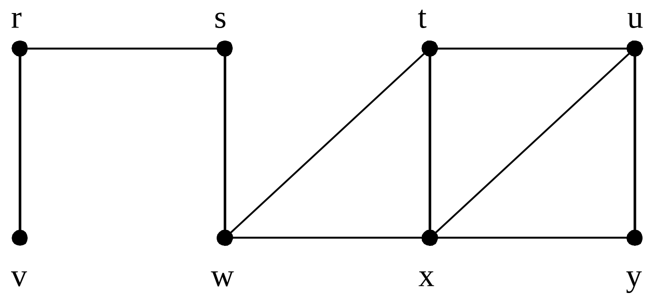
CMPS 101
Summer 2010
Homework Assignment 5 Solutions

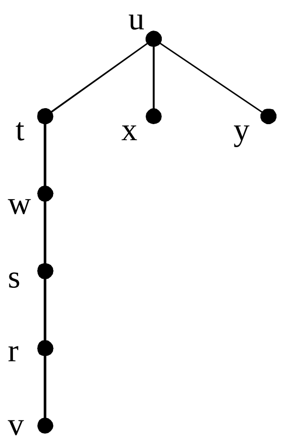
1. (3 Points) p. 538: 22.2-2
Show the d and π values that result from running breadth-first search on the undirected graph below using the following vertices as source. For each source, show the order in which vertices are added to the Queue, and show the state of the BFS tree after execution completes. Assume adjacency lists are processed in alphabetical order.



a. (1 Point) Let vertex u be the source
Solution:

vertex	**adjacency list**	**d**	π
r	s v	4	s
s	r w	3	w
t	u 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

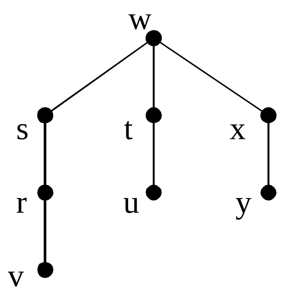
Queue: u t x y w s r v

BFS Tree


b. (1 Point) Let vertex w be the source
Solution:

vertex	**adjacency list**	**d**	π
r	s v	2	s
s	r w	1	w
t	u w x	1	w
u	t 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

BFS Tree


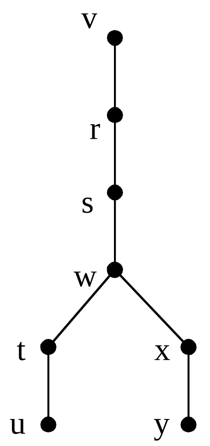
c. (1 Point) Let vertex v be the source

Solution:

vertex	**adjacency list**	**d**	π
r	s v	1	v
s	r w	2	r
t	u w x	4	w
u	t 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

BFS Tree



2. (1 Point) p. 538: 22.2-6
There are two types of professional wrestlers: “good guys” and “bad guys.” Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an $O(n+r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as good guys and the remainder as bad guys such that each rivalry is between a good guy and a bad guy. If it is possible to perform such a designation, your algorithm should produce it. (Hint: figure out how to use BFS to solve this problem.)

Solution:
Let the n wrestlers be the vertices of an undirected graph in which two vertices are joined by an edge if and only if there is a rivalry between the corresponding wrestlers. The question asks whether it is possible to partition the vertex set into two subsets *Good* and *Bad* such that all edges join Good vertices to Bad vertices. Such a graph is called *bipartite*. More precisely, an undirected graph $G=(V,E)$ is called bipartite if its vertex set V can be partitioned into two subsets X and Y such that every edge has one end in X and the other end in Y. In other words, $(u,v)\in E$ implies that either $u\in X$ and $v\in Y$, or $u\in Y$ and $v\in X$. The pair of sets (X, Y) is called a *bipartition* of the vertex set V (See B.4 p.1083). Note that if $(v_0, v_1, v_2, \dots, v_k)$ is any path in a bipartite graph, then the vertices v_i ($i=0,\dots,k$) must lie alternately in X and Y. Therefore any path from X to Y must have odd length, while any path from X to X, or Y to Y must have even length.

Let s be any vertex in a connected graph G, and consider the distance fields $d[u]$ for all $u\in V$ after BFS(G, s) has been run. If G is bipartite, then it’s vertex set V admits a bipartition (X, Y) as in the preceding paragraph, and we may assume for definiteness that $s\in X$. It follows from the above discussion that every vertex in X has an even distance from s, and every vertex in Y has an odd distance. (What is true for all paths must also be true for shortest paths.) Thus if G is bipartite, the bipartition must be given by

$$X = \{u\in V \;:\; d[u] \text{ is even }\} \quad\text{and}\quad Y = \{u\in V \;:\; d[u] \text{ is odd }\}.$$

Conversely, if G is not bipartite, then no bipartition of $V(G)$ exists. Thus if we define X and Y as above, there must either exist an edge joining X to X, or an edge joining Y to Y. We have proved the following: *G* is bipartite if and only if no two adjacent vertices have the same mod 2 distance from *s*. The following algorithm exploits this fact. However it works only in the special case that *G* is a connected graph.

```
isBipartite(G)  (Pre: G is a connected graph)
1.  pick any $s \in V[G]$
2.  BFS($G$, $s$)
3.  for all  $u \in V[G]$
4.          for all  $v \in \text{adj}[u]$
5.                  if  $d[u] \equiv d[v] (\text{mod } 2)$
6.                          print "no partition of wrestlers is possible"
7.                          return
8.  print "The good guys are:"
9.  print the set of vertices which are an even distance from $s$
```

The cost of this algorithm is obviously $O(|V|+|E|)$ in worst case. Of course, there is no reason to expect that the graph corresponding to wrestler rivalries would be connected. A moments thought shows that a disconnected graph is bipartite if and only if all it's connected components are bipartite. In the modified algorithm below, *T* and *X* denote subsets of *V*(*G*), and $G-T$ denotes the subgraph of *G* obtained by removing all vertices in *T*, along with all incident edges.

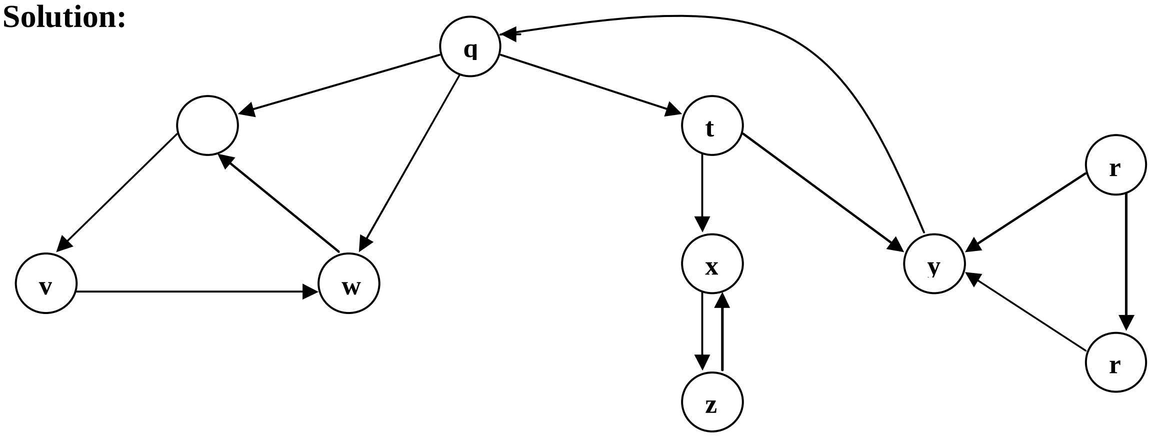
```
IsBipartite(G)
1.  $T \leftarrow X \leftarrow \varnothing$
2.  while $V - T \neq \varnothing$
3.          pick $s \in V - T$
4.          BFS($G-T$ , $s$)
5.          $T \leftarrow T \cup$ {black vertices}
6.          $X \leftarrow X \cup$ {black vertices whose distance from s is even}
7.          for all black vertices $u$
8.                  for all $v \in \text{adj}[u]$
9.                          if $d[u] \equiv d[v] (\text{mod } 2)$
10.                                 print "no such partition of the wrestlers is possible"
11.                                 return
12. print "the good guys are:"
13. print the members of $X$
```

3. (1 Point) p.547: 22.3-1
Make a 3-by-3 chart with row and column labels WHITE, GRAY, and BLACK. In each cell (i, j), indicate whether, at any point during a depth-first search of a directed graph, there can be an edge from a vertex of color *i* to a vertex of color *j*. For each possible edge, indicate what types it can be.
Solution:

	WHITE	GRAY	BLACK
WHITE	Yes: tree, back, forward, cross	Yes: back, cross	Yes: cross
GRAY	Yes: tree, forward	Yes: tree, back, forward	Yes: tree, forward, cross
BLACK	No	Yes: back	Yes: tree, back, forward, cross

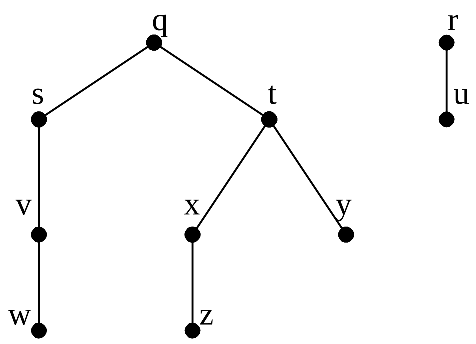
4. (1 Point) p.547: 22.3-2
Show how depth-first search works on the graph of Figure 22.6 (p.548). Assume that the **for** loop of lines 5-7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discover and finishing times for each vertex, and show the classification of each edge.

Solution:



Vertex	Adj	Discover	Finish
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

DFS Forest



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