Min Feedback Arc Set

vertices: teams

edges: $u \rightarrow v$ (team $u$ lost to team $v$)

interesting problems:

- is there a cycle? (easy)
- if not, find an ordering of vertices so all edges in same direction (hard)
- else find ordering that minimizes number of wrong-way edges

brute force: for each ordering, count wrong-way edges, keeping track of min

no one has done significantly better
no one has proven that significantly better is impossible

NP-complete
Graph Representation

Adjacency Matrix
to

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>G</th>
<th>D</th>
<th>Pr</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Pr</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>H</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

$O(1)$ if labels are integers $0, \ldots, n-1$

$\text{has-edge}(H, G)$: $O(1)$ expected

map vertex label $\rightarrow$ indices

Adjacency List – for each vertex $v$, a list of vertices it has edges to

Y :  
G :  
D :  
Pr :  
H :  

$\text{has-edge}(u, v)$: get list for $u$ by indexing into array of adjacency lists

symbol search in that list for $v$

$O(n^2)$ worst case

$O(n + m)$ space

$O(m + n \log n)$ if edges are sorted

$O(n)$ for a lot of apps (sparse)

<table>
<thead>
<tr>
<th>n</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

for each vertex $u$

for each edge $u \rightarrow v$

$\sum_{1 \leq i \leq n} 1 = O(n)$

for each vertex $u$

$\sum_{v \in \text{neighbors}(u)} 1 = O(n)$

by only looking at edges

$O(n + m)$ linear
<table>
<thead>
<tr>
<th></th>
<th>Adj Matrix</th>
<th>Adj List</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>space</strong></td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>worst $0 \leq m \leq n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O(n+m)$</td>
</tr>
<tr>
<td><strong>has_edge</strong></td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>worst case</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O(\text{outdegree}(n))$ directed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O(\text{degree}(n))$ undirected</td>
</tr>
<tr>
<td><strong>add_edge</strong></td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>worst case</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>if enforcing one edge/dir/pair (call has_edge first)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O(1)$ if allowing multiple edges between a single pair</td>
</tr>
<tr>
<td><strong>for_each_neighbor</strong></td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>worst case</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O(\text{degree}(n))$ undirected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$O(\text{outdegree}(n))$ directed</td>
</tr>
<tr>
<td><strong>for each vertex v</strong></td>
<td>$O(n^-)$</td>
<td>$O(n^-)$</td>
<td>worst case</td>
</tr>
<tr>
<td>for each neighbor of v</td>
<td></td>
<td></td>
<td>$O(n+m)$</td>
</tr>
</tbody>
</table>
BFS(G, s)

for each vertex u in graph_vertices(G)
    color[u] <- WHITE
d[u] <- infinity
    pred[u] <- NIL
    color[s] <- GRAY
    d[s] <- 0
    pred[s] <- NIL
Q <- [s]

while not queue_is_empty(Q)
    u <- queue_dequeue(Q)
    for each v in graph_adjacent(u)
        if color[v] = WHITE
            color[v] <- GRAY
            d[v] <- d[u] + 1
            pred[v] <- u
            queue_enqueue(Q, v)
    color[u] <- BLACK

while not queue_is_empty(Q)
    u <- queue_dequeue(Q)
    for each v in graph_adjacent(u)
        if color[v] = WHITE
            color[v] <- GRAY
            d[v] <- d[u] + 1
            pred[v] <- u
            queue_enqueue(Q, v)
    color[u] <- BLACK