

CSE 326: Data Structures
Assignment #6
November 17, 2008
due: Wednesday, November 26, 11:30 a.m.

The purpose of this programming assignment is to implement Dijkstra's algorithm using heaps. Here are the details.

1. Implement a heap. You may assume that the heap never has more than 500 entries. You may assume that the Key and Info fields each have type integer. You will implement all the priority queue operations of Section 9.1, plus the extra operation $\text{DecreasePriority}(i, K)$, which decreases the key at heap index i to the value K , making any necessary changes to the heap as a result of this decrease. (If K is greater than the current key at this index, the operation does nothing.) Be sure that your implementation of this new operation runs in time $O(\log N)$, where N is the current heap size.
2. Implement Algorithm 12.6 on page 449 using a heap to manage the Distance values for the members of U . Your implementation should run in time $O((n + e) \log n)$, where n and e are the numbers of vertices and edges, respectively, of the input graph. To do this you will need to represent the graph by its adjacency lists. You may assume that vertices are numbered consecutively starting with 0 and that vertex 0 is the source, so that you are computing the least cost paths from vertex 0 to every vertex in the graph.

The command line that runs your program should take two filenames as parameters, the first being the input file and the second the output file. The input file that describes the directed graph will have the following format:

```
0:2,15;5,3;
1:
2:0,3;
...
```

where, in this example, the neighbors of vertex 0 are vertices 2 and 5, with $c(0,2)=15$ and $c(0,5) = 3$, and vertex 1 has no neighbors.

The output file will have the following format:

```
1:Distance(1),Parent(1)
2:Distance(2),Parent(2)
3:Distance(3),Parent(3)
...
```

where $\text{Distance}(i)$ is the cost of a least cost path from 0 to i , and $\text{Parent}(i)$ is the vertex that precedes i on that least cost path. (From this representation, one can easily reconstruct the entire least cost path by working backwards from i to 0.) If there is no path from 0 to i , the i -th output line should read "i:infinity,-".