CPSC-629 Analysis of Algorithms

Fall 2014

Instructor: Dr. Jianer Chen
Office: HRBB 315C
Phone: 845-4259
Email: chen@cse.tamu.edu
Office Hours: T,Th 4:00pm-5:00pm
Teaching Assistants:
Folami Alamudun (fola@cse.tamu.edu): Office Hours: M,Th 3:00-4:00pm (TEAG 327C)
Jiacheng Gu (jcgu@tamu.edu): Office Hours: M,W 3:00-4:00pm (HRBB 315A)

Course Project

Network optimization has been an important area in the current research in computer science and computer engineering. In this course project, you will implement a network routing protocol using the data structures and algorithms we have studied in class. This provides you with an opportunity to translate your theoretical understanding into a real-world practical computer program. Translating algorithmic ideas at a "higher level" of abstraction into real implementations in a particular programming language is not at all always trivial. The implementations often force you to work on more details of the algorithms, which sometimes may lead to a much better understanding.

Your implementation should include the following parts:

1. Random Graph Generation. Write subroutines that generate two kinds of "random" graphs of 5000 vertices.

- In the first graph G_1, every vertex has degree exactly 6;
- In the second graph G_2, each vertex has edges going to about 20% of the other vertices;
- Randomly assign positive weights to edges in the graphs.

2. Heap Structure Write subroutines for the min-heap structure. In particular, your implementation should include subroutines for MINIMUM, INSERT, and DELETE. See Homework #2, Problem 3.

Since the heap structure you implement will be used for a Dijkstra-style algorithm in the routing protocol, we suggest the following data structures in your implementation:

- The vertices of a graph is named by integers 1, 2, ..., n;
- The heap is given by an array $H[1..5000]$, where each element $H[i]$ gives the *name* of a vertex in the graph;

- The vertex "values" are given in another array $D[1..5000]$. Thus, to find the value of a vertex $H[i]$ in the heap, we can use $D[H[i]]$.

3. Routing Algorithms Your algorithms are to solve the MAX-CAPACITY-PATH problem for which you need to find a path of the maximum capacity (i.e., the maximum *bandwidth*) between two vertices in a given weighted undirected graph. See Homework #2, Problem 1. You should have three different versions of implementations:

- An algorithm for MAX-CAPACITY-PATH based on a modification of Dijkstra's algorithm *without* using a heap structure;
- An algorithm for MAX-CAPACITY-PATH based on a modification of Dijkstra's algorithm using a heap structure for fringes;
- An algorithm for MAX-CAPACITY-PATH based on a modification of Kruskal's algorithm, in which the edges using sorted by HeapSort. See Homework #3, Problem 3.

4. Testing. Test you routing algorithms on 5 pairs of graphs, randomly generated using your subroutines implemented in Step 1. For each generated graph, pick at least 5 pairs of randomly selected source-destination vertices. For each source-destination pair (s, t) on a graph G, do the following:

- add a path from s to t that goes through all vertices in the graph G — this is to ensure that there are always paths connecting s and t, and randomly assign positive weights to the new edges on the path;
- Run each of the three algorithms on the pair (s, t) and the graph G, and record their running time (you should find a proper way to "count" the running time of an algorithm).

5. Report. Write a report of at least 5 typed pages, which explains your implementation details, and discusses and analyzes the performance of your routing algorithms on different kinds of input graphs. The data you record in Step 4 for the algorithm performance should also be given here. Also, if possible, discuss any possible further improvements on data structures, algorithms, and implementations.

Project Submission Deadline. December 2, 2014