

Projeto e Análise de Algoritmos

Prof. Me. Jonas Lopes de Vilas Boas

O que é um Algoritmo?

O que é um Problema?

Exemplo de problema

Imagine que você é um minerador buscando pedras preciosas com seu carrinho e sua picareta. Seu carrinho é versátil e fácil de transportar, mas ele só consegue carregar um volume máximo de um metro cúbico e um peso máximo de 400 quilos.

Depois de muito procurar e quase desistir, você encontra uma caverna com várias pedras de diferentes tamanhos e do lado delas uma lista com os volumes, pesos e preço de cada uma delas.

Exemplo de problema

Pedra	Volume (m ³)	Peso (kg)	Preço (R\$)
Diamante	0,455	263	500
Esmeralda	0,521	127	410
Topázio	0,857	254	320
Rubi	0,065	134	315
Jade	0,012	111	280

Você sabe que dependendo da carga que levar, pode ser que você não consiga voltar para buscar o resto.

Quais pedras você levaria?

Exemplo de problema

Quando você está analisando a lista para tentar descobrir qual a carga de pedras você vai levar, você percebe que é uma das páginas de um caderno com inúmeras folhas, descrevendo várias outras pedras.

Ao olhar em volta, você vê que realmente existem muitas outras pedras disponíveis.

Como você analisaria esse caderno em busca das melhores pedras para compor sua carga?

Tipos de problemas

Ordenação

Busca

Processamento de Strings

Grafos

Combinatórios (Otimização)

Geométricos

Numéricos

Estratégias de Projetos de Algoritmos

Força Bruta (Brute Force)

Dividir e Conquistar (Divide and Conquer)

Diminuir e Conquistar (Decrease and Conquer)

Transformar e Conquistar (Transform and Conquer)

Compromisso Tempo–Espaço (Space and Time Tradeoffs)

Estratégia Gulosa (Greedy)

Programação Dinâmica (Dynamic Programming)

Voltando Atrás (Backtracking)

Ramificar e Limitar (Branch and Bound)

Algoritmos Aproximados

Força Bruta

Força Bruta

Também conhecida como busca exaustiva.

Resolve o problema avaliando (quase) todas as possibilidades.

Realiza uma varredura completa (ou parcial) do espaço de busca.

Uma das estratégias mais fáceis de aplicar.

Pode ser usada em uma ampla variedade de problemas.

Algoritmo Geral de Força Bruta

Listar todas as soluções potenciais para o problema de uma maneira sistemática.

Todas as soluções estão eventualmente listadas;

Nenhuma solução é repetida;

Avaliar as soluções, uma a uma, talvez, desqualificando as não práticas e mantendo a melhor encontrada até o momento.

Quando a busca terminar, retornar a solução encontrada.

Exemplo: Busca Sequencial

Dado uma lista com vários elementos, encontrar um determinado elemento na lista ou chegar ao fim da lista sem encontrar o elemento.

```
int busca(int n, int *vet, int elem) {  
    int i; for (i=0; i<n; i++){  
        if (elem == vet[i]) return i;  
    }  
    return -1;  
}
```

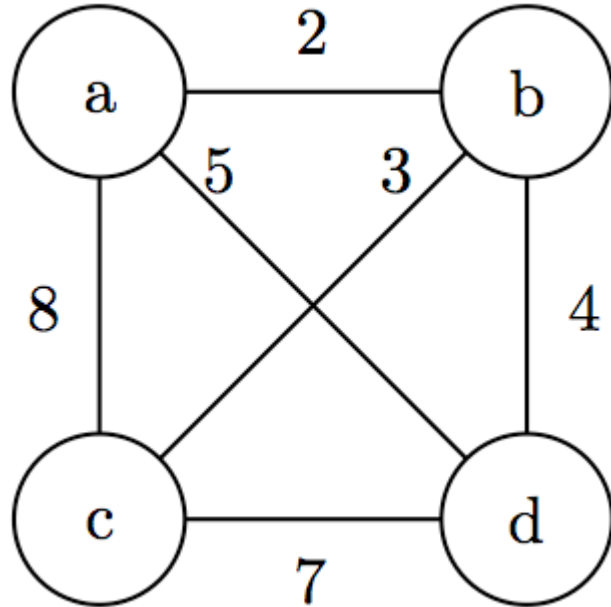
Exemplo: Busca Sequencial

Dado uma lista com vários elementos, retornar quantos elementos similares a um determinado elemento existem na lista.

```
int conta_ocorrencias(int n, int *vet, int elem) {  
    quantidade = 0;  
    int i; for (i=0; i<n; i++){  
        if (elem == vet[i]) quantidade++;  
    }  
    return quantidade;  
}
```

Exemplo: Problema do caixeiro viajante

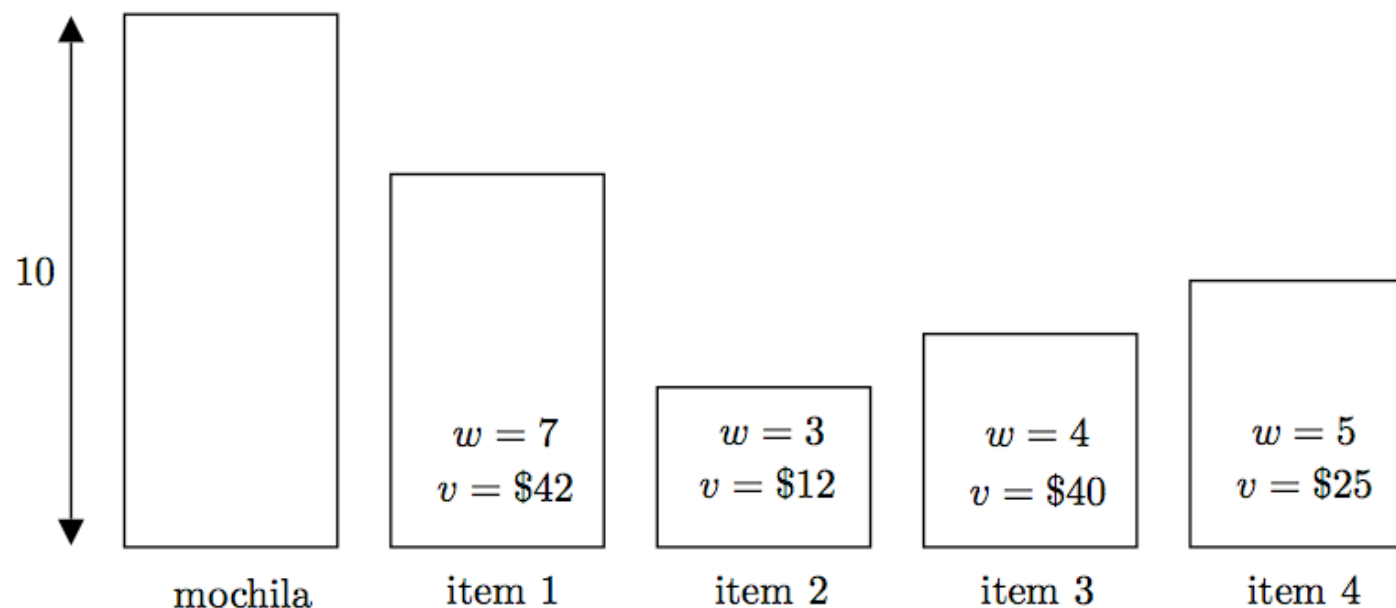
Dadas n cidades com distâncias conhecidas entre cada par, encontrar o trajeto mais curto que passe por todas as cidades exatamente uma vez antes de retornar a cidade de origem.



Trajeto	Custo
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow a$	$2 + 3 + 7 + 5 = 17$
$a \rightarrow b \rightarrow d \rightarrow c \rightarrow a$	$2 + 4 + 7 + 8 = 21$
$a \rightarrow c \rightarrow b \rightarrow d \rightarrow a$	$8 + 3 + 4 + 5 = 20$
$a \rightarrow c \rightarrow d \rightarrow b \rightarrow a$	$8 + 7 + 4 + 2 = 21$
$a \rightarrow d \rightarrow b \rightarrow c \rightarrow a$	$5 + 4 + 3 + 8 = 20$
$a \rightarrow d \rightarrow c \rightarrow b \rightarrow a$	$5 + 7 + 3 + 2 = 17$

Exemplo: Problema da mochila

Dados n itens (Pesos: w_1, w_2, \dots, w_n ; e Valores: v_1, v_2, \dots, v_n ;) e uma mochila de capacidade W , encontrar o subconjunto mais valioso de itens que caibam dentro da mochila.



Solução para o
problema da mochila
por força bruta

Subconjunto	Peso Total	Valor total
ϕ	0	\$0
{1}	7	\$42
{2}	3	\$12
{3}	4	\$40
{4}	5	\$25
{1, 2}	10	\$36
{1, 3}	11	excedeu o peso
{1, 4}	12	excedeu o peso
{2, 3}	7	\$52
{2, 4}	8	\$37
{3, 4}	9	\$65
{1, 2, 3}	14	excedeu o peso
{1, 2, 4}	15	excedeu o peso
{1, 3, 4}	16	excedeu o peso
{2, 3, 4}	12	excedeu o peso
{1, 2, 3, 4}	19	excedeu o peso

Conclusão

Vantagens:

- Ampla aplicabilidade;
- Simplicidade (geralmente baseada diretamente no enunciado);
- Fornece algoritmos razoáveis para alguns problemas.
- Em alguns casos, busca exaustiva (ou variação) é a única solução conhecida.

Desvantagens:

- Raramente fornece algoritmos eficientes;
- Alguns algoritmos força bruta são inaceitavelmente vagarosos;
(quantidade de tempo realística somente para instâncias muito pequenas)
- Não tão construtivo/criativo quanto outras técnicas de projeto de algoritmos;
- Em muitos casos existem alternativas muito melhores!

Estratégia Gulosa

Estratégia Gulosa

Escolhe, em cada iteração, o objeto mais apetitoso que vê pela frente.

Toma decisões com base nas informações disponíveis na iteração corrente, sem olhar as consequências que essas decisões terão no futuro.

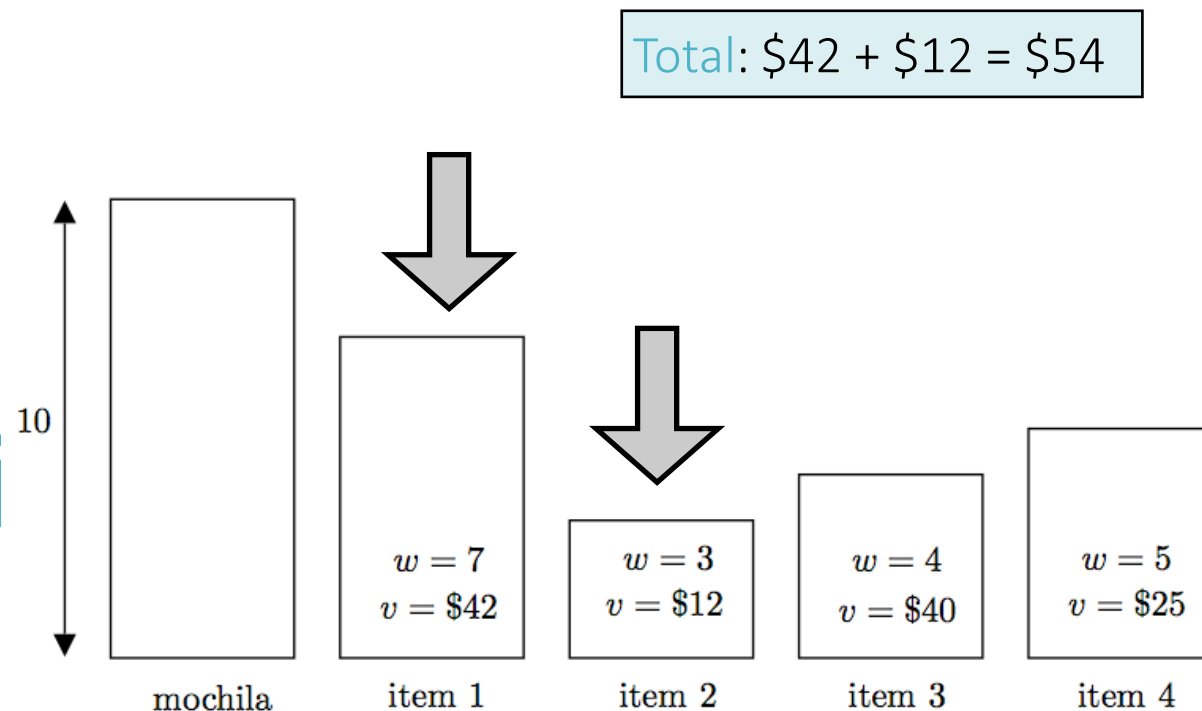
Jamais se arrepende ou volta atrás.

Parece entregar respostas corretas, mas a prova de sua correção é, em geral, difícil e sutil.

São muito eficientes.

Problemas que admitem solução gulosa são raros.

Solução para o problema da mochila por método guloso











Exemplo: Problema do pendrive

Tenho n arquivos digitais no meu computador. Cada arquivo i tem t_i megabytes. Quero gravar o maior número possível de arquivos num pendrive que tem capacidade para c megabytes.

Encontrar o maior subconjunto X do intervalo $1..n$ que satisfaça a restrição $\sum_{i \in X} t_i \leq c$.



$c = 90$

							
10	15	20	20	30	35	40	50
✓	✓	✓	✓	-	-	-	-
✓	✓	✓	-	✓	-	-	-
✓	-	✓	✓	-	✓	-	-
-	✓	✓	✓	-	✓	-	-

Exemplo: Problema do pendrive

PENDRIVE $(t, n, c) \triangleright t_1 \leq \dots \leq t_n$

1 $i := 1$

2 enquanto $i \leq n$ e $t_i \leq c$

3 $c := c - t_i \triangleright$ escolhe i

4 $i := i + 1$

5 devolva $\{1, \dots, i-1\}$

10	15	20	20	30	35	40	50
✓	✓	✓	✓	-	-	-	-
✓	✓	✓	-	✓	-	-	-
✓	-	✓	✓	-	✓	-	-
-	✓	✓	✓	-	✓	-	-

Busca exaustiva

vs.

Algoritmos gulosos

- Garante o **ótimo global**.
- São lentos para encontrar uma solução.

- Não garante o **ótimo global**.
- São rápidos para encontrar uma solução.