CS 59000-NMC, 8 September

Please answer the following questions. You may not use any outside references or technology. Justify and explain all answers. This quiz is for my own evaluation, so that I can provide better instruction in the course.

Question

Consider the following matrix-based algorithm for breadth-first search

```
Input:
  A - adjacency matrix
  v - starting vertex

d(i) <- Inf for 1 <= i <= n
x <- e_v % i.e. x is a vector of all zeros with a 1 in the $v$th's position

for step = 1 to n
  y = A*x % do a mat-vec
  newv = {i : y(i) != 0 && d(i) == Inf } % set of newly found vertices
  d(newv) <- step % assign the current distance
  x <- x | y % set x
  if |newv| == 0 then quit
~~~~
```

```
What is the best bound you can put on the runtime of the algorithm,
using any graph property you wish?

Solution
--------

First, each iteration of the for-loop takes $O(|V| + |E|)$ work
because a mat-vec is $O(|E|)$ work and the other operations
take $O(|V|)$ work.  Consequently, the question is really asking
how many iterations the algorithm will perform.  The best
bound for this would be the _eccentricity_ of the starting
vertex $v$.  Eccentricity is the longest-shortest path from
a given vertex to any other vertex.  The diameter of the graph
is the largest eccentricity of any vertex.  Hence, ignoring
the identity of the starting vertex, the algorithm could run for
diameter iterations.

The runtime is then bounded by
$O(d \cdot (|V| + |E|))$ where $d$ is the diameter.

Here are a few special cases for the runtime

Chain graph
: Suppose your graph is just a chain of n connected vertices.
  Then the diameter is n, and each iteration of the loop takes O(n)
  work, for a total runtime of O(n^2).

Clique graph
```

: Suppose your graph is a clique, then there are n^2 edges, but the
  diameter is 1.

Clique and chain graph
: Suppose the graph is a clique on $n/2$ vertices, and a single chain
  of $n/2$ vertices. Then there are $O(n^2)$ edges and a diameter of $O(n)$.
  Yielding a runtime of $O(n^3)$.

Specializing the algorithm.

A number of students tried to _optimize_ the algorithm by using additional
properties to get back to the standard $O(|V| + |E|)$ runtime. e.g.
changing the algorithm so that ‘x = e_newv‘, or just the non-zeros
for the new set of vertices, and then using a sparse-matrix-sparse-vector
product ‘A*x‘ will do this. This was greatly insightful, but was a
bit outside the scope of the answers. Such thinking will be the core
of how to do the forthcoming _local_ algorithms, however.