Problem Statement

There are two types of professional wrestlers: “babyfaces” (“good guys”) and “heels” (“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 babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.

Main Idea

The main idea of the algorithm is to partition the graph into two sets such that only nodes in different sets have edges between them. If such a partition is not possible, then no valid assignment exists. This problem is equivalent to checking whether the graph is bipartite, and thus can be framed as the bipartite graph problem.

To solve this problem, we can use Depth-First Search (DFS) to attempt to color the graph. The approach is to assign each node one of two colors: color 0 represents a ”babyface” and color 1 represents a ”heel”. During the DFS traversal, if we can assign every node a color such that each neighbor of a node has a different color, then the graph is bipartite and the desired assignment is possible. In fact, the colors assigned during the DFS represent the final assignment of wrestlers.

Thus, the algorithm proceeds as follows:

- Represent the wrestlers and rivalries as a graph, where each wrestler is a vertex and each rivalry is an undirected edge.
- Perform a DFS traversal, attempting to color the graph with two colors.
- If the graph can be successfully colored, then the bipartite condition is satisfied, and an assignment is possible.
- If at any point a node has the same color as one of its neighbors, the graph is not bipartite, and no valid assignment exists.

In summary, the problem of assigning wrestlers as ”babyfaces” and ”heels” is reduced to determining whether the rivalry graph is bipartite, which can be efficiently checked using DFS with two colors.

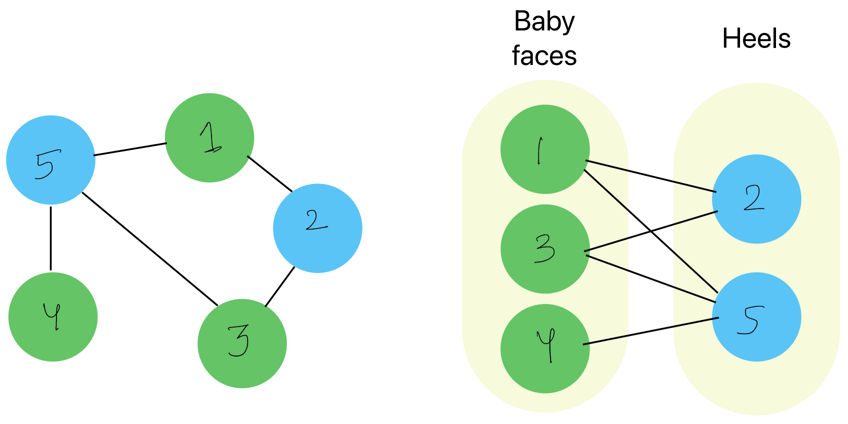

Figure 1: Example of Bipartite Graph

In figure 1 we define a graph G to represent the wrestlers and their rivalries.

- **Wrestlers**: 5 wrestlers labeled as W_1, W_2, W_3, W_4, W_5
- **Rivalries**: rivalries between of wrestlers: $(W_1, W_2), (W_2, W_3), (W_3, W_5), (W_5, W_4), (W_1, W_5)$

Definitions:

Let $G = (V, E)$, where:

- $V = \{W_1, W_2, W_3, W_4, W_5\}$ is the set of wrestlers (vertices).
- $E = \{(W_1, W_2), (W_2, W_3), (W_3, W_5), (W_5, W_4), (W_1, W_5)\}$ is the set of rivalries (edges).
- $n = 5$ (the number of wrestlers).
- $r = 5$ (the number of rivalries).

Explanation:

In this example, the graph G can be divided into two sets:

- **Babyfaces**: $\{W_1, W_3, W_4\}$
- **Heels**: $\{W_2, W_5\}$

Every rivalry is between a wrestler from the "Babyfaces" set and a wrestler from the "Heels" set, satisfying the requirement. Therefore, the graph is bipartite, and the wrestlers can be successfully divided into the two groups.

Pseudocode

Algorithm 1 Wrestler Designation Algorithm

```
procedure CanDesignateWrestlers($G, n$)
    Create an array color of size $n$, initialized to $-1$
    Initialize isBipartite to True
    for each wrestler $w = 0$ to $n - 1$ do
        if color[$w$] $= -1$ then
            Call DFS($G$, $w$, color, 0)
        end if
    end for
    if isBipartite then
        return Assignment Possible: Babyfaces: color 0 and Heels: color 1
    else
        return No Valid Assignment Exists
    end if
end procedure
procedure DFS($G, v, color, c$)
    color[$v$] $\leftarrow c$
    for each neighbor $u$ of $v$ do
        if color[$u$] $= -1$ then
            Call DFS($G$, $u$, color, $1 - c$)
        else if color[$u$] $=$ color[$v$] then
            Set isBipartite to False
            return
        end if
    end for
end procedure
```

Variable	**Description**
G	The graph representing wrestlers and rivalries. Vertices represent wrestlers, and edges represent rivalries.
n	The total number of wrestlers (vertices in the graph).
`color`	An array of size n, where `color`$[v]$ is either 0 (babyface), 1 (heel), or -1 (uncolored).
w	A wrestler (vertex) in the graph.
`isBipartite`	A boolean variable that indicates whether the graph can be partitioned into two sets (babyfaces and heels). It is initially set to **True**.
v	The current wrestler (vertex) being processed in the DFS.
u	A neighboring wrestler (vertex) of v.
c	The current color being assigned in the DFS (0 for babyface, 1 for heel).

Table 1: Variable Definitions for the Wrestler Designation Algorithm

Proof of Correctness

Let $G = (V, E)$ be a graph representing the wrestlers and rivalries, where:

- V is the set of vertices (wrestlers),
- E is the set of edges (rivalries between pairs of wrestlers).

Our objective is to determine if the graph G is bipartite. A graph is bipartite if we can partition V into two disjoint sets V_1 (babyfaces) and V_2 (heels) such that no edge in E has both endpoints in the same set. This is equivalent to asking whether G is 2-colorable.

We aim to prove that:

1. **Part 1**: If the algorithm outputs a valid 2-coloring, then the graph G is bipartite.
2. **Part 2**: If the graph G is bipartite, then the algorithm will output a valid 2-coloring.

Part 1

Claim: If the algorithm outputs a valid 2-coloring, then G is bipartite.
Proof:

- Assume the algorithm successfully assigns colors 0 or 1 to each vertex in V, and that for every edge $(u, v) \in E$, we have $\texttt{color}(u) \neq \texttt{color}(v)$.
- This implies that no two adjacent vertices are colored the same. Therefore, the vertices can be partitioned into two sets:
$$V_1 = \{u \in V \mid \texttt{color}(u) = 0\}$$
and
$$V_2 = \{v \in V \mid \texttt{color}(v) = 1\}$$
where all edges in E have one endpoint in V_1 and the other endpoint in V_2.
- By definition, a graph where such a partition exists is bipartite. Hence, if the algorithm finds a valid 2-coloring, G is bipartite, and we have successfully partitioned the wrestlers into "babyfaces" (colored 0) and "heels" (colored 1).

Thus, the algorithm produces a valid assignment whenever it outputs a 2-coloring.

Part 2

Claim: If G is bipartite, the algorithm will find a valid 2-coloring.
Proof:

- Assume that G is bipartite. By definition, there exists a partition of the vertices V into two sets V_1 and V_2 such that no two vertices within the same set are adjacent, i.e., $\forall (u, v) \in E$, either $u \in V_1$ and $v \in V_2$, or $u \in V_2$ and $v \in V_1$.
- Let the DFS algorithm start from any uncolored vertex u_0. Without loss of generality, assume $u_0 \in V_1$, and color u_0 with 0.
- The DFS proceeds by visiting every adjacent vertex v of u_0. Since G is bipartite, v must belong to V_2, so the algorithm will correctly color v with 1 (the opposite color of u_0).
- The DFS continues this process for every vertex and its neighbors. Since G is bipartite, this alternating coloring of vertices will succeed, as every vertex will be assigned the correct color based on the bipartite partition.

- If at any point the DFS detects that a vertex u and its neighbor v share the same color, it contradicts the assumption that G is bipartite. Therefore, such a scenario will not occur, and the algorithm will successfully 2-color the graph.

Thus, if G is bipartite, the DFS algorithm will find a valid 2-coloring, which corresponds to the valid designation of wrestlers as "babyfaces" and "heels."

By proving both soundness and completeness, we have shown that the algorithm correctly determines whether the graph is bipartite, and if so, finds a valid 2-coloring of the graph. Therefore, the algorithm correctly determines if the wrestlers can be designated as "babyfaces" and "heels" such that each rivalry is between wrestlers of different types.

Time Complexity

The time complexity of the algorithm is $O(n+r)$, where n represents the number of wrestlers (nodes in the graph), and r represents the number of rivalries (edges in the graph). The algorithm involves two main components: graph construction and coloring using Depth-First Search (DFS).

- **Graph Construction**: Constructing the graph takes $O(n + r)$ time. We use an adjacency list to represent the graph, where each wrestler is a vertex, and each rivalry corresponds to an undirected edge between two vertices. Adding each of the r rivalries to the graph requires linear time relative to the number of rivalries.
- **DFS Traversal**: In the worst case, the DFS traversal visits each vertex and edge exactly once. During the DFS, each node is assigned a color, and each edge is processed once. Since there are n vertices and r edges, the total time for the DFS traversal is $O(n + r)$.

Thus, the overall time complexity of the algorithm is $O(n+r)$, as both graph construction and DFS traversal are linear in the number of nodes and edges. This ensures that the algorithm can handle the problem efficiently with respect to the input size.