Consider the following greedy strategy for finding a shortest path from

vertex *start*to vertex *goal*in a given connected graph.

1: Initialize *path*to *start*.

2: Initialize set *visited*to {*start*}.

3: If *start=goal*, return *path*and exit. Otherwise, continue.

4: Find the edge (*start,v*) of minimum weight such that *v*is adjacent to

*start*and *v*is not in *visited*.

5: Add *v*to *path*.

6: Add *v*to *visited*.

7: Set *start*equal to *v*and go to step 3.

Does this greedy strategy always find a shortest path from *start*to *goal*?

Either explain intuitively why it works, or give a counterexample.

Step 1:

This avaricious approach does not identify the "shortest path" from beginning to end since there is no assurance that the choice that seems optimal at the moment (affected by only incomplete information) will actually be the best one. We must have comprehensive knowledge of the paths in order to compare them with regard to their weights; only then can we compute and contrast the weights of the paths.

Step 2:

Consider the following basic graph, where the edges are represented by the triplets initial vertex, final vertex, and weight.

(start, u, 2)

(start, p, 5)

(u, v, 5)

(p, goal, 2)

(v, goal, 3)

Thus, there are two ways to get from here to there:

start -> u -> v -> total weight (2+5+3) = 10 as the goal

start, p, and total weight goal (5+2) all equal 7.

We will ultimately choose u as the next vertex in step 4 if we follow the greedy strategy, which means we will never be able to find the shortest path using this method.

The following would be a smaller counter example:

(start, u, 2)

(start, goal, 3)

(u, goal, 3)

Start -> u -> goal with total weight (2 + 3) = 5 is the first path.

Path 2: begin to the objective with the total weight 3.

We will, however, choose u as the next vertex on the (supposedly shortest) path from start to goal in step 4 due to our greedy strategy, and we will never be able to obtain the shortest path from start to goal.