Alocação de Registradores e Coloração de Grafos

Alexsandro Santos Soares
prof.asoares@gmail.com

Sumário

Sumário

Sumário

Sumário

Sumário

Coloração de grafos

- O problema de alocação de registradores pode ser formulado como um problema de coloração do grafo de interferência com k cores.
  - k é o número de registradores
- Usaremos um algoritmo aproximado com tempo linear, cujas principais fases são: Construa, Simplifique, Derrame e Selecione [1].

Coloração por simplificação

1. Construa:
   - Construir o grafo de interferência.
2. Simplifica
   - Ideia: suponha que o grafo G tenha um nó m com menos de k vizinhos, onde k é o número de registradores. Faça $G' = G - m$. Se G' pode ser colorido com k cores, G também pode.
   - Repetidamente:
     - Remova nós de grau menor que K
     - Coloque na pilha
     - Cada remoção diminui o grau dos nós em G, dando oportunidades para novas remoções.

Coloração por simplificação

1. Derramamento (*spill*):
   - Em algum momento não temos um nó com grau ¡ k, a heurística falha.
   - Temos que marcar algum nó para spill
   - A escolha desse nó é também uma heurística:
     - Nó que reduza o grau do maior número de outros nós
     - Nó com menor custo relacionado as operações de memória

Coloração por simplificação

1. Seleciona:
   - Atribui as cores
   - Reconstrói o grafo G adicionando os nós na ordem determinada pela pilha
   - Quando adicionamos um nó, devemos ter uma cor para ele dado o critério de seleção usado para remover
   - Isso não vale para os nós empilhados marcados como spill.
     - Se todos os vizinhos já usarem k cores, não adicionamos no grafo
     - Continua o processo

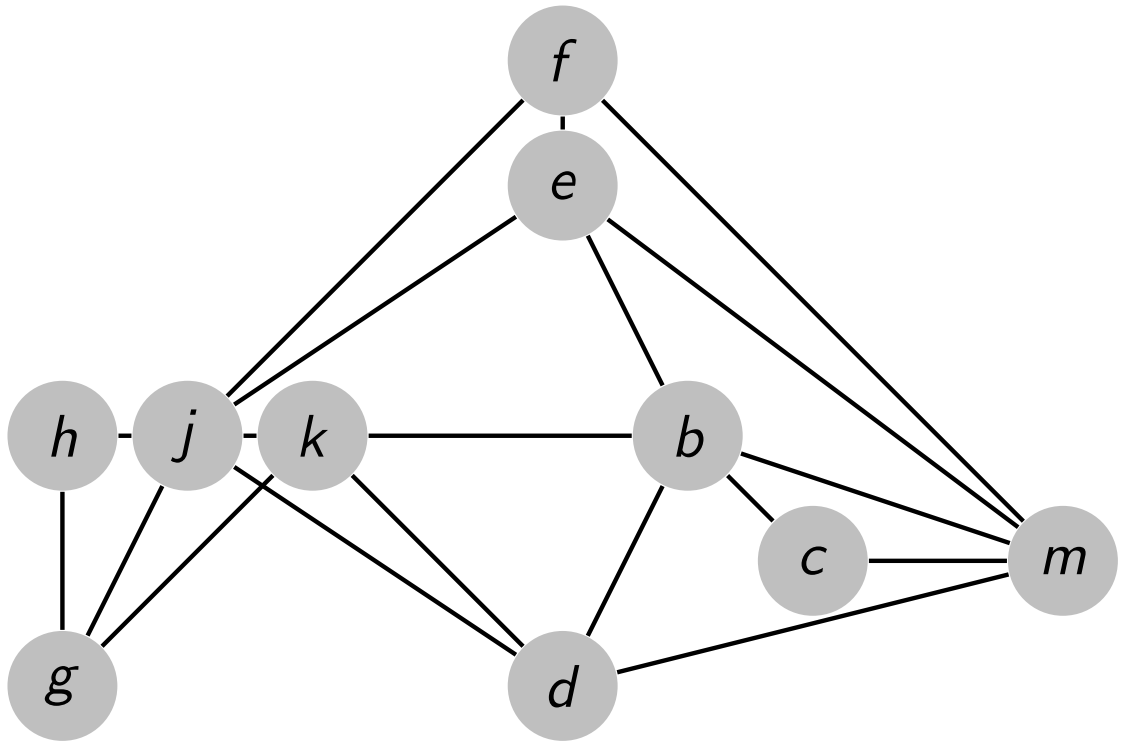
Coloração por simplificação

1. Recomeça:
   - Pode ser que Seleciona não consiga atribuir uma cor a algum nó
   - Reescreve o programa para pegar esse valor da memória antes de cada uso e armazená-lo de volta após o uso
   - Isso gera novos temporários, mas com tempos de vida mais curtos.
   - O algoritmo é repetido desde a construção do Grafo de Interferência
   - O processo acaba quando Seleciona tiver sucesso para todos os vértices

Exemplo de coloração

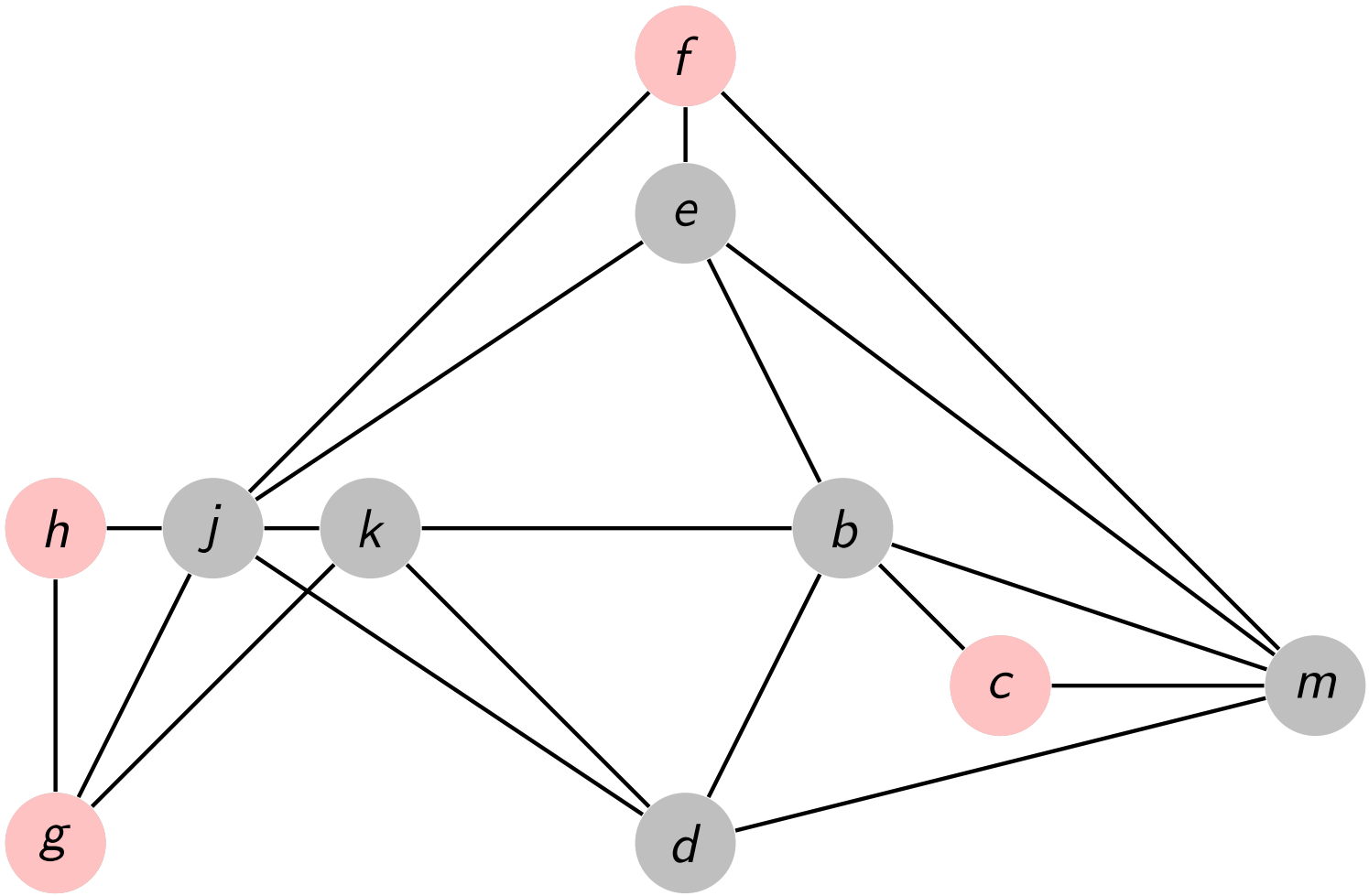
- Suponha que o programa abaixo e seu grafo de interferência sejam dados.
- Suponha também que temos apenas 4 registradores.

```
vivas na entrada: k j
    g := mem[j+12]
    h := k - 1
    f := g * h
    e := mem[j+8]
    m := mem[j+16]
    b := mem[f]
    c := e + 8
    d := c
    k := m + 4
    j := b
vivas na saída: d k j
```



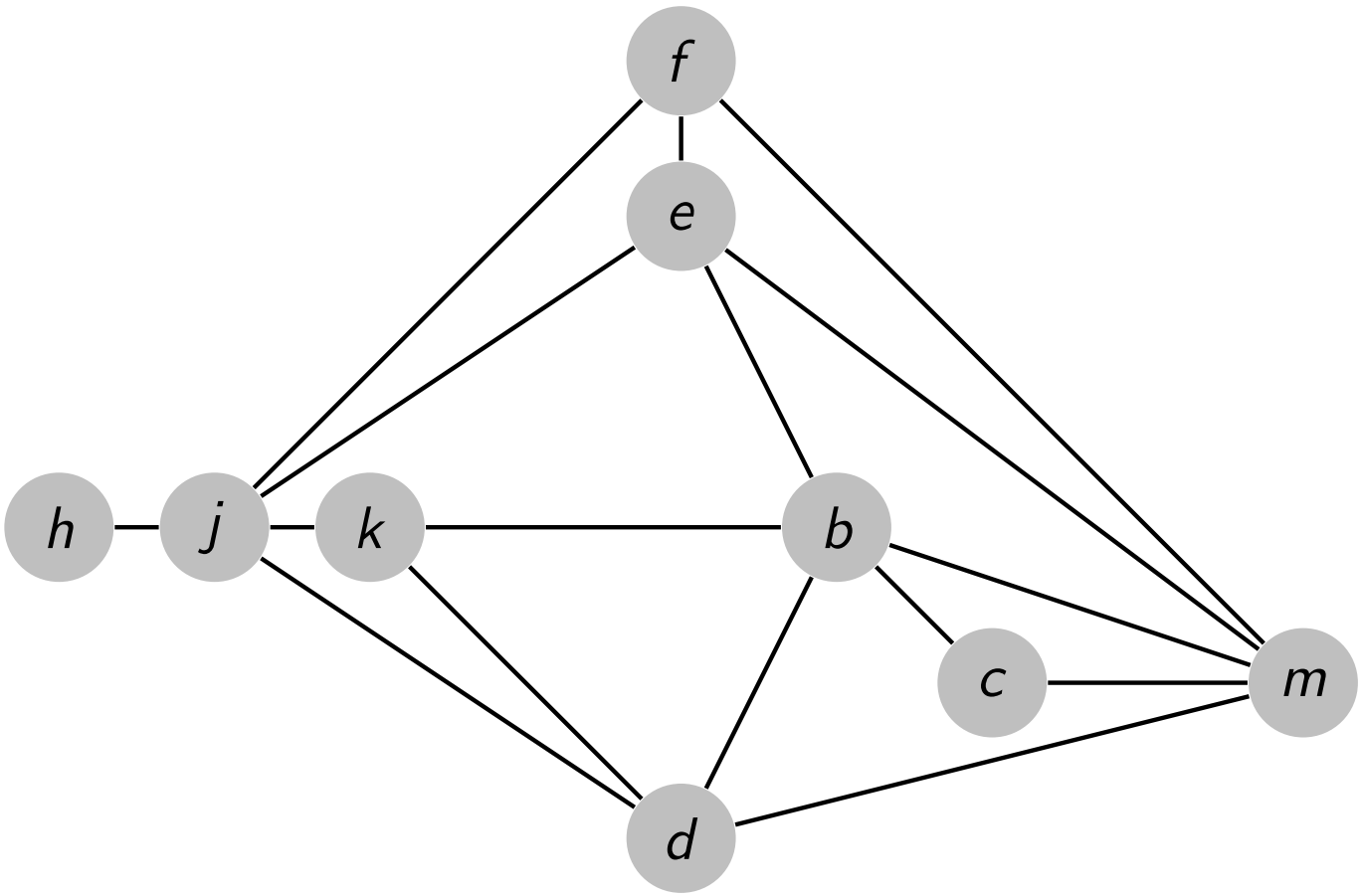
Exemplo de coloração: simplifica

- Podemos iniciar com os nós `g`, `h`, `c` e `f` pois cada um possui menos que 4 vizinhos.



Exemplo de coloração: simplifica

- O algoritmo inicia removendo g



g

pilha reg

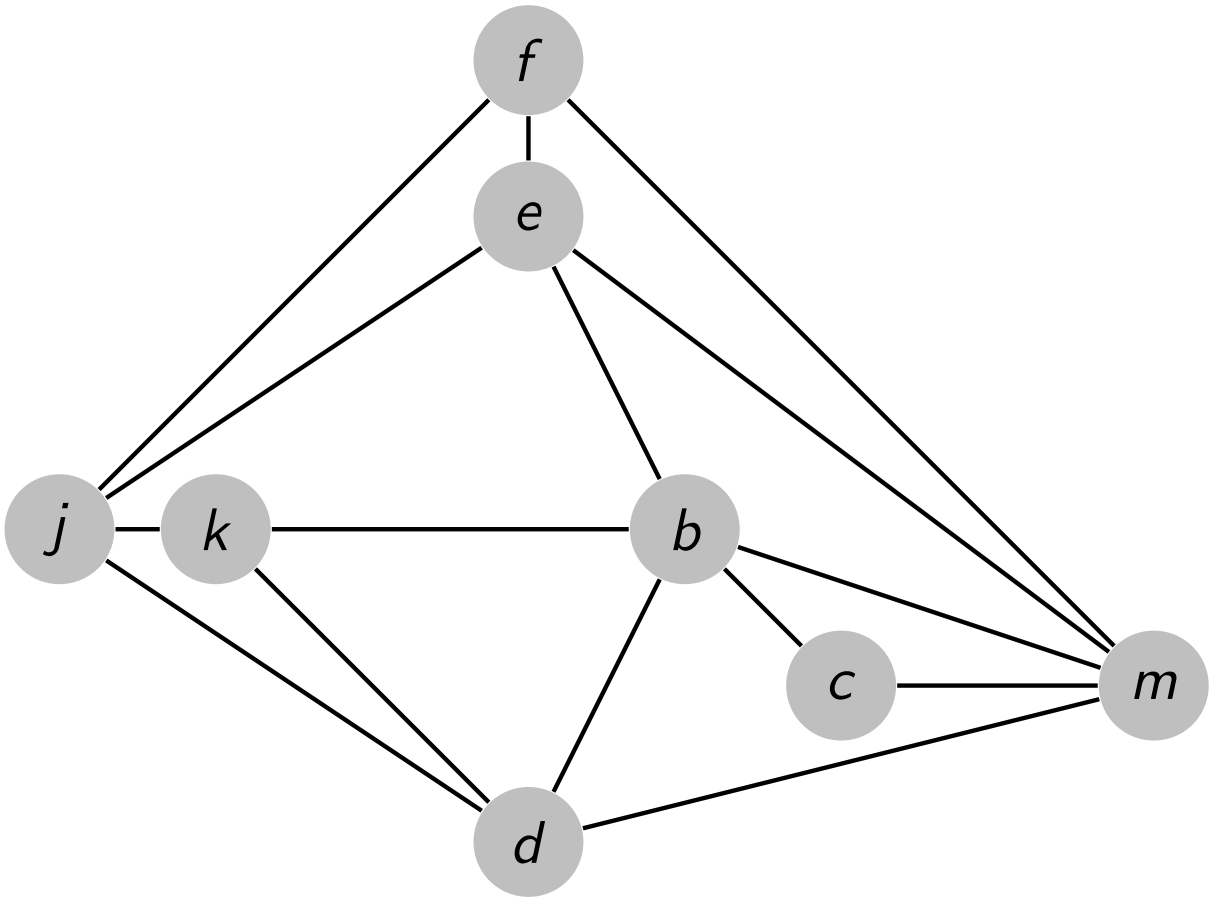
Exemplo de coloração: simplifica

- Agora é a vez de `h`

h

g

pilha	reg

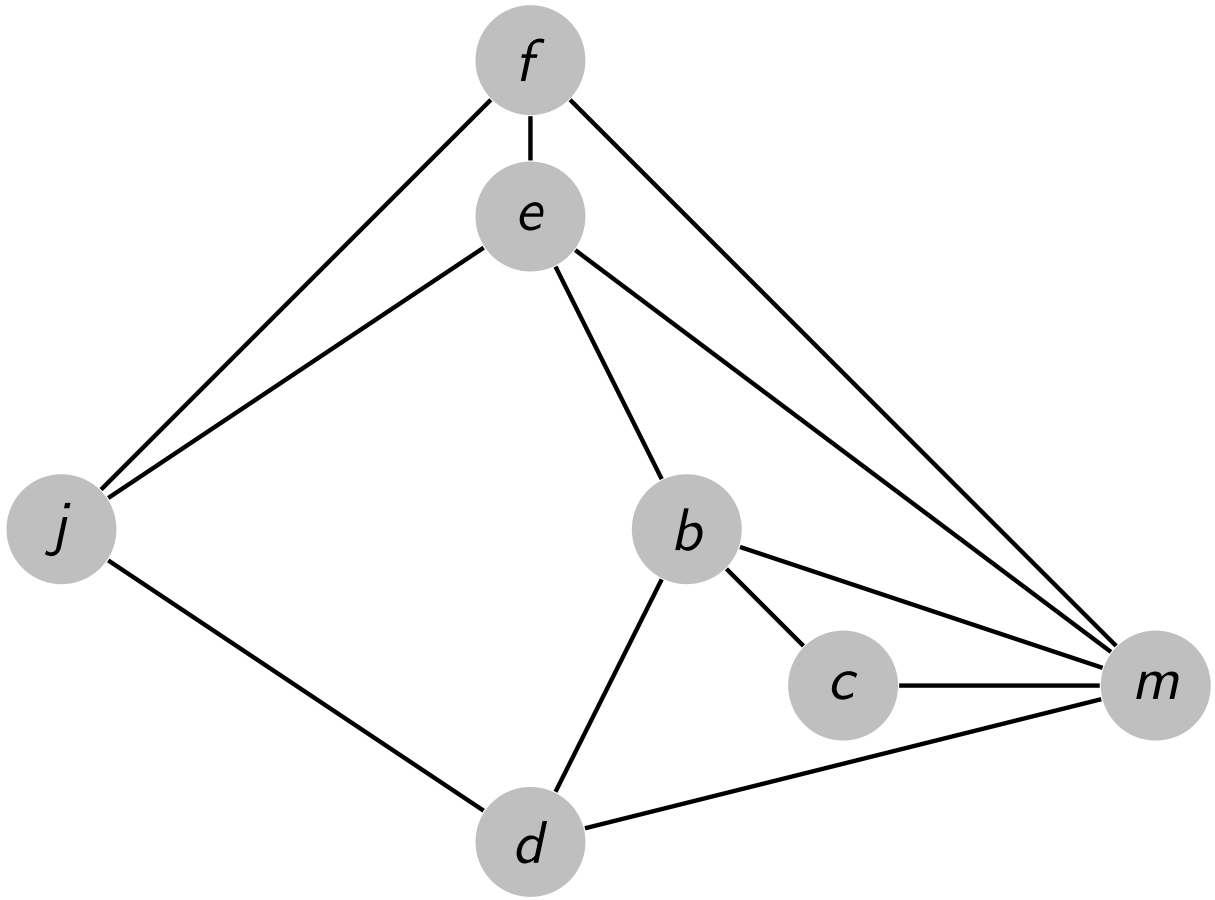


Exemplo de coloração: simplifica

- O algoritmo remove `k`

k
h
g

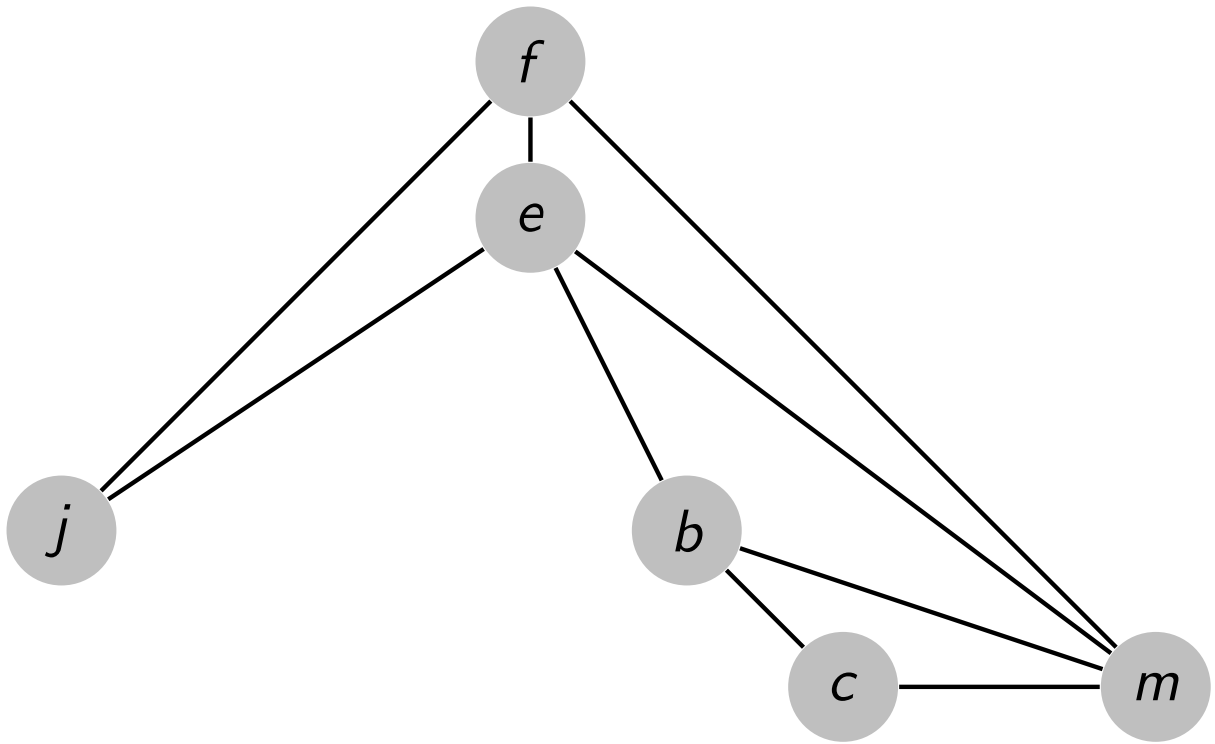
pilha	reg



Exemplo de coloração: simplifica

- O algoritmo remove d

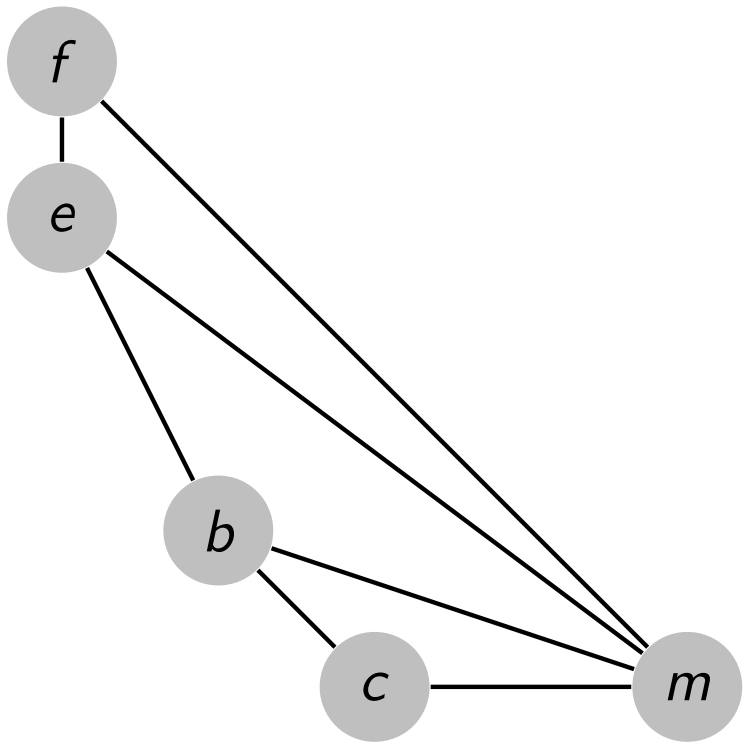
pilha	reg
d	
k	
h	
g	



Exemplo de coloração: simplifica

- O algoritmo remove `j`

pilha	reg
j	
d	
k	
h	
g	

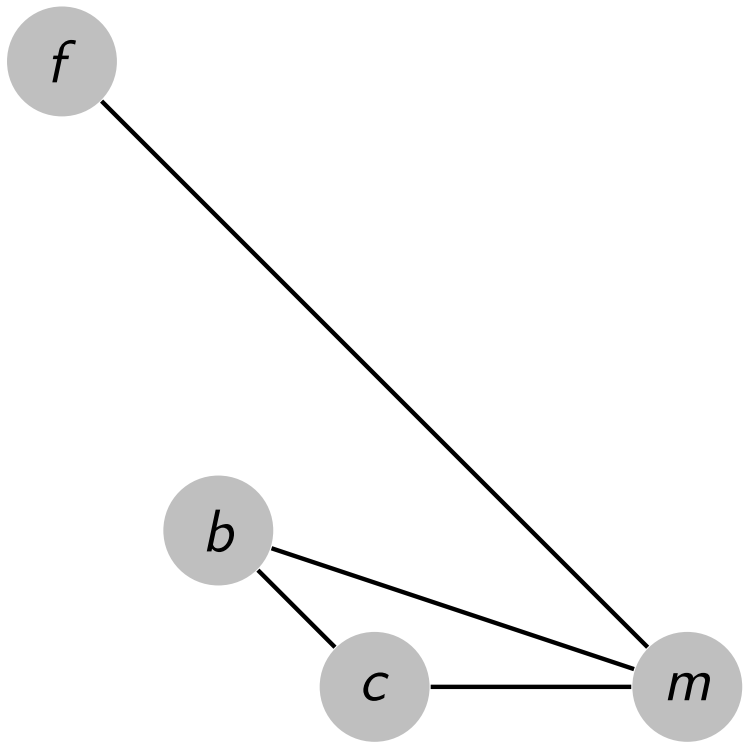


Exemplo de coloração: simplifica

- O algoritmo remove e

e
j
d
k
h
g

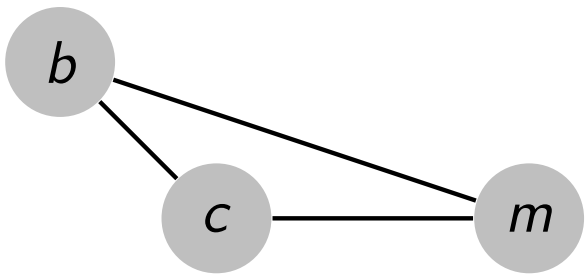
pilha reg



Exemplo de coloração: simplifica

- O algoritmo remove `f`

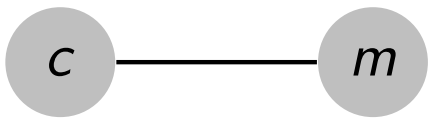
pilha	reg
f	
e	
j	
d	
k	
h	
g	



Exemplo de coloração: simplifica

- O algoritmo remove b

pilha	reg
b	
f	
e	
j	
d	
k	
h	
g	



Exemplo de coloração: simplifica

- Remove o nó c e depois o nó `m`

pilha	reg
m	
c	
b	
f	
e	
j	
d	
k	
h	
g	

Exemplo de coloração: seleciona

- Os nós são desempilhados e o grafo original é reconstruído e colorido ao mesmo tempo.
- Iniciando com `m`, uma cor é escolhida arbitrariamente, pois o grafo neste instante consiste apenas de um único nó.

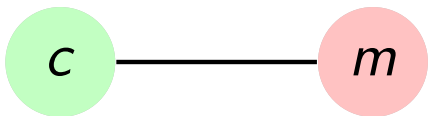
c
b
f
e
j
d
k
h
g

pilha reg

Exemplo de coloração: seleciona

- Na sequência os demais nós são desempilhados e uma cor é atribuída a cada um deles respeitando a configuração atual do grafo.

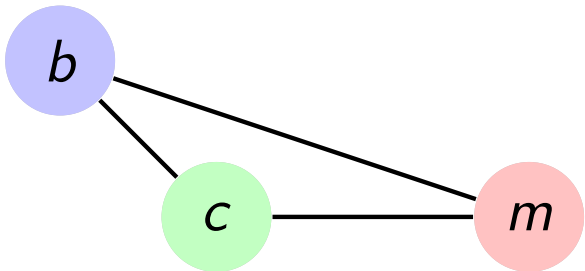
pilha	reg
b	
f	
e	
j	
d	
k	
h	
g	



Exemplo de coloração: seleciona

f
e
j
d
k
h
g

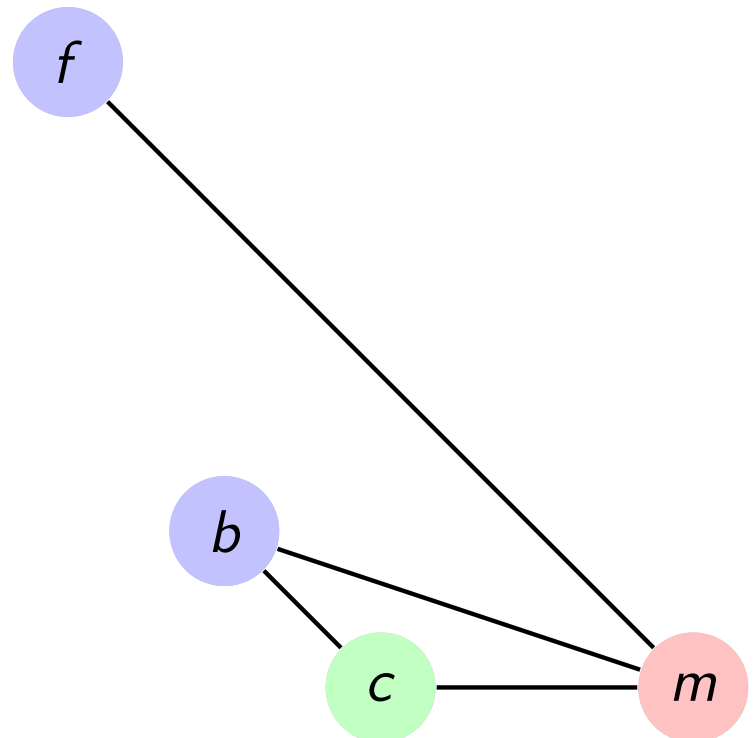
pilha reg



Exemplo de coloração: seleciona

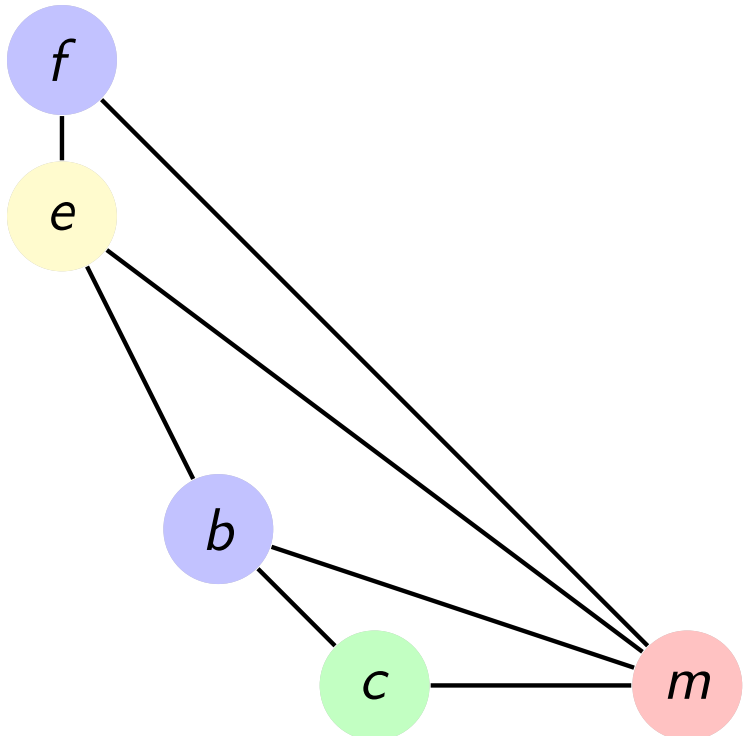
e
j
d
k
h
g

pilha reg



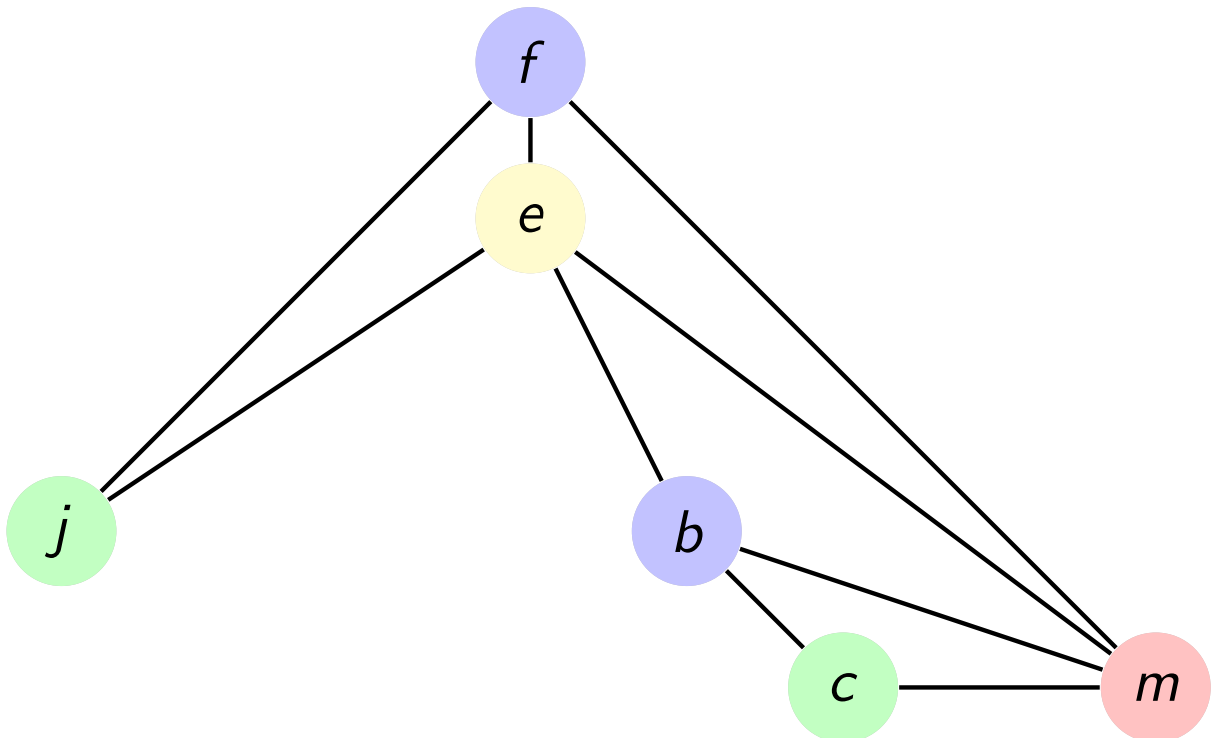
Exemplo de coloração: seleciona

pilha	reg
j	
d	
k	
h	
g	



Exemplo de coloração: seleciona

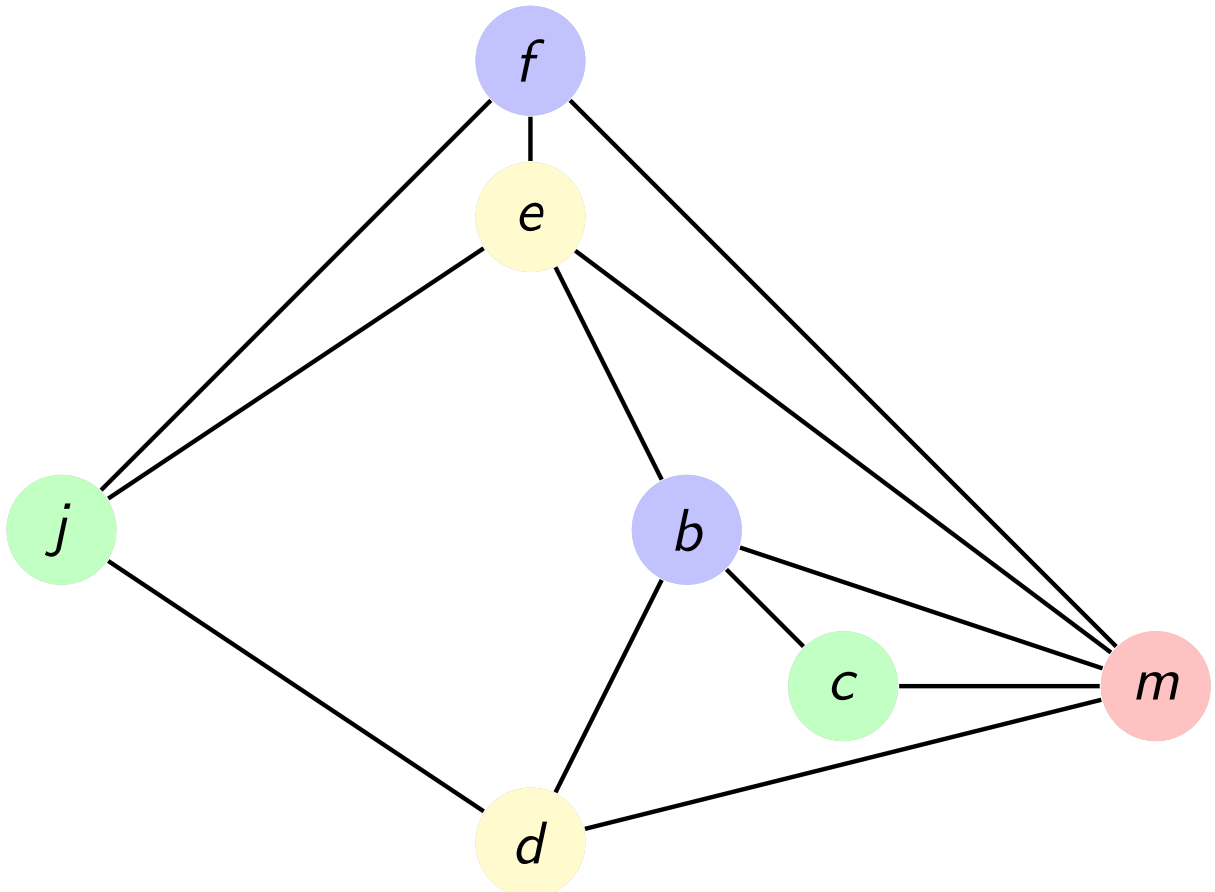
pilha	reg
d	
k	
h	
g	



Exemplo de coloração: simplifica

k
h
g

pilha	reg

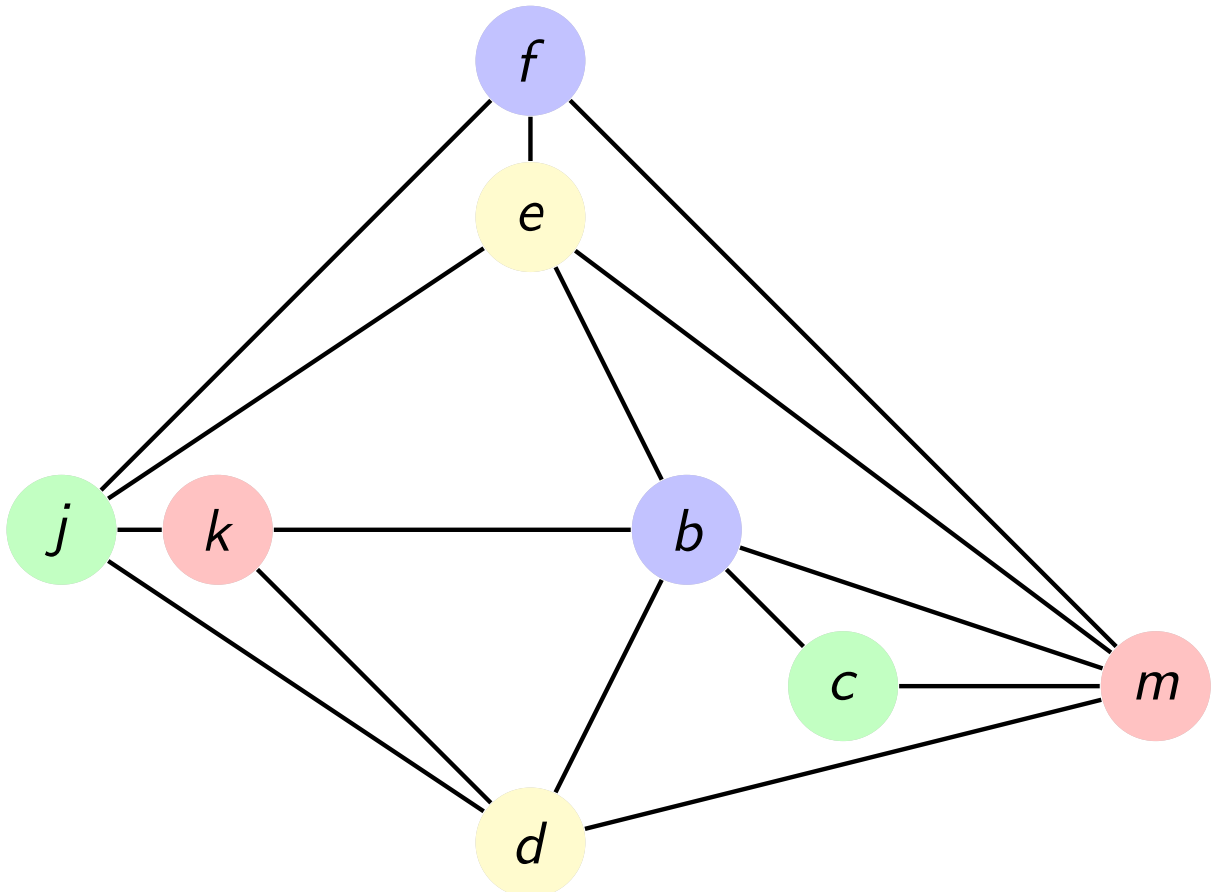


Exemplo de coloração: seleciona

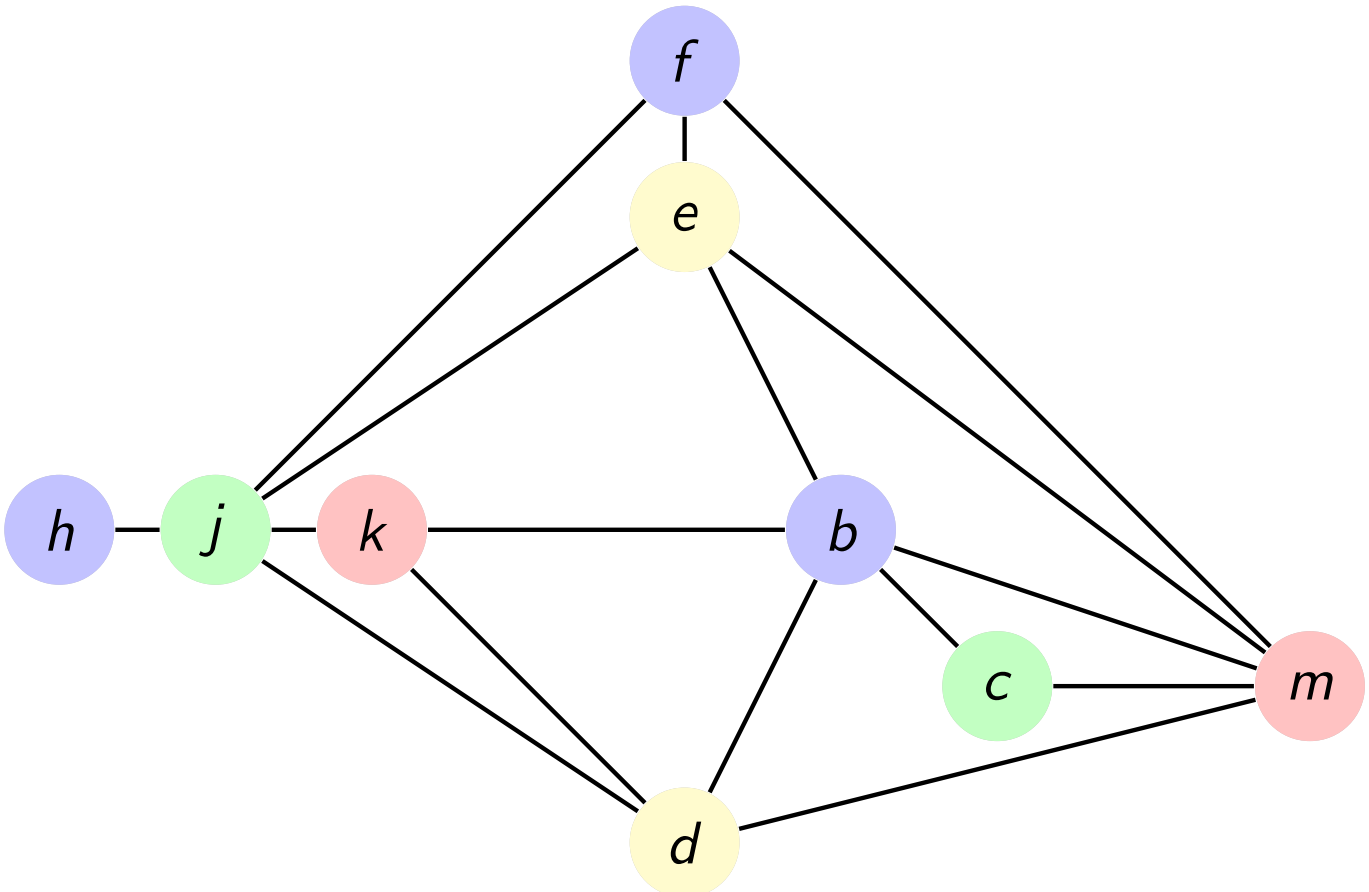
h

g

pilha	reg



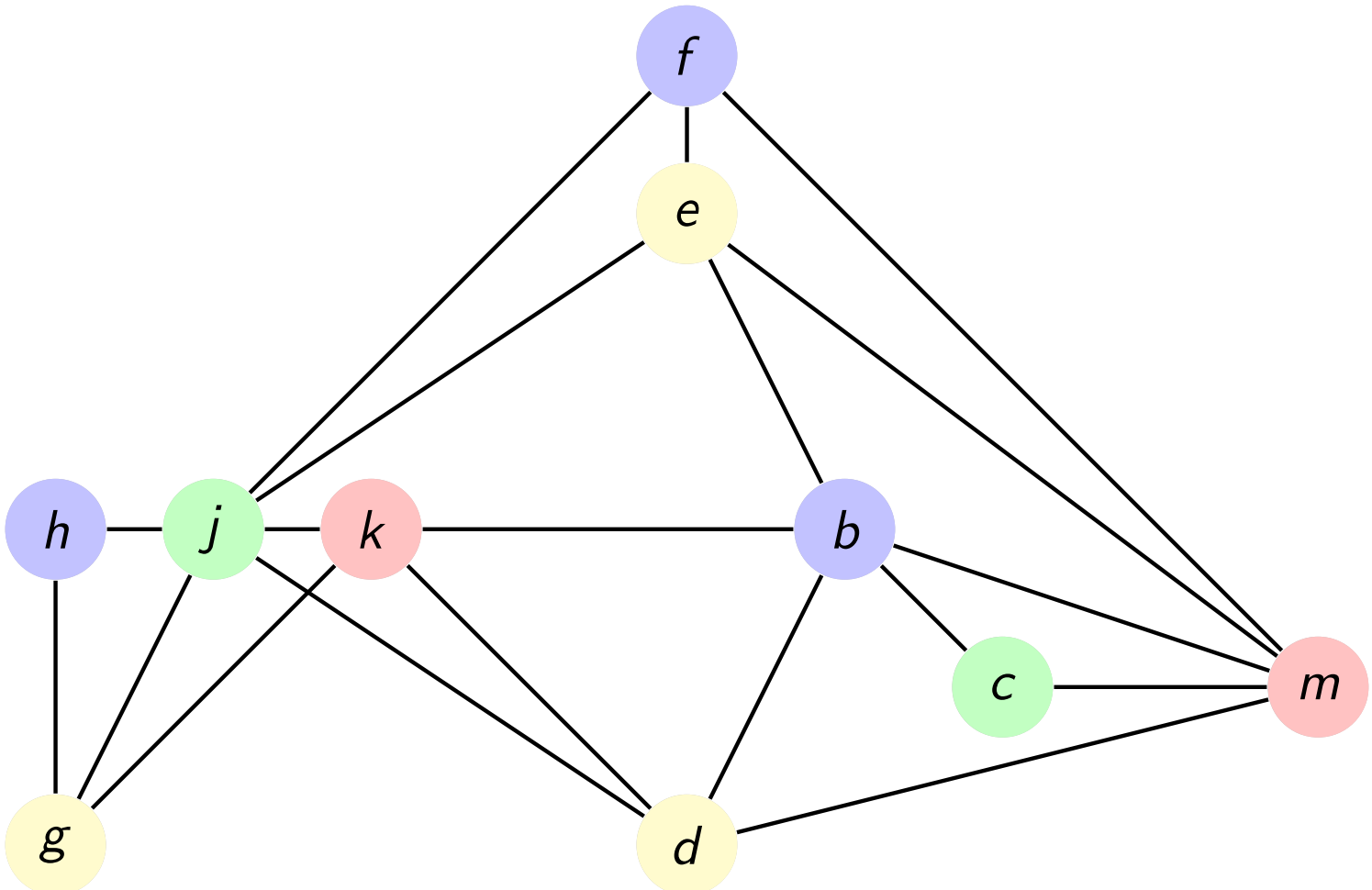
Exemplo de coloração: seleciona



g

pilha reg

Exemplo de coloração: seleciona



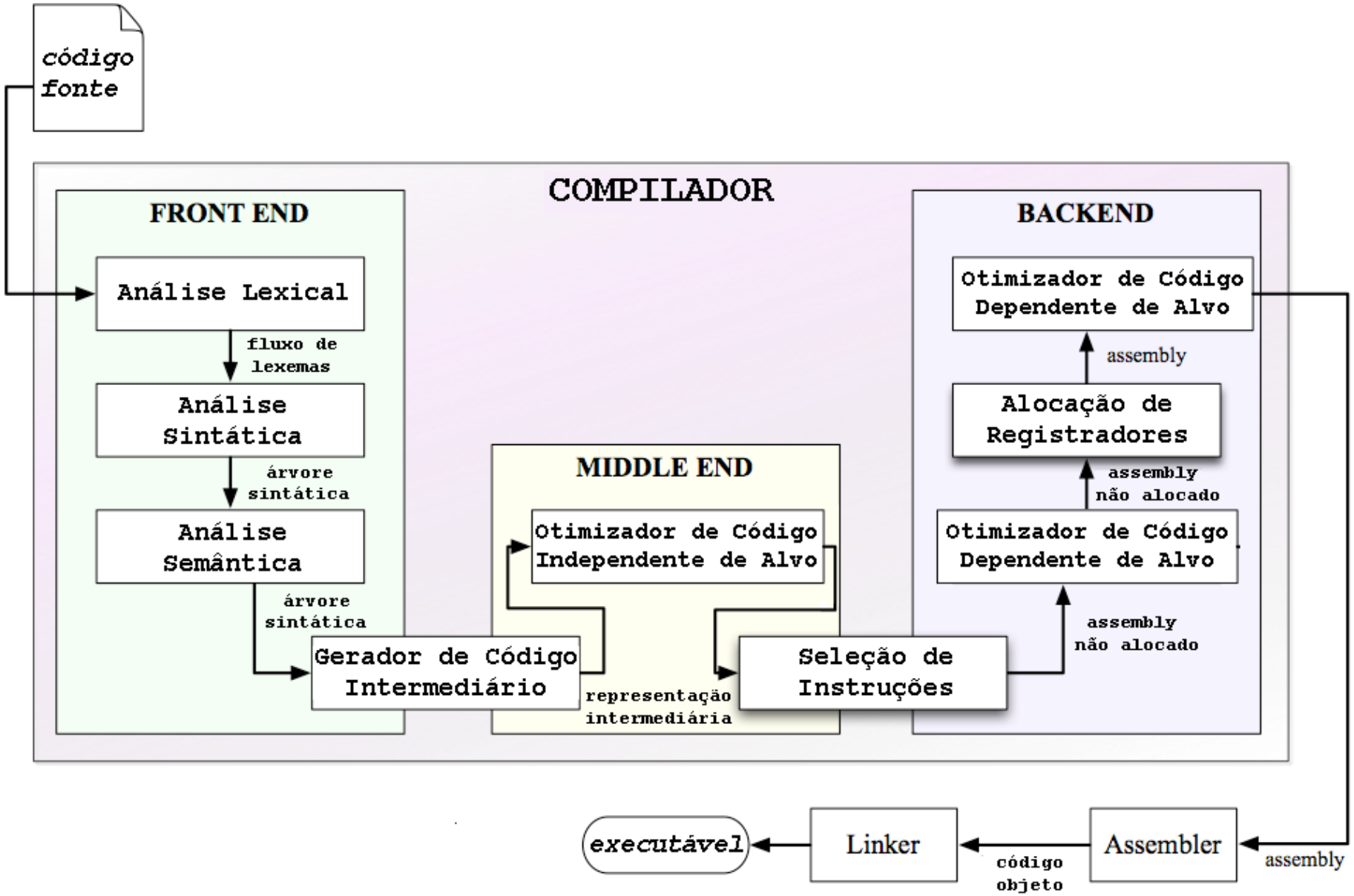
Escalonamento de instruções

- A ordem em que as computações são efetuadas pode afetar a eficiência do código objeto.
- Algumas ordens exigem menos registradores para manter resultados intermediários que outras.
- A seleção da melhor ordem no caso geral é um problema NP-completo [2].
- Encontrar uma boa ordem para uma determinada arquitetura é a tarefa do escalonador de instruções

Outras otimizações

- Existem muitas outras optimizações:
  - Loop unrolling (existe uma série de otimizações ao nível de loops)
  - propagação de constantes
  - redução de força (strength reduction)
  - avaliação de expressões com constantes
  - eliminação de sub-expressões comuns,
  - eliminação de acessos a memória redundantes
  - eliminação de código morto
  - movimento de código

Módulos e interfaces de compilador



Referências


APPEL, A. W. *Modern Compiler Implementation in Java, 2nd edition*. [S.l.]: Cambridge University Press, 2002.


AHO, A. V. et al. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. [S.l.]: Addison Wesley, 2006. Hardcover.