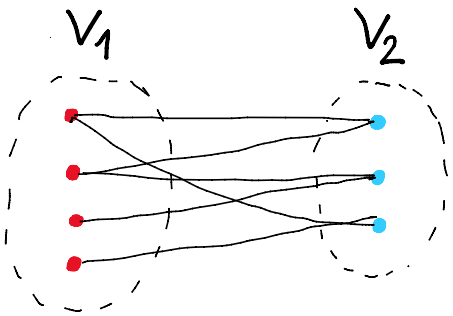
Zad 2.

wtorek, 17 stycznia 2023 12:37

2. Zastosuj przeszukiwanie grafu w głąb do sprawdzania, czy graf jest dwudzielny. Twoja procedura powinna mieć złożoność $O(m+n)$.

korzystamy z faktu, że graf dwudzielny można dwu-kolorować (pokolorować wierzchołki grafu na dwa kolory tak, że żadne dwa sąsiadujące ze sobą nie są tego samego koloru).



zakładamy, że graf jest spójny

```cpp
#include <iostream>
#include <vector>
using namespace std;

#define N 10 // liczba wierzchołków w grafie

vector<int> graph[N]; // graf reprezentowany jako listy sąsiadów
vector<bool> visited(N, false); // false - nieodwiedzony, true - odwiedzony wierzcholek
vector<int> colors(N, -1); // -1 - brak koloru, 0 - czerwony, 1 - niebieski
int color = 1; // kolor startowy
bool flag = true; // odpowiedź

void dfs(int u) {
    visited[u] = true; // oznaczenie wierzcholka jako odwiedzony
    colors[u] = color; // przypisanie koloru
    color = (color + 1) % 2; // zmiana koloru
    for (int v : graph[u]) {
        if (!visited[v]) dfs(v);
        else if (colors[u] == colors[v]) flag = false;
        // skoro dwa sąsiadujące wierzchołki są tego samego koloru, to graf nie jest dwudzielny
    }
}

int main() {
    dfs(0);
    cout << (flag ? "Graf jest dwudzielny\n" : "Graf nie jest dwudzielny\n");
}
```

przeszukiwanie w głąb ma złożoność O(n+m) tak jak chcieliśmy