Tim Cheeseman
CS 260-003
Summer 2014
Assignment 8 Sample Solution

6.6)

The program *Dijkstra* is a greedy algorithm; each time we add a vertex *w* to *S* we are concluding that we have found the shortest path from the source to *w*. This conclusion is easily seen to be true if we have non-negative arc costs: if the path from the source to all other nodes is longer than the path from the source to *w*, then a path to *w* that goes through the other longer paths could not possibly be shorter. If, however, there could be negative arc costs, then it is entirely possible that one of the other paths could improve to become a shorter path to *w*, but the greedy nature of *Dijkstra* will not find those paths.

7.1)

```
# assume ajd_list is a list of lists of integers, where v and w are
# in 1..n for a graph with n vertices.

def insert_edge(v, w):
  adj_list[v].insert(w) # insert w at the front of v's list, O(1)
  adj_list[w].insert(v) # insert v at the front of w's list, O(1)

def delete_edge(v, w):
  adj_list[v].remove(w) # remove w from v's list, O(n)
  adj_list[w].remove(v) # remove v from w's list, O(n)
```

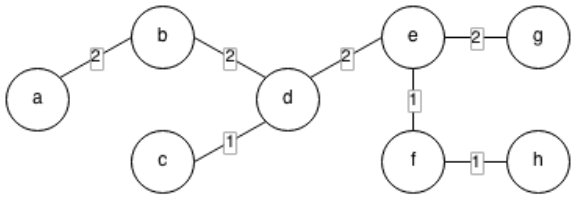
7.2)

```
# assume ajd_list is a list of lists of tuples of the form (z, e),
# where z is the other vertex in the edge and e is a reference to the
# complementary edge

def insert_edge(v, w):
  # create two edges that reference each other, O(1)
  e1 = (w, None)
  e2 = (v, e1)
  e1[1] = e2

  # insert the edges into the adjacency list
  adj_list[v].insert(e1) # insert w at the front of v's list, O(1)
  adj_list[w].insert(e2) # insert v at the front of w's list, O(1)
```

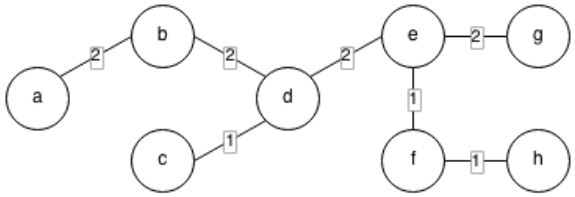
```
def delete_edge(v, w):
  # assume remove can take a reference to an element in the linked list
  # and delete it in O(1) time; this doesn't exist in python's native list
  # but is easily done in a linked list implementation

  # find w's edge in v's list, O(n) (this is O(1) if v->w is the first
  # edge in v's adjacency list)
  e1 = adj_list[v].find(w)
  e2 = e1[1]

  adj_list[v].remove(e1) # remove w from v's list, O(1)
  adj_list[w].remove(e2) # remove v from w's list, O(1)
```

7.3)

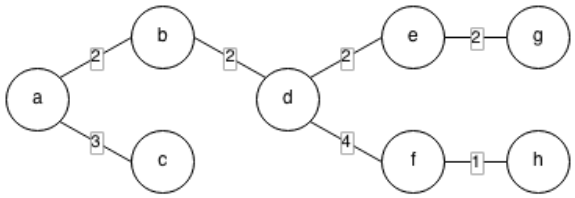
a)



b)



d i)



d ii)