Homework Assignment 6

1. I. Base case:

Let $d=0$

Then A^0 = the identity matrix. If $i=j$ then there is one walk, which is the trivial walk. If $i \neq j$, then there are none, which lines up with the identity matrix

II. Let $d \geq 0$. Assume the ijth entry of A^d is the number of walks in G of length d from i to j. We must show the ijth entry of A^{d+1} is the number of walks in G of length $d+1$ from i to j.

$$(A^{d+1})_{ij}$$
$$= (A^d \cdot A)_{ij}$$
$$= \sum_{k=1}^{n} A^d_{ik} \cdot A_{kj}$$

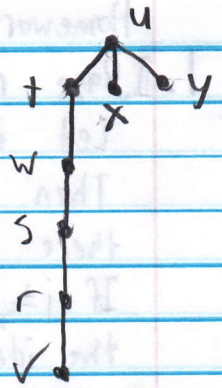
In length $d+1$, it takes two parts to walk from i to j: A walk from i to some vertex k and then an edge from k to j. i to k has length d via induction hypothesis. k to j only exists if there is an edge that connects them and gives an additional length making i to j length $d+1$ which is $A^d_{ik} \cdot A_{kj}$. So iterating over all k shows the ijth entry of A^{d+1} is the # of walks of length $d+1$ from i to j.

2. From the last problem, $A^d_{ij} = 0$ iff the number of walks of length d from i to j is also 0. So for A^0_{ij}, A^1_{ij}, A^2_{ij}, ... you eventually come across a walk from i to j whose length is the ijth term. Since its the first term, no other walks from i to j are shorter so, $\min\{d \mid (A^d)_{ij} > 0\}$

Tree

3. a

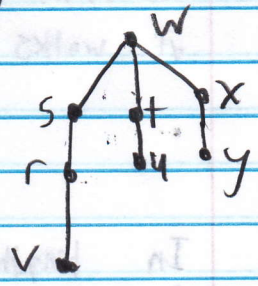
ver.	adj.	dist.	par.
r	s, v	4	s
s	r, w	3	w
t	y, w, x	1	u
u	t, x, y	0	nil
v	r	5	r
w	s, t, x	2	t
x	t, u, w, y	1	u
y	u, x	1	u



Queue u t x y w s r v

b.

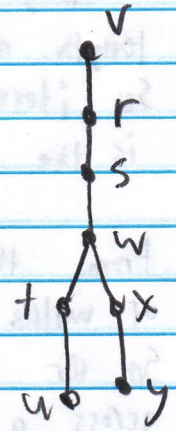
r	s, v	2	s
s	r, w	1	w
t	u, w, x	1	w
u	s, x, y	2	t
v	r	3	r
w	s, t, x	0	nil
x	t, u, w, y	1	w
y	u, x	2	x



Queue: w s t x r u y v

c.

r	s, v	1	v
s	r, w	2	r
t	u, w, x	4	w
u	s, x, y	5	t
v	r	0	nil
w	s, t, x	3	s
x	t, u, w, y	4	w
y	u, x	5	x



Queue: v r s w t x u y

4. BFS (G, s)
for all $x \in V$
color [x] = white
distance [x] = INF
parent [x] = NIL
Color [x] = GRAY
distance [s] = 0
parent [s] = NIL
Queue = new queue()
enqueue (Queue, s)
While Queue.length is not 0
v = dequeue (Queue)
for y = 1 to V.length
if A[V][y] = 1 and color [y] = white
Color [y] = gray
distance = distance [v] + 1
parent [y] = v
enqueue
color [v] = black

The cost is relatively the same until the while loop which executes n times and the inner for loop, which executes another n times for a total of $\Theta(n^2)$.

5. Using BFS, you would have to divide vertices into 2 subsets: "good" and "bad." A path from good-good or bad-bad is always even and every other case is odd, since edges can only connect vertices of the opposite type. So you would use BFS and check the distances of the vertices from its neighbors. If any are even, then you can't assign them. If they are all odd, then you can assign "good guys". The algorithm is as follows:

Pick source $\in V[G]$
BFS(G,s)
for all $x \in V[G]$
    for all $y \in$ distance[G]
        if distance[x] = distance[y] Y. 2
            return not possible

Assign good guys
return possible

6.

ver.	adj.	Disc.	Fin
q	s, t, w	1	16
r	u, y	17	20
s	v	2	7
t	x, y	8	15
u	y	18	19
v	w	3	6
w	s	4	5
x	z	9	12
y	q	13	14
z	x	10	11

Edges (q,s) tree
(q,w) forward
(q,t) tree
(s,v) tree
(v,w) tree
(w,s) back
(t,x) tree
(x,z) tree
(z,x) back
(t,y) tree
(y,q) back
(r,u) tree
(r,y) cross
(u,y) cross