CPS 430/590.06 **Prof. John Reif**
Design and Analysis of Algorithms **Fall 2013**
Homework 5 **Matt Dickenson**

Question 1

We can develop an algorithm to find the shortest path between vertices a and b in a weighted-edge (possibly negative) directed graph, with a guarantee that there will be at most k edges in the shortest path. To develop this algorithm we can use a "greedy" design.

The algorithm takes as input the graph, represented by its vertices and edges, as well as the vertices of interest a and b. We can use two arrays (d and p) to store the distance and path information. We also rely on the special values `null` to represent the empty set and `MAX` to represent $+\text{inf}$.

```
findShortestPath(a, b, edges, vertices):

# initialize with infinite distances from a to each node
predecessor = [null] * vertices.length
distance = [MAX] * vertices.length
distance[a] = 0

# greedy step
for i in 1..k:
  for e in edges:
    if distance[e.source] + e.weight < distance[e.destination]:
      distance[e.destination] = distance[e.source] + e.weight
      predecessor[e.destination] = e.source
```

This algorithm works because we are guaranteed to find the shortest path that is no longer than k edges by the greedy step. Our double loop takes $k \times |E|$ iterations, for a running time of $O(k|E|)$ as desired.

Question 2

Question 3

Question 4