know that which node have the shortest distance from itself to the start node. We can sort this dynamic array by the weight value, and then choose the smallest weight and not visited.(a node have recorded the distance from all of the connected node to this node called is visited)

Third, choose the smallest weight and not visited node, and doing the same thing likes the second step.

Finally, when all the node in connectivity component have visited, the Dijkstra’s algorithm done. Printf the WEIGHT TABLE to screen.