EECS 477. HOMEWORK 4.

Due on Thursday 10/17/2002 before noon
in mailbox labeled 477 in room 2420 EECS

YOUR NAME: ____

You must show all work to receive credit!
Please read the statement below and sign your name; otherwise, your homework will not be graded. The text of the College of Engineering's Honor Code can be found at http://honor.personal.engin.umich.edu/

I hereby acknowledge that I understand the College of Engineering's Honor Code and have pledged to uphold it and abide by it.

Signature: ____

1. Random Graph Generation and Visualization (10pts)

In this assignment you will need to generate random undirected graphs and visualize them.

First, generate N random points uniformly distributed within the unit square $[0,1] \times [0,1]$. These points will form the vertex set V of your graph $G = (V, E)$. Then randomly generate edges so that the probability of an edge connecting two vertices v and v' is given as:

$$P(\{v, v'\} \in E) = \begin{cases} H, & \text{when } \rho(v, v') < D, \\ He^{A(D-\rho(v,v'))} & \text{when } \rho(v, v') \geq D. \end{cases}$$

Here $\rho(v, v')$ denotes the Euclidean distance between the points v and v'. $0 \leq H \leq 1$, $A > 0$, and $D > 0$ are constants.

Submit the printout of your algorithm and the graph plots with parameters set to $N = 1000$, $H = 0.8$, $A = 25$ and different values of $D = 0, D = 0.1, D = 0.5, D = 1$. You may use `gnuplot` for the visualization.

2. k-ary heap Dijkstra algorithm(25pts)

Implement Dijkstra algorithm from Section 6.4 (pp.189–202) with k-ary heap (see Problem 6.16 for the details). You will need to implement your own k-ary heap functionality (see Problem 5.23). The input graphs for the Dijkstra algorithm will come from the first part of this homework: you will use undirected graphs with edge lengths generated randomly so that they are distributed uniformly in $[0, 1]$.

Submit a printout of your program together with two of example runs on some simple graphs: assume that the first vertex is the starting one, then the result will be the distance assignment from that vertex.

Analyze the asymptotic runtime of your algorithm both theoretically and experimentally. Plot the runtime of the algorithm varying the number of vertices and edges in your graph (one way to leave the number of edges approximately constant is to double the number of vertices and at the same time divide the constant H by four in your graph generation procedure).

Change the value of the heap parameter k between 2 and $k' = \max(2, \lfloor a/n \rfloor)$ (do it gradually in ten increments of the size $\lfloor (k' - 2)/10 \rfloor$. Do you see the performance boost as expected from the description in Problem 6.16? Plot the resulting performance plot (runtime against k) for a graph with the number of vertices greater than 5,000 and the number of edges greater than 100,000. Try to match your asymptotic analysis and the observed performance.

3. Linear inhomogeneous recurrencies (20pts)

Solve the following recurrencies exactly and express your answer as simply as possible using Θ notation.

A.

$$t(n) = 5 * t(n-1) - 4 * t(n-2) + 3 * 2^n,$$
$$t(0) = 1,$$
$$t(1) = 2.$$

B.

$$t(n) = 5 * t(n-1) - 4 * t(n-2) + (n+1) * 4^n,$$
$$t(0) = 1,$$
$$t(1) = 2.$$

4. Master theorem (20pts)

Analyze the following recurrencies using the master theorem and express your answer as simply as possible using Θ notation.

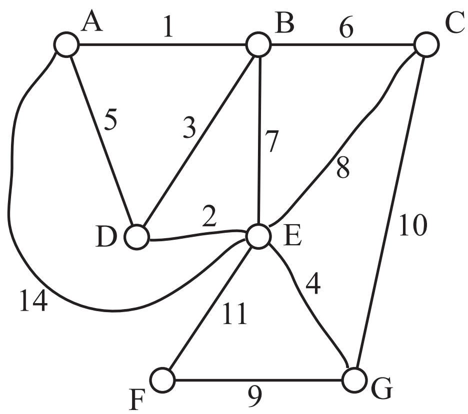
A. $t(n) = 4t(n/3) + n^2$

B. $t(n) = 4t(n/3) + n$

C. $t(n) = 3t(n/3) + \log n$

D. $t(n) = 2t(n/3) + \log n$

5. MST and Union-find (25pts)



In this part, we will be looking for the minimum spanning tree of the undirected graph above.

Give the order in which edges will be added to the MST by Prim's algorithm. Start the algorithm from vertex A.

Repeat for Kruskal's algorithm.

Show the final configuration of the disjoint set structure tree at the end of Kruskal's algorithm (use the disjoint set structure with path compression, and in case of ties, use the alphabetically larger node as the root).