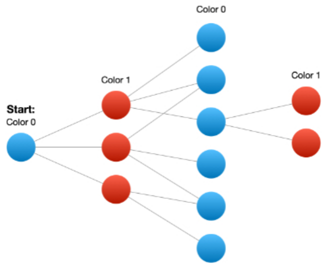
★★★★☆ Average Rating: 4.52 (77 votes) Premium

Approach #1: Coloring by Depth-First Search [Accepted]

Intuition

Color a node blue if it is part of the first set, else red. We should be able to greedily color the graph if and only if it is bipartite: one node being blue implies all it's neighbors are red, all those neighbors are blue, and so on.

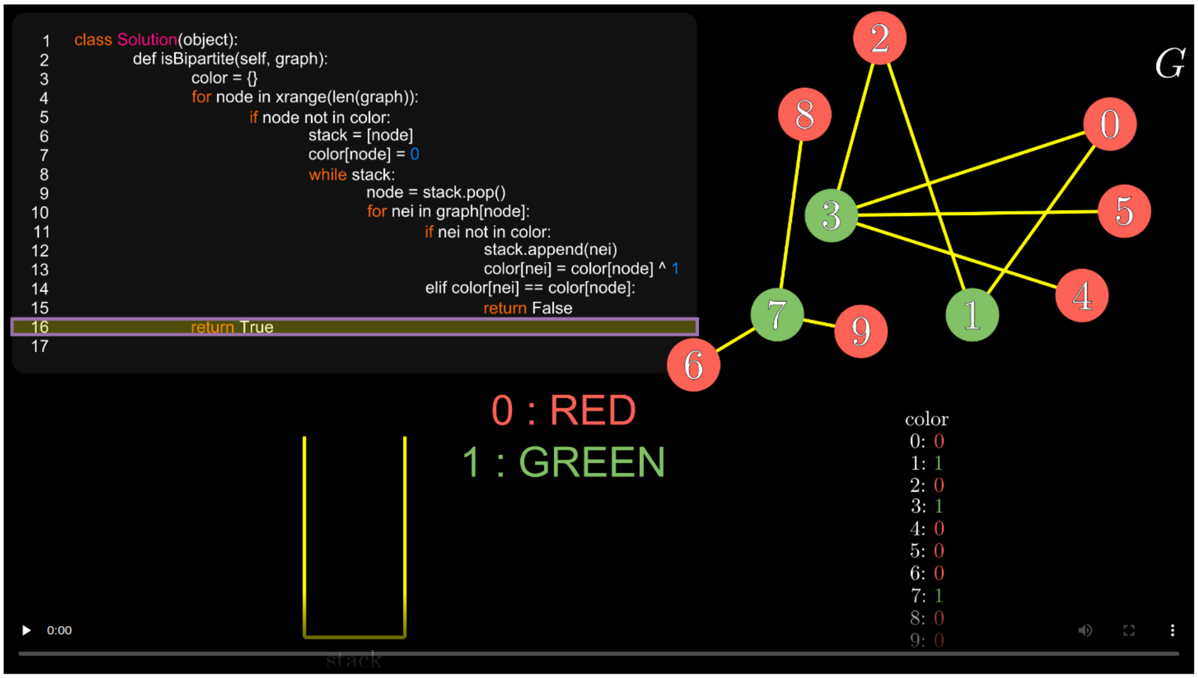


Algorithm

We'll keep an array (or hashmap) to lookup the color of each node: `color[node]`. The colors could be `0`, `1`, or uncolored (`-1` or `null`).

We should be careful to consider disconnected components of the graph, by searching each node. For each uncolored node, we'll start the coloring process by doing a depth-first-search on that node. Every neighbor gets colored the opposite color from the current node. If we find a neighbor colored the same color as the current node, then our coloring was impossible.

To perform the depth-first search, we use a `stack`. For each uncolored neighbor in `graph[node]`, we'll color it and add it to our `stack`, which acts as a sort of "todo list" of nodes to visit next. Our larger loop `for start...` ensures that we color every node. Here is a visual dry-run of the algorithm whose Python code is below.



```
class Solution(object):
    def isBipartite(self, graph):
        color = {}
        for node in xrange(len(graph)):
            if node not in color:
                stack = [node]
                color[node] = 0
                while stack:
                    node = stack.pop()
                    for nei in graph[node]:
                        if nei not in color:
                            stack.append(nei)
                            color[nei] = color[node] ^ 1
                        elif color[nei] == color[node]:
                            return False
        return True
```

C++ Java Python Copy

```
class Solution {
    public boolean isBipartite(int[][] graph) {
        int n = graph.length;
        int[] color = new int[n];
        Arrays.fill(color, -1);

        for (int start = 0; start < n; ++start) {
            if (color[start] == -1) {
                Stack<Integer> stack = new Stack();
                stack.push(start);
                color[start] = 0;

                while (!stack.empty()) {
                    Integer node = stack.pop();
                    for (int nei: graph[node]) {
                        if (color[nei] == -1) {
                            stack.push(nei);
                            color[nei] = color[node] ^ 1;
                        } else if (color[nei] == color[node]) {
                            return false;
                        }
                    }
                }
            }
        }

        return true;
    }
```

Complexity Analysis

- Time Complexity: $O(N + E)$, where N is the number of nodes in the graph, and E is the number of edges. We explore each node once when we transform it from uncolored to colored, traversing all its edges in the process.

C++

Autocomplete

```
// https://leetcode.com/problems/is-graph-bipartite/

#include<iostream>
#include<vector>
#include <algorithm>
#include <unordered_set>
#include <unordered_map>
#include <map>
using namespace std;

typedef vector<vector<int>> GRAPH;

class Solution {
private:
    bool is_color_conflict { false };

    void dfs(GRAPH &graph, int node, vector<int> &colors, int assign_color = 1) {
        if (colors[node] == 0)  // NOT visited
            colors[node] = assign_color;
        else {
            if (colors[node] != assign_color)
                is_color_conflict = true;
            return;
        }

        for (int neighbour : graph[node])
            dfs(graph, neighbour, colors, 3 - assign_color);
    }
public:
    bool isBipartite(GRAPH &graph) {
        int nodes = graph.size();
        vector<int> colors(nodes);

```

Your previous code was restored from your local storage

Console Contribute

- Space Complexity: $O(N)$, the space used to store the `color`.

Report Article Issue

Comments: 41

Best   Most Votes   Newest to Oldest   Oldest to Newest

Type comment here... (Markdown is supported)

Post

ursbhaskar ★ 113 September 7, 2020 1:09 PM

The problem is missing one specific information that the input can produce disconnected multiple graphs, which should be a huge point. I could not figure out why my answer was marked wrong until I looked up the solution.

Surprising thing is others providing answers as if that is not a concern.

89   Show 6 replies   Reply   Share   Report

Werber-Zeng ★ 33 October 6, 2018 11:52 AM

This is actually variant of Hungarian Algorithm.

27   Show 6 replies   Reply

Atoosa ★ 148 Last Edit: October 12, 2018 9:15 AM

Isn't this approach a BFS? The title says DFS, but I'm really confused as it appears to be BFS

38   Show 7 replies   Reply

calvinchankf ★ 6160 Last Edit: December 29, 2020 7:57 PM

Similar logic. I did it in both BFS and the recursive DFS. Know both to crack the coding interviews.💪

```
"""
    1st approach: BFS + nodes coloring

    Time    O(V+E)
    Space   O(V)
    156 ms, faster than 62.09%
"""
```

Read More

11   Show 1 reply   Reply

undefitied ★ 333 March 17, 2020 8:20 AM

Thanks for the article!
For some reason I assumed, that all nodes are connected, and started with a just queue(/stack), missing the first for-loop.

8   Reply

sschangi ★ 292 Last Edit: September 30, 2018 5:22 PM

So is there any difference between the time and space complexity of the DFS approach and the BFS approach?

7   Show 3 replies   Reply

cpjo ★ 26 Last Edit: October 2, 2018 5:05 PM

Why do you use stack?

11   Show 9 replies   Reply

cpshilpa ★ 5 August 14, 2020 6:37 PM

Could someone please explain how the graph is represented in the description. How does [[1,3], [0,2], [1,3], [0,2]] turn to
0----1
| |
| |
3----2

5   Show 3 replies   Reply

thegreenquaga ★ 67 February 13, 2020 12:26 AM

There is no need to create new stack for every iteration.

5   Reply

crisjul09 ★ 4 February 17, 2020 1:57 PM

Here is the bfs and dfs solutions.
DFS:

```
class Solution {
    public boolean isBipartite(int[][] graph) {
        Set<Integer> A = new HashSet<>();
        Set<Integer> B = new HashSet<>();
        for (int i = 0; i < graph.length; i++) {
            if (!A.contains(i) && !B.contains(i)) {
                A.add(i);
                if (!dfs(graph, i, A, B)) {
                    return false;
                }
            }
        }
        return true;
    }

    boolean dfs(int[][] graph, int node, Set<Integer> A, Set<Integer> B) {
        for (int i = 0; i < graph[node].length; i++) {
            if (!A.contains(graph[node][i]) && !B.contains(graph[node][i])) {
                if (A.contains(node)) {
                    B.add(graph[node][i]);
                }
                else {
                    A.add(graph[node][i]);
                }
                if (!dfs(graph, graph[node][i], A, B)) {
                    return false;
                }
            }
            else {
                if ((A.contains(node) && A.contains(graph[node][i])) || (B.contains(node) && B.contains(graph[node][i]))) {
                    return false;
                    return false;
                }
            }
        }
        return true;
    }
}
```

BFS:

```
class Solution {
    public boolean isBipartite(int[][] graph) {
        int[] color = new int[graph.length];
        Arrays.fill(color, -1);
        for (int i = 0; i < graph.length; i++) {
            if (color[i] == -1 && !bfs(graph, color, i)) {
                return false;
            }
        }
        return true;
    }

    boolean bfs(int[][] graph, int[] color, int node) {
        Queue<Integer> queue = new LinkedList<Integer>();
        color[node] = 0;
        queue.add(node);
        while (!queue.isEmpty()) {
            int n = queue.poll();
            for (int i = 0; i < graph[n].length; i++) {
                if (color[graph[n][i]] == -1) {
                    color[graph[n][i]] = color[n] ^ 1;
                    queue.add(graph[n][i]);
                }
                else {
                    if (color[graph[n][i]] == color[n]) {
                        return false;
                    }
                }
            }
        }
        return true;
    }
}
```

▲ 4 ▼ Show 1 reply Reply

‹ 1 2 3 4 5 ›

Problems Pick One ‹ Prev 18/56 Next › ▶ Run Code ^ Submit