CS69003: Computing Systems Lab 1
Autumn 2006

Assignment 2

Implementing a Graph ADT and some graph algorithms

Due: August 16

This assignment has two parts.

Part 1:
In the first part of this assignment, you will write a C program that will generate a random undirected weighted graph. The program will take as command line arguments four parameters (in this order) – no. of nodes (*n*), a probability (*p*), maximum weight ($w > 0$), and a file name (*f*). The program will generate a undirected weighted graph with *n* nodes numbered from 1 to *n*, with an edge between each pair of nodes *i* and *j* generated with probability *p*, and the weight of an edge being a positive integer value randomly chosen between 1 and *w*. The generated graph will be written in the file named *f* in exactly the following format:

line 1 - <no. of nodes>
line 2 - <edge probability used>
line 3 - <max. weight used>
Followed by all the edges, with each edge in a separate line written as the two node ids followed by the weight (i.e., a sequence of 3 integers in each line)

Name the file <your roll no>_graph_gen.c (for example, 06CS1004_graph_gen.c).

Part 2
In this part, you will implement an ADT called GRAPH that can store a graph of arbitrary number of nodes and edges. The GRAPH ADT will support the following operations:

1. *int CreateGraph(GRAPH *G, char *inp_file)* – reads in a graph from the file named *inp_file* in the graph *G*. The file *inp_file* should have a graph in the above format. Returns 0 if graph is read successfully, -1 otherwise (for ex., file not present)
2. *int NoOfConnComp(GRAPH G)* – returns the number of connected components of *G*
3. *int SizeOfLargestComp(GRAPH G)* – returns the size (number of nodes) of the largest connected component of *G*
4. *int IsConnected(GRAPH G)* – returns 1 if *G* is connected, 0 otherwise
5. *int IsTree(GRAPH G)* – returns 1 if *G* is a tree, 0 otherwise
6. *void MST(GRAPH G)* – if the graph is connected, finds the MST using Kruskal's algorithm and prints the edges on the screen (one edge on each line, with each edge printed with three integers as earlier). If the graph is not connected, a

message is printed saying that “the graph is not connected and hence no spanning tree exists”.

7. *void Destroy(GRAPH *G)* – releases any allocated memory for G. if no memory is allocated, does nothing and returns.

Design appropriate data structures to define GRAPH and implement the above functions. Your final output for this second part of the assignment will be two files:

1. A .h file containing the type definition for GRAPH and any other type definition you may need. Name the file <your roll no.>_graph.h (for ex. 06CS1004_graph.h)
2. A .c file containing the implementation of the above functions that can be compiled into a static library. This file will also contain any other function that you may write to implement the above functions. For information on how to create a static library, look up the linux manpage for the “ar” command. You do not need to actually submit the static library, we will create it while evaluating. Name the file <your roll no.>_graph.c (for ex., 06CS1004_graph.c).

We will test your program by creating another C file that has a main function to call the above functions. This C program will be linked with your static library while compiled. You should think about what all needs to be tested in your program and write a main function to test it appropriately.

It is very important that you follow the above file naming conventions and function prototypes EXACTLY as the evaluation will be done by a program that will assume these. Any error arising out of deviations from above will incur severe penalty in marks.