Análise de Algoritmos

Parte destes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Algoritmos gulosos (*greedy*)

CLRS 16.2

Problema fracionário da mochila

Problema: Dados (w, v, n, W), encontrar uma mochila ótima.

Problema fracionário da mochila

Problema: Dados (w, v, n, W), encontrar uma mochila ótima.

Exemplo: $W = 50$, $n = 4$

	1	2	3	4	
w	40	30	20	10	
v	840	600	400	100	
x	1	0	0	0	valor $= 840$
x	1	0	0	1	valor $= 940$
x	0	1	1	0	valor $= 1000$
x	1	$1/3$	0	0	valor $= 1040$

A propósito . . .

O problema fracionário da mochila é um problema de programação linear (PL): encontrar um vetor x que

$$\begin{array}{rlll} \text{maximize} & x \cdot v & & \\ \text{sob as restrições} & x \cdot w & \leq W & \\ & x[i] & \geq 0 & \text{para } i = 1, \ldots, n \\ & x[i] & \leq 1 & \text{para } i = 1, \ldots, n \end{array}$$

A propósito . . .

O problema fracionário da mochila é um problema de programação linear (PL): encontrar um vetor x que

$$\begin{array}{rll} \text{maximize} & x \cdot v & \\ \text{sob as restrições} & x \cdot w & \leq W \\ & x[i] & \geq 0 \quad \text{para } i = 1, \ldots, n \\ & x[i] & \leq 1 \quad \text{para } i = 1, \ldots, n \end{array}$$

PL's podem ser resolvidos por

SIMPLEX: no pior caso consome tempo exponencial
na prática é muito rápido

ELIPSÓIDES: consome tempo polinomial
na prática é lento

PONTOS-INTERIORES: consome tempo polinomial
na prática é rápido

Subestrutura ótima

Suponha que $x[1\,..\,n]$ é mochila ótima para o problema (w, v, n, W).

Subestrutura ótima

Suponha que $x[1\,..\,n]$ é mochila ótima para o problema (w, v, n, W).

Se $x[n] = \delta$

então $x[1\,..\,n-1]$ é mochila ótima para

$$(w, v, n-1, W - \delta\, w[n])$$

Subestrutura ótima

Suponha que $x[1\,..\,n]$ é mochila ótima para o problema (w, v, n, W).

Se $x[n] = \delta$

então $x[1\,..\,n{-}1]$ é mochila ótima para

$$(w, v, n - 1, W - \delta\, w[n])$$

NOTA. Não há nada de especial acerca do índice n. Uma afirmação semelhante vale para qualquer índice i.

Escolha gulosa

Suponha $w[i] \neq 0$ para todo i.

Escolha gulosa

Suponha $w[i] \neq 0$ para todo i.

Se $v[n]/w[n] \geq v[i]/w[i]$ para todo i

então EXISTE uma mochila ótima $x[1\,..\,n]$ tal que

$$x[n] = \min\left\{1, \frac{W}{w[n]}\right\}$$

Algoritmo guloso

Esta propriedade da escolha gulosa sugere um algoritmo que atribui os valores de $x[1\,..\,n]$ supondo que os dados estejam em ordem decrescente de "valor específico":

$$\frac{v[1]}{w[1]} \leq \frac{v[2]}{w[2]} \leq \cdots \leq \frac{v[n]}{w[n]}$$

Algoritmo guloso

Esta propriedade da escolha gulosa sugere um algoritmo que atribui os valores de $x[1\,..\,n]$ supondo que os dados estejam em ordem decrescente de "valor específico":

$$\frac{v[1]}{w[1]} \leq \frac{v[2]}{w[2]} \leq \cdots \leq \frac{v[n]}{w[n]}$$

É nessa ordem "mágica" que está o segredo do funcionamento do algoritmo.

Algoritmo guloso

Devolve uma mochila ótima para (w, v, n, W).

```
MOCHILA-FRACIONÁRIA (w, v, n, W)
 0  ordene w e v de tal forma que
          v[1]/w[1] ≤ v[2]/w[2] ≤ ··· ≤ v[n]/w[n]
 1  para i ← n decrescendo até 1 faça
 2      se w[i] ≤ W
 3          então x[i] ← 1
 4                W ← W − w[i]
 5          senão x[i] ← W/w[i]
 6                W ← 0
 7  devolva x
```

Algoritmo guloso

Devolve uma mochila ótima para (w, v, n, W).

MOCHILA-FRACIONÁRIA (w, v, n, W)

```
0  ordene w e v de tal forma que
       v[1]/w[1] ≤ v[2]/w[2] ≤ ··· ≤ v[n]/w[n]
1  para i ← n decrescendo até 1 faça
2      se w[i] ≤ W
3          então x[i] ← 1
4                W ← W − w[i]
5          senão x[i] ← W/w[i]
6                W ← 0
7  devolva x
```

Consumo de tempo da linha 0 é $\Theta(n \lg n)$.
Consumo de tempo das linhas 1–7 é $\Theta(n)$.

Invariante

Seja W_0 o valor original de W.
No início de cada execução da linha 1 vale que

Invariante

Seja W_0 o valor original de W.
No início de cada execução da linha 1 vale que

(i0) $x' = x[i{+}1\,..\,n]$ é mochila ótima para

$$(w', v', n', W_0)$$

onde

$$w' = w[i{+}1\,..\,n]$$
$$v' = v[i{+}1\,..\,n]$$
$$n' = n - i$$

Invariante

Seja W_0 o valor original de W.
No início de cada execução da linha 1 vale que

(i0) $x' = x[i{+}1\,..\,n]$ é mochila ótima para

$$(w', v', n', W_0)$$

onde

$$w' = w[i{+}1\,..\,n]$$
$$v' = v[i{+}1\,..\,n]$$
$$n' = n - i$$

Na última iteração $i = 0$ e
portanto $x[1\,..\,n]$ é mochila ótima para (w, v, n, W_0).

Conclusão

O consumo de tempo do algoritmo MOCHILA-FRACIONÁRIA é $\Theta(n \lg n)$.

Escolha gulosa

Precisamos mostrar que se $x[1\,..\,n]$ é uma mochila ótima, então podemos supor que

$$x[n] = \alpha := \min\left\{1, \frac{W}{w[n]}\right\}$$

Escolha gulosa

Precisamos mostrar que se $x[1\,..\,n]$ é uma mochila ótima, então podemos supor que

$$x[n] = \alpha := \min\left\{1, \frac{W}{w[n]}\right\}$$

Depois de mostrar isto, indução faz o resto do serviço.

Escolha gulosa

Precisamos mostrar que se $x[1\,..\,n]$ é uma mochila ótima, então podemos supor que

$$x[n] = \alpha := \min\left\{1, \frac{W}{w[n]}\right\}$$

Depois de mostrar isto, indução faz o resto do serviço.

Técnica: transformar uma solução ótima em uma solução ótima 'gulosa'.

Escolha gulosa

Precisamos mostrar que se $x[1\,..\,n]$ é uma mochila ótima, então podemos supor que

$$x[n] = \alpha := \min\left\{1, \frac{W}{w[n]}\right\}$$

Depois de mostrar isto, indução faz o resto do serviço.

Técnica: transformar uma solução ótima em uma solução ótima 'gulosa'.

Esta transformação é semelhante ao processo de pivotação do algoritmo SIMPLEX para programação linear.

Escolha gulosa

Seja $x[1\,..\,n]$ uma mochila ótima para (w, v, n, W) tal que $x[n]$ é máximo. Se $x[n] = \alpha$, não há o que mostrar.

Escolha gulosa

Seja $x[1\,..\,n]$ uma mochila ótima para (w, v, n, W) tal que $x[n]$ é máximo. Se $x[n] = \alpha$, não há o que mostrar.

Suponha $x[n] < \alpha$.

Escolha gulosa

Seja $x[1\,..\,n]$ uma mochila ótima para (w, v, n, W) tal que $x[n]$ é máximo. Se $x[n] = \alpha$, não há o que mostrar.

Suponha $x[n] < \alpha$.

Podemos supor que $x \cdot w = W$. (Podemos mesmo?)

Escolha gulosa

Seja $x[1\,..\,n]$ uma mochila ótima para (w, v, n, W) tal que $x[n]$ é máximo. Se $x[n] = \alpha$, não há o que mostrar.

Suponha $x[n] < \alpha$.

Podemos supor que $x \cdot w = W$. (Podemos mesmo?)

Seja i em $[1\,..\,n-1]$ tal que $x[i] > 0$.
(Quem garante que existe um tal i?).

Escolha gulosa

Seja $x[1\,..\,n]$ uma mochila ótima para (w, v, n, W) tal que $x[n]$ é máximo. Se $x[n] = \alpha$, não há o que mostrar.

Suponha $x[n] < \alpha$.

Podemos supor que $x \cdot w = W$. (Podemos mesmo?)

Seja i em $[1\,..\,n{-}1]$ tal que $x[i] > 0$.
(Quem garante que existe um tal i?).

Seja

$$\delta := \min \left\{ x[i], (\alpha - x[n]) \frac{w[n]}{w[i]} \right\}$$

e

$$\beta := \delta \, \frac{w[i]}{w[n]} \, .$$

Escolha gulosa

Seja $x[1\,..\,n]$ uma mochila ótima para (w, v, n, W) tal que $x[n]$ é máximo. Se $x[n] = \alpha$, não há o que mostrar.

Suponha $x[n] < \alpha$.

Podemos supor que $x \cdot w = W$. (Podemos mesmo?)

Seja i em $[1\,..\,n{-}1]$ tal que $x[i] > 0$.
(Quem garante que existe um tal i?).

Seja

$$\delta := \min \left\{ x[i], (\alpha - x[n]) \frac{w[n]}{w[i]} \right\}$$

e

$$\beta := \delta \, \frac{w[i]}{w[n]} \, .$$

Note que $\delta > 0$ e $\beta > 0$.

Mais escolha gulosa

Seja $x'[1\,..\,n]$ tal que

$$x'[j] := \begin{cases} x[j] & \text{se } j \neq i \text{ e } j \neq n, \\ x[i] - \delta & \text{se } j = i, \\ x[n] + \beta & \text{se } j = n. \end{cases}$$

Mais escolha gulosa

Seja $x'[1\,..\,n]$ tal que

$$x'[j] := \begin{cases} x[j] & \text{se } j \neq i \text{ e } j \neq n, \\ x[i] - \delta & \text{se } j = i, \\ x[n] + \beta & \text{se } j = n. \end{cases}$$

Verifique que $0 \leq x[j] \leq 1$ para todo j.

Mais escolha gulosa

Seja $x'[1\,..\,n]$ tal que

$$x'[j] := \begin{cases} x[j] & \text{se } j \neq i \text{ e } j \neq n, \\ x[i] - \delta & \text{se } j = i, \\ x[n] + \beta & \text{se } j = n. \end{cases}$$

Verifique que $0 \leq x[j] \leq 1$ para todo j.
Além disso, temos que

$$\begin{aligned} x' \cdot w &= x'[1]w[1] + \cdots + x'[i]w[i] + \cdots + x'[n]w[n] \\ &= x[1]w[1] + \cdots + (x[i] - \delta)w[i] + \cdots + (x[n] + \beta)w[n] \\ &= x[1]w[1] + \cdots + (x[i] - \delta)w[i] + \cdots + \left(x[n] + \delta\,\frac{w[i]}{w[n]}\right) w[n] \\ &= x[1]w[1] + \cdots + x[i]w[i] - \delta w[i] + \cdots + x[n]w[n] + \delta w[i] \\ &= W\,. \end{aligned}$$

Mais escolha gulosa ainda

Temos ainda que

$$
\begin{aligned}
x' \cdot v &= x'[1]v[1] + \cdots + x'[i]v[i] + \cdots + x'[n]v[n] \\
&= x[1]v[1] + \cdots + (x[i] - \delta)v[i] + \cdots + (x[n] + \beta)v[n] \\
&= x[1]v[1] + \cdots + (x[i] - \delta)v[i] + \cdots + \left(x[n] + \delta \frac{w[i]}{w[n]}\right) v[n] \\
&= x[1]v[1] + \cdots + x[i]v[i] - \delta v[i] + \cdots + x[n]v[n] + \delta w[i] \frac{v[n]}{w[n]} \\
&= x \cdot v + \delta \left(w[i] \frac{v[n]}{w[n]} - v[i] \right) \\
&\geq x \cdot v + \delta (w[i] \frac{v[i]}{w[i]} - v[i]) \quad \text{(devido à escolha gulosa!)} \\
&= x \cdot v.
\end{aligned}
$$

Escolha gulosa: epílogo

Assim, x' é uma mochila tal que $x' \cdot v \geq x \cdot v$.

Escolha gulosa: epílogo

Assim, x' é uma mochila tal que $x' \cdot v \geq x \cdot v$.
Como x é mochila ótima, concluímos que $x' \cdot v = x \cdot v$ e que x' é uma mochila ótima que contradiz a nossa escolha de x, já que

$$x'[n] = x[n] + \beta > x[n].$$

Conclusão

Se $v[n]/w[n] \geq v[i]/w[i]$ para todo i
e x é uma mochila ótima para (w, v, n, W)
com $x[n]$ máximo, então

$$x[n] = \min \left\{ 1, \frac{W}{w[n]} \right\}.$$

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 onde
$\pi(i)$ é a posição em que a tarefa i é executada.

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 onde
$\pi(i)$ é a posição em que a tarefa i é executada.

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 $L = \max_i \ell_i$.

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

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

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 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 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.