Técnicas de Projeto de Algoritmos – Algoritmo Guloso

Kleber Jacques F. de Souza

Algoritmos Gulosos

- São aqueles que, a cada decisão:
  - Sempre escolhem a alternativa que parece mais promissora **naquele instante**
  - **Nunca reconsideram** essa decisão

Algoritmos Gulosos

- Uma escolha que foi feita **nunca é revista**
- Não há **Retrocesso**
- Por fazer a escolha que parece ser a melhor a cada iteração, diz-se que a escolha é feita de acordo com um **critério guloso - decisão localmente ótima!**

Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de otimização.
- Uma característica comum dos problemas onde se aplicam algoritmos gulosos é a existência de **subestrutura ótima**

Algoritmos Gulosos

- **Subestrutura ótima:** quando uma solução ótima para o problema contém em seu interior soluções ótimas para subproblemas.
- Existe uma **função** que verifica se um conjunto de candidatos produz uma solução - sem ter uma visão de todo o problema. **Uma decisão local!**

Algoritmos Gulosos

- A função de seleção é geralmente relacionada com a função objetivo.
- maximizar ➔ candidato que proporcione o **maior ganho individual**.
- minimizar ➔ candidato que proporcione o **menor custo**.

Algoritmos Gulosos - Algoritmo Geral

1. Definir a subestrutura ótima
    A. Definir quais subproblemas podem ser resolvido de maneira ótima para alcançar a solução ótima do problema original total

Algoritmos Gulosos - Algoritmo Geral

2. Definir o critério guloso
   A. Tipicamente, um dos “segredos” dos algoritmos gulosos é a escolha de como o conjunto de entrada será ordenado.
   B. Como será definido a melhor escolha local.

Algoritmos Gulosos - Algoritmo Geral

3. Após uma sequência de decisões, uma solução para o problema é alcançada.
    A. Nessa sequência de decisões, nenhum elemento é examinado mais de uma vez: **ou ele fará parte da solução, ou será descartado.**

Problema do Caixeiro Viajante

- Um caixeiro viajante deseja **visitar *N* cidades** e entre cada par de cidades existe uma rota;
- Cada rota possui uma distância (ou o **custo necessário**) para percorrê-la;

Problema do Caixeiro Viajante

- O caixeiro viajante deseja encontrar:
  - um caminho que **passe por cada cidade apenas uma vez**,
  - o caminho que tenha o **menor custo possível**.

Problema do Caixeiro Viajante

- Solução:
  1. achamos todas as rotas possíveis,
  2. calculamos o comprimento de cada uma delas,
  3. Selecionamos a melhor rota.

Problema do Caixeiro Viajante!

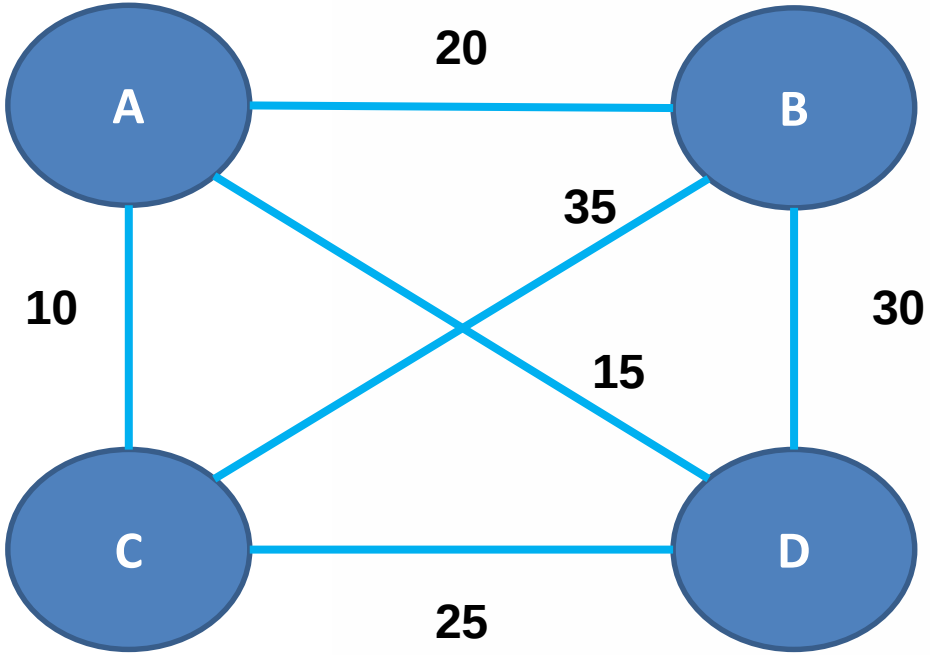
Usando um computador muito veloz, capaz de fazer 1 bilhão de adições por segundo!

# cidades	(n-1)!	Tempo
5	24	insignificante
10	362 880	0.003 segundos
15	87 bilhões	20 minutos
20	$1,2 \times 10^{17}$	73 anos
25	$6,2 \times 10^{23}$	470 milhões de anos

Algoritmos Gulosos: Problema do Caixeiro Viajante

- Critério Guloso: Vizinho mais próximo
  1. Selecione arbitrariamente uma cidade inicial
  2. Selecione a menor rota até qualquer cidade.
  3. Repita até todas as cidades terem sido visitadas.

Algoritmos Gulosos: Problema do Caixeiro Viajante



A → C → D → B → A = 85

B → A → C → D → B = 85

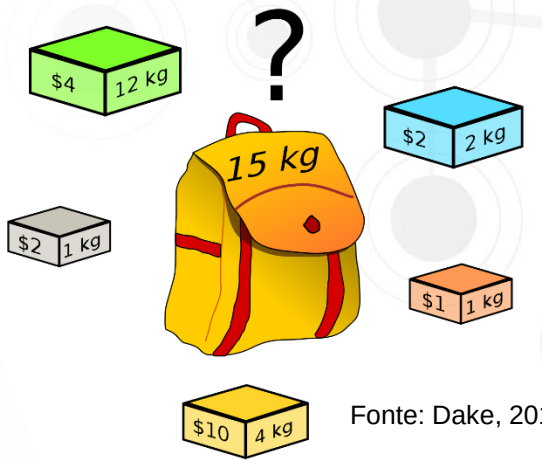
C → A → D → B → C = 90

D → A → C → B → D = 90

Ótimo Local ≠ Ótimo Global

Problema da Mochila

- Dados **n** itens
  - Pesos: p_1, p_2, ..., p_n
  - Valores: v_1, v_2, ..., v_n
  - Uma mochila de capacidade C
- Problema:
  - Encontrar o subconjunto mais valioso de itens que caibam dentro da mochila.



Fonte: Dake, 2017

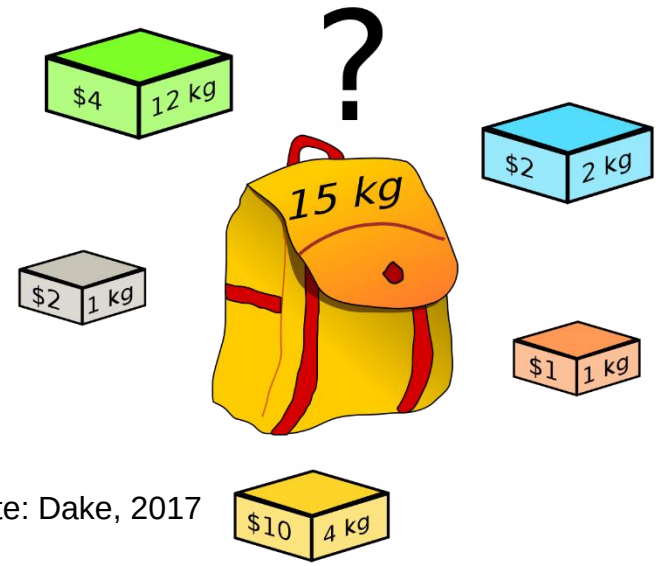
Problema da Mochila

- Como o número de subconjuntos de um conjunto de *n* elementos é 2^n, a busca exaustiva leva a um algoritmo $O(2^n)$.
- Assim, a busca exaustiva leva a algoritmos que são extremamente ineficientes

Algoritmos Gulosos: Problema da Mochila

- Objetivo:
  - maximizar ➔ candidato que proporcione o **maior ganho individual**.
- Critério Guloso:
  - Selecionar o item com o maior valor possível.

Força Bruta - Problema da Mochila



Fonte: Dake, 2017

Item	Peso	Valor
1	12	4
2	2	2
3	1	1
4	4	10
5	1	2

Força Bruta - Problema da Mochila

Item	Peso	Valor
1	12	4
2	2	2
3	1	1
4	4	10
5	1	2

1. Selecionar item 4 - Valor Total = 10
   - Peso Restante = 11
2. Selecionar item 2 - Valor Total = 12
   - Peso Restante = 9
3. Selecionar item 5 - Valor Total = 14
   - Peso Restante = 8
4. Selecionar item 3 - **Valor Total = 15**
   - Peso Restante = 7

Força Bruta - Problema da Mochila

Item	Peso	Valor
1	11	4
2	2	2
3	1	1
4	4	10
5	1	2

1. Selecionar item 4 - Valor Total = 10
   - Peso Restante = 11
2. Selecionar item 1 - **Valor Total = 14**
   - Peso Restante = 0

Algoritmos Gulosos: Problema da Mochila

- Objetivo:
  - maximizar ➔ candidato que proporcione o **maior ganho individual**.
- Critério Guloso:
  - Selecionar o item com **a maior proporção Valor/Peso** valor possível.

Força Bruta - Problema da Mochila

Item	Peso	Valor	V/P
1	11	4	0.36
2	2	2	1
3	1	1	1
4	4	10	2.5
5	1	2	2

1. Selecionar item 4 - Valor Total = 10
   - Peso Restante = 11
2. Selecionar item 5 - Valor Total = 12
   - Peso Restante = 10
3. Selecionar item 2 - Valor Total = 14
   - Peso Restante = 8
4. Selecionar item 3 - **Valor Total = 15**
   - Peso Restante = 7

Referências Bibliográficas

Ziviani, Nivio. **Projeto de Algoritmos:** com implementações em JAVA e C++. CENGACE Learning, 2012. (Livro Eletrônico)

CORMEN, Thomas H. et al. **Algoritmos**: teoria e prática. Elsevier, RJ, 2012.

Dake. **File:Knapsack.svg**. Disponível em: commons.wikimedia.org/wiki/File:Knapsack.svg, 2017.