


# Problem 2: Not Another Pandemic

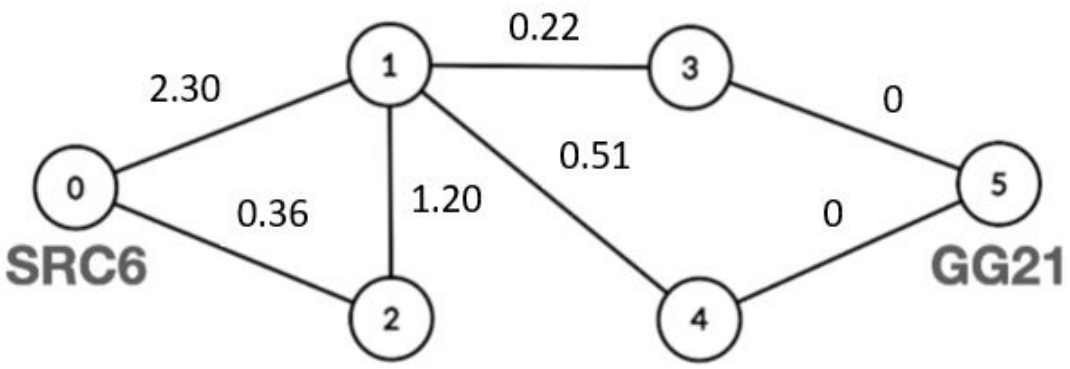
By Nalin Ranjan

This problem wasn't meant to be difficult, but in hind sight we see that the problem statement may have been too verbose and hard to parse. Abstracting all the specifics of the setting away, we just want to be able to find the path from 0 to $N$ that maximizes the product of every other edge weight, which are given to all be in the range $[0.0001, 0.9999]$.

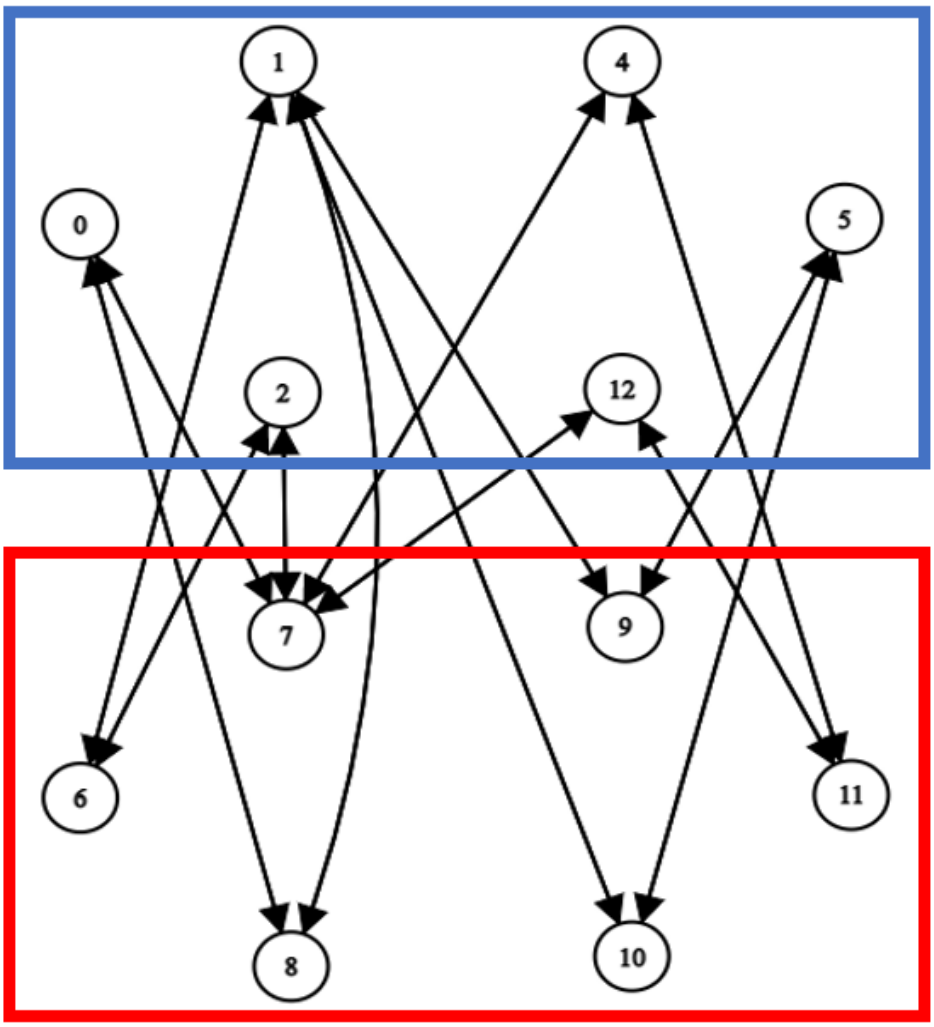
You would have first read in the input as an *undirected* graph, where the vertices are the proteins (and GG21) and the edges are the interactions. To solve the problem, we need to transform the input so as to make this an ordinary shortest paths problem.

- First, you were given the probability of *failing* for each edge, but the quantity we wish to calculate is actually the product of probabilities of *succeeding* for every alternate edge. So we should first replace every edge weight with one minus itself.
- Second, as maximizing the product of the weights is equivalent to maximizing the sum of their logarithms, we the replace each edge weight with its logarithm.
- However, taking the logarithm of any weight yields a negative number (since all weights are less than 1). To maximize the sum of logarithms, we need to find the *least negative* path, which is equivalent to finding the shortest path if we negate all the edge weights.

The effect of these transformations is to take each edge weight $w$ of the original graph, and replace it with $-\ln(1-w)$. After these transformations, example 0 looks like (edge weights are approximate)



At this point, we just need to find the path that minimizes the sum of every alternate edge weight, starting with the first one. We can turn this into an ordinary *directed* shortest paths problem on a graph with twice the vertices and four times the edges. Our new graph will in effect "duplicate" each vertex, and for each edge $(u, v)$ with weight $-\ln(1-w)$ in the previous graph, we add four *directed* edges: 1) an edge from $u$ to $v+N$ with edge weight $-\ln(1-w)$, 2) an edge from $v$ to $u+N$ with edge weight $-\ln(1-w)$, 3) an edge from $u+N$ to $v$ with edge weight 0, and 4) an edge from $v+N$ to $u$ with edge weight 0. So the graph from above would become






You can see that this new graph is *bipartite*, i.e. every directed edge is from a vertex in the blue (top) box to the red (bottom) box, or vice-versa. We've omitted the edge weights so as not to clutter the graph — just remember that whenever you go from a vertex in the blue (top) box, you incur a nonzero cost, but when you go from a vertex in the red (bottom) box to the blue (top) box, you incur a zero cost. (Here cost = edge weight.)

Every path in this new bipartite graph that starts at vertex 0 and ends at either vertex $N-1$ or $2N-1$ corresponds directly to a path in the original graph. For example, the path

$$0 \longrightarrow 7 \longrightarrow 4 \longrightarrow 11$$

on the bipartite graph corresponds to the path

$$0 \longrightarrow 1 \longrightarrow 4 \longrightarrow 5$$

in the original. All we've done is replace any vertex $v$ with label greater than or equal to $N$ with $v - N$.

Thus, if run a shortest paths algorithm (e.g. Dijkstra) that starts at vertex 0 in the new graph and stop once we see either vertex $N-1$ or vertex $2N-1$ (which are the two "copies" of the vertex corresponding to GG21), we can see that the sum of weights of the path we took is exactly the sum of every other edge weight of the corresponding path in the original graph.

Once we've found the shortest path starting from vertex 0 and ending in either vertex $N-1$ or $2N-1$ in this bipartite graph, all we need to return is the negative of its weight. This will be exactly the product of every other edge weight of the most resilient path in the original graph.

**Time Complexity**: $\mathcal{O}(M \log N)$, with the bottleneck being Dijkstra's algorithm.

**Plaudits**:

- Congratulations to Antonio Molina (grad) for being the first to solve this problem in an impressive 32 minutes!