Algorithms PA3 Report

B08901207 EE3 呂俐君

（A）Code Design Concept

This time I will introduce my code by different classes. They are edge, vertex, uGraph, dGraph, and Sort.

- edge
  there are 3 integer variables:
  s— representing the vertex number that the edge is out from.
  t — representing the vertex number that the edge is going into.
  w— representing the weight of this edge.

- Vertex
  there are 5 integer variables, an integer vector, and two functions.
  Vn— recording the number of the vertex.
  set— Used in undirected graph, for Kruskal algorithm make set&find set.

  pre— recording the predecessor number of that vertex when running DFS.
  postInAdj— recording the successor's index in adj list of the vertex.
  color— representing the color the vertex when running DFS. 0=white, 1=gray, 2=black
  adj— the adj list of the vertex.

  makeSet(num)—make the set number = num;
  init( )— set the pre=-1(none); color=0(white)

- uGraph/dGraph common variables&functions
  undirected graph.
  V— number of vertices.
  E— number of edges.
  Vlist— an array of vertices, recording the vertices of the graph.
  Elist— a vector of edges, recording the edges of the graph.
  minCost— accumulating the minimum cost of deleting cycles.
  A/B— an edge vector recording the deleted edges (result).
  ConstructEdge( )— get inputs and put the given edges into Elist and build adj list.
  Print( )— output the final minCost and deleted edges.

- uGraph
  undirected graph.
  Kruskal( )— run Kruskal algorithm to fund "Maximum Spanning Tree", and put the left edges into A.

- dGraph
  directed graph.
  foundCycle— a boolean variable to record if there is still cycle in the graph.
  lMC— local minimum cost: recording the cost of min weighted edge in a cycle.
  lME— local minimum edge: recording the position of the min weighted edge in a cycle.
  Ematrix[i][j]— a 2D array recoding the number of edge from vertex I to vertex j.

init( )— initialize IMC, IME, and the color of vertices after breaking an edge.
DFS(Vn)— run Depth First Search to look for cycle. Check the color of each vertex making sure that none of them is ignored.

DFS_Help(Vn)— if the vertex detected in the previous function is white, run DFS_Help from that vertex Vn, to see if there exist any backedge that forms a cycle. If yes, call ListCycle.
ListCycle— from the re-visited vertex, list out the cycle from calling back their predecessors. During the process, compare the edges' weights and delete the min in the end.

- Sort
  Doing Heap Sort in Kruskal algorithm to sort the edges by their weights descendingly.

- Main

(a) Basic Settings
By #include <fstream> and int main(int argc, char* argv[]), we are able to read data from the specified file and create another file to store the outputs.

(b) Inputs
In the beginning, we f-in the first line getting character "u/d" to decide which kind of graph to build.
Second, we create a corresponding graph with V vertices and E edges ,and construct the edges information.

(c) Error Checking
Since the last input is a single zero, we simply read it as a checking message to see if the data input properly.

(d) Calculating& Outputs
By each function introduced above, we found the resulting answer to the problem, then print them out to the output file.

（B）Reflection

This is a cool problem that we can applied the theorem we learnt in class. I tried to implement the algorithms with slightly modification, from which found some difficulties to realize the part seems to be simple in pseudo code. For example, make set and find set. Instead of giving several vertex-sets and recombine them, I assigned "set number" to each vertex, and used a 2D vector to record the elements' indices of each set.
Another challenging part is DFS. I need to break all the loops when a cycle is found. However, because of the iterative implementation, I could only break a single loop. In the end, I accomplished this job with the help of boolean variable- foundCycle.
I like doing programming assignments though it really takes a lot of time debugging and sometimes being very frustrating.