Bipartite Graph Check

Khritish Kumar Behera

July 29, 2024

Introduction

A bipartite graph is a graph whose vertex set can be divided into two disjoint sets such that every edge connects a vertex in one set with a vertex in the other set. In simpler terms, it is possible to assign two different colors to the vertices in such a way that no two adjacent vertices share the same color.

Algorithm

To determine if a graph is bipartite, we can employ a graph coloring approach using Breadth-First Search (BFS):

1. **Initialization:** Assign a color of -1 to all vertices, indicating uncolored.
2. **BFS Traversal:**
   (a) Start a BFS from any uncolored vertex and assign it color 0
   (b) For each neighbor of the current vertex:
      i. If the neighbor is uncolored, assign it the opposite color of the current vertex.
      ii. If the neighbor is already colored and has the same color as the current vertex, the graph is not bipartite.
3. **Coloring Continuation:** Continue the BFS process, assigning alternating colors to adjacent vertices.
4. **Bipartite Check:** If the entire graph can be colored with two colors without conflicts, the graph is bipartite. Otherwise, it is not.

Note: If at any point during the BFS, a conflict arises (adjacent vertices with the same color), the algorithm can terminate early, as the graph is definitively not bipartite.

Implementation Considerations

- For efficiency, consider using an array or a hash table to store vertex colors
- The BFS can be optimized by using a queue to manage the vertices to be visited

By following these steps and handling potential conflicts, we can accurately determine whether a given graph is bipartite.

Psudocode

Assume Graph, G:{ 0...n-1} represented as ADJ_LIST

```
queue = [0] // any un-color node
node_color = [-1] * n
current_color = 0
while queue:
    conected_nodes = []
    while queue:
        poped_node = queue.pop()
        if node_color[poped_node] == -1:
            node_color[poped_node] = current_color
        elif node_color[poped_node] != current_color:
            Return False
        conected_nodes += ADJ_LIST[poped_node]
    queue = connected_nodes
    current_color = not current_color // 0/1 -> 1/0
Return True
```