Aditya Shirode, Rhythm Shah, Shrenuj Gandhi

1. [2 points each for (a) and (b), 6 points for (c)]

   Suppose that we wish to maintain the transitive closure of a directed graph $G = (V, E)$ (with $n = |V|$) dynamically in response to edge insertions. That is, after each edge is inserted we want to update the transitive closure of the graph having the edges inserted so far. Assume that G starts out with no edges and that the transitive close is represented as a Boolean matrix.

   (a) Prove that the transitive closure $G^* = (V, E^*)$ of graph $G = (V, E)$ can be updated in time $O(n^2)$ when a new edge is inserted. That is, describe an algorithm for doing the update within the desired time bound.

   (b) Describe an infinite collection of examples of graphs G and edges uv such that, for some constant c, at least cn^2 operations are required to update the transitive closure of G in response to inserting edge uv. In other words, show that the update time for inserting a single edge is $\Omega^{\infty}(n^2)$.

   (c) Describe an efficient algorithm for updating the transitive closure as edges are inserted into a graph. For any sequence of k insertions, your algorithm should run in total time $\sum_{i=1}^{k} t_i \in O(n^3)$, where t_i is the time taken by the i-th insertion. This does not contradict the result of (b), which only applies to a single insertion, even if $k \in \omega(n)$. Also, note that part (a) does not help if $k \in \omega(n)$.

Answer:

(a) Let G^* be the boolean matrix that represents the transitive closure of a directed graph $G(V, E)$. Since G starts out with no edges, initially G^* is identity matrix. This implies that there is a path from each vertex itself.

The algorithm to update G^* when a new edge (u, v) is inserted is as follows -

```
Algorithm(G*, u, v):
        G*[u, v] ← 1

        for row←1 to n, do:
                if G*[row, u] = 1, then:
                        for col←1 to n, do:
                                if G*[v, col] = 1, then:
                                        G*[row, col] ← 1
end
```

The first loop gives us all the vertices which have 1 in the column u.
The second loop gives us all the vertices which have 1 in the row v.
The new edges in the transitive closure will then be the pairing of all such vertices.

Because of the two nested loops, the complexity of above function is $O(n^2)$, where $n = |V|$.

(b) Consider our graph to consist of two disjoint graphs, each of size $|V|/2$.
Assume that each subgraph is complete *i.e.* there are edges from each vertex in the subgraph to every other vertex in the same subgraph.

If an edge (u, v) is then added that connects a vertex from first subgraph to a vertex in second subgraph, then we need to update the transitive closure for all vertices in first and second subgraph.

The first loop will give us $|V|/2$ vertices and the second loop will also give us $|V|/2$ vertices.
Hence there will be $n^2/4$ updates in the transitive closure.

Thus regardless of the algorithm, we can come up with infinite such examples where we can add an edge from first connected subgraph to second connected subgraph, giving us an update time of $\Omega^{\infty}(n^2)$.

(c) Initially, the graph G contains n vertices and no edges. In other words, G contains n strongly connected components.

To make n-strongly connected components into one strongly connected component, we need $2(n-1)$ edges for n directed graph.

Now consider the following algorithm -

```
Update(G*, u, v):
        if  G*[u, v]! = 1:
                Algorithm(G*, u, v)                    ▷Ref.(a)
end
```

Here, at most $2n-2$ edges will go through the algorithm discussed in (a).
The rest of the edges will take O(1) to detect their presence in G^*.

So even for k insertions, total time will be -

$(2n-2)n^2 + k \in \mathrm{O}(n^3)$ where $k < n(n-1)$.