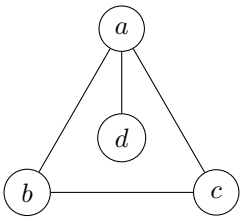
Exercises, chapter 2, solutions

---

1. If we let $V_1 = \{A, C, F, H\}$ and $V_2 = \{B, D, E, G, I\}$ then every edge will have one endpoint in V_1 and one endpoint in V_2. Thus, the requirement in the definition of "bipartite" is satisfied.

2. The following graph with four vertices is not bipartite:



   To see why, we give a proof by contradiction. Suppose the graph is bipartite. Consider any partition of the vertices into two sets V_1 and V_2 such that every edge has one endpoint in V_1 and one endpoint in V_2, and assume without loss of generality that vertex a belongs to V_1. First note that vertex b must belong to V_2 because of the edge $\{a, b\}$. Similarly, because of the edge $\{b, c\}$, vertex c must belong to V_1. Finally, the edge $\{a, c\}$ forces a to belong V_2, but this is a contradiction because a cannot belong to both V_1 and V_2. Hence, the graph is not bipartite.

3. (a) Our strategy will be to try to assign one of the two colors *red* and *blue* to each vertex in V in such a way that there is no edge between any two red vertices and no edge between any two blue vertices. If this is possible, then the graph is bipartite since we can just let the red and blue vertices correspond to V_1 and V_2, respectively, in the definition of "bipartite".

   To accomplish this, the algorithm will select an arbitrary starting vertex v_0 and assign it the color *red*. Next, all neighbors of v_0 are assigned the color *blue*. After that, neighbors of neighbors of v_0 that haven't been treated yet are colored *red*, etc. If the algorithm discovers that a neighbor w of a vertex v that is being considered has previously been assigned the same color as v, then it answers "NO" and stops, but if all vertices can be colored without any such conflicts then it answers "YES".

   Pseudocode:

```
while V contains at least one uncolored vertex do
    Choose any uncolored vertex v_0 ∈ V.
    Assign v_0 the color red and let Q be a queue initially consisting of v_0.
    while Q is not empty do
        Remove the first element from Q, and call it v.
        for each neighbor w of v do
            if w is uncolored then
                Assign w the opposite color of v (red ⇝ blue, and vice versa).
                Insert w at the end of Q.
            else if w has the same color as v then return "NO"
return "YES"
```

(b) To prove the correctness of the algorithm, consider the following two possible cases.

- If the algorithm answers "YES" then for every edge $\{x, y\} \in E$, it holds that one of x and y has been colored *red* and the other one *blue*. By definition, G is bipartite.
- If the algorithm answers "NO" then G has a vertex z that is reachable from v_0 by a path $\langle v_0, x_1, x_2, \ldots, x_i, z\rangle$ of even length as well as a path $\langle v_0, y_1, y_2, \ldots, y_j, z\rangle$ of odd length. This means G contains a cycle of odd length. Denote it by $\langle a_1, a_2, \ldots, a_\ell, a_1\rangle$, where ℓ is an odd number. Now, if G is bipartite and a_1 is *red* then a_2 must be *blue*, a_3 must be *red*, $\ldots$, a_ℓ must be *red*, and a_1 must be *blue*. However, a_1 cannot be both *red* and *blue*, so this would lead to a contradiction. In other words, G cannot be bipartite.

(c) **Time complexity analysis:** Each vertex is assigned a color and put into the queue Q once. Also, each vertex is removed from Q and treated once. Every edge $\{v, w\} \in E$ is examined twice. Therefore, the algorithm's time complexity is $O(|V|)+O(|V|)+O(2\cdot|E|) = O(|V|+|E|)$.

4. To solve the Two-Clique Problem, proceed as follows.

> 1) Given any input graph $G = (V, E)$, create a graph $\overline{G} = (V, \overline{E})$, where $\overline{E}$ is the complement of E (i.e., for each $u, v \in V$ with $u \neq v$, let $\{u, v\} \in \overline{E}$ if and only if $\{u, v\} \notin E$).
>
> 2) Run the algorithm for the Bipartite Graph Problem from question 3 above on input $\overline{G}$.
>
> 3) **if** the answer is "YES" then **return** "YES"; otherwise, **return** "NO".

This method works because:

- If the answer to the Two-Clique Problem on input G is "YES" then G can be partitioned into two cliques C_1 and C_2. This means that $\overline{G}$ has no edges between two vertices in C_1 or between two vertices in C_2, so the answer to the Bipartite Graph Problem on input $\overline{G}$ is "YES".
- If the answer to the Bipartite Graph Problem on input $\overline{G}$ is "YES" then V can be partitioned into V_1 and V_2 with no edges in $\overline{G}$ between two vertices in V_1 or between two vertices in V_2. In G, each pair of vertices in V_1 is therefore connected by an edge, and similarly for V_2. Since V_1 and V_2 form two cliques in G, the answer to the Two-Clique Problem on input G is "YES".

Time complexity analysis: To build $\overline{G}$ in step 1) takes $O(|V|^2)$ time, and running the algorithm from question 3 in step 2) takes $O(|V|+|\overline{E}|)$ time. Step 3) takes $O(1)$ time. Since $|\overline{E}| = O(|V|^2)$, the total time complexity is $O(|V|^2)+O(|V|+|\overline{E}|)+O(1) = O(|V|^2)+O(|V|+|V|^2)+O(1) = O(|V|^2)$, which is polynomial in $|V|$.