Câu 2:

số lượng đồ thị vô hướng khác nhau với *V* đỉnh và *E* cạnh là tổ hợp chập E của (V2)

Câu 3:

#include <iostream>

#include <unordered\_map>

#include <vector>

using namespace std;

// Custom comparator for pairs to handle unordered\_map key comparison

struct *PairHash* {

    template <class *T1*, class *T2*>

*size\_t* operator() (const pair<*T1*, *T2*>& *p*) const {

        auto hash1 = hash<*T1*>{}(*p*.first);

        auto hash2 = hash<*T2*>{}(*p*.second);

        // Combine hash values without considering order

        return hash1 ^ hash2;

    }

};

struct *PairEqual* {

    template <class *T1*, class *T2*>

    bool operator() (const pair<*T1*, *T2*>& *p1*, const pair<*T1*, *T2*>& *p2*) const {

        return (*p1*.first == *p2*.first && *p1*.second == *p2*.second) ||

               (*p1*.first == *p2*.second && *p1*.second == *p2*.first);

    }

};

// Function to count parallel edges in a graph

int countParallelEdges(const vector<pair<int, int>>& *edges*) {

    unordered\_map<pair<int, int>, int, *PairHash*, *PairEqual*> edgeCount;

    // Count the number of edges between each pair of vertices

    for (const auto& edge : *edges*) {

        edgeCount[edge]++;

    }

    // Count the number of parallel edges

    int parallelEdges = 0;

    for (const auto& it : edgeCount) {

        if (it.second > 1) {

            parallelEdges += it.second - 1;

        }

    }

    return parallelEdges;

}

int main() {

    // Example usage

    vector<pair<int, int>> edges = {{1, 2}, {2, 3}, {1, 2}, {3, 4}, {2, 3}, {1, 4}, {2, 1}};

    int parallelEdges = countParallelEdges(edges);

    cout << "Number of parallel edges: " << parallelEdges << endl;

    return 0;

}

Câu 4

Để chứng minh rằng một đồ thị là đồ thị hai màu (bipartite) khi và chỉ khi nó không chứa chu trình độ dài lẻ, ta sẽ sử dụng phương pháp chứng minh bằng phản chứng.

**Phần 1: Nếu đồ thị là hai màu thì nó không chứa chu trình độ dài lẻ**

Giả sử bạn có một đồ thị hai màu. Nếu có một chu trình trong đồ thị này, chu trình đó phải xen kẽ giữa các đỉnh thuộc hai nhóm màu khác nhau. Bởi vì không có hai đỉnh cùng màu nào được nối với nhau, chu trình phải "nhảy" từ một nhóm màu sang nhóm màu kia sau mỗi cạnh. Điều này có nghĩa là chu trình phải có độ dài chẵn, vì nó sẽ kết thúc ở nhóm màu bắt đầu. Do đó, không thể có chu trình độ dài lẻ trong một đồ thị hai màu.

**Phần 2: Nếu đồ thị không chứa chu trình độ dài lẻ thì nó là hai màu**

Giả sử bạn có một đồ thị không chứa chu trình độ dài lẻ nhưng không phải là đồ thị hai màu. Điều này có nghĩa là có ít nhất một cạnh trong đồ thị nối hai đỉnh cùng màu.

Tuy nhiên, nếu bạn theo dõi đường đi từ một đỉnh qua cạnh này đến đỉnh kia và quay trở lại đỉnh ban đầu qua các cạnh khác (điều này luôn có thể xảy ra vì đồ thị là liên thông), bạn sẽ tạo thành một chu trình. Vì hai đỉnh cuối cùng này cùng màu, chu trình này sẽ có độ dài lẻ (vì nó phải "nhảy" số lẻ lần giữa các màu để quay lại một đỉnh cùng màu). Điều này mâu thuẫn với giả thiết ban đầu rằng đồ thị không chứa chu trình độ dài lẻ. Do đó, mọi đồ thị không chứa chu trình độ dài lẻ đều phải là đồ thị hai màu.

Như vậy, ta đã chứng minh rằng một đồ thị là đồ thị hai màu khi và chỉ khi nó không chứa chu trình độ dài lẻ.

Câu 5

Chứng minh rằng một đồ thị không có điểm articulation là một đồ thị biconnected

Bây giờ, chúng ta sẽ chứng minh rằng một đồ thị không có điểm articulation là một đồ thị biconnected:

* **Bước 1**: Xét một đồ thị *G* không có điểm articulation. Điều này có nghĩa là việc loại bỏ bất kỳ đỉnh nào cũng không làm mất tính liên thông của đồ thị.
* **Bước 2**: Lấy hai đỉnh bất kỳ *s* và *t* trong đồ thị. Vì *G* là liên thông, tồn tại ít nhất một đường đi từ *s* đến *t*.
* **Bước 3**: Giả sử có một đường đi *P* từ *s* đến *t*. Vì không có điểm articulation, việc loại bỏ bất kỳ đỉnh nào trên đường đi *P* không làm mất tính liên thông của đồ thị.
* **Bước 4**: Nếu ta loại bỏ một đỉnh *v* trên đường đi *P*, vẫn phải tồn tại một đường đi khác từ *s* đến *t* không qua *v* (do tính liên thông không bị mất).
* **Bước 5**: Áp dụng luận điểm trên cho mỗi đỉnh trên đường đi *P*, ta có thể tìm ra một đường đi khác từ *s* đến *t* không giao nhau với *P*.

Kết luận: Từ các bước trên, ta thấy rằng cho mọi cặp đỉnh *s* và *t* trong *G*, luôn tồn tại ít nhất hai đường đi không giao nhau nối *s* và *t*. Do đó, *G* là một đồ thị biconnected. Điều này chứng minh rằng mọi đồ thị không có điểm articulation đều là đồ thị biconnected.

Câu 8