Homework 01

Ben Zuckier

Network Theory

Problem 02

a)

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

The adjacency matrix of a bipartite network is a block diagonal matrix whose nonzero blocks consist of the incidence matrix $\mathbf{B}$ and its inverse, $\mathbf{B}^{\mathrm{T}}$. $\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & \mathbf{0} \end{pmatrix}$. The matrix $\mathbf{B}$ can be thought of as answering the question "does element $j \in$ V belong to element $i \in$ V.

The reason this is the case is that the only links in such a network are between nodes from group U to group V (undirected), and therefore also no self edges. As such, the only nonzero elements of the matrix are found at i, j where $|\mathrm{U}| < j \leq |\mathrm{V}|$ if node $i \in U$, else $0 < j \leq |U|$.

b)

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

c) The average number of edges is given by total number of edges connected to the nodes, divides by the number of nodes. In a bipartite network, this is $\frac{m}{n_\alpha}$ where m is the number of edges in one partition of the network (same in both partitions) and n_α is the number of nodes in partition α.

For the purple nodes this is $c_p = \frac{10}{6} = .6^{-1} = 1.\bar{6}$. For the green nodes it's $c_g = \frac{10}{5} = 2$.

d) For the network projections, the average degree is given by $c_\alpha = \frac{2m_\alpha}{n_\alpha}$

For the purple projection this is $c_P = \frac{16}{6} = .375^{-1} = 2.\bar{6}$. For the green projection $c_G = \frac{10}{5} = 2$.

It's not surprising that these result differ from the results in part c). There are fewer nodes in the projections vs their partition. Additionally, the network projections lose a lot of information present in the original graph.

Problem 02

a) Adjacency matrix of (a)

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

b) cocitation of (a)

$$\begin{pmatrix} 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

c) Incidence of (b), assuming that the filled circles on top are the "groups".

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

d) (Unweighted) projection matrix of (b) onto the black vertices

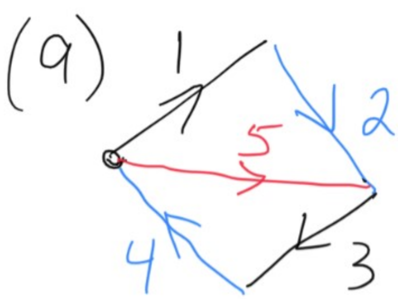
$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

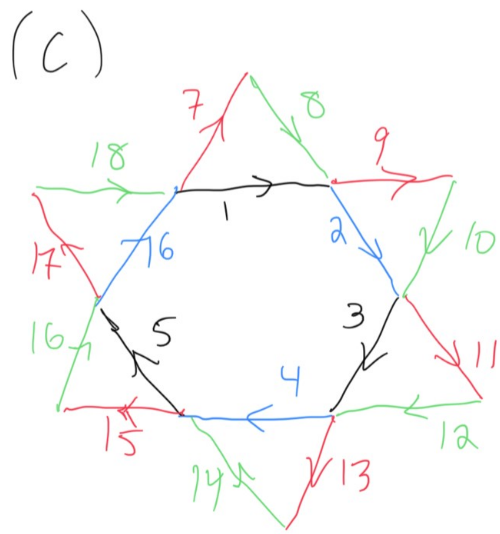
Problem 03

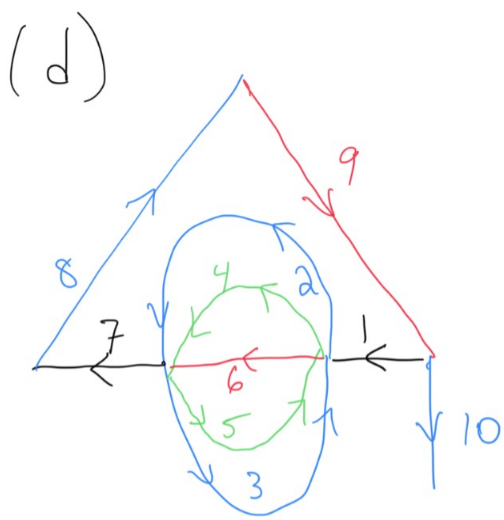
a) (a), (c), and (d)

b) Because they all have fewer than two nodes with an odd degree. The only way you can have an odd number of edges connected to a node is if you either start there or end there, and therefore two nodes of odd degree is the max.

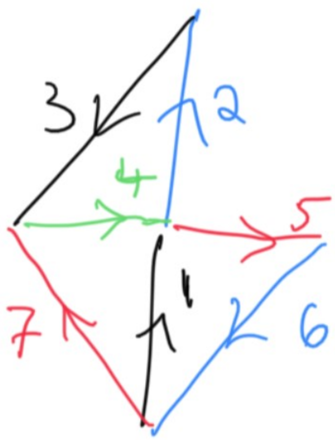
c) The paths are drawn below for (a), (c), and (d)







d) By removing one of the segments of the perimeter of (b), we can draw the resulting shape:



Problem 04

a) We know that every edge in the network goes from one partition to the other, by definition of the bipartite network. Therefore the number of edges m in each partition is the same, since every edge has one side in each partition. The average degree of these partitions, therefore, is $c_1 = \frac{m}{n_1}$, $c_2 = \frac{m}{n_2}$.
Then we solve for m: $m = c_2 n_2$, $m = c_1 n_1$. Setting these equal:

$$c_2 n_2 = c_1 n_1 \Rightarrow c_2 = c_1 \frac{n_1}{n_2}$$

$\square$

b) We know that the number of ends of edges in a network is always 2 times the number of edges m (sometimes referred to as the handshaking lemma). The degree of a node is nothing but how many ends of edges it is connected to, so the sum of the degree of the nodes is equal to the number of ends of edges. And if every node has degree 3, the sum of the degrees which is the number of ends of edges is equal to $3n \Rightarrow 3n = 2m$. We can see that $2m$ is of the form $2k$, the definition of an even number, therefore $3n$ must also be even. We know that 3 is odd, and only an even number times this odd number results in an even number. Therefore n is even. $\square$

Problem 05

a) $\mathbf{A1}$

b) $\mathbf{1}^{\mathrm{T}}\mathbf{A1}$

c) $\mathbf{A}^2$

d) Trace($\mathbf{A}^3$)/6

Problem 06

1. The edge connectivity of A and C is y. It definitely is not lower than y since it takes at least that number many cuts to separate A and B, and B to C has a higher connectivity, so "worst case" A could go to C through B.

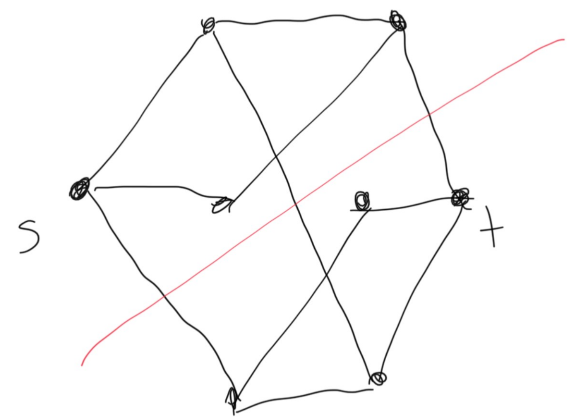
2. It's also not more than y. Let us assume that the connectivity of A and C is greater than y, toward a contradiction. This means that there is an additional path from A to C (ie $y + 1$ connectivity) that is distinct from all the y paths from A to B because if we were to cut the y paths from A to B, this path would have to still exist to make A and C have connectivity $y + 1$. If this is the case, however, then why didn't A and B include this path in their connectivity (remember, this path is distinct from all the paths A to B), namely this route from A to C and then take one of the $x - y$ "extra paths" from C to B (since $x > y$). We have a contradiction, $\Rightarrow$ the negation of "the connectivity of A and C is greater than y" is true, ie the connectivity of A and C is less than or equal to y.

3. We proved in 1 that the connectivity between A and C is $\not< y$, and we proved in 2 that it is $\leq y$. Therefore the edge connectivity of A and B is equal to y.
   $\square$

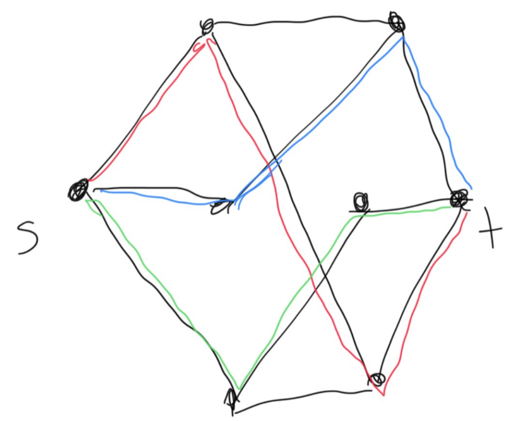
Problem 07

The minimum vertex cut set is 3 ($k = 3$).

a) One cut of size 3



b) One set of 3 independent paths



Similar to Menger's Theorem (Newman Networks pp.140).
Both are needed because in part a, the paths selected might be inefficient meaning k' is always less than k. In part b, one might select to cut too many edges when fewer would have worked, $k'' > k$. Iff both part a and part b are equal, $k' = k''$, does it equal k the true min cut set. This is also why it works. $k' \leq k$ and $k'' \geq k$ so if they are equal the it must be the case that $k' = k'' = k$, where k is the true min cut set.
$\square$

Problem 08

a) It was shown in lecture that the number of walks of length n in a network $\mathbf{A}$ is equal to $\mathbf{A}^n$, and the number of walks of length n that connect some node a to node b is A^n_{ab}. We then derived that $\mathbf{M} = \sum_n \alpha^n \sum_j (\mathbf{A}^n)_j = 1 + \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \dots$.

With some algebraic manipulation, we can show that $\mathbf{M} = \dfrac{1}{\mathbf{I} - \alpha\mathbf{A}}$:

$$\begin{aligned} \mathbf{M} &= 1 + \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \alpha^3\mathbf{A}^3 \dots \\ \alpha\mathbf{A}\mathbf{M} &= \quad\;\; \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \alpha^3\mathbf{A}^3 \dots \\ \Rightarrow \mathbf{M} - \alpha\mathbf{A}\mathbf{M} &= 1 \end{aligned}$$

We can also pull a factor of $\mathbf{M}$ to the right from the LHS: $\mathbf{M} - \alpha\mathbf{A}\mathbf{M} = (\mathbf{I} - \alpha\mathbf{A})\,\mathbf{M}$

$$(\mathbf{I} - \alpha\mathbf{A})\,\mathbf{M} = 1, \quad \Rightarrow\; \mathbf{M} = \frac{1}{\mathbf{I} - \alpha\mathbf{A}}$$

It follows (as stated above) that the sum of the weights of all the paths from a to b with that weight is given by $\mathbf{M}_{ab}$, or the ab element of $\frac{1}{\mathbf{I} - \alpha\mathbf{A}}$.

b) This is discussed in Newman page 164. We start α at zero, meaning $\mathbf{M} = \frac{1}{\mathbf{I}}$. As we increase α, the centralities diverge when $\frac{1}{\mathbf{I} - \alpha\mathbf{A}}$ diverges, namely when $\det(\mathbf{I} - \alpha\mathbf{A}) = 0$. If we multiply on the left by α^{-1} we get $\det(\mathbf{I} - \alpha\mathbf{A}) = \det(\alpha^{-1}\mathbf{I} - \mathbf{A})$ which is nothing but the characteristic polynomial $p_{\mathbf{A}}(\alpha^{-1})$. The determinant will first cross zero when α^{-1} is equal to κ_1 the largest eigenvalue of $\mathbf{A}$, i.e. when $\alpha = \kappa_1^{-1}$.

Therefore, for our sum to converge, we should pick a value $0 < \alpha < \kappa_1^{-1}$.

c) To show that $l_{ab} = \lim\limits_{\alpha \to 0} \dfrac{\partial \log(M_{ab})}{\partial \log(\alpha)}$ equals the length of the geodesic path from a to b if it exists, we start with the summation definition of $\mathbf{M}$ as in part a, where $M_{ab} = \sum_n (\alpha^n \mathbf{A}^n)_{ab} = (\alpha^0\mathbf{A}^0)_{ab} + (\alpha\mathbf{A})_{ab} + (\alpha^2\mathbf{A}^2)_{ab} + \dots$
The first l terms of this sum will be zero since there is no path from node a to node b (the properties of random walks corresponding to powers of the adjacency), and the first nonzero term will be $\alpha^l\mathbf{A}^l$. The terms after may or may not be nonzero. Thus, our sum simplifies to $M_{ab} = 0 + \dots + 0 + \left(\alpha^l\mathbf{A}^l\right)_{ab} + \left(\alpha^{l+1}\mathbf{A}^{l+1}\right)_{ab} + \dots$. We can factor out $\left(\alpha^l\mathbf{A}^l\right)_{ab}$ giving us

$$\begin{aligned} M_{ab} &= \left(\alpha^l\mathbf{A}^l\right)_{ab}\left[1 + (\alpha\mathbf{A})_{ab} + \left(\alpha^2\mathbf{A}^2\right)_{ab} + \dots\right] && \text{taking log:} \\ \log(M_{ab}) &= \log\!\left(\left(\alpha^l\mathbf{A}^l\right)_{ab}\left[1 + (\alpha\mathbf{A})_{ab} + \left(\alpha^2\mathbf{A}^2\right)_{ab}\right] + \dots\right) \\ &= \log\!\left(\alpha^l\right) + \log\!\left(\left(\mathbf{A}^l\right)_{ab}\right) + \log\!\left(1 + (\alpha\mathbf{A})_{ab} + \left(\alpha^2\mathbf{A}^2\right)_{ab} + \dots\right) \\ &= l \cdot \log(\alpha) + \log\!\left(\left(\mathbf{A}^l\right)_{ab}\right) + \log\!\left(1 + (\alpha\mathbf{A})_{ab} + \left(\alpha^2\mathbf{A}^2\right)_{ab} + \dots\right) \end{aligned}$$

Now we take the partial derivative with respect to $\log(\alpha)$:

$$
\begin{aligned}
\frac{\partial \log(M_{ab})}{\partial \log(\alpha)} &= l \cdot \frac{\partial \log(\alpha)}{\partial \log(\alpha)} + \frac{\partial \log\big((\mathbf{A}^l)_{ab}\big)}{\partial \log(\alpha)} + \frac{\partial \log(1 + (\alpha \mathbf{A})_{ab} + (\alpha^2 \mathbf{A}^2)_{ab} + \ldots)}{\partial \log(\alpha)} \\
&= l \cdot \cancelto{1}{\frac{\partial \log(\alpha)}{\partial \log(\alpha)}} + \cancelto{0}{\frac{\partial \log\big((\mathbf{A}^l)_{ab}\big)}{\partial \log(\alpha)}} + \frac{\partial \log(1 + (\alpha^1 \mathbf{A})_{ab} + (\alpha^2 \mathbf{A}^2)_{ab} + \ldots)}{\partial \log(\alpha)} \\
\lim_{\alpha \to 0} \left(\frac{\partial \log(M_{ab})}{\partial \log(\alpha)} \right) &= l + \lim_{\alpha \to 0} \frac{\partial \log\left(1 + \left(\cancelto{0}{\alpha^1} \mathbf{A}\right)_{ab} + \left(\cancelto{0}{\alpha^2} \mathbf{A}^2\right)_{ab} + \cancelto{0}{\ldots}\right)}{\partial \log(\alpha)} \\
&= l + \frac{\partial \log(1)}{\partial \log(\alpha)} = l + \frac{0}{\partial \log(\alpha)} = l
\end{aligned}
$$

$$
\Rightarrow \lim_{\alpha \to 0} \left(\frac{\partial \log(M_{ab})}{\partial \log(\alpha)} \right) = l
$$

Problem 09

Using Mathematica, the graph Laplacian was entered. The eigenvectors were normalized and made sure to be orthogonal. For each of the 6 starting positions (100% of the commodity at node 1, …, 100% of the commodity at node 6) the coefficient vector a was found and $\Psi(t)$ was defined as $\Psi(t) = a_1(0)\exp(-\lambda_1 t)\vec{V}_1 + a_2(0)\exp(-\lambda_2 t)\vec{V}_2 + \cdots + a_6(0)\exp(-\lambda_6 t)\vec{V}_6$. The solution for t to $\Psi(t) = .1$ was calculated for each node (except the node where the commodity started in each iteration). Here is the calculation:

```
time10p[psiN_, start_] :=
 Module[{y, iter},
    iter = Complement[{1, 2, 3, 4, 5, 6}, {start}];
    y = t /. FindRoot[.1 == psiN[[#]], {t, 0, 10}] & /@ iter
     (* mt = Ordering[y, -1];
      {mt, y[[mt]]} *)
] (* time for each node except start to hit 10% *)

time10p[psi1, 1]
time10p[psi2, 2]
time10p[psi3, 3]
time10p[psi4, 4]
time10p[psi5, 5]
time10p[psi6, 6]

{0.13984, 0.620015, 0.153507, 0.614987, 0.13927}

{0.13984, 0.13925, 0.13927, 0.620015, 0.85229}

{0.620015, 0.13925, 0.13927, 0.13984, 0.85229}

{0.153507, 0.13927, 0.13927, 0.153507, 0.65968}

{0.614987, 0.620015, 0.13984, 0.153507, 0.13927}

{0.13927, 0.85229, 0.85229, 0.65968, 0.13927}
```

1. 1 or 5
   We took the min of the max of each starting position: `min`(.620, .852, .852, .660, .620, .852).

2. $t \approx .620016$

3. 2, 3 or 6.
   This time we took the max of the max of each pos: `max`(.620, .852, .852, .660, .620, .852).

4. $t \approx .85229$

5. Medium match with my intuition:

   At first I thought that starting at 4 would be the fastest to ten percent since it has the most connections, but I realized that 6 is by far the hardest node to get to since it's the only node with only two edges so it makes sense that starting at 1 or 5 which are connected directly to 6 and also are connected both to node 4 and to another node as well.

   It first startled me that the slowest configuration is starting at either 2, 3, or 6 since I would think that starting at 6 would be the slowest since it's so poorly connected. But then I thought that it's just as hard to get the resource *from* node 6 to nodes 2 and 3, as it is to get the resource from either node 2 or 3 *to* node 6.

6. Time equal sharing where t_i is the time to equal with all the resource starting at i:
   $t_1 = 2.36137$, $t_2 = 3.0698$, $t_3 = 3.0698$, $t_4 = 2.77539$, $t_5 = 2.36137$, $t_6 = 3.47853$

   For each of the starting states, a solution was found for each node for t for when $\Psi(t) = \left(\frac{1}{6}\right) \pm .01 \left(\frac{1}{6}\right)$. This is shown here, with the example of the first configuration at the claimed time where everything is equal to within 1%, and a short time before where it is not the case that every node is within 1% of $\left(\frac{1}{6}\right)$.

```
N[Reduce[(1.01/6) > psi1[[#]] && (.99/6) < psi1[[#]]  && 0 ≤ t ≤ 50, t] &/@{1, 2, 3, 4, 5, 6}]
N[Reduce[(1.01/6) > psi2[[#]] && (.99/6) < psi2[[#]]  && 0 ≤ t ≤ 50, t] &/@{1, 2, 3, 4, 5, 6}]
N[Reduce[(1.01/6) > psi3[[#]] && (.99/6) < psi3[[#]]  && 0 ≤ t ≤ 50, t] &/@{1, 2, 3, 4, 5, 6}]
N[Reduce[(1.01/6) > psi4[[#]] && (.99/6) < psi4[[#]]  && 0 ≤ t ≤ 50, t] &/@{1, 2, 3, 4, 5, 6}]
N[Reduce[(1.01/6) > psi5[[#]] && (.99/6) < psi5[[#]]  && 0 ≤ t ≤ 50, t] &/@{1, 2, 3, 4, 5, 6}]
N[Reduce[(1.01/6) > psi6[[#]] && (.99/6) < psi6[[#]]  && 0 ≤ t ≤ 50, t] &/@{1, 2, 3, 4, 5, 6}]
{2.33692 < t ≤ 50., 0.389684 < t < 0.423739 || 1.56405 < t ≤ 50., 2.35813 < t ≤ 50., 1.58812 < t ≤ 50., 2.17398 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50.}

{0.389684 < t < 0.423739 || 1.56405 < t ≤ 50., 2.73964 < t ≤ 50., 0.367993 < t < 0.393656 || 2.56251 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50., 2.35813 < t ≤ 50., 3.0698 < t ≤ 50.}

{2.35813 < t ≤ 50., 0.367993 < t < 0.393656 || 2.56251 < t ≤ 50., 2.73964 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50., 0.389684 < t < 0.423739 || 1.56405 < t ≤ 50., 3.0698 < t ≤ 50.}

{1.58812 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50., 2.08808 < t ≤ 50., 1.58812 < t ≤ 50., 2.77539 < t ≤ 50.}

{2.17398 < t ≤ 50., 2.35813 < t ≤ 50., 0.389684 < t < 0.423739 || 1.56405 < t ≤ 50., 1.58812 < t ≤ 50., 2.33692 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50.}

{0.369496 < t < 0.395768 || 2.36137 < t ≤ 50., 3.0698 < t ≤ 50., 3.0698 < t ≤ 50., 2.77539 < t ≤ 50., 0.369496 < t < 0.395768 || 2.36137 < t ≤ 50., 3.47853 < t ≤ 50.}

N[psi1[t = 2.36137]]
Clear[t]
N[psi1[t = 2.35]]
Clear[t]
{0.168243, 0.166616, 0.165011, 0.16617, 0.165627, 0.168333}[2.36137]
{0.168284, 0.166621, 0.164972, 0.166161, 0.165597, 0.168365}[2.35]
```

Problem 10

We inject a commodity into some node α in our complete graph of $n = 15$. There are only two equations $f_i(t)$ needed to tell us the state of any node i at time t. This is due to symmetry – all the nodes started as identical and now there are only two types of nodes:

the node we injected and all the other nodes. As such we only need 2 equations: one for the node we injected and one for any of the nodes not injected.

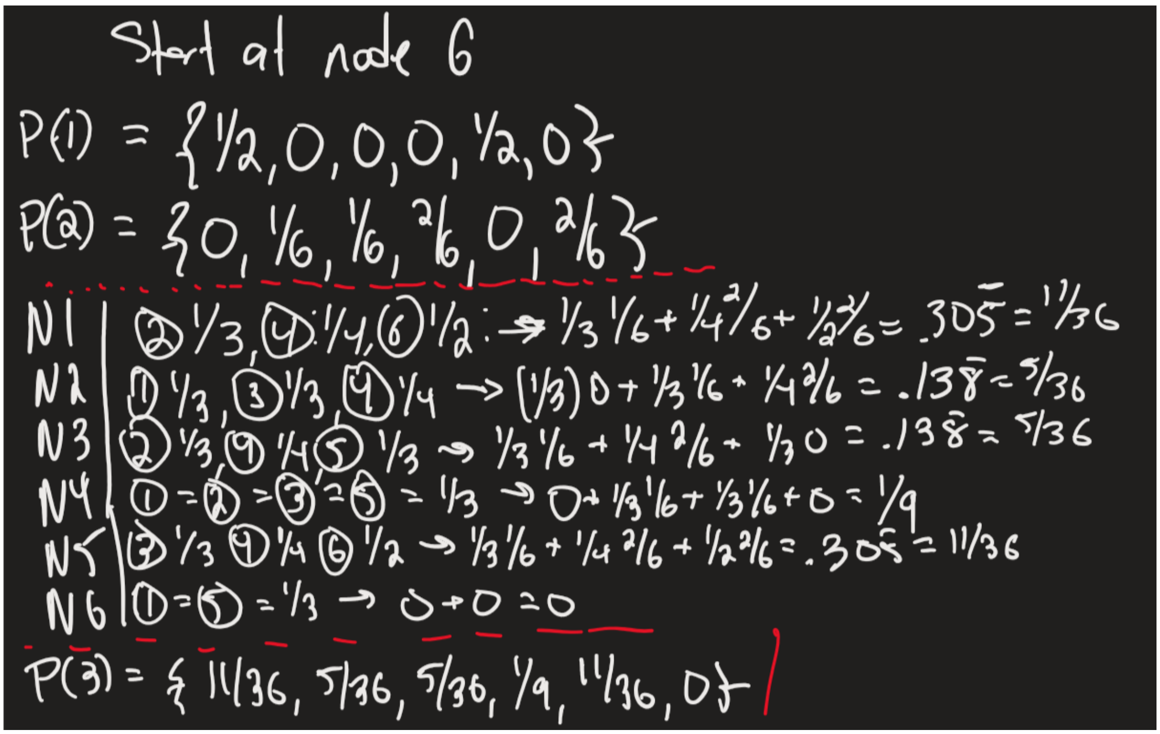
$$f_i(t) = \begin{cases} \dfrac{1}{15}\left(1 + 14\exp(-15t)\right) & \text{if } i = \alpha \\ \dfrac{1}{15}\left(1 - \exp(-15t)\right) & \text{if } i \neq \alpha \end{cases}$$

Note: Many patterns and features of a complete graph made this less tedious, like the fact that an N complete graph has one eigenvalue 0 and $N-1$ eigenvalues of value N. The eigenvectors (before orthonormalization) consist of $(N-1)$vectors $j \equiv 1 \to N-1$ where the first element of j is -1, the $N-j-1$ element is 1, and all the other elements are 0, and one $\mathbf{1}$ vector of all ones.

Problem 11

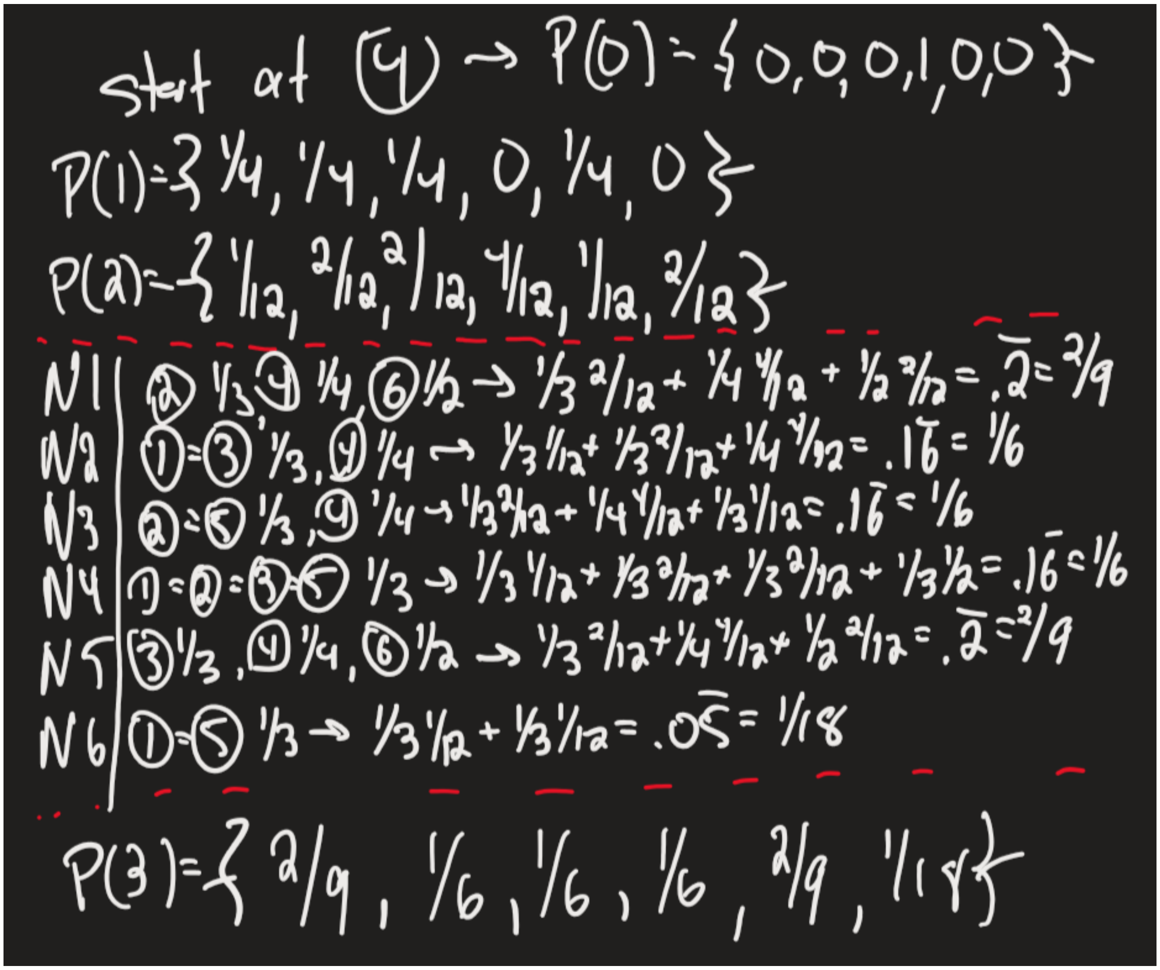
1. By hand, starting from node 6:

   We start with the vector $\mathbf{P_0} = \{0, 0, 0, 0, 0, 1\}$. The following is the working out for $t : 1 \to 3$.



2. By hand, starting from node 4:

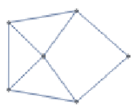
   We start with the vector $\mathbf{P_0} = \{0, 0, 0, 1, 0, 0\}$. The following is the working out for $t : 1 \to 3$.



3. Mathematica setup:

   The following method was employed $\vec{p}(t) = (\mathbf{AD}^{-1})^t\,\vec{p}(t-1)$. I was unsure how to use the method with eigenvectors and eigenvalues since I was unsure of the value α in the equations to find $\vec{p}$ at some time t. Here is the setup:

```
a = {{0, 1, 0, 1, 0, 1}, {1, 0, 1, 1, 0, 0}, {0, 1, 0, 1, 1, 0}, {1, 1, 1, 0, 1, 0}, {0, 0, 1, 1, 0, 1}, {1, 0, 0, 0, 1, 0}};
M = AdjacencyGraph[a]
dInv = DiagonalMatrix[1 / VertexDegree[M]];
P = a.dInv;
MatrixForm[P]
```



rixForm=

$$\begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{4} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \end{pmatrix}$$

4. Distribution after 50 steps:

   To find this, we just take the matrix $(\mathbf{AD}^{-1})$ to the 50^{th} power. The resulting matrix multiplied by any starting position will give the (same) distribution for time $t = 50$.

```
P50 = MatrixPower[P, 50];
P50 // N // MatrixForm
```

latrixForm=

$$\begin{pmatrix} 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 \\ 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 \\ 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 \\ 0.222222 & 0.222222 & 0.222222 & 0.222222 & 0.222222 & 0.222222 \\ 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 & 0.166667 \\ 0.111111 & 0.111111 & 0.111111 & 0.111111 & 0.111111 & 0.111111 \end{pmatrix}$$

5. Confirm walks done by hand:

   For the same starting positions (6 and 4) from above, the distribution for times $t : 1 \to 3$ were found (respectively). One can easily verify the results of both computations are the same.

```
p0i4 = {0, 0, 0, 1, 0, 0};
p0i6 = {0, 0, 0, 0, 0, 1};

p1i4 = MatrixPower[P, 1, p0i4]
p2i4 =  MatrixPower[P, 2, p0i4]
p3i4 =  MatrixPower[P, 3, p0i4]
p1i6 = MatrixPower[P, 1] . p0i6
p2i6 = MatrixPower[P, 2] . p0i6
p3i6 = MatrixPower[P, 3] . p0i6
```

$$\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0\right\}$$

$$\left\{\frac{1}{12}, \frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{12}, \frac{1}{6}\right\}$$

$$\left\{\frac{2}{9}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{2}{9}, \frac{1}{18}\right\}$$

$$\left\{\frac{1}{2}, 0, 0, 0, \frac{1}{2}, 0\right\}$$

$$\left\{0, \frac{1}{6}, \frac{1}{6}, \frac{1}{3}, 0, \frac{1}{3}\right\}$$

$$\left\{\frac{11}{36}, \frac{5}{36}, \frac{5}{36}, \frac{1}{9}, \frac{11}{36}, 0\right\}$$

6. Large t, same result:

   We now show that for sufficiently large value t, our probability distribution is the same, regardless of the starting position of the walker.

```
In[21]:= p1000i4 = N[MatrixPower[P, 1000, p0i4], 100];
         p1000i6 = N[MatrixPower[P, 1000, p0i6], 100];
         p1000i6 == p1000i4

Out[23]= True
```