CSCE-629 Analysis of Algorithms

Fall 2017

Instructor: Dr. Jianer Chen
Office: HRBB 315C
Phone: 845-4259
Email: `chen@cse.tamu.edu`
Office Hours: T,Th 2:00pm-3:30pm

Teaching Assistant: Qin Huang
Office: HRBB 315A
Phone: 402-6216
Email: `huangqin@email.tamu.edu`
Office Hours: T,Th 9:00am-12:00noon

Course Project

(Due November 30, 2017)

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 much better understanding.

Your implementation should include the following parts:

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

- In the first graph G_1, the average vertex degree is 8;
- In the second graph G_2, each vertex is adjacent to about 20% of the other vertices, which are randomly chosen;
- Randomly assign positive weights to edges in the graphs.

Your graphs should be "random" enough. Therefore, in the graph G_1, the pairs of vertices that are adjacent should be chosen randomly, and in the graph G_2, the number of neighbors and the neighbors of a vertex should be chosen randomly. To make sure that your graphs are connected, I suggest that you start with a cycle that contains all vertices of the graphs, then add the rest edges randomly.

2. Heap Structure Write subroutines for the max-heap structure. In particular, your implementation should include subroutines for MAXIMUM, INSERT, and DELETE. See Homework #1, Problem 4.

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 are 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-BANDWIDTH-PATH problem for which you need to find a path of the maximum bandwidth between two vertices in a given weighted undirected graph. You should have three different versions of implementations:

- An algorithm for MAX-BANDWIDTH-PATH based on a modification of Dijkstra's algorithm *without* using a heap structure;
- An algorithm for MAX-BANDWIDTH-PATH based on a modification of Dijkstra's algorithm using a heap structure for fringes;
- An algorithm for MAX-BANDWIDTH-PATH based on a modification of Kruskal's algorithm, in which the edges are sorted by HeapSort.

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, 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.

6. Further Research. You may consider also implementing, testing, and analyzing the linear-time algorithm for MAX-BANDWIDTH-PATH, which we have discussed in detail in class. This requires significant extra work, including implementing Median-Finding, Connected-Component, and other necessary algorithms. This item is *not* required, but will be interesting for the study of practical performance of this theoretically best algorithm for the problem.