Alguns comentários

- Segunda prova
- Programação dinâmica em grafos
- Guloso em grafos

Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar coleção máxima de intervalos disjuntos dois a dois.

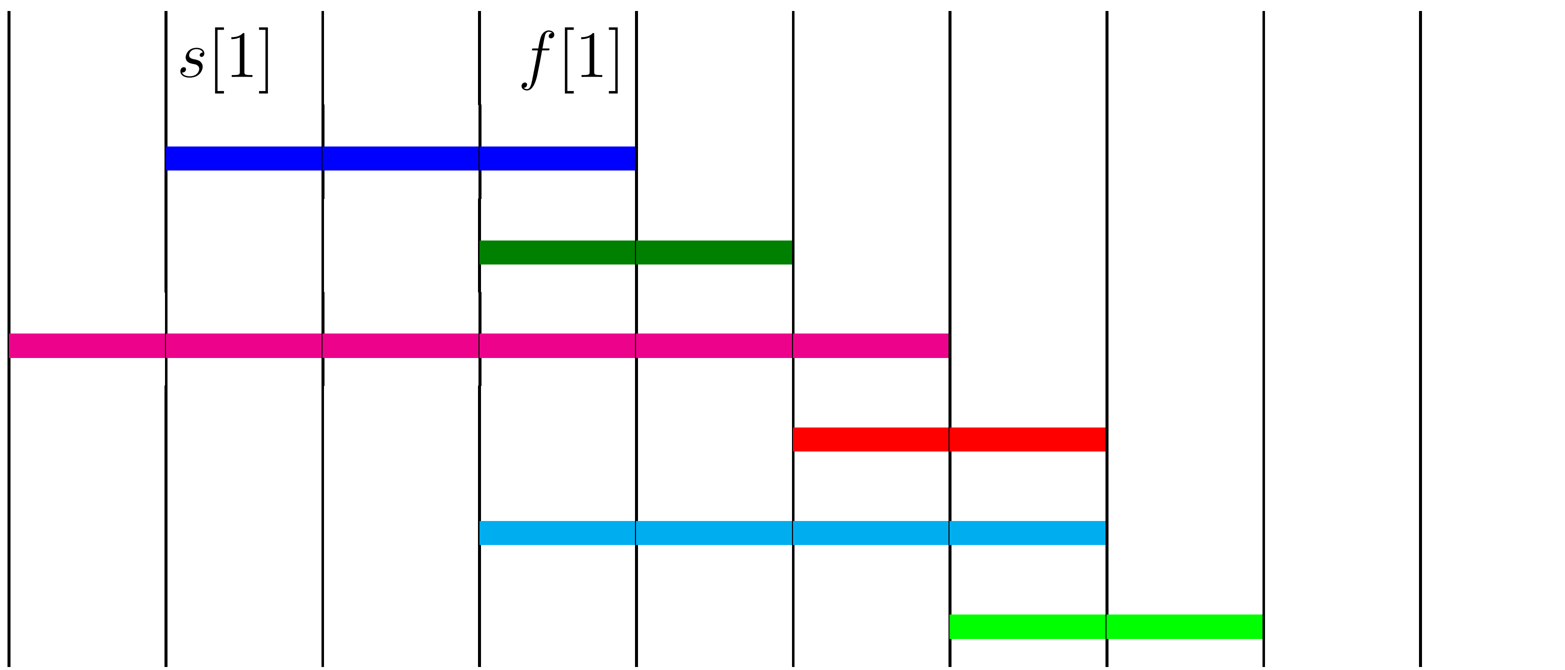
Solução é um subconjunto A de $\{1, \ldots, n\}$.

Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar coleção máxima de intervalos disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \ldots, n\}$.

Exemplo:

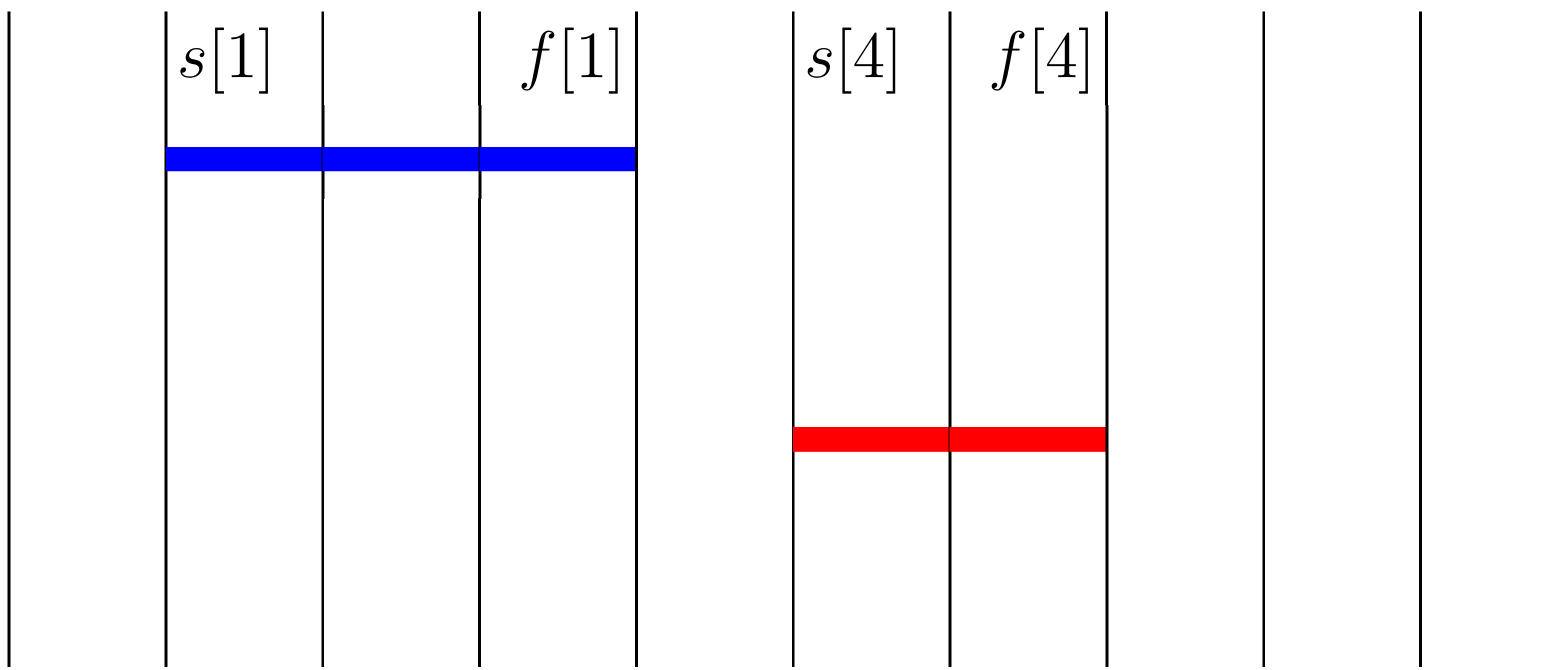


Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar coleção máxima de intervalos disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \ldots, n\}$.

Solução:

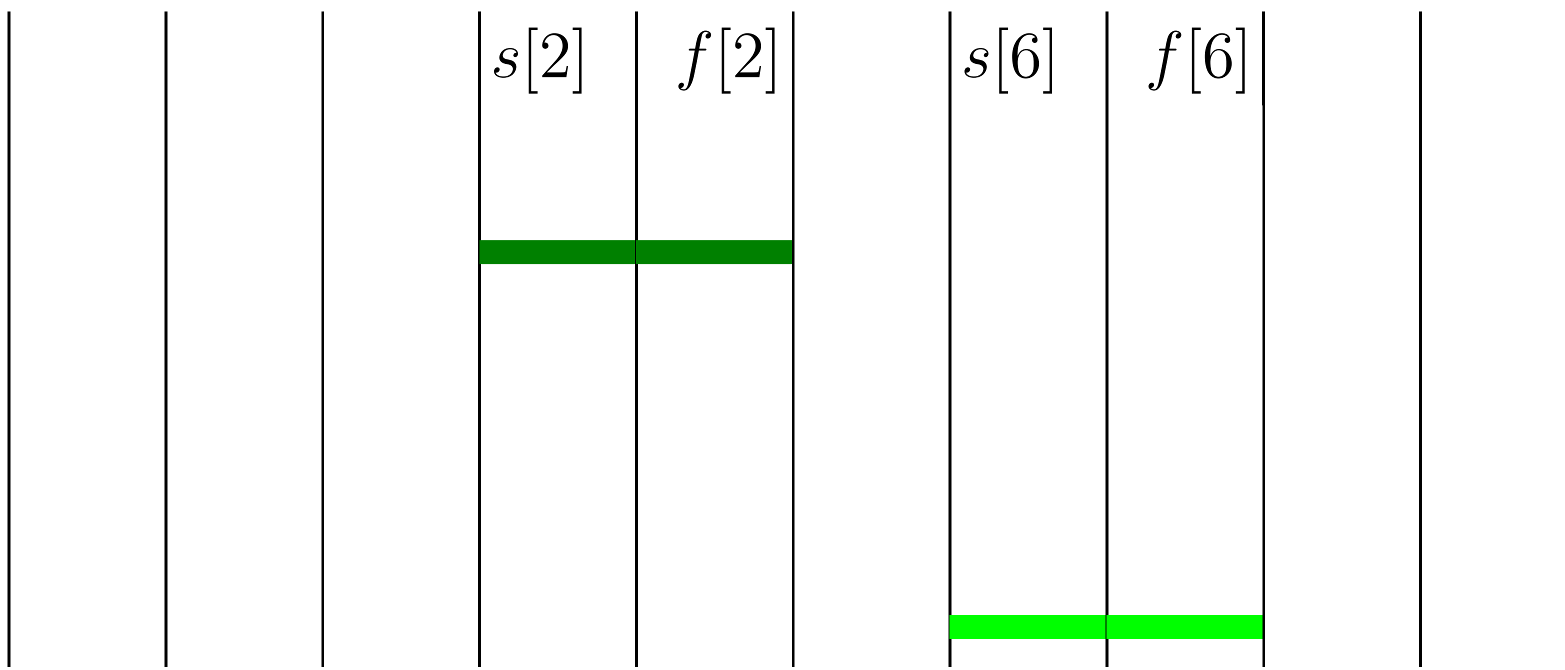


Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar coleção máxima de intervalos disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \ldots, n\}$.

Solução:



Escolha gulosa

Intervalos $S := \{1, \ldots, n\}$

Se $f[i]$ é mínimo em S,

então EXISTE uma solução ótima A tal que $i \in A$.

Demonstração dessa propriedade na aula.

Algoritmo guloso

Devolve coleção máxima de intervalos disjuntos dois a dois.

INTERVALOS-DISJUNTOS (s, f, n)

ordene s e f de tal forma que $f[1] \leq f[2] \leq \cdots \leq f[n]$
$A \leftarrow \{1\}$
$i \leftarrow 1$
para $j \leftarrow 2$ **até** n **faça**
    se $s[j] \geq f[i]$
        então $A \leftarrow A \cup \{j\}$
                $i \leftarrow j$
devolva A

Algoritmo guloso

Devolve coleção máxima de intervalos disjuntos dois a dois.

INTERVALOS-DISJUNTOS (s, f, n)

```
0   ordene s e f de tal forma que f[1] ≤ f[2] ≤ ··· ≤ f[n]
1   A ← {1}
2   i ← 1
3   para j ← 2 até n faça
4       se s[j] ≥ f[i]
5           então A ← A ∪ {j}
6                 i ← j
7   devolva A
```

Na linha 3 vale que

(i0) A é coleção máxima de intervalos disjuntos de $(s, f, j-1)$

Algoritmo guloso

Devolve coleção máxima de intervalos disjuntos dois a dois.

INTERVALOS-DISJUNTOS (s, f, n)
0 ordene s e f de tal forma que $f[1] \leq f[2] \leq \cdots \leq f[n]$
1 $A \leftarrow \{1\}$
2 $i \leftarrow 1$
3 **para** $j \leftarrow 2$ **até** n **faça**
4   **se** $s[j] \geq f[i]$
5   **então** $A \leftarrow A \cup \{j\}$
6                 $i \leftarrow j$
7 **devolva** A

Consumo de tempo da linha 0 é $\Theta(n \lg n)$.

Consumo de tempo das linhas 1–7 é $\Theta(n)$.

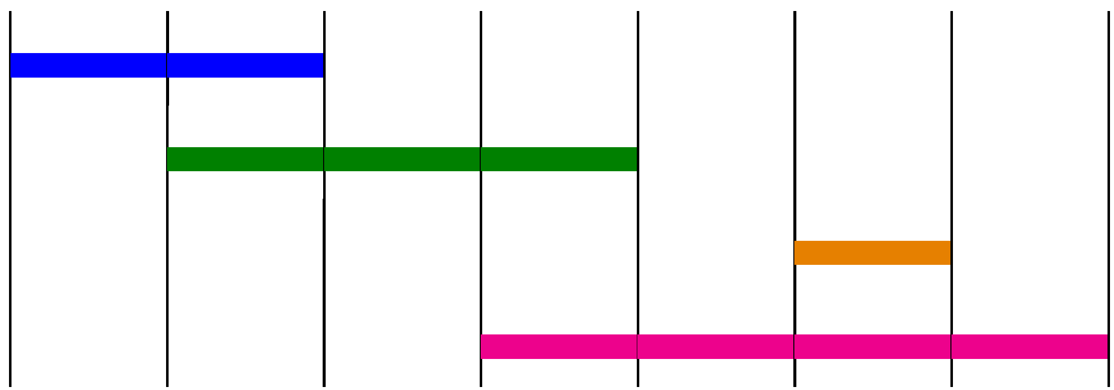
Coloração de intervalos

Problema: Dados intervalos de tempo $[s_1, f_2), \ldots, [s_n, f_n)$, encontrar uma coloração dos intervalos com o menor número possível de cores em que dois intervalos de mesma cor sempre sejam disjuntos.

Coloração de intervalos

Problema: Dados intervalos de tempo $[s_1, f_2), \ldots, [s_n, f_n)$, encontrar uma coloração dos intervalos com o menor número possível de cores em que dois intervalos de mesma cor sempre sejam disjuntos.

Solução: partição de $\{1, \ldots, n\}$ em coleções de intervalos dois a dois disjuntos.

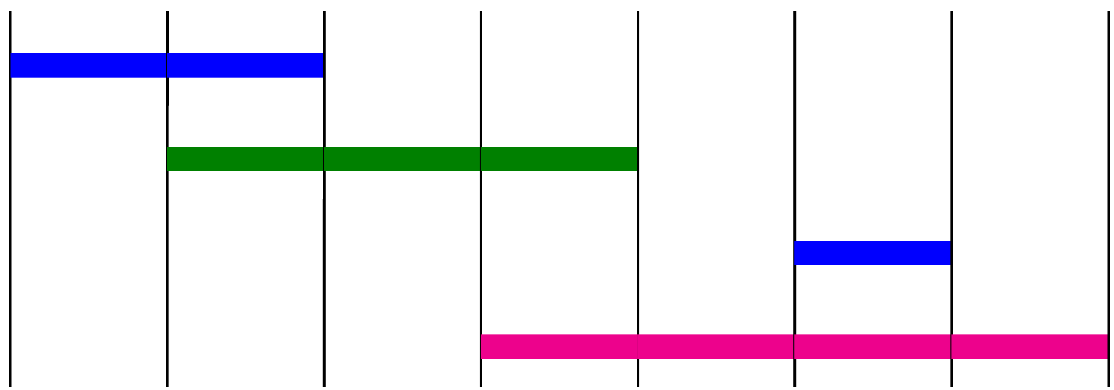


4 cores

Coloração de intervalos

Problema: Dados intervalos de tempo $[s_1, f_2), \ldots, [s_n, f_n)$, encontrar uma coloração dos intervalos com o menor número possível de cores em que dois intervalos de mesma cor sempre sejam disjuntos.

Solução: partição de $\{1, \ldots, n\}$ em coleções de intervalos dois a dois disjuntos.

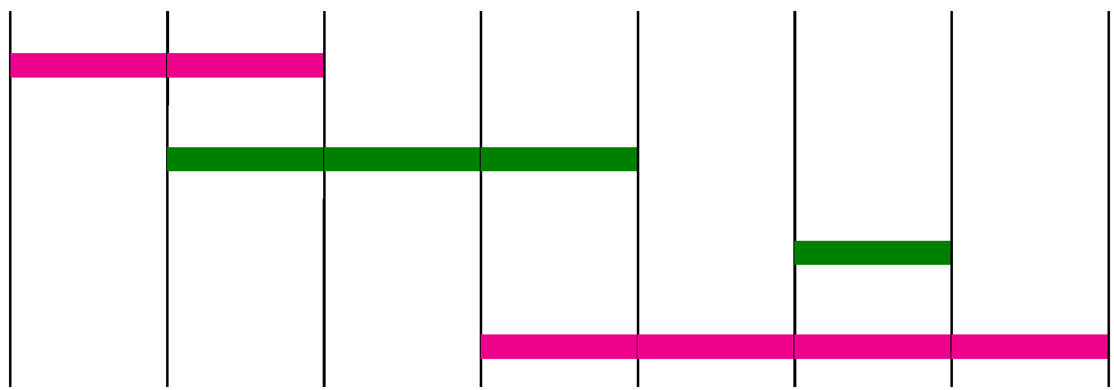


3 cores

Coloração de intervalos

Problema: Dados intervalos de tempo $[s_1, f_2), \ldots, [s_n, f_n)$, encontrar uma coloração dos intervalos com o menor número possível de cores em que dois intervalos de mesma cor sempre sejam disjuntos.

Solução: partição de $\{1, \ldots, n\}$ em coleções de intervalos dois a dois disjuntos.



2 cores!

Motivação

Queremos distribuir um conjunto de atividades no menor número possivel de salas.

Motivação

Queremos distribuir um conjunto de atividades no menor número possivel de salas.

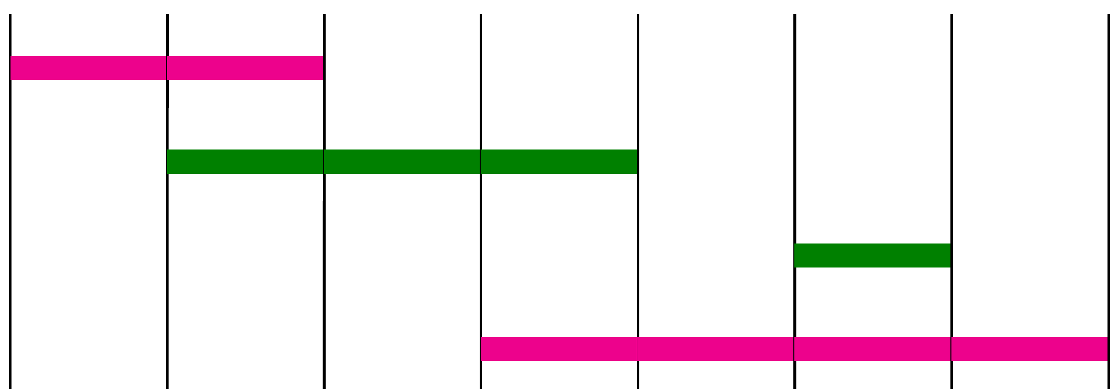
Cada atividade a_i ocupa certo intervalo de tempo $[s_i, f_i)$ e duas atividades podem ficar na mesma sala somente se os correspondentes intervalos são disjuntos.

Motivação

Queremos distribuir um conjunto de atividades no menor número possivel de salas.

Cada atividade a_i ocupa certo intervalo de tempo $[s_i, f_i)$ e duas atividades podem ficar na mesma sala somente se os correspondentes intervalos são disjuntos.

Cada sala corresponde a uma cor. Queremos usar o menor número possível de cores para pintar todos os intervalos.



Coloração de intervalos

Estratégias gulosas:

- Encontre uma coleção disjunta máxima de intervalos, pinte com a próxima cor disponível e repita a idéia para os intervalos restantes.
- Ordene as atividades de maneira que $f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.
- Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Coloração de intervalos

Estratégias gulosas:

- Encontre uma coleção disjunta máxima de intervalos, pinte com a próxima cor disponível e repita a idéia para os intervalos restantes.
- Ordene as atividades de maneira que $f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.
- Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Coloração de intervalos

Estratégias gulosas:

- Encontre uma coleção disjunta máxima de intervalos, pinte com a próxima cor disponível e repita a idéia para os intervalos restantes.
- Ordene as atividades de maneira que $f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.
- Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Coloração de intervalos

Estratégias gulosas:

- Encontre uma coleção disjunta máxima de intervalos, pinte com a próxima cor disponível e repita a idéia para os intervalos restantes.
- Ordene as atividades de maneira que $f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.
- Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Quais destas estratégias funcionam?
Quais não funcionam?

Estratégia 1

Encontre uma coleção disjunta máxima de intervalos,
pinte com a próxima cor disponível
e repita a idéia para os intervalos restantes.

Estratégia 1

Encontre uma coleção disjunta máxima de intervalos,
pinte com a próxima cor disponível
e repita a idéia para os intervalos restantes.

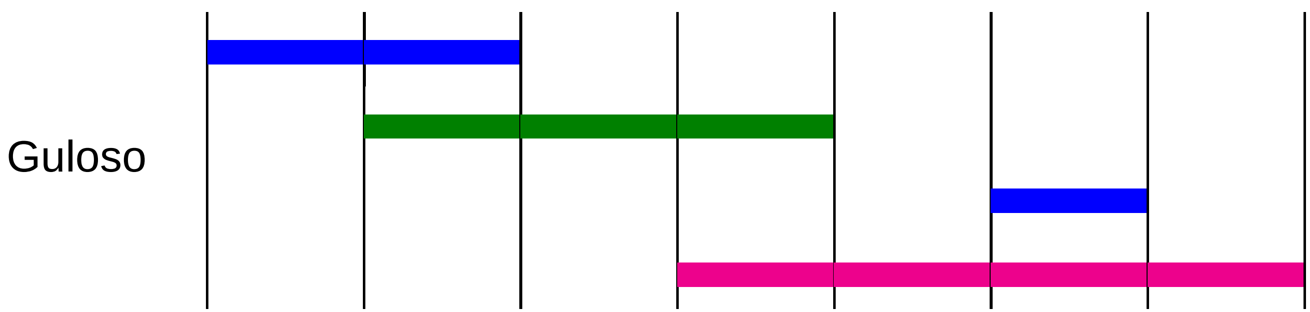
Não funciona...

Estratégia 1

Encontre uma coleção disjunta máxima de intervalos,
pinte com a próxima cor disponível
e repita a idéia para os intervalos restantes.

Não funciona...

Exemplo:

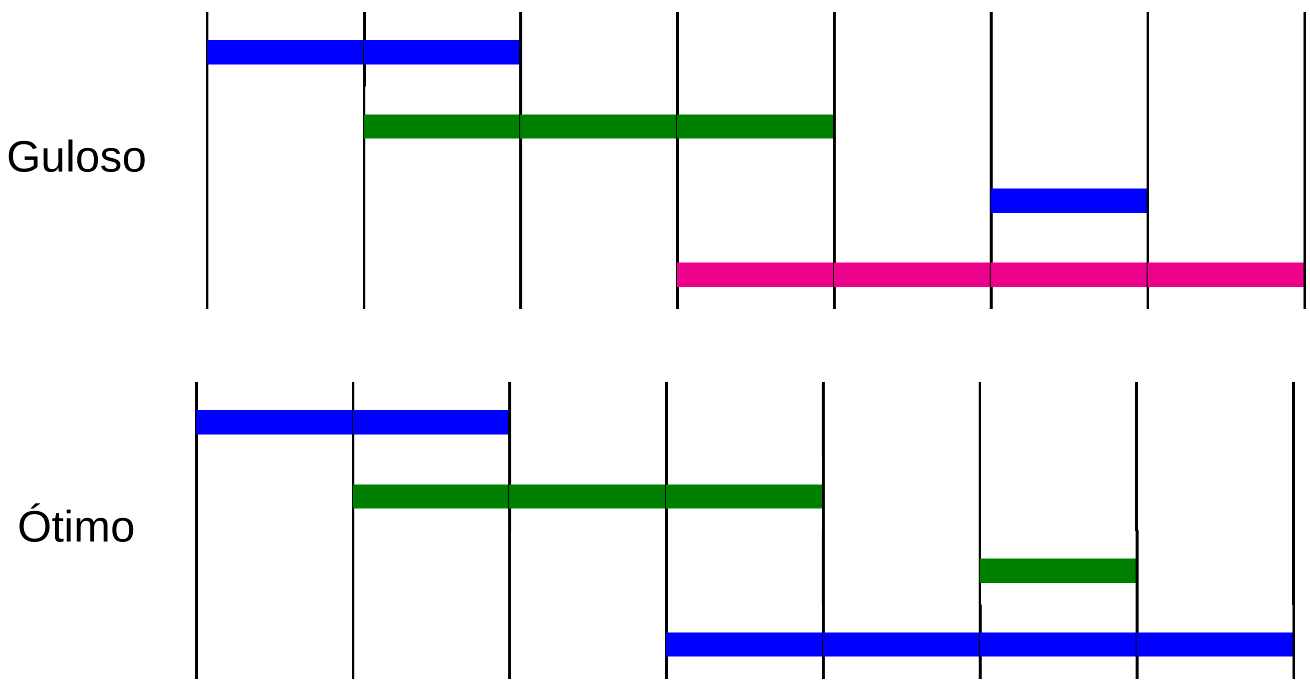


Estratégia 1

Encontre uma coleção disjunta máxima de intervalos,
pinte com a próxima cor disponível
e repita a idéia para os intervalos restantes.

Não funciona...

Exemplo:



Estratégia 2

Ordene as atividades de maneira que $f[1] \le f[2] \le \cdots \le f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Estratégia 2

Ordene as atividades de maneira que
$f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

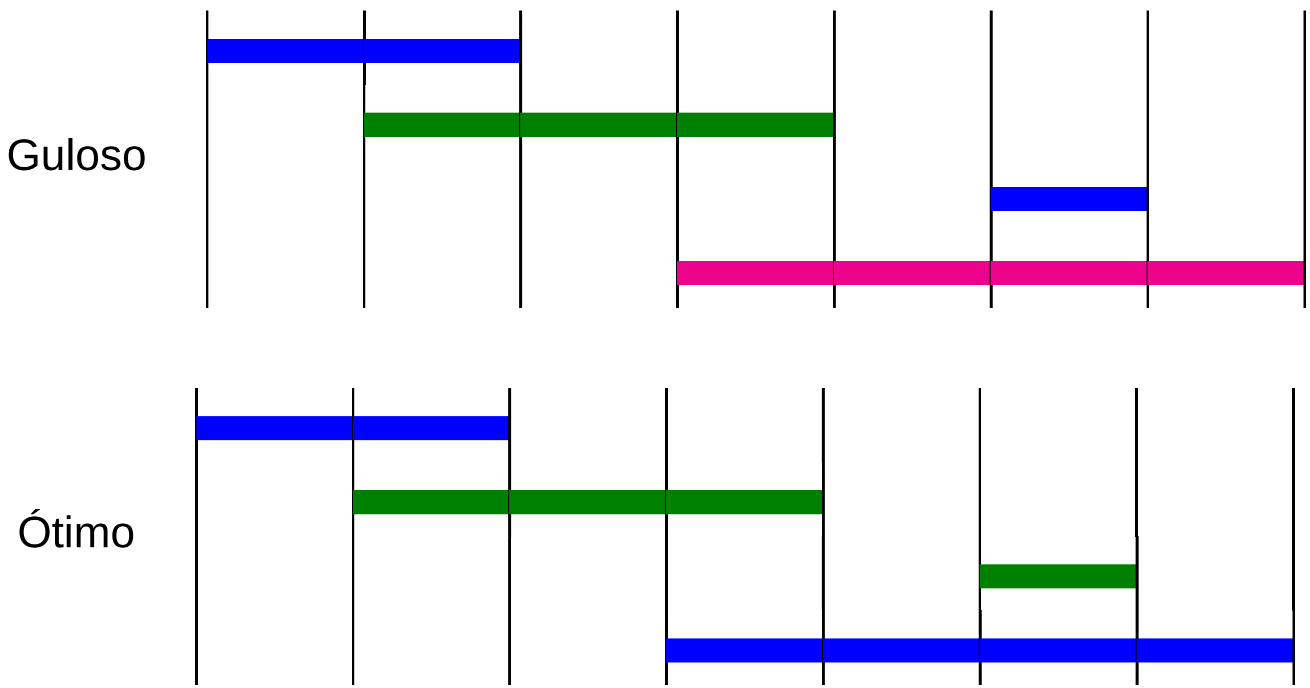
Não funciona de novo...

Estratégia 2

Ordene as atividades de maneira que
$f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Não funciona de novo...

Mesmo exemplo:



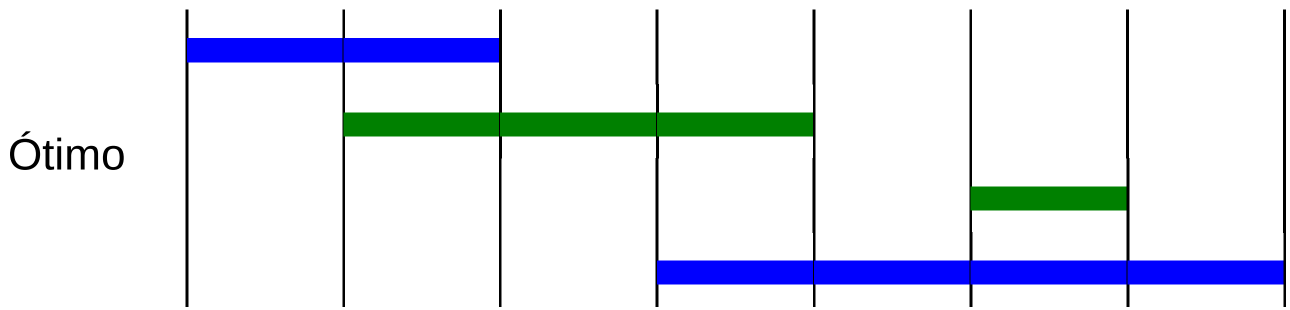
Estratégia 3

Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Estratégia 3

Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

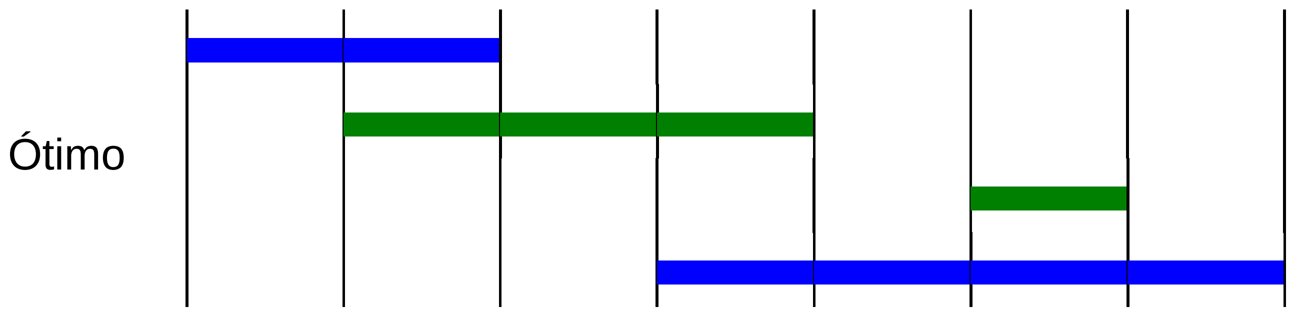
Pelo menos funciona para o exemplo...



Estratégia 3

Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Pelo menos funciona para o exemplo...



De fato, funciona sempre!

A seguir, apresentamos o algoritmo guloso obtido.

No algoritmo, para uma cor j, o número $\ell[j]$ indica o final da última tarefa que foi colorida com a cor j.

Algoritmo guloso

Devolve coloração dos intervalos com menor número de cores.

COLORAÇÃO-INTERVALOS (s, f, n)

ordene s e f de tal forma que $s[1] \leq s[2] \leq \cdots \leq s[n]$
$k \leftarrow 0$
para $i \leftarrow 1$ **até** n **faça**
    $j \leftarrow 1$
    enquanto $j \leq k$ **e** $\ell[j] > s[i]$ **faça** $j \leftarrow j + 1$
    se $j > k$
        então $k \leftarrow k + 1$
                $D[k] \leftarrow \{i\}$
                $\ell[k] \leftarrow f[i]$
        senão $D[j] \leftarrow D[j] \cup \{i\}$
                $\ell[j] \leftarrow f[i]$
devolva $D[1], \ldots, D[k]$

Consumo de tempo

Observe que o algoritmo consome tempo $\mathrm{O}(n^2)$.

Como observado na aula, é possível obter um certificado de que o algoritmo apresentado utiliza o menor número de cores.

Exercício: Modifique o algoritmo para que ele devolva um certificado de que sua coloração usa um número mínimo de cores.

Demonstração

Seja $B[1], \ldots, B[k^*]$ uma solução ótima que coincide com a solução gulosa $D[1], \ldots, D[k]$ quando restrita aos intervalos em $\{1, \ldots, i-1\}$, com i o maior possível.

Se $i = n+1$, então não há nada a provar.
Senão, seja j tal que $i \in D[j]$ e j^* tal que $i \in B[j^*]$.

Seja X o conjunto $B[j] \cap \{i, \ldots, n\}$ e
Y o conjunto $B[j^*] \cap \{i, \ldots, n\}$.
Seja B' a partição de $\{1, \ldots, n\}$ tal que

$$\begin{array}{rcl} B'[t] & = & B[t] \quad \text{se } t \neq j \text{ e } t \neq j^* \\ B'[j] & = & B[j] \setminus X \cup Y \\ B'[j^*] & = & B[j^*] \setminus Y \cup X \end{array}$$

B' é uma solução ótima que contraria a escolha de B.

Problema de escalonamento

Considere n tarefas indicadas pelos números $1, \ldots, n$

Problema de escalonamento

Considere n tarefas indicadas pelos números $1, \ldots, n$

t_i: duração da tarefa i

d_i: prazo de entrega da tarefa i

Problema de escalonamento

Considere n tarefas indicadas pelos números $1, \ldots, n$

t_i: duração da tarefa i

d_i: prazo de entrega da tarefa i

Escalonamento: permutação de 1 a n

Para um escalonamento π, o tempo de início da tarefa i é

$$s_i = \sum_{j=1}^{\pi(i)-1} t_{\pi^{-1}(j)}$$

(soma da duração das tarefas anteriores a i).

Problema de escalonamento

Considere n tarefas indicadas pelos números $1, \ldots, n$

t_i: duração da tarefa i

d_i: prazo de entrega da tarefa i

Escalonamento: permutação de 1 a n

Para um escalonamento π, o tempo de início da tarefa i é

$$s_i = \sum_{j=1}^{\pi(i)-1} t_{\pi^{-1}(j)}$$

(soma da duração das tarefas anteriores a i).

O tempo de término da tarefa i é $f_i = s_i + t_i$.

Problema de escalonamento

Para um escalonamento π, o tempo de início da tarefa i é soma da duração das tarefas anteriores a i.

O tempo de término da tarefa i é $f_i = s_i + t_i$.

Problema de escalonamento

Para um escalonamento π, o tempo de início da tarefa i é soma da duração das tarefas anteriores a i.

O tempo de término da tarefa i é $f_i = s_i + t_i$.

O atraso da tarefa i é o número

$$\ell_i = \begin{cases} 0 & \text{se } f_i \leq d_i \\ f_i - d_i & \text{se } f_i > d_i. \end{cases}$$

Problema de escalonamento

Para um escalonamento π, o tempo de início da tarefa i é soma da duração das tarefas anteriores a i.

O tempo de término da tarefa i é $f_i = s_i + t_i$.

O atraso da tarefa i é o número

$$\ell_i = \begin{cases} 0 & \text{se } f_i \leq d_i \\ f_i - d_i & \text{se } f_i > d_i. \end{cases}$$

Problema: Dados $t_1, \ldots, t_n$ e $d_1, \ldots, d_n$, encontrar um escalonamento com o menor atraso máximo.

Ou seja, que minimize $L = \max_i \ell_i$.

Estratégias gulosas

- escalonar primeiro as tarefas mais curtas
  (ignoro o prazo de entrega)

Estratégias gulosas

- escalonar primeiro as tarefas mais curtas
  (ignoro o prazo de entrega)

  Não funciona...

  Exemplo: $t_1 = 1$, $d_1 = 10$, $t_2 = 8$, $d_2 = 8$

  Guloso: 1, 2, $L = 1$ Solução: 2, 1, $L = 0$

Estratégias gulosas

- escalonar primeiro as tarefas mais curtas
  (ignoro o prazo de entrega)

  Não funciona...

  Exemplo: $t_1 = 1$, $d_1 = 10$, $t_2 = 8$, $d_2 = 8$

  Guloso: 1, 2, $L = 1$ Solução: 2, 1, $L = 0$

- escalonar primeiro as tarefas com menos $d_i - t_i$
  (tarefas com menor folga)

Estratégias gulosas

- escalonar primeiro as tarefas mais curtas
  (ignoro o prazo de entrega)

  Não funciona...

  Exemplo: $t_1 = 1$, $d_1 = 10$, $t_2 = 8$, $d_2 = 8$

  Guloso: $1, 2, L = 1$ Solução: $2, 1, L = 0$

- escalonar primeiro as tarefas com menos $d_i - t_i$
  (tarefas com menor folga)

  Não funciona...

  Exemplo: $t_1 = 1$, $d_1 = 2$, $t_2 = 10$, $d_2 = 10$

  Guloso: $2, 1, L = 9$ Solução: $1, 2, L = 1$

Estratégias gulosas

- escalonar primeiro as tarefas mais curtas
(ignoro o prazo de entrega)

  Não funciona...

  Exemplo: $t_1 = 1$, $d_1 = 10$, $t_2 = 8$, $d_2 = 8$

  Guloso: $1, 2$, $L = 1$ Solução: $2, 1$, $L = 0$

- escalonar primeiro as tarefas com menos $d_i - t_i$
(tarefas com menor folga)

  Não funciona...

  Exemplo: $t_1 = 1$, $d_1 = 2$, $t_2 = 10$, $d_2 = 10$

  Guloso: $2, 1$, $L = 9$ Solução: $1, 2$, $L = 1$

- escalonar primeiro as tarefas com menor prazo
(ignoro a duração)

Algoritmo guloso

Devolve escalonamento com atraso máximo mínimo.

ESCALONAMETO (t, d, n)

1 seja π a permutação de 1 a n tal que

$$d[\pi[1]] \leq d[\pi[2]] \leq \cdots \leq d[\pi[n]]$$

2 **devolva** π

Algoritmo guloso

Devolve escalonamento com atraso máximo mínimo.

ESCALONAMETO (t, d, n)

1 seja π a permutação de 1 a n tal que

$$d[\pi[1]] \leq d[\pi[2]] \leq \cdots \leq d[\pi[n]]$$

2 **devolva** π

Consumo de tempo: $\mathrm{O}(n \lg n)$.

Algoritmo guloso

Devolve escalonamento com atraso máximo mínimo.

ESCALONAMETO (t, d, n)
1 seja π a permutação de 1 a n tal que

$$d[\pi[1]] \le d[\pi[2]] \le \cdots \le d[\pi[n]]$$

2 **devolva** π

Consumo de tempo: $\mathrm{O}(n \lg n)$.

Na aula, vimos a prova de que este algoritmo está correto (devolve um escalonamento com atraso máximo mínimo).

Ingredientes da prova

Uma inversão em um escalonamento π é um par (i, j) tal que $i < j$ e $d[\pi[i]] > d[\pi[j]]$.

Ingredientes da prova

Uma inversão em um escalonamento π é um par (i, j) tal que $i < j$ e $d[\pi[i]] > d[\pi[j]]$.

Mostre que dois escalonamentos que não têm inversões têm o mesmo atraso máximo.

Ingredientes da prova

Uma inversão em um escalonamento π é um par (i, j) tal que $i < j$ e $d[\pi[i]] > d[\pi[j]]$.

Mostre que dois escalonamentos que não têm inversões têm o mesmo atraso máximo.

Mostre que se um escalonamento ótimo tem uma inversão, então ele tem uma inversão do tipo $(i, i+1)$.

Ingredientes da prova

Uma inversão em um escalonamento π é um par (i, j) tal que $i < j$ e $d[\pi[i]] > d[\pi[j]]$.

Mostre que dois escalonamentos que não têm inversões têm o mesmo atraso máximo.

Mostre que se um escalonamento ótimo tem uma inversão, então ele tem uma inversão do tipo $(i, i+1)$.

Mostre então que, se trocarmos $\pi[i]$ e $\pi[i+1]$, obteremos um escalomento com atraso máximo não superior ao atraso máximo de π.

Exercícios

Exercício 22.A [CLRS 16.1-1]
Escreva um algoritmo de programação dinâmica para resolver o problema dos intervalos disjuntos. (Versão simplificadda do exercício: basta determinar o *tamanho* de uma coleção disjunta máxima.) Qual o consumo de tempo do seu algoritmo?

Exercício 22.B
Prove que o algoritmo guloso para o problema dos intervalos disjuntos está correto. (Ou seja, prove a propriedade da subestrutura ótima e a propriedade da escolha gulosa.)

Exercício 22.C [CLRS 16.1-2]
Mostre que a seguinte idéia também produz um algoritmo guloso correto para o problema da coleção disjunta máxima de intervalos: dentre os intervalos disjuntos dos já escolhidos, escolha um que tenha instante de início máximo. (Em outras palavras, suponha que os intervalos estão em ordem decrescente de início.)

Exercício 22.D [CLRS 16.1-4]
Nem todo algoritmo guloso resolve o problema da coleção disjunta máxima de intervalos. Mostre que nenhuma das três idéias a seguir resolve o problema. Idéia 1: Escolha o intervalo de menor duração dentre os que são disjuntos dos intervalos já escolhidos. Idéia 2: Escolha um intervalo seja disjunto dos já escolhidos e intercepte o menor número possível de intervalos ainda não escolhidos. Idéia 3: Escolha o intervalo disjunto dos já selecionados que tenha o menor instante de início.

Mais exercícios

Exercício 22.F [Pares de livros]
Suponha dado um conjunto de livros numerados de 1 a n. Suponha que o livro i tem peso $p[i]$ e que $0 < p[i] < 1$ para cada i. Problema: acondicionar os livros no menor número possível de envelopes de modo que cada envelope tenha no máximo 2 livros e o peso do conteúdo de cada envelope seja no máximo 1. Escreva um algoritmo guloso que calcule o número mínimo de envelopes. O consumo de tempo do seu algoritmo deve ser $\mathrm{O}(n \lg n)$. Mostre que seu algoritmo está correto (ou seja, prove a "greedy-choice property" e a "optimal substructure" apropriadas). Estime o consumo de tempo do seu algoritmo.

Exercício 22.G [Bin-packing]
São dados objetos $1, \ldots, n$ e um número ilimitado de "latas". Cada objeto i tem "peso" w_i e cada lata tem "capacidade" 1: a soma dos pesos dos objetos colocados em uma lata não pode passar de 1. Problema: Distribuir os objetos pelo menor número possível de latas. Programe e teste as seguinte heurísticas. Heurística 1: examine os objetos na ordem dada; tente colocar cada objeto em uma lata já parcialmente ocupada que tiver mais "espaço" livre sobrando; se isso for impossível, pegue uma nova lata. Heurística 2: rearranje os objetos em ordem decrescente de peso; em seguida, aplique a heurística 1. Essas herísticas resolvem o problema? Compare com o exercício 22.F.
Para testar seu programa, sugiro escrever uma rotina que receba $n \leq 100000$ e $u \leq 1$ e gere $w_1, \ldots, w_n$ aleatoriamente, todos no intervalo $(0, u)$.

Mais exercícios ainda

Exercício 22.H [parte de CLRS 16-4, modificado]
Seja $1, \ldots, n$ um conjunto de *tarefas*. Cada tarefa consome um dia de trabalho; durante um dia de trabalho somente uma das tarefas pode ser executada. Os dias de trabalho são numerados de 1 a n. A cada tarefa t está associado um *prazo* p_t: a tarefa deveria ser executada em algum dia do intervalo $1 \,..\, p_t$. A cada tarefa t está associada uma *multa* não-negativa m_t. Se uma dada tarefa t é executada depois do prazo p_t, sou obrigado a pagar a multa m_t (mas a multa não depende do número de dias de atraso). Problema: Programar as tarefas (ou seja, estabelecer uma bijeção entre as tarefas e os dias de trabalho) de modo a minimizar a multa total. Escreva um algoritmo guloso para resolver o problema. Prove que seu algoritmo está correto (ou seja, prove a "greedy-choice property" e a "optimal substructure" apropriadas). Analise o consumo de tempo.