

Algoritmos de aproximação

CLRS 35

Transparências de Cristina Gomes Fernandes
MAC5727 Algoritmos de Aproximação

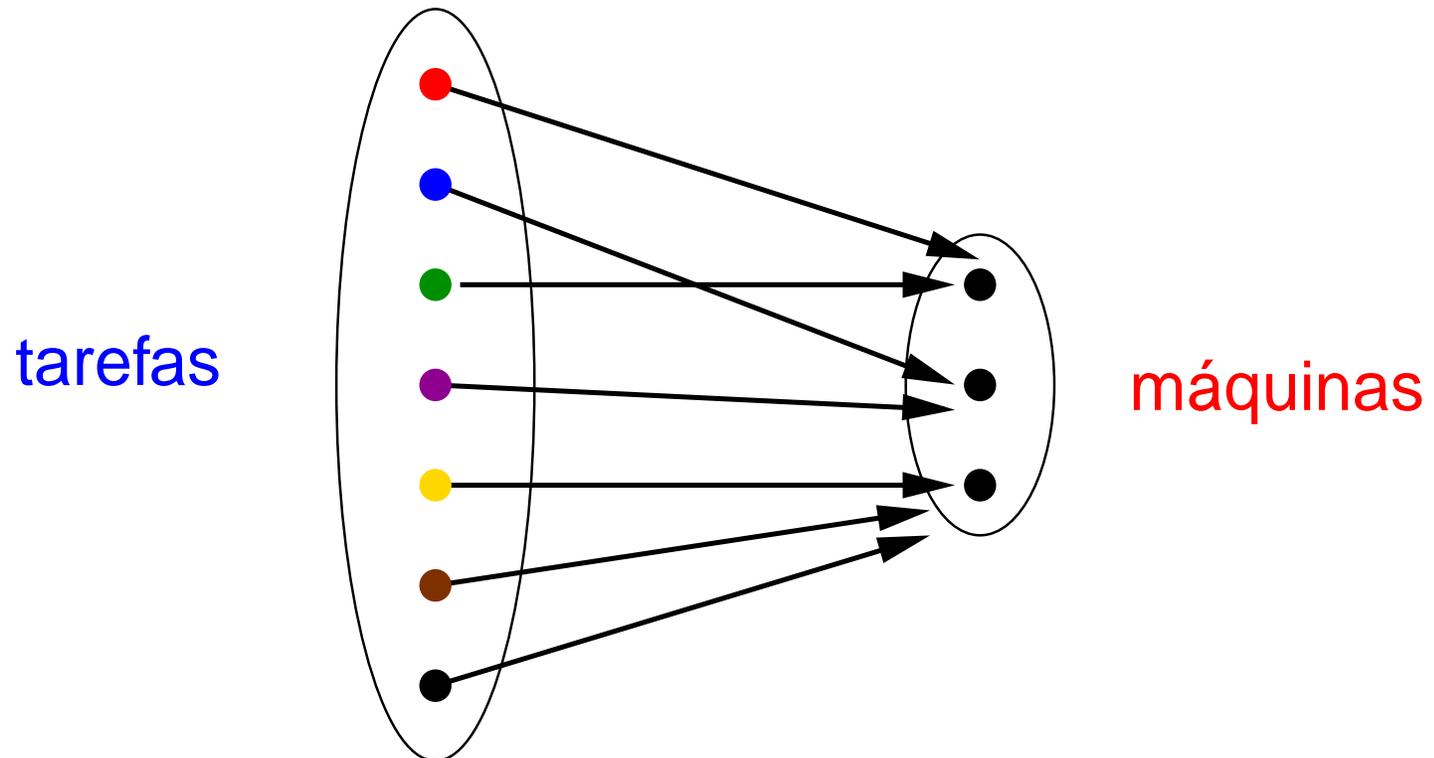
Escalonamento de máquinas idênticas

Dados: m máquinas

t tarefas

duração $d[i]$ da tarefa i ($i = 1, \dots, t$)

um **escalonamento** é uma **partição** $\{M[1], \dots, M[m]\}$
de $\{1, \dots, t\}$



Exemplo 1

$$m = 3 \quad t = 7$$

						
$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$
3	2	7	5	1	6	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$M[1]$	yellow	yellow	yellow	brown	brown	brown	brown	brown	black	black				
$M[2]$	blue	blue	purple											
$M[3]$	red	red	red	red	red	red	red	green	green	green	green	green	green	

$\{\{1, 4, 7\}, \{2, 5\}, \{3, 6\}\} \Rightarrow$ Tempo de conclusão = 13

Exemplo 2

$$m = 3 \quad t = 7$$

						
$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$
3	2	7	5	1	6	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$M[1]$	yellow	yellow	yellow	blue	blue	red	red	red	red	red	red	red		
$M[2]$	brown	brown	brown	brown	brown	purple								
$M[3]$	green	green	green	green	green	green	black	black						

$\{\{1, 2, 3\}, \{4, 5\}, \{6, 7\}\} \Rightarrow$ Tempo de conclusão = 12

Problema

Encontrar um escalonamento com tempo de conclusão **mínimo**.



$d[1]$
3



$d[2]$
2



$d[3]$
7



$d[4]$
5



$d[5]$
1



$d[6]$
6



$d[7]$
2

1

2

3

4

5

6

7

8

9

10

11

12

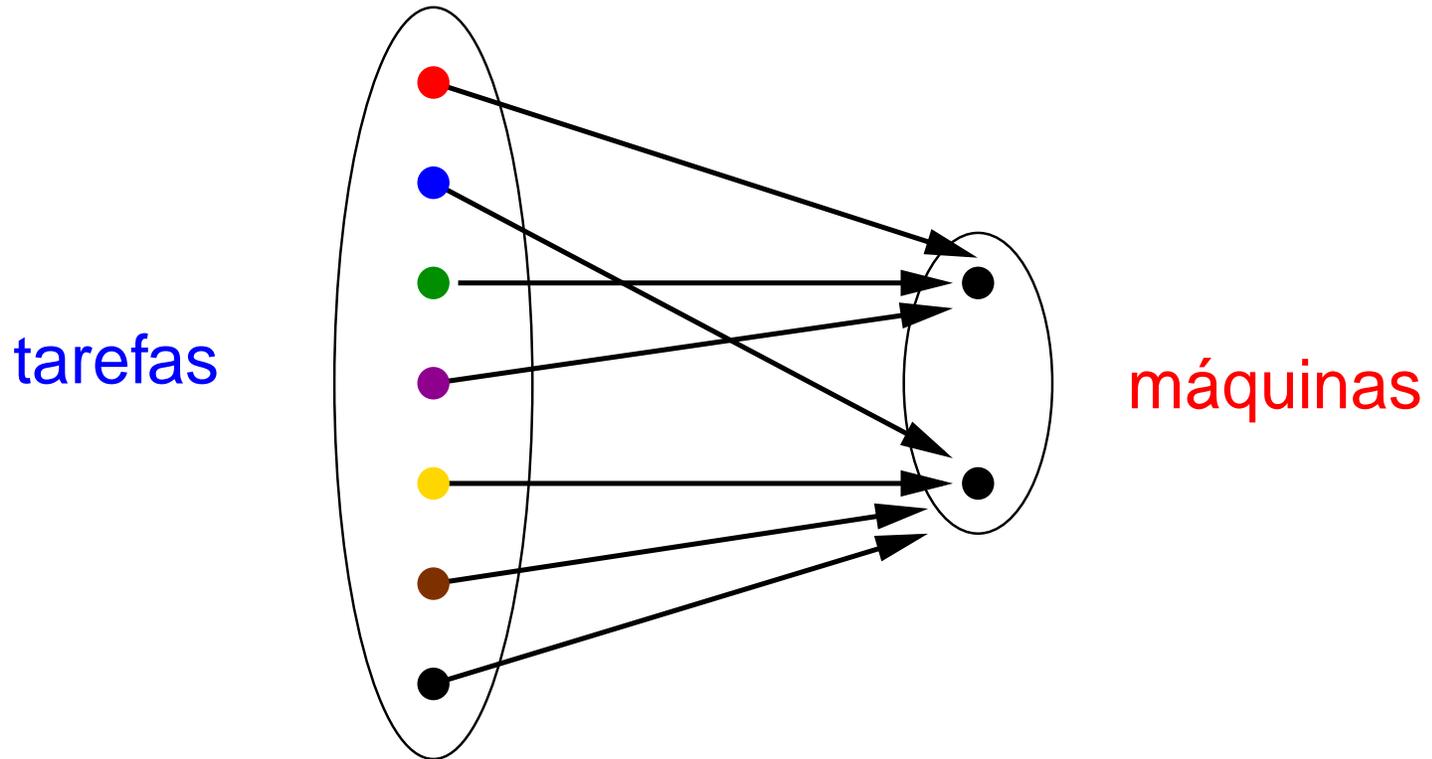
13

14

$M[1]$	Yellow	Yellow	Yellow	Brown	Brown	Brown	Brown	Brown							
$M[2]$	Blue	Blue	Red	Red	Red	Red	Red	Red	Red						
$M[3]$	Purple	Green	Green	Green	Green	Green	Green	Black	Black						

$\{\{1, 4\}, \{2, 3\}, \{5, 6, 7\}\} \Rightarrow$ Tempo de conclusão = 9

NP-difícil mesmo para $m = 2$



Algoritmo: testa todo $M[1] \subseteq \{1, \dots, t\}$ e escolhe melhor 2^t subconjuntos \Rightarrow **exponencial**

NP-difícil \Rightarrow é improvável que exista algoritmo **polinomial** que resolva o problema (se existir, **P = NP**)

Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



$d[1]$
3



$d[2]$
2



$d[3]$
7



$d[4]$
5



$d[5]$
1



$d[6]$
6



$d[7]$
2

1

2

3

4

5

6

7

8

9

10

11

12

13

14

$M[1]$															
$M[2]$															
$M[3]$															

Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



$d[1]$
3



$d[2]$
2



$d[3]$
7



$d[4]$
5



$d[5]$
1



$d[6]$
6



$d[7]$
2

1

2

3

4

5

6

7

8

9

10

11

12

13

14

$M[1]$															
$M[2]$															
$M[3]$															

Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



$d[1]$
3



$d[2]$
2



$d[3]$
7



$d[4]$
5



$d[5]$
1



$d[6]$
6



$d[7]$
2

1

2

3

4

5

6

7

8

9

10

11

12

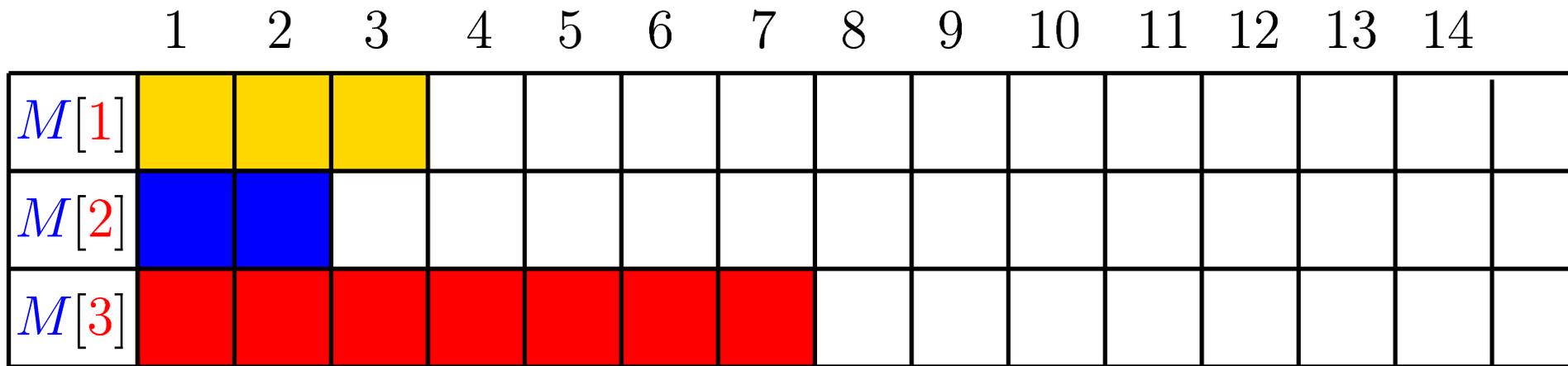
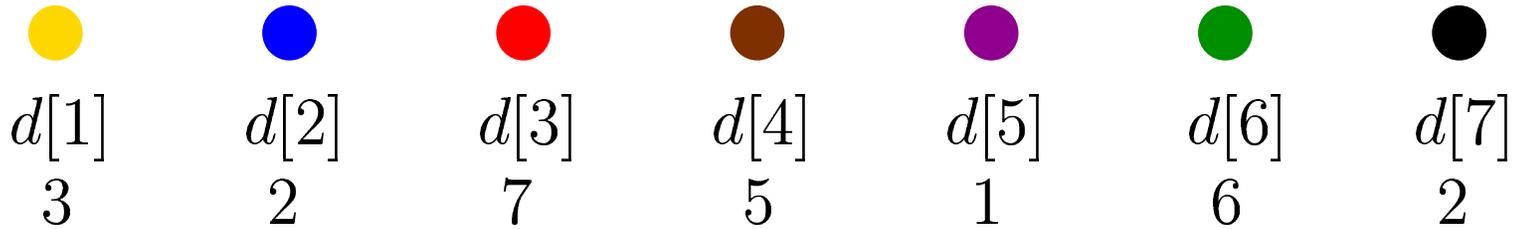
13

14

$M[1]$															
$M[2]$															
$M[3]$															

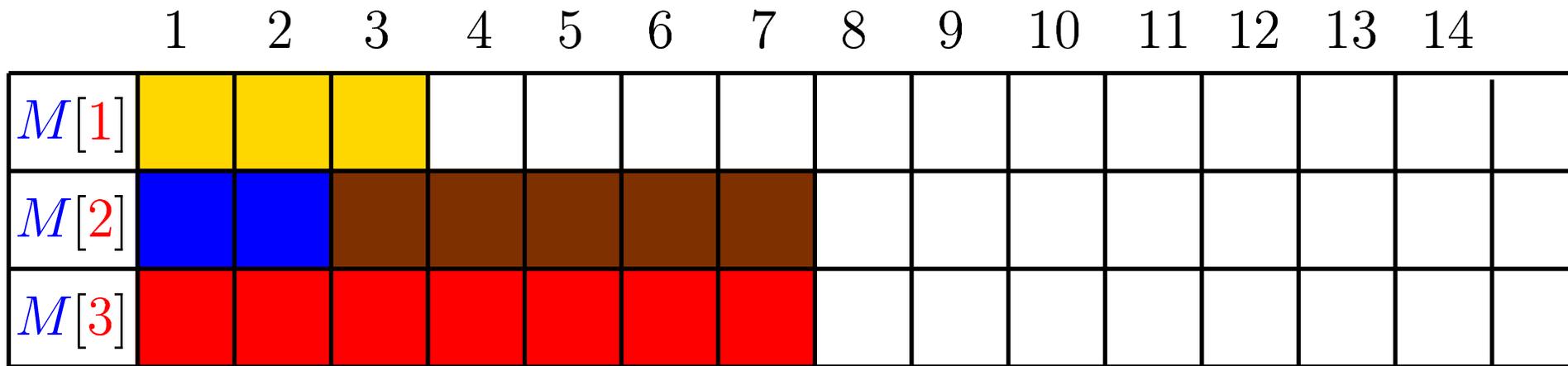
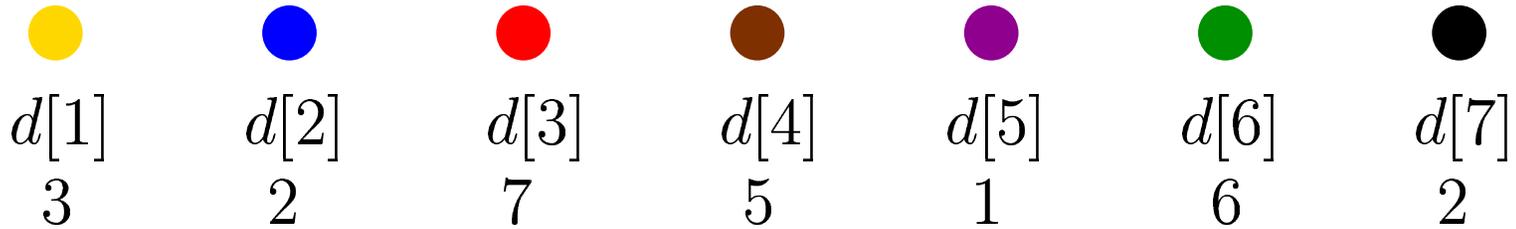
Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



$d[1]$
3



$d[2]$
2



$d[3]$
7



$d[4]$
5



$d[5]$
1



$d[6]$
6



$d[7]$
2

1

2

3

4

5

6

7

8

9

10

11

12

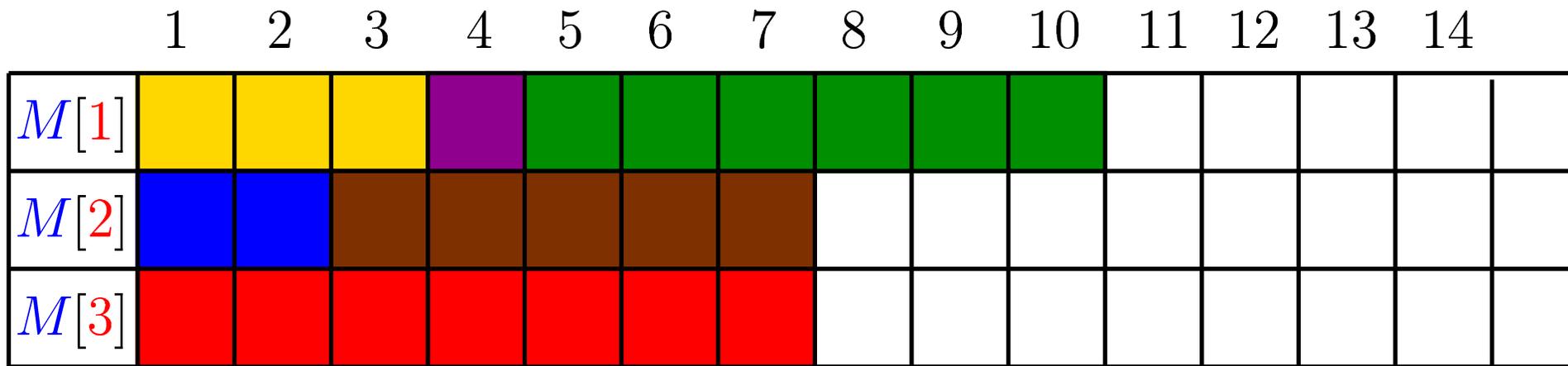
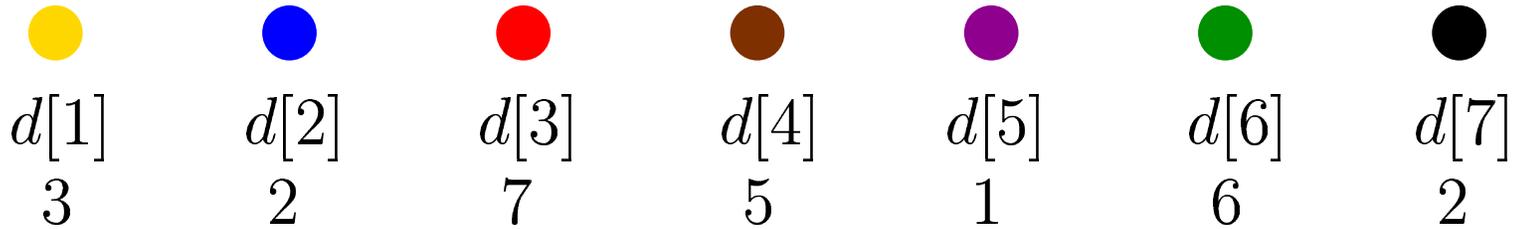
13

14

$M[1]$	Yellow	Yellow	Yellow	Purple										
$M[2]$	Blue	Blue	Brown	Brown	Brown	Brown	Brown							
$M[3]$	Red	Red	Red	Red	Red	Red	Red							

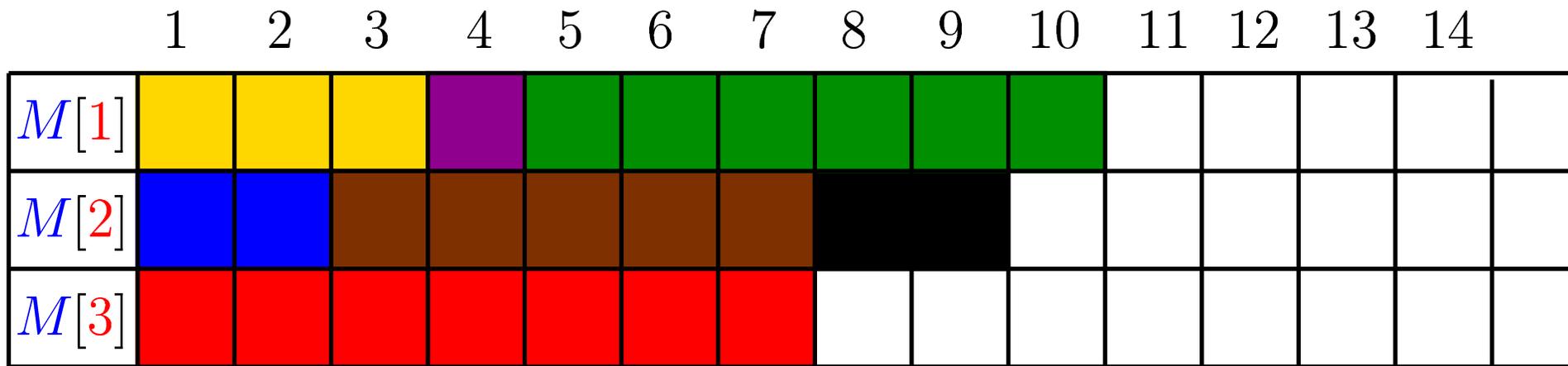
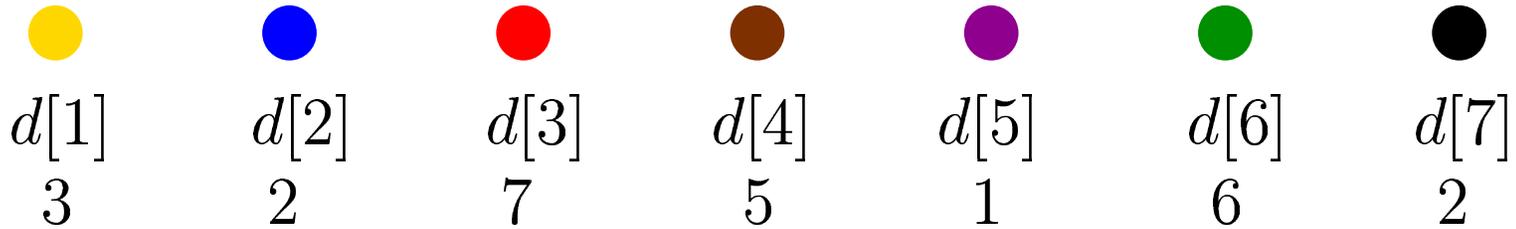
Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



Algoritmo de Graham

Atribui, uma a uma, cada tarefa à máquina menos ocupada.



Escalonamento-Graham

Recebe números inteiros positivos m e t e um vetor $d[1..t]$ e **devolve** um escalonamento de $\{1, \dots, t\}$ em m máquinas.

ESCALONAMENTO-GRAHAM (m, t, d)

1 **para** $j \leftarrow 1$ **até** m **faça**

2 $M[j] \leftarrow \emptyset$

3 $T[j] \leftarrow 0$

4 **para** $i \leftarrow 1$ **até** t **faça**

5 **seja** k **tal que** $T[k]$ **é mínimo**

6 $M[k] \leftarrow M[k] \cup \{i\}$

7 $T[k] \leftarrow T[k] + d[i]$

8 **devolva** $\{M[1], \dots, M[m]\}$

Consumo de tempo

linha consumo de **todas** as execuções da linha

1 $\Theta(m)$

2 $\Theta(m)$

3 $\Theta(m)$

4 $\Theta(t)$

5 $\Theta(mt)$

6 $\Theta(t)$

7 $\Theta(t)$

8 $\Theta(t)$

total $3\Theta(m) + \Theta(mt) + 3\Theta(t) = \Theta(mt)$

Escalonamento-Graham

Se utilizarmos uma **fila com prioridades (min-heap)** para representar as m máquinas onde a prioridade da máquina i é $T[i]$ teremos:

ESCALONAMENTO-GRAHAM (m, t, d)

```
1  para  $j \leftarrow 1$  até  $m$  faça
2       $M[j] \leftarrow \emptyset$ 
3       $T[j] \leftarrow 0$ 
4  BUILD-MIN-HEAP ( $T, m$ )
5  para  $i \leftarrow 1$  até  $t$  faça
6       $k \leftarrow$  HEAP-MIN ( $T, m$ )
7       $M[k] \leftarrow M[k] \cup \{i\}$ 
8      HEAP-INCREASE-KEY ( $T, k, T[k] + d[i]$ )
9  devolva  $\{M[1], \dots, M[m]\}$ 
```

Consumo de tempo

linha consumo de **todas** as execuções da linha

1-4 $\Theta(m)$

5 $\Theta(t)$

6 $\Theta(t)$

7 $\Theta(t)$

8 $O(t \lg m)$

9 $\Theta(t)$

total $\Theta(m) + 4\Theta(t) + O(t \lg m) = O(m + t \lg m)$

O consumo de tempo do algoritmo
ESCALONAMENTO-GRAHAM é $O(m + t \lg m)$.

Delimitações para OPT

OPT = menor tempo de conclusão de um escalonamento

- Duração da tarefa mais longa:

$$\text{OPT} \geq \max\{d[1], d[2], \dots, d[t]\}$$

- Distribuição balanceada:

$$\text{OPT} \geq \frac{d[1] + d[2] + \dots + d[t]}{m}$$

Algoritmos de aproximação

Algoritmo **ALG** é **de aproximação** se existe $\alpha > 0$ tal que

$$\text{valor de } \mathbf{ALG}(I) \leq \alpha \cdot \mathbf{OPT}(I)$$

para toda instância I do problema.

α é o fator de aproximação

Objetivo: α tão perto de 1 quanto possível

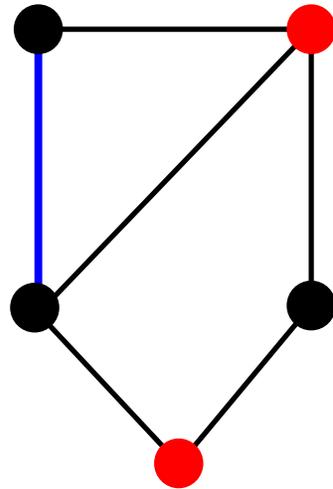
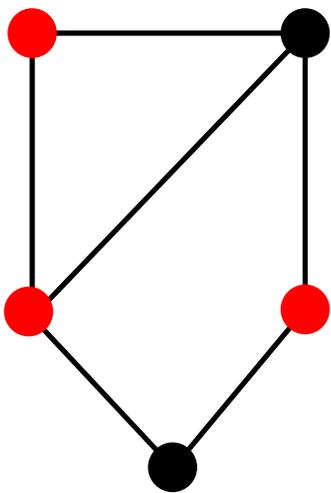
Conclusão

O algoritmo **ESCALONAMENTO-GRAHAM** é uma **2**-aproximação polinomial.

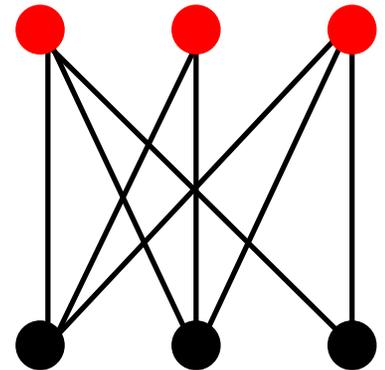
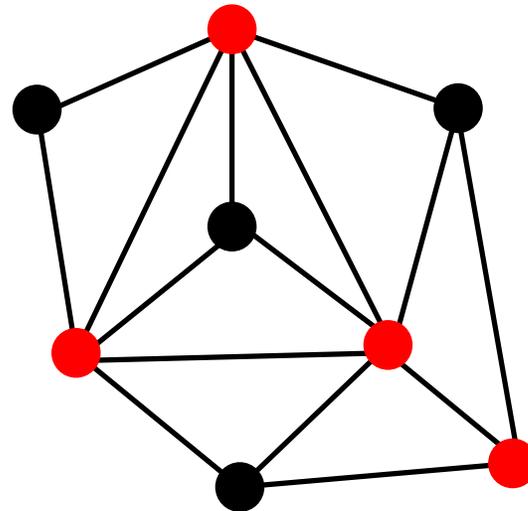
Cobertura por vértices

Um conjunto de vértices C é uma **cobertura** por vértices de G se cada aresta tem pelo menos uma ponta em C

Exemplos:



não é



Cobertura por vértices

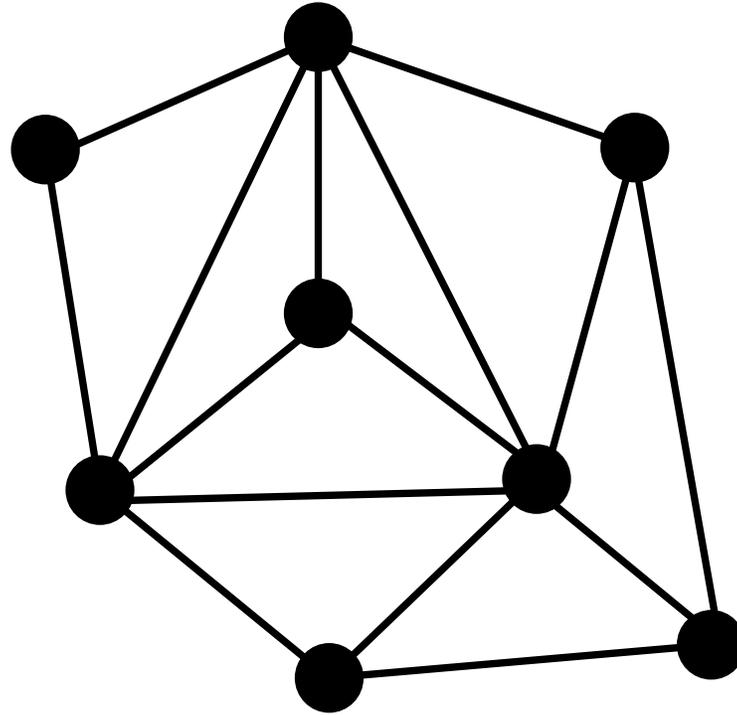
Problema: Dado um grafo G , encontrar um cobertura mínima.

Problema é **NP**-difícil.

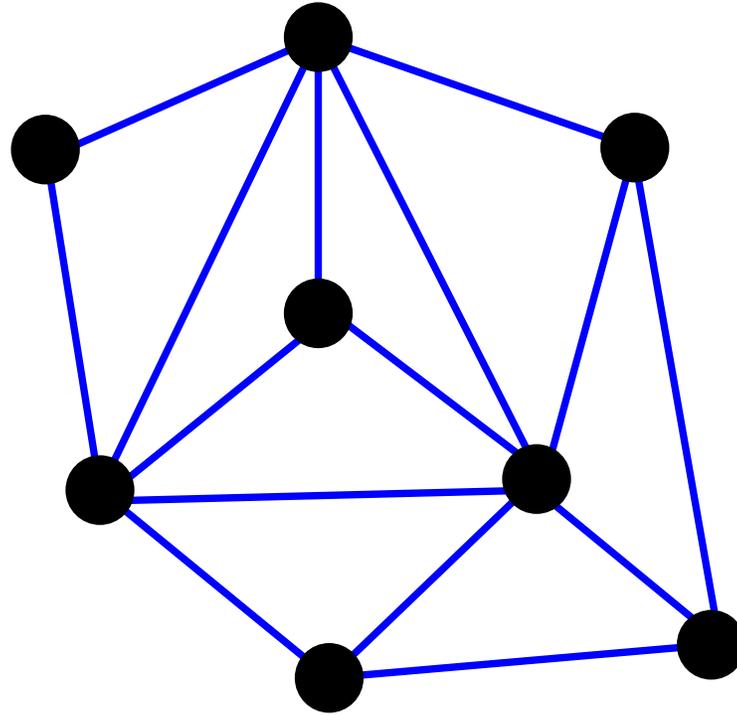
APPROX-VERTEX-COVER (G)

- 1 $C \leftarrow \emptyset$
- 2 $E \leftarrow \{\text{arestas de } G\}$
- 3 **enquanto** $E \neq \emptyset$ **faça**
- 4 seja (u, v) uma aresta qualquer em E
- 5 $C \leftarrow C \cup \{u, v\}$
- 6 $E \leftarrow E - \{\text{arestas incidentes a } u \text{ ou } v\}$
- 7 **devolva** C

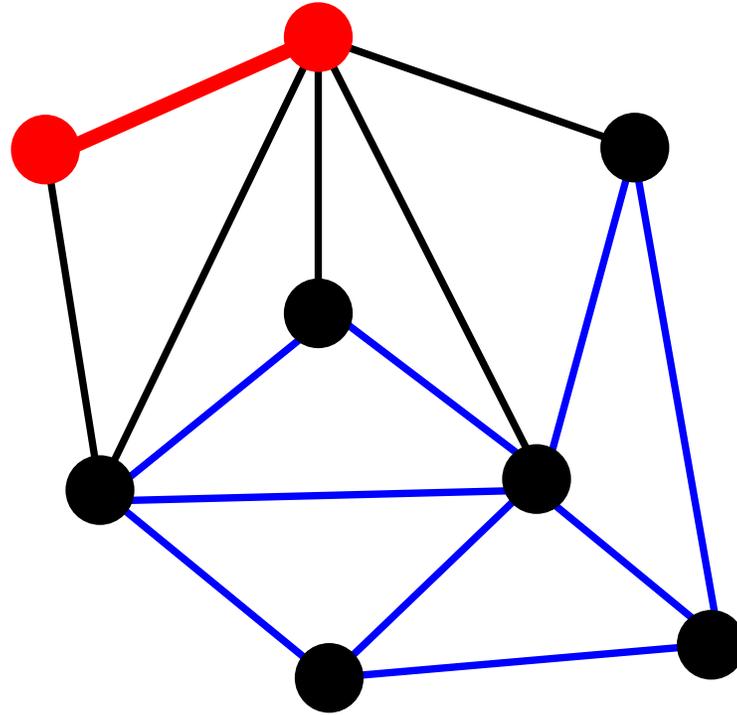
Exemplo



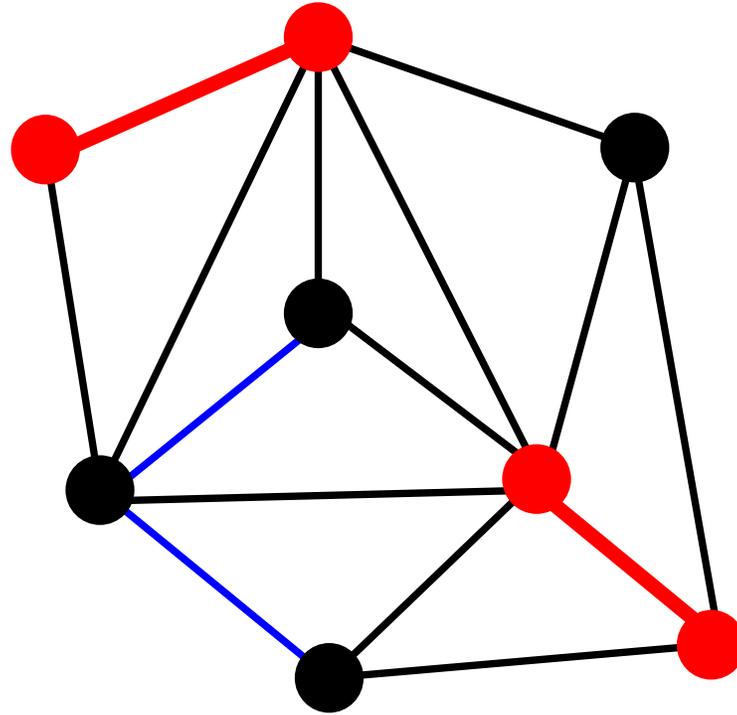
Exemplo



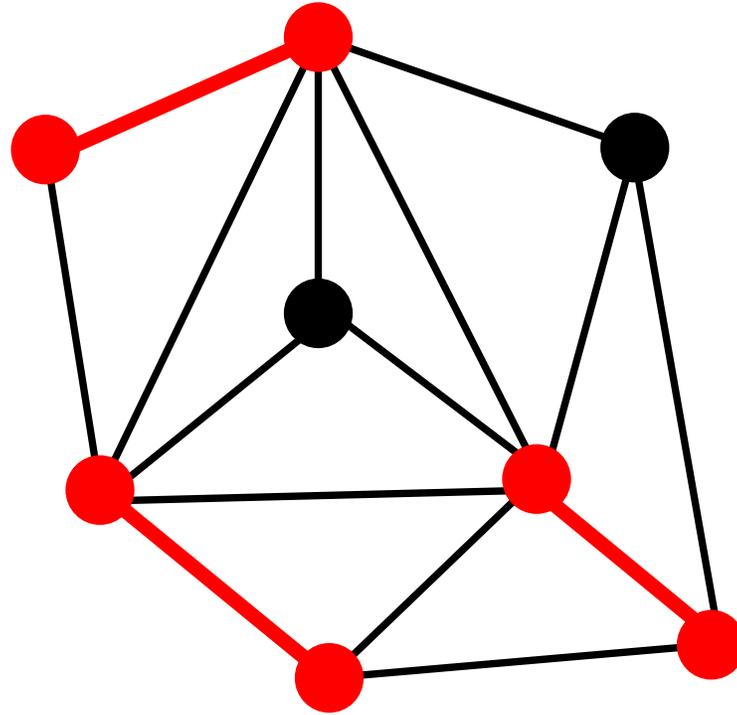
Exemplo



Exemplo



Exemplo



Consumo de tempo

E representado através de listas de adjacência.

n = número de vértices

m = número de arestas

linha consumo de **todas** as execuções da linha

1 $\Theta(1)$

2 $\Theta(n + m)$

3 $O(m)$

4 $O(m)$

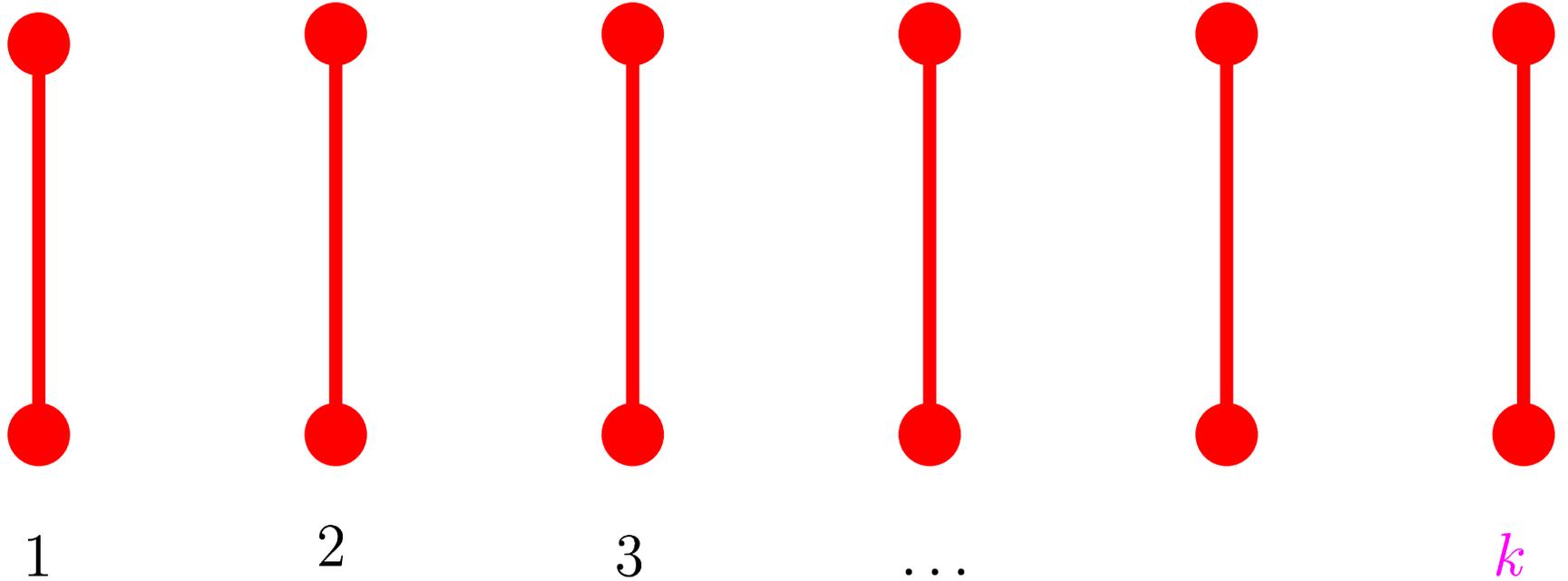
5 $O(n)$

6 $O(m)$

7 $O(n)$

total $\Theta(1) + \Theta(n + m) + 3O(m) + 2O(n) = \Theta(n + m)$

Delimitações para OPT

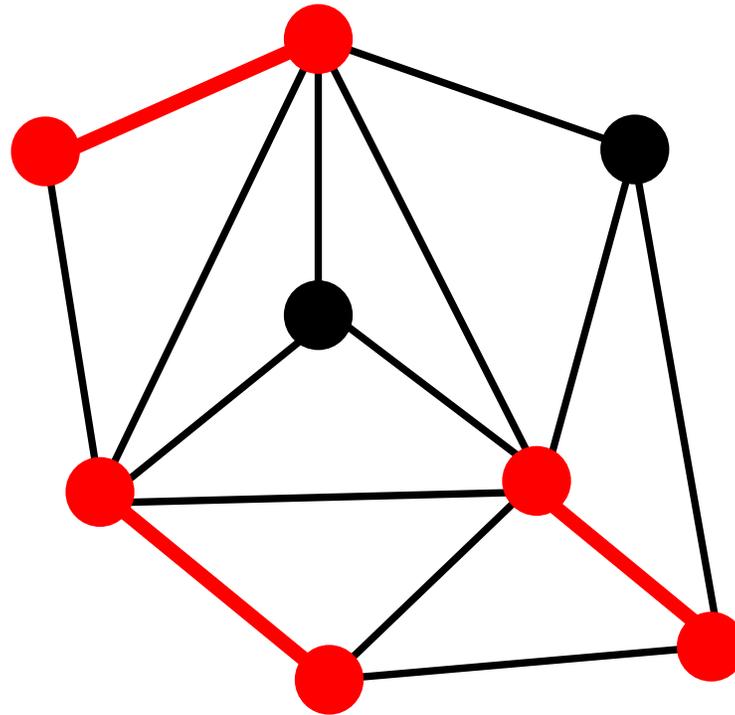


Se G possui um emparelhamento com k arestas, então

$$\text{OPT} \geq k.$$

OPT = número **mínimo** de vértices de uma cobertura por vértices

Fator de aproximação



C = cobertura devolvida pelo algoritmo.

G possui um emparelamento com $|C|/2$ arestas.

Logo,

$$\frac{|C|}{2} \leq \text{OPT} \quad \Rightarrow \quad |C| \leq 2 \text{OPT}.$$

Conclusão

O algoritmo **APPROX-VERTEX-COVER** é uma **2**-aproximação polinomial.