Quando Algoritmos Gulosos Falham

Problema do Troco e Limitações

Prof. Me. Lucas Bruzzone

Aula 12

Relembrando: Algoritmos Gulosos

Estratégia gulosa:

- Fazer escolha localmente ótima em cada etapa
- Nunca reconsiderar decisões
- Esperar que isso leve ao ótimo global

Quando funcionam?

- Propriedade da escolha gulosa
- Subestrutura ótima

Questão de hoje:

E quando essas propriedades NÃO estão presentes?

Problema do Troco - Versão Clássica

Problema:

- Dar troco de valor V usando mínimo número de moedas
- Disponíveis: quantidades ilimitadas de cada denominação

Sistema monetário brasileiro:

$$\text{Moedas: } \{1, 5, 10, 25, 50, 100\} \text{ centavos}$$

Algoritmo guloso:

1. Ordenar moedas em ordem decrescente
2. Para cada moeda:
   - Usar o máximo possível dessa moeda
   - Reduzir valor restante
3. Repetir até completar o troco

Exemplo: Troco com Moedas Brasileiras

Dar troco de 87 centavos

Aplicando estratégia gulosa:

1. 50 centavos: usa 1 moeda $\rightarrow$ resta 37
2. 25 centavos: usa 1 moeda $\rightarrow$ resta 12
3. 10 centavos: usa 1 moeda $\rightarrow$ resta 2
4. 5 centavos: não pode usar $\rightarrow$ resta 2
5. 1 centavo: usa 2 moedas $\rightarrow$ resta 0

Solução gulosa:

$$\{50, 25, 10, 1, 1\} = \textbf{5 moedas}$$

Esta é a solução ótima? Sim! ✓

Por que Funciona com Moedas Canônicas?

Sistema canônico:

- Cada moeda é múltiplo ou sub-múltiplo de outras
- Propriedade especial: guloso sempre produz solução ótima

Exemplos de sistemas canônicos:

- Brasil: $\{1, 5, 10, 25, 50, 100\}$
- EUA: $\{1, 5, 10, 25, 50, 100\}$
- Euro: $\{1, 2, 5, 10, 20, 50, 100, 200\}$

Por que foram projetados assim?

- Facilitar cálculo mental
- Permitir algoritmo guloso simples
- Minimizar número médio de moedas no troco

Sistema Não-Canônico: Primeiro Exemplo

Moedas disponíveis: $\{1, 3, 4\}$ **Troco desejado:** 6

Solução gulosa:

1. 4: usa 1 moeda $\rightarrow$ resta 2
2. 3: não pode usar $\rightarrow$ resta 2
3. 1: usa 2 moedas $\rightarrow$ resta 0

Resultado guloso: $\{4, 1, 1\} =$ **3 moedas**

Solução ótima:

- $\{3, 3\} =$ **2 moedas** ✓

Guloso falhou! Não encontrou a solução ótima.

Analisando a Falha

Por que a estratégia gulosa falhou?

Problema:

- Escolher 4 (localmente ótimo) bloqueou a escolha de 3+3
- Falta de "visão do futuro"
- Decisão inicial impediu solução global ótima

Violação da propriedade:

- Escolha localmente ótima NÃO levou ao ótimo global
- Propriedade da escolha gulosa não vale aqui

Lição importante:

O algoritmo guloso funciona quando é projetado para funcionar!

Outro Exemplo de Falha

Moedas: $\{1, 5, 6, 9\}$ **Troco:** 11

Solução gulosa:

- 9: usa 1 $\rightarrow$ resta 2
- 6: não pode $\rightarrow$ resta 2
- 5: não pode $\rightarrow$ resta 2
- 1: usa 2 $\rightarrow$ resta 0
- **Total:** $\{9, 1, 1\} =$ **3 moedas**

Solução ótima:

- $\{5, 6\} =$ **2 moedas** ✓

Novamente: Escolha gulosa bloqueou solução melhor

Exemplo Mais Dramático

Moedas: $\{1, 10, 20, 25\}$ **Troco:** 40

Solução gulosa:

- 25: usa 1 $\rightarrow$ resta 15
- 20: não pode $\rightarrow$ resta 15
- 10: usa 1 $\rightarrow$ resta 5
- 1: usa 5 $\rightarrow$ resta 0
- **Total:** $\{25, 10, 1, 1, 1, 1, 1\} =$ **7 moedas**

Solução ótima:

- $\{20, 20\} =$ **2 moedas** ✓

Diferença de 5 moedas! Falha significativa.

Tabela Comparativa de Resultados

Moedas	Valor	Guloso	Ótimo
$\{1, 5, 10, 25\}$	30	6	6
$\{1, 5, 10, 25\}$	87	5	5
$\{1, 3, 4\}$	6	3	2
$\{1, 5, 6, 9\}$	11	3	2
$\{1, 10, 20, 25\}$	40	7	2
$\{1, 7, 10\}$	14	5	2

Observação:

- Sistemas canônicos: guloso $=$ ótimo
- Sistemas não-canônicos: guloso pode falhar drasticamente

Como Identificar Sistemas Não-Canônicos?

Não há fórmula simples, mas sinais de alerta:

Indícios de sistema não-canônico:

- Moedas sem relação de múltiplos claros
- Valores "estranhos"ou irregulares
- Lacunas grandes entre denominações
- Moedas próximas em valor mas sem padrão

Exemplos problemáticos:

- $\{1, 3, 4\}$ - valores consecutivos sem padrão
- $\{1, 5, 6, 9\}$ - 6 quebra a sequência esperada
- $\{1, 10, 20, 25\}$ - 25 não se encaixa bem com 20

Teste definitivo:

- Testar exaustivamente valores pequenos
- Se guloso falhar em algum, sistema não é canônico

Como Resolver Corretamente?

Problema: Guloso não garante solução ótima

Solução: Programação Dinâmica

- Explorar todas as possibilidades
- Armazenar soluções de subproblemas
- Garantir solução ótima

Ideia básica:

- $dp[i]$ = mínimo de moedas para valor i
- Para cada valor, testar todas as moedas possíveis
- Escolher a que resulta em menos moedas

Recorrência:

$$dp[i] = \min_{m \in \text{moedas}} \{dp[i-m] + 1\}$$

Algoritmo de Programação Dinâmica

Pseudocódigo:

```
TrocoDP(moedas[], valor):
    dp = array[0..valor] inicializado com infinito
    dp[0] = 0

    para i de 1 ate valor:
        para cada moeda m em moedas:
            se i >= m:
                dp[i] = min(dp[i], dp[i-m] + 1)

    retornar dp[valor]
```

Complexidade:

- Tempo: $O(V \times M)$ onde V = valor, M = número de moedas
- Espaço: $O(V)$

Como a PD Resolve o Problema - A Ideia

Base do raciocínio: Construir soluções de baixo para cima

- Se sei dar troco de 5, posso calcular troco de 6
- Se sei dar troco de 3, posso calcular troco de 6
- Se sei dar troco de 2, posso calcular troco de 6

A pergunta chave: Para dar troco de valor i:

- "Qual moeda devo usar por última?"
- Testo todas as moedas possíveis
- Escolho a que resulta no menor total

Exemplo concreto: Troco de 6 com moedas $\{1, 3, 4\}$

- Posso usar moeda 1: preciso de troco de 5 + esta moeda
- Posso usar moeda 3: preciso de troco de 3 + esta moeda
- Posso usar moeda 4: preciso de troco de 2 + esta moeda
- **Escolho a melhor opção entre as três**

Passo a Passo da Construção

Tabela sendo preenchida: Moedas $\{1, 3, 4\}$, Valor $= 6$

- $dp[0] = 0$ (troco de $0 = 0$ moedas)
- $dp[1] = 1$ (só posso usar moeda 1)
- $dp[2] = 2$ (só posso usar 1+1)
- $dp[3] = 1$ (posso usar moeda 3 - melhor que 1+1+1)
- $dp[4] = 1$ (posso usar moeda 4 - melhor que outras)
- $dp[5] = 2$ (testo moeda 1: $dp[4] + 1 = 2$ e moeda 4: $dp[1] + 1 = 2$)

Calculando dp[6]: Testo todas as moedas

- Usar moeda 1: $dp[5] + 1 = 3$
- Usar moeda 3: $dp[3] + 1 = 2 \leftarrow$ **melhor!**
- Usar moeda 4: $dp[2] + 1 = 3$

Resultado: $dp[6] = 2$ **(solução: 3+3)**

Comparação: Guloso vs PD

Critério	**Guloso**	**PD**
Complexidade	$O(M \log M + V/m_{max})$	$O(V \times M)$
Espaço	$O(1)$	$O(V)$
Garantia de ótimo	Só canônico	Sempre
Simplicidade	Alta	Média
Uso prático	Comum	Quando necessário

Trade-off clássico:

- Guloso: Rápido e simples, mas nem sempre correto
- PD: Sempre correto, mas mais custoso

Decisão de projeto:

- Sistema canônico conhecido: use guloso
- Sistema arbitrário ou não confiável: use PD

Como Reconhecer Quando Usar Cada Técnica?

Use Guloso quando:

- Sistema é conhecido e canônico
- Velocidade é mais importante que perfeição
- Há escolha localmente ótima clara

Use PD quando:

- Solução ótima é obrigatória
- Guloso falha em contra-exemplos
- Problema tem subestrutura ótima

Na prática:

- Teste guloso primeiro em casos pequenos
- Compare com força bruta se possível
- Se falhar, implemente PD
- Guloso pode servir como heurística rápida

Exercícios para Praticar

1. **Implementar ambas soluções**
   - Guloso para troco
   - PD para troco
   - Comparar com moedas $\{1, 3, 4\}$ e valor 6

2. **Encontrar contra-exemplos**
   - Moedas $\{1, 7, 10\}$: encontrar valor onde guloso falha
   - Testar valores de 1 a 20

3. **Criar instância problemática**
   - Inventar sistema de 3 moedas não-canônico
   - Mostrar valor onde diferença é máxima

Próxima Aula

Tema: Introdução à Programação Dinâmica

O que veremos:

- Princípios da Programação Dinâmica
- Subestrutura ótima e sobreposição de subproblemas
- Memoização vs Tabulação
- Primeiros exemplos clássicos
- Como identificar problemas que precisam de PD

Para casa:

- Implementar solução PD para o troco
- Resolver os exercícios propostos
- Pensar: quais outros problemas vistos podem precisar de PD?

Conclusões

Principais lições da aula:

- Algoritmos gulosos são poderosos, mas limitados
- Funcionam apenas quando propriedades específicas existem
- Sistemas monetários reais são projetados para guloso funcionar
- Quando guloso falha, PD geralmente é a solução
- Importante: sempre validar se estratégia gulosa é adequada

Mensagem final:

A simplicidade do guloso é atraente,
mas a correção deve sempre vir primeiro!