## **Assignment 1**: Solutions

TOTAL MARKS: 0

1. Solutions:

   (a) Problem a1898629
   Edge list: (2,1) (2,4) (6,4) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

   (b) Problem a1897413
   Edge list: (2,4) (2,6) (4,3) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

   (c) Problem a1932791
   Edge list: (2,4) (4,2) (6,1) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

   (d) Problem a1871789
   Edge list: (1,4) (1,6) (5,4) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

   (e) Problem a1734046
   Edge list: (1,2) (1,3) (4,1) (5,2) (6,2) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(f) Problem a1820798
Edge list: (3,2) (4,2) (6,4) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(g) Problem a1825938
Edge list: (1,4) (4,2) (5,6) (6,1) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(h) Problem a1813487
Edge list: (1,3) (3,6) (5,2) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(i) Problem a1871781
Edge list: (2,1) (3,2) (3,6) (5,2) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(j) Problem a1907611
Edge list: (1,2) (1,5) (5,3) (6,2) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(k) Problem a1709218
Edge list: (2,1) (3,1) (3,6) (6,5) Adjacency matrix:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

2. For a fully-meshed (undirected) network with $|N|$ nodes the number of links is $|N|(|N|-1)/2$. Determine the number of loop-free paths $|P|$.

   **Solution:** Let us consider the number of paths $|P_{ij}|$ between one pair of nodes $i$ and $j$. We consider these paths in order of length. For convenience of notation, let $n = |N|$.

   - **length 1:** there is exactly one path of length 1 (the direct path).
   - **length 2:** for $n \geq 3$ we may use each of the nodes $k \neq i, j$ only once on the path, and so there are $n-2$ possible such paths.
   - **length 3:** for $n \geq 4$ we may use any pair of the nodes $k \neq i, j$ in a path, in any order, and so the number of paths of length three is the number of 2 element permutations of $n-2$ nodes, i.e. $\frac{(n-2)}{(n-2-2)!} = (n-2)(n-3)$.
   - **length 4:** for $n \geq 5$ we may use any three of the nodes $k \neq i, j$ in a path, and so there are $\frac{(n-2)}{(n-2-3)!} = (n-2)(n-3)(n-4)$ possible paths.
   - **$\geq$ 4:** we can see the repeating pattern in the above series. The longest possible path has $n-2$ nodes ($n-1$ links) with $(n-2)!$ possible orderings.

   To obtain the total number of paths we must add all of these terms, which gives

   $$|P_{ij}| = \sum_{m=0}^{n-2} \frac{(n-2)!}{(n-2-m)!} = (n-2)! \sum_{m=0}^{n-2} \frac{1}{(n-2-m)!} = (n-2)! \sum_{m=0}^{n-2} \frac{1}{m!}$$

   Now, the sum $\sum_{m=0}^{n-2} \frac{1}{m!} \to e$, very quickly, and so the total number of paths goes like

   $$|P_{ij}| \sim e(n-2)!$$

   The network is completely connected, and therefore the above result holds for any pair of nodes. There are exactly $n(n-1)$ such pairs of nodes, and so the total number of paths in the network will be $|P| = |P_{ij}| n(n-1) = en!$, which, as you might know is a very large number, even for relatively small values of $n$.

   We can also see that a lot of the paths come from the longer paths. There are many more longer paths than shorter, and so by using shortest-path like routing, we eliminate many possible paths from consideration.

   We could approximate the above expression further using Stirling's approximation: for large $n$

   $$\ln n! \simeq n \ln n - n,$$

   and hence

   $$en! \simeq \frac{n^n}{e^{n-1}}.$$

   I would also accept that from the expression above, the largest term is $n!$, and so the asymptotic limit is $O(n!)$, but this is not quite as good as the above.

3. Computational time is usually calculated using the worst case performance:

- Edge list: in order to remove the edge, we must search for it in the edge list. Presuming this list is stored as a list (not a sorted tree or some other more efficient means) finding the correct entry will take $|E|$ test in the worst case. Removing an item from a list is an $O(1)$ operation, so the time is dominated by the search, and is $O(|E|)$.
- Adjacency matrix: removing an edge requires that we access the appropriate term in the matrix, which requires indexing into the matrix (an $O(1)$ task) and changing a 1 to a 0 (and $O(1)$ task).
- Neighbourhood list: Presume a reference to each node is stored in a sorted array, we can access the $i$th element by indexing into this array (an $O(1)$) operation. Then we need to find the destination node in the list of neighbours, but in the worst case all of the other nodes are neighbours of $i$, and hence we need to search a list which is $O(|N|)$ long. Removing the link is $O(1)$ and so the total will be dominated by the search, *i.e.,* $O(|N|)$.