Universidade Federal da Fronteira Sul
Ciência da Computação
Grafos
Professor: Andrei de Almeida Sampaio Braga
Discentes: Jonathan Gotz Correa e Luna Matana Fortes

Trabalho 1 - Grafos Bipartidos

Durante a produção do projeto de Grafos Bipartidos para a matéria de Grafos selecionamos três testes (Testes 2, 3 e 10) nos quais o método eh_bipartido_1 retorna *falso* (NAO), enquanto o método eh_bipartido_2 retorna *verdadeiro* (SIM), discorrerei adiante sobre:

Funcionamento do eh_bipartido_1

O método eh_bipartido_1 realiza uma abordagem **recursiva e ingênua de particionamento**, baseada na tentativa de alocar vértices em dois conjuntos (conjunto_a e conjunto_b) de forma **sequencial**. Ele percorre os vértices na ordem em que aparecem e tenta encaixá-los em um dos conjuntos, com base nos vizinhos já alocados.

Porém, essa abordagem assume que existe **uma única forma possível** de montar os conjuntos bipartidos e não leva em conta todas as combinações possíveis. Se em algum momento o vértice atual não puder ser colocado em nenhum dos dois conjuntos com base nos já visitados, o método retorna false, **sem tentar voltar atrás e explorar outras possibilidades** (sem *backtracking*). Com isso, grafos que *são* bipartidos podem falhar nesse teste, simplesmente porque a ordem de processamento dos vértices não favorece a montagem correta dos conjuntos.

Funcionamento do eh_bipartido_2

Já o método eh_bipartido_2 utiliza uma **busca em profundidade iterativa com pilha**, alternando os conjuntos a cada nível de profundidade. Ao visitar um vértice, ele tenta colocá-lo em um conjunto alternado ao de seus vizinhos, como é esperado em uma bipartição. O método garante que **vizinhos não fiquem no mesmo conjunto** e, se isso for impossível, retorna false.

Apesar de também não usar backtracking, essa abordagem é mais robusta para verificar bipartição, pois impõe restrições locais (entre vértices adjacentes) de forma sistemática e

independente da ordem inicial dos vértices, o que evita muitos dos conflitos que afetam o eh_bipartido_1.

Explicação por testes

Teste 2

input

7 10

0 1 A

0 3 A

0 4 A

0 5 A

1 2 A

1 3 N

2 3 A

4 5 N

4 6 A

5 6 A

Output

NAO

SIM

Universidade Federal da Fronteira Sul
Ciência da Computação
Grafos
Professor: Andrei de Almeida Sampaio Braga
Discentes: Jonathan Gotz Correa e Luna Matana Fortes

Teste 3

Input

7 11

0 1 N

6 5 N

5 4 A

2 4 A

5 1 A

3 6 A

5 0 A

6 5 N

1 3 A

3 4 A

3 1 N

Output

NAO

SIM

Universidade Federal da Fronteira Sul
Ciência da Computação
Grafos
Professor: Andrei de Almeida Sampaio Braga
Discentes: Jonathan Gotz Correa e Luna Matana Fortes

Teste 10

Input

9 23
1 8 A
1 6 A
7 2 N
1 4 A
0 3 A
2 7 A
0 8 A
4 3 N
2 7 N
4 5 A
5 6 A
2 5 N
7 4 A
5 3 A
2 0 N
5 8 A
7 6 A
7 3 A

Universidade Federal da Fronteira Sul
Ciência da Computação
Grafos
Professor: Andrei de Almeida Sampaio Braga
Discentes: Jonathan Gotz Correa e Luna Matana Fortes

0 2 N

0 2 A

2 5 A

0 6 A

1 3 A

Output

NAO

SIM

Explicação:

Basicamente em todos os testes o método eh_bipartido_1 percorre os vértices em ordem e tenta encaixá-los em um dos conjuntos com base em decisões anteriores. Como não há *backtracking*, ao encontrar um vértice que não pode ser alocado diretamente (por causa das conexões anteriores), ele retorna false. Já eh_bipartido_2, ao alternar dinamicamente os conjuntos durante a DFS, consegue alocar todos corretamente e retorna true.

Concluimos que o método eh_bipartido_1 falha nos testes mencionados por ser uma solução recursiva limitada, que não explora diferentes caminhos ou combinações possíveis de particionamento. Por outro lado, o método eh_bipartido_2 adota uma abordagem mais apropriada e generalizada para esse tipo de verificação, utilizando uma lógica semelhante à coloração de grafos em duas cores.

Por isso, para uma verificação confiável da bipartição em grafos, o eh_bipartido_2 é o método mais adequado.