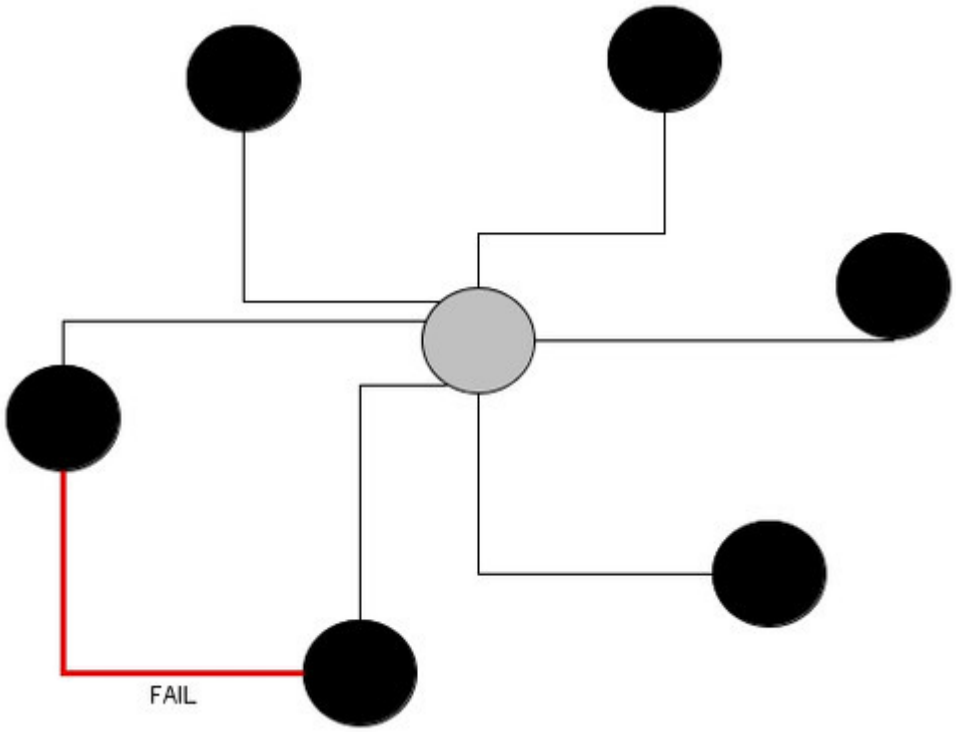
Bicoloração

Problema da Coloração de Vértices

- Dado um grafo não-dirigido, encontrar uma coloração dos vértices com número mínimo de cores.

- É fácil produzir uma coloração com muitas cores: basta atribuir uma cor diferente para cada vértice. É mais difícil obter uma coloração com poucas cores.

O Problema



O Problema

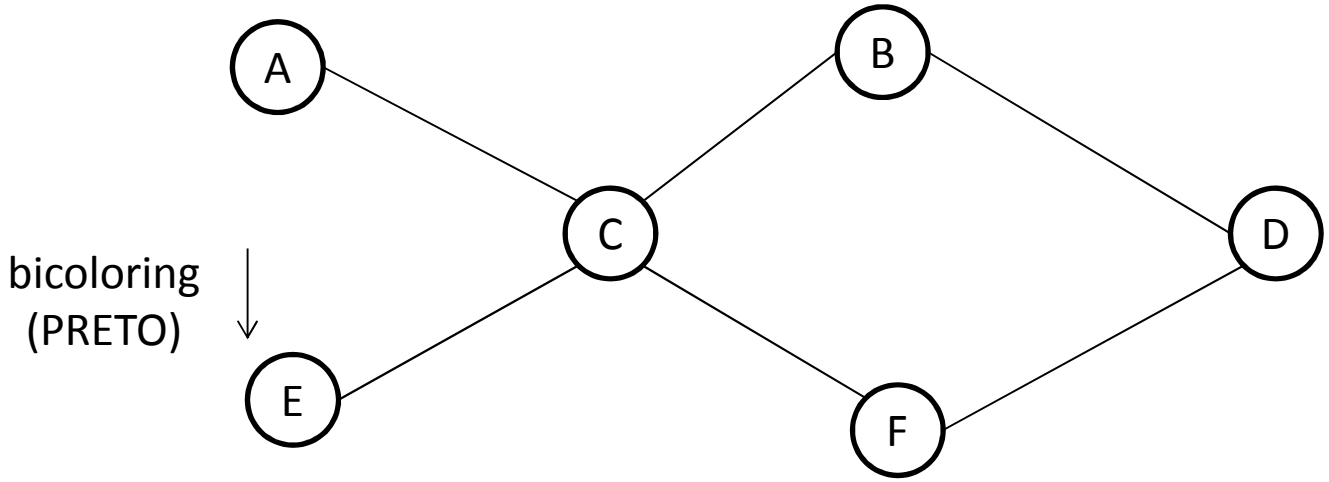
- Um grafo não-dirigido é bicromático (ou bipartido) se admite uma coloração dos vértices com não mais que 2 cores.
- Grafos bicromáticos são usualmente desenhados de modo que os vértices de uma das cores fiquem alinhados na parte superior da figura e os da outra cor fiquem alinhados na parte inferior da figura.

O Problema

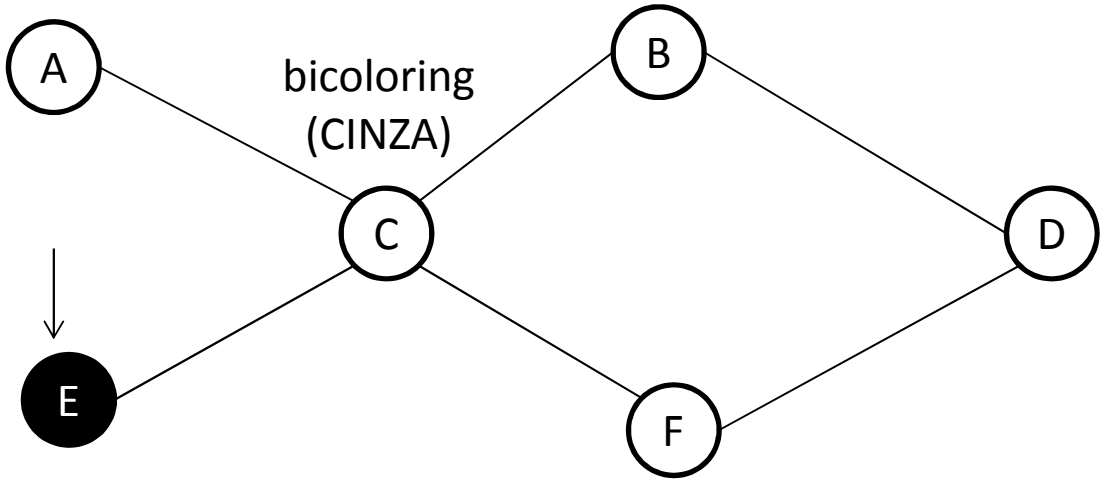
- TEOREMA: Um grafo é bicromático se e somente se não tem ciclo de comprimento ímpar.

- Portanto, para provar que um dado grafo não é bicromático, basta exibir um ciclo ímpar no grafo.

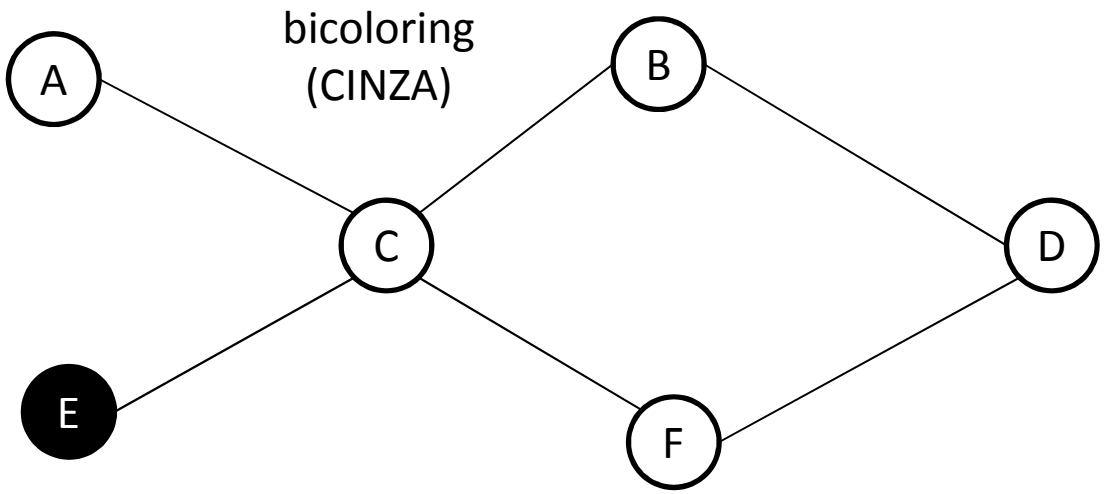
Algoritmo de Bicoloração



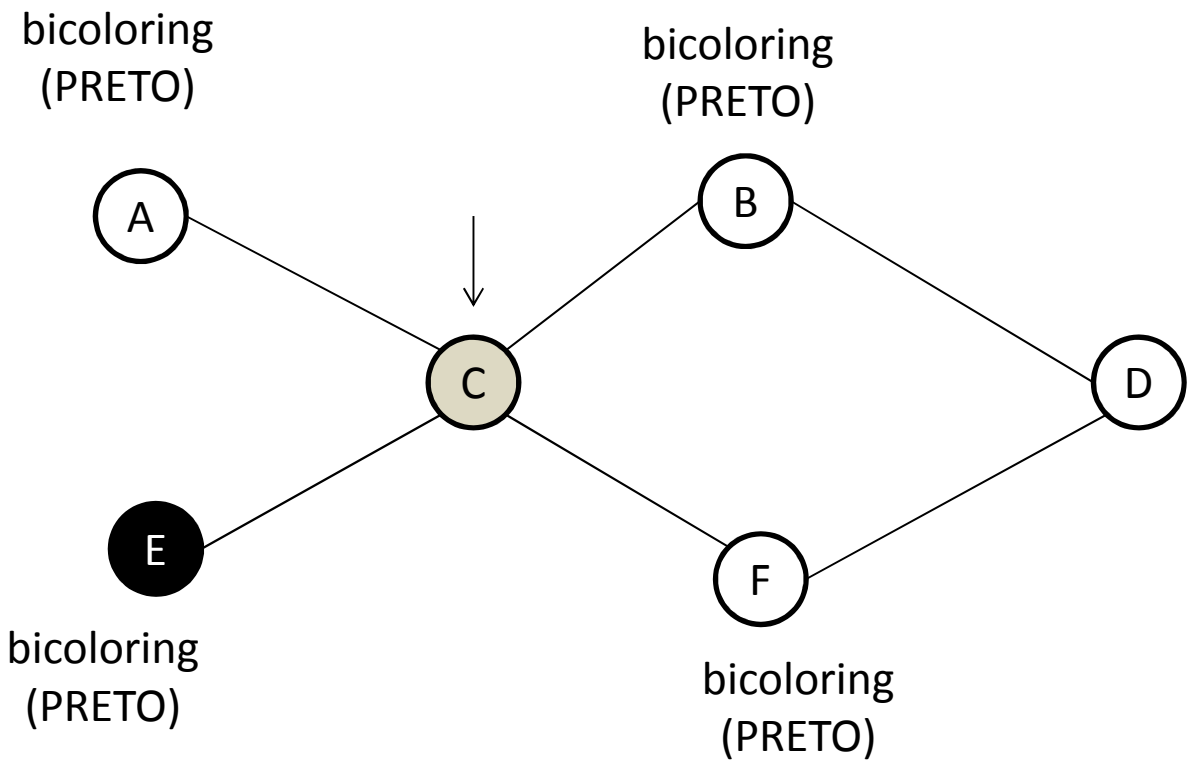
Algoritmo de Bicoloração



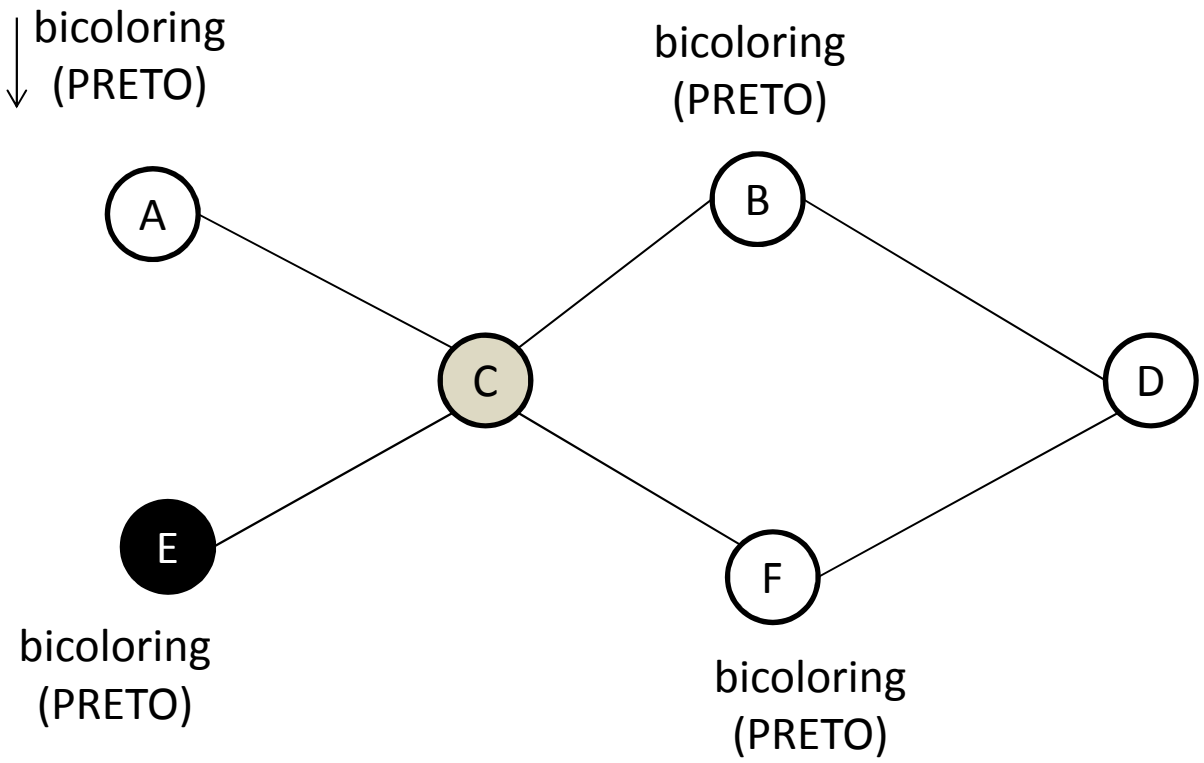
Algoritmo de Bicoloração



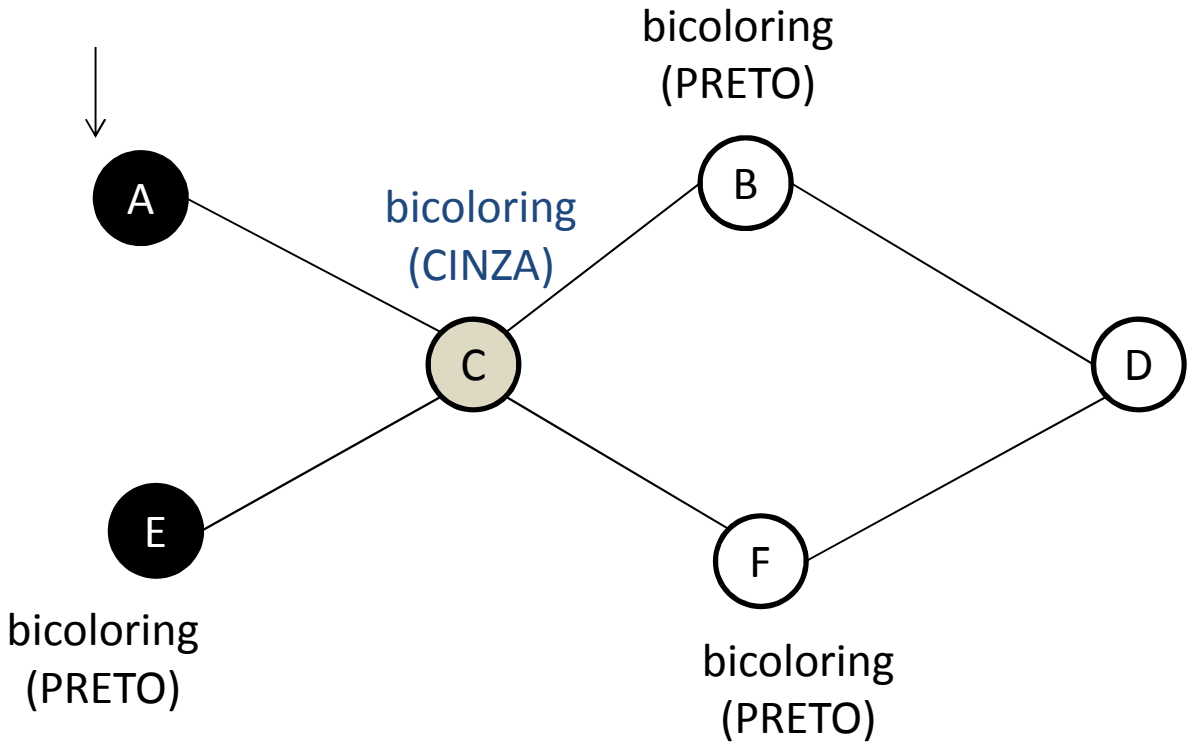
Algoritmo de Bicoloração



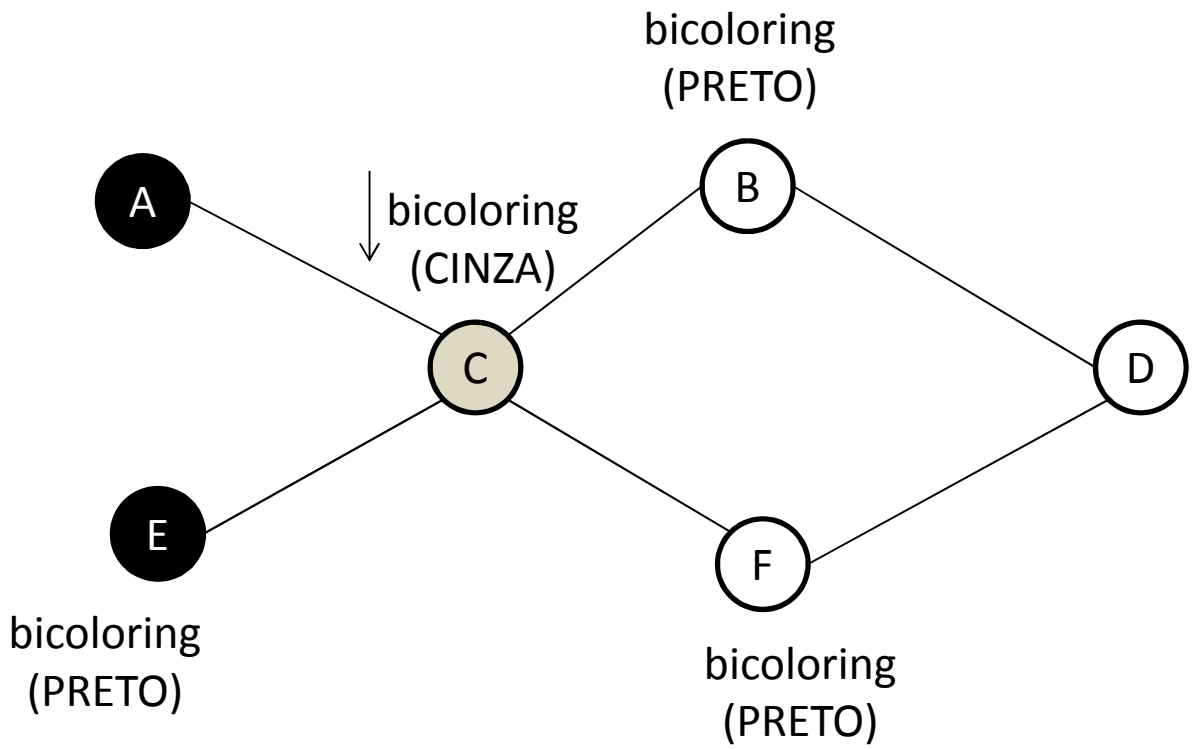
Algoritmo de Bicoloração



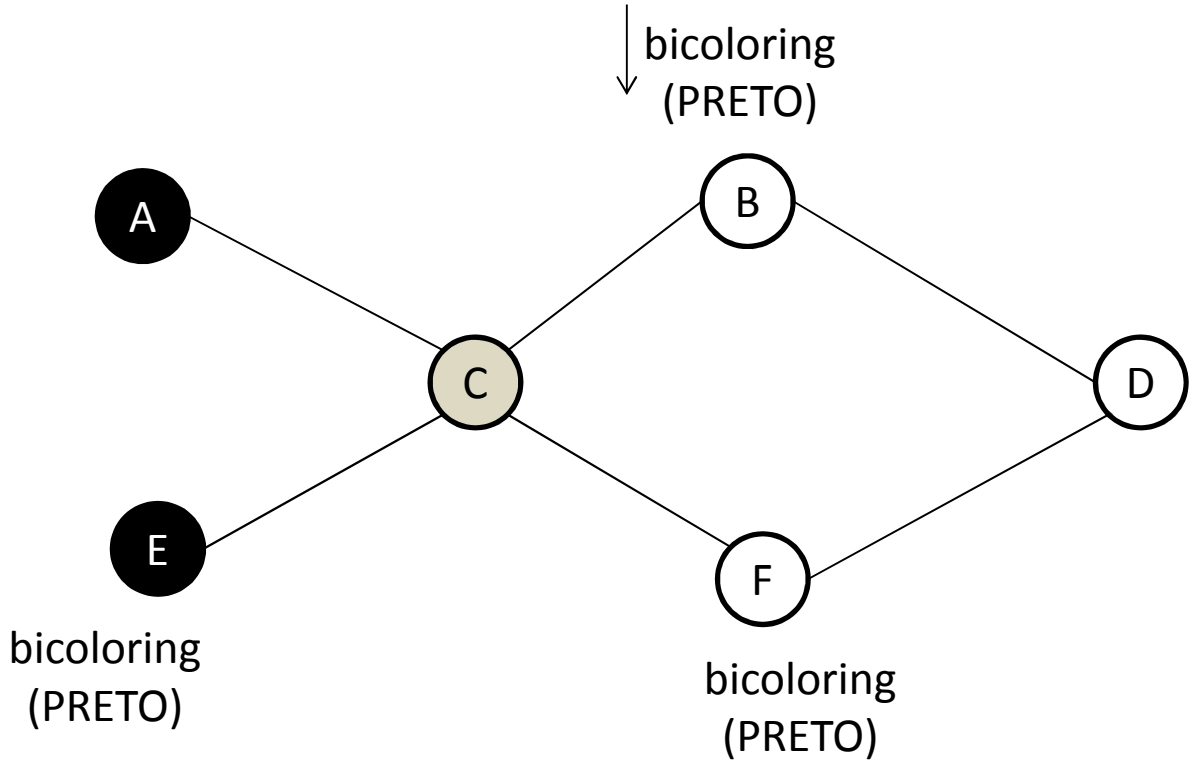
Algoritmo de Bicoloração



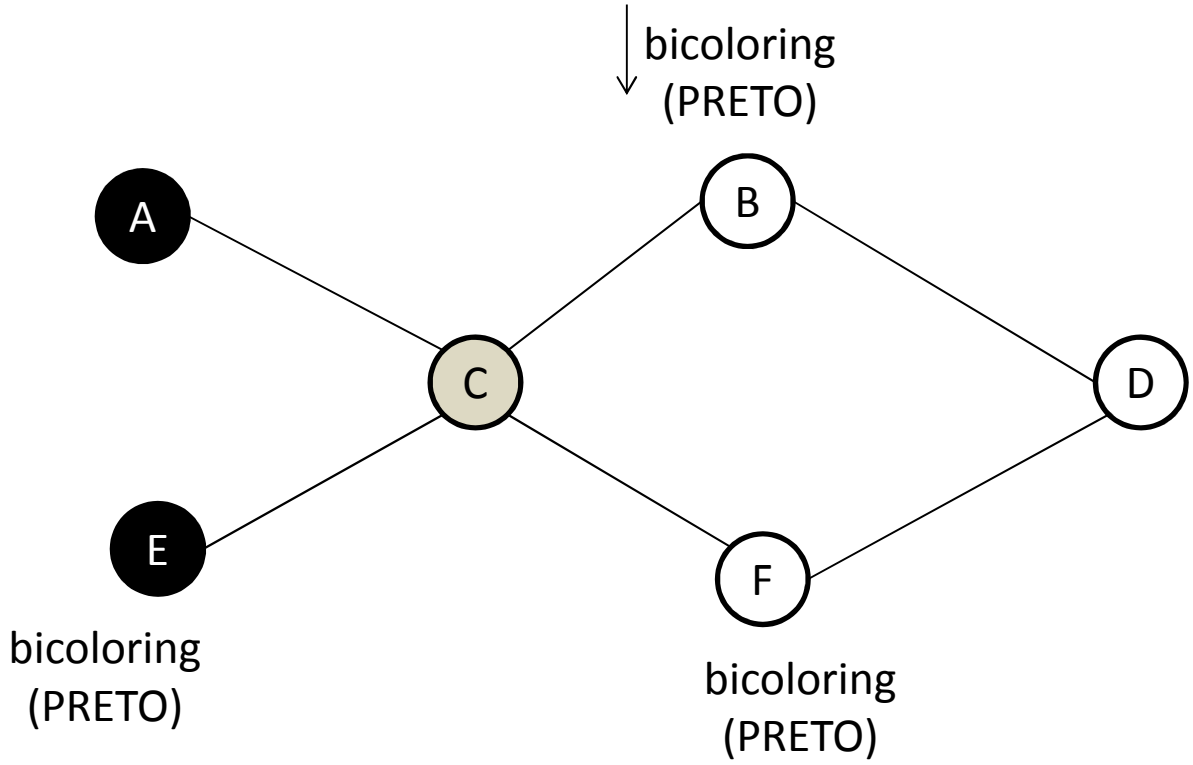
Algoritmo de Bicoloração



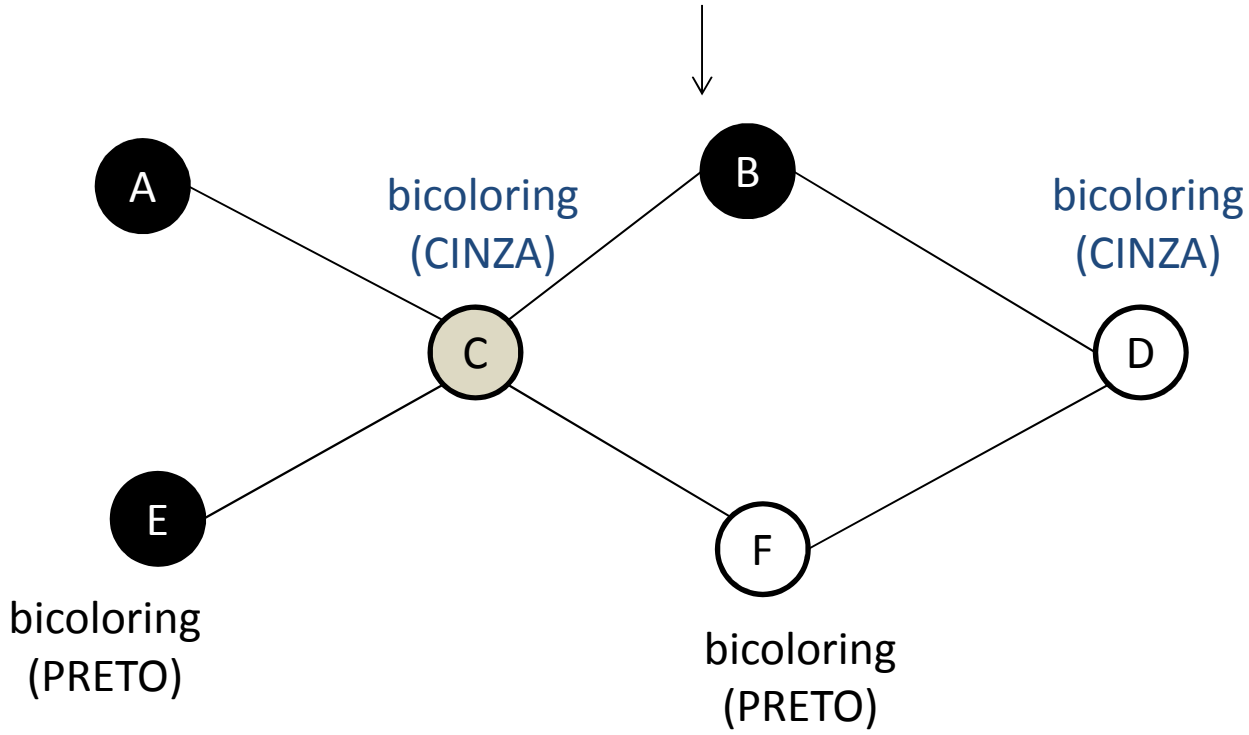
Algoritmo de Bicoloração



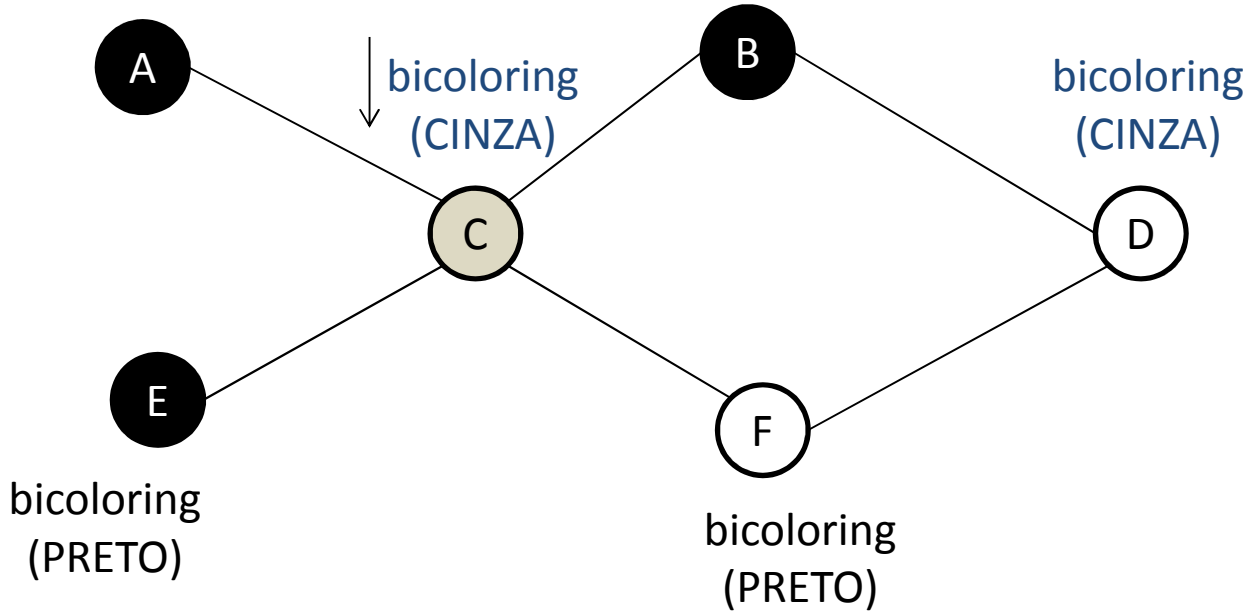
Algoritmo de Bicoloração



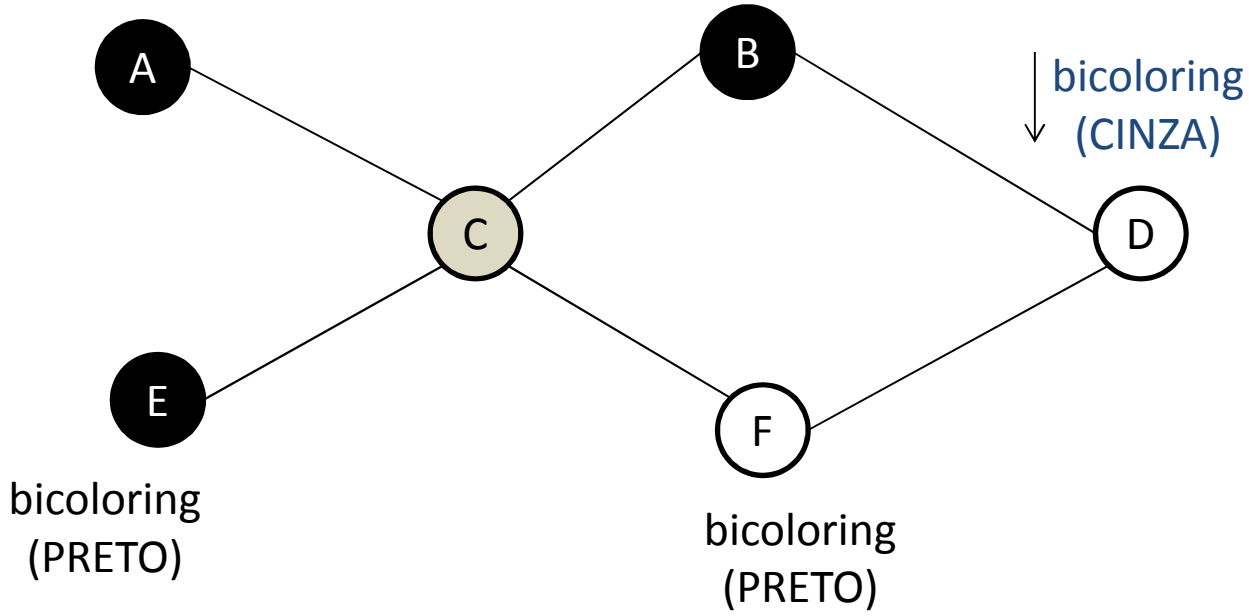
Algoritmo de Bicoloração



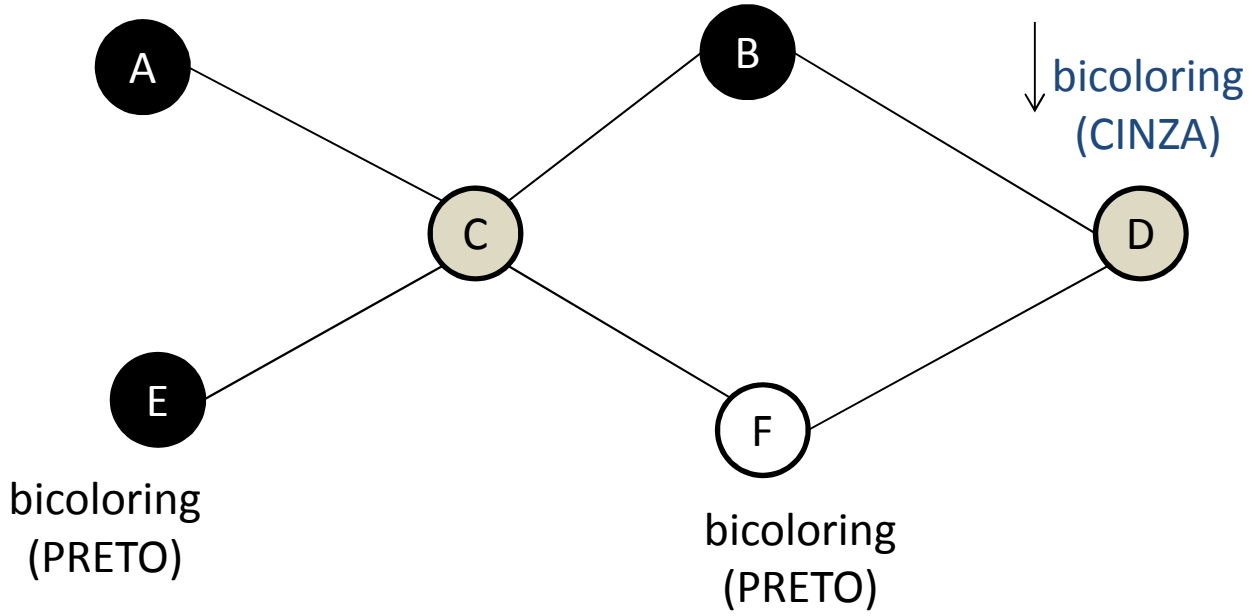
Algoritmo de Bicoloração



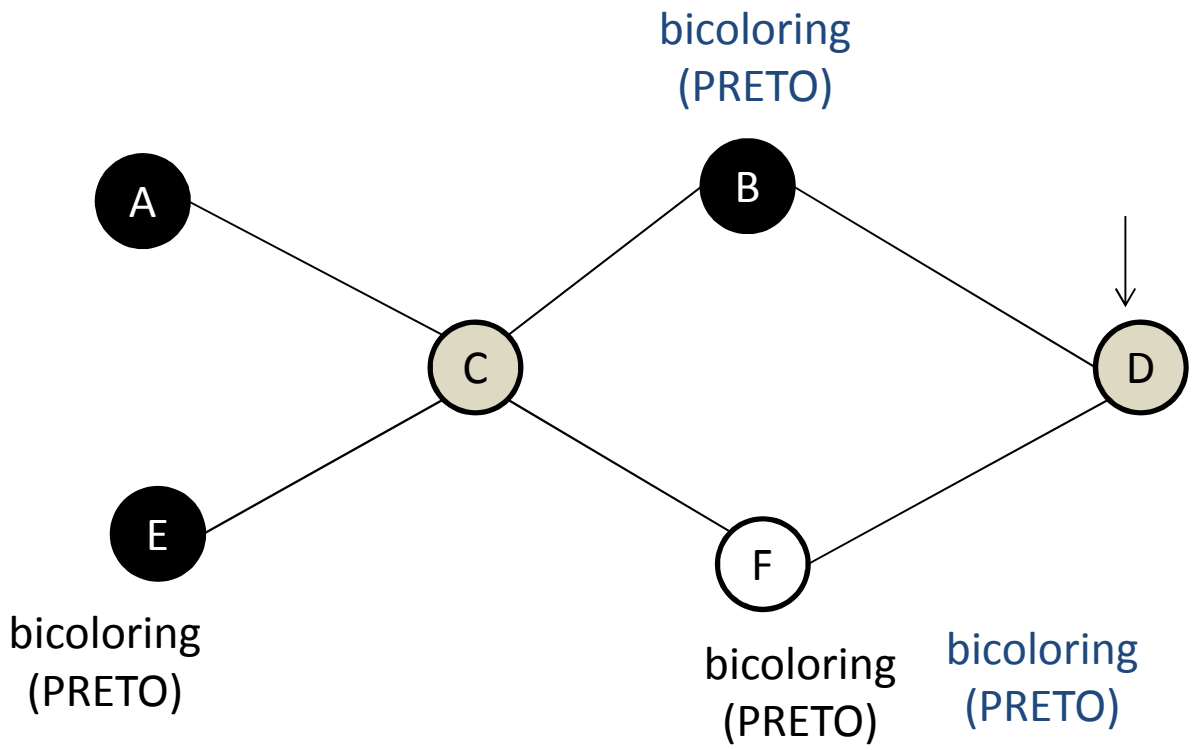
Algoritmo de Bicoloração



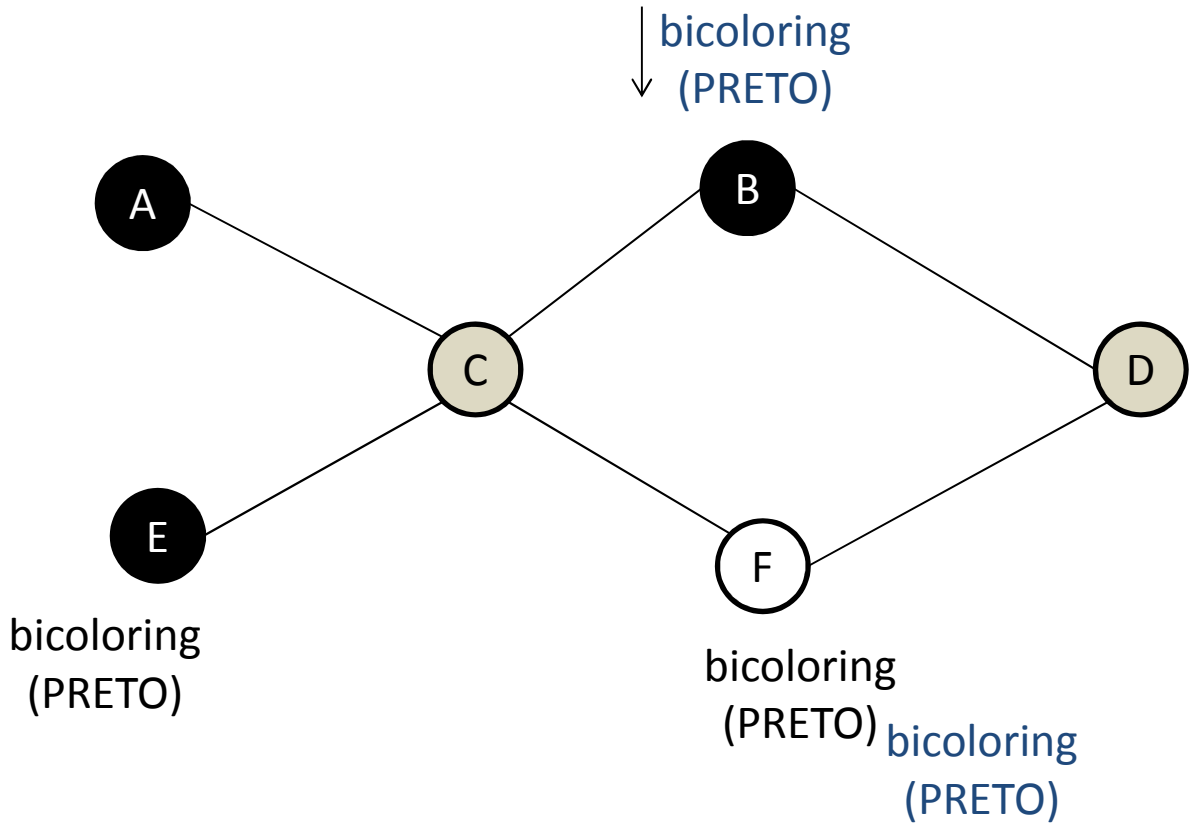
Algoritmo de Bicoloração



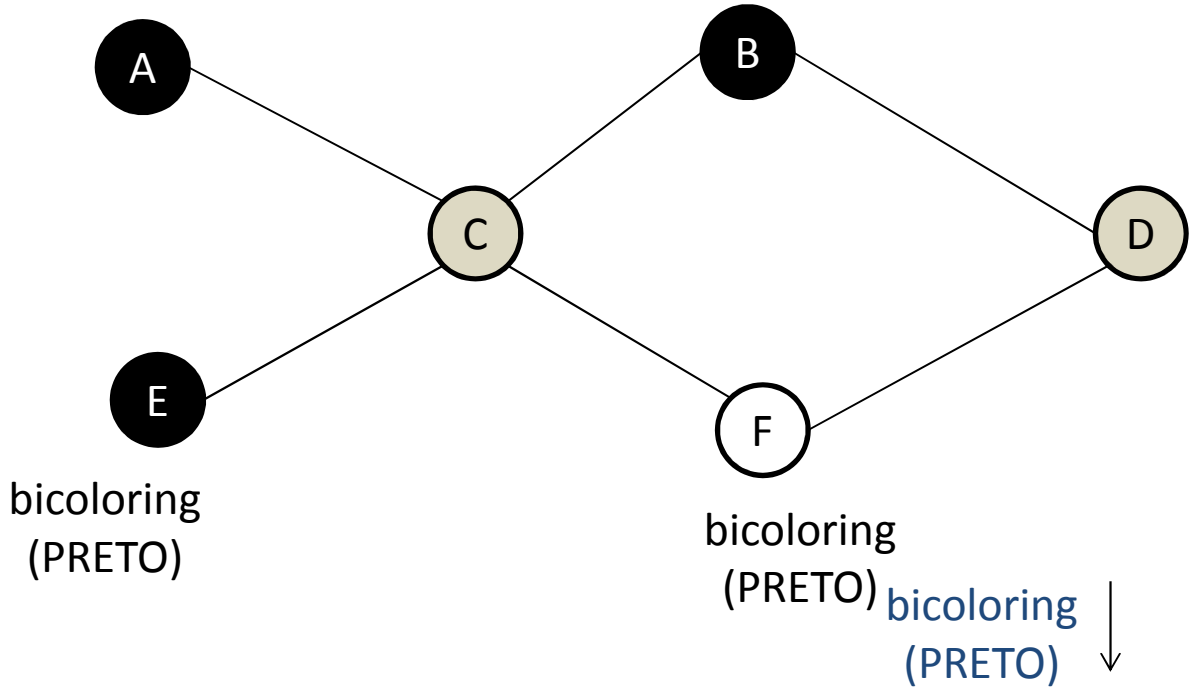
Algoritmo de Bicoloração



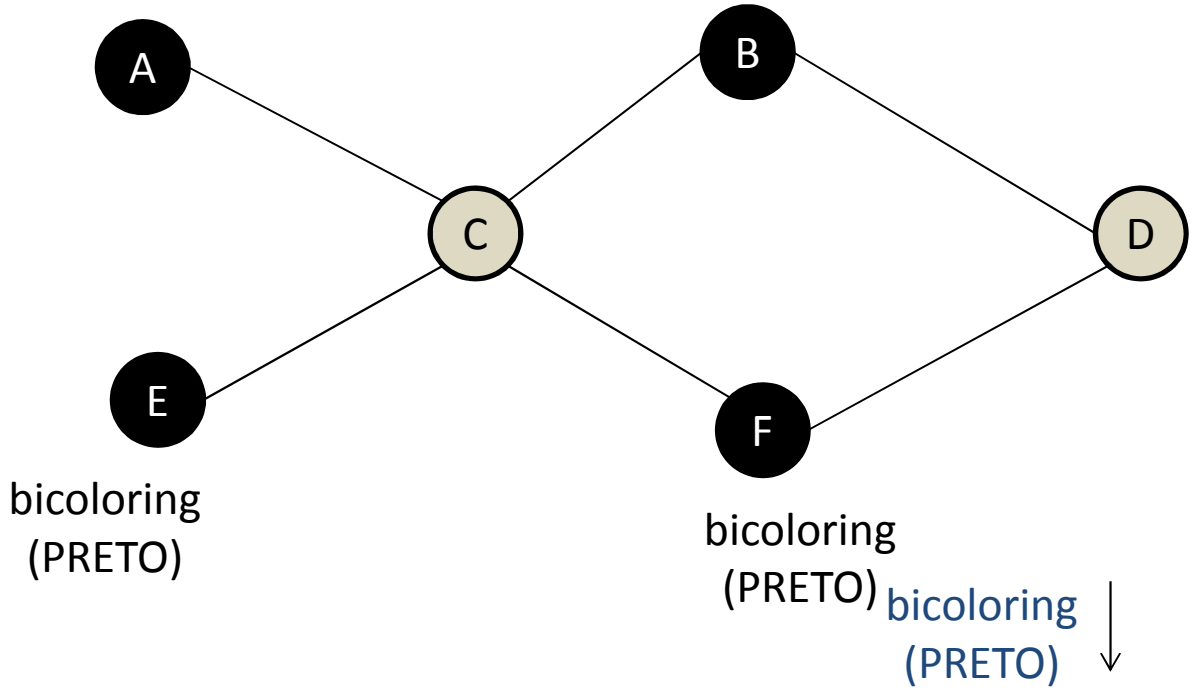
Algoritmo de Bicoloração



Algoritmo de Bicoloração



Algoritmo de Bicoloração



Bicoloração

```
bicoloring(Node u, Color c)
    se u.color=c então
        retorna VERDADEIRO;
    se u.color<>BRANCO então
        retorna FALSO;
    u.color = c;
    para todo v vizinho de u faça
        bicoloring(v, otherColor(c))
fim
```

Bicoloração

- Neste problema, você é desafiado a resolver um problema similar. Você tem que decidir se dado um grafo conexo qualquer, ele pode ser bicolorido. Ou seja, se alguém pode atribuir cores (de uma paleta de duas) aos nós de forma que nenhum nó adjacente tenha a mesma cor. Para simplificar, você pode supor que:
  - Nenhum nó terá uma aresta pra si mesmo.
  - O grafo será não-direcionado, ou seja, se um nó *a* está conectado a um nó *b*, então você deve assumir que *b* está conectado a *a*.
  - O grafo será fortemente conexo, ou seja, há pelo menos um caminho entre quaisquer dois nós distintos do grafo.

Entrada

A entrada consiste de muitos casos de teste. Cada caso começa com uma linha contendo um número n ($1 < n < 200$) de diferentes nós. A segunda linha contém o número de arestas l.

Depois disso, as l linhas seguintes possuirão dois números indicando que há uma aresta entre os dois nós numerados. Um nó no grafo será marcado com um número a ( ).

Uma entrada com $n = 0$ ditará o fim da entrada e não deve ser processada.

Saída

Você deve decidir se o grafo da entrada pode ser bicolorido e imprimir ``BICOLORABLE.", ou não, imprimindo ``NOT BICOLORABLE.", conforme o exemplo de saída mostrado abaixo.

Exemplo de Entrada

```
3
3
0 1
1 2
2 0
9
8
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
0
```

Exemplo de Saída

NOT BICOLORABLE.

BICOLORABLE.