

Algoritmos e Estruturas de Dados II

Backtracking

Prof. Tiago Eugenio de Melo

tmelo@uea.edu.br

www.tiagodemelo.info

Observações

- As palavras com a fonte `Courier` indicam as palavras-reservadas da linguagem de programação.

Referências

- **Fundamentals of Python From First Programs Through Data Structures.** Kenneth A. Lambert. CENGAGE Learning, 2010.
- **Algorithms in a Nutshell.** George T. Heineman, Gary Pollice, Stanley Selkow. O'Reilly Media, 2009.
- **Projetos de Algoritmos – com implementações em Pascal e C.** Nivio Ziviani. 2ª edição. Thomson, 2005.

Força Bruta

Força Bruta

Força Bruta

- Também conhecido como busca exaustiva.

Força Bruta

- Também conhecido como busca exaustiva.
- É um tipo de estratégia para resolução de problemas que consiste em enumerar todos os possíveis candidatos de uma solução e verificar se cada um satisfaz o problema.

Força Bruta

- Também conhecido como busca exaustiva.
- É um tipo de estratégia para resolução de problemas que consiste em enumerar todos os possíveis candidatos de uma solução e verificar se cada um satisfaz o problema.
- A implementação é simples e sempre encontrará uma solução.

Força Bruta

- Também conhecido como busca exaustiva.
- É um tipo de estratégia para resolução de problemas que consiste em enumerar todos os possíveis candidatos de uma solução e verificar se cada um satisfaz o problema.
- A implementação é simples e sempre encontrará uma solução.
- Porém, o custo computacional é proporcional ao número de candidatos a solução que, em problemas reais, tende a crescer exponencialmente.

Força Bruta

- Estratégia comumente usada em problemas cujo tamanho é limitado ou quando não se conhece um algoritmo mais eficiente.

Força Bruta

Força Bruta

- Problema Clique¹

Força Bruta

- Problema Clique¹

- Considere um conjunto P de n pessoas e uma matriz M de tamanho $n \times n$, tal que

- $M[i, j] = M[j, i] = 1$, se as pessoas i e j se conhecem e $M[i, j] = M[j, i] = 0$, caso contrário.

Força Bruta

- Problema Clique¹

- Considere um conjunto P de n pessoas e uma matriz M de tamanho $n \times n$, tal que $M[i, j] = M[j, i] = 1$, se as pessoas i e j se conhecem e $M[i, j] = M[j, i] = 0$, caso contrário.
- Problema: existe um subconjunto C (Clique), de r pessoas escolhidas de P , tal que qualquer par de pessoas de C se conhecem?

Força Bruta

- Problema Clique¹

- Considere um conjunto P de n pessoas e uma matriz M de tamanho $n \times n$, tal que $M[i, j] = M[j, i] = 1$, se as pessoas i e j se conhecem e $M[i, j] = M[j, i] = 0$, caso contrário.
- Problema: existe um subconjunto C (Clique), de r pessoas escolhidas de P , tal que qualquer par de pessoas de C se conhecem?
- Solução por FB: verificar, para todas as combinações simples (sem repetições) C de r pessoas escolhidas entre as n pessoas do conjunto P , se todos os pares de pessoas de C se conhecem.

Força Bruta

Força Bruta

- Considere um conjunto \mathbb{P} de 8 pessoas representado pela matriz abaixo (8 x 8):

Força Bruta

- Considere um conjunto P de 8 pessoas representado pela matriz abaixo (8 x 8):

x	1	2	3	4	5	6	7	8
1	1	0	1	1	1	1	1	0
2	0	1	0	0	1	0	0	1
3	1	0	1	1	0	1	1	1
4	1	0	1	1	1	1	1	1
5	1	1	0	1	1	0	0	0
6	1	0	1	1	0	1	1	1
7	1	0	1	1	0	1	1	0
8	0	1	1	1	0	1	0	1

Força Bruta

- Considere um conjunto P de 8 pessoas representado pela matriz abaixo (8 x 8):

x	1	2	3	4	5	6	7	8
1	1	0	1	1	1	1	1	0
2	0	1	0	0	1	0	0	1
3	1	0	1	1	0	1	1	1
4	1	0	1	1	1	1	1	1
5	1	1	0	1	1	0	0	0
6	1	0	1	1	0	1	1	1
7	1	0	1	1	0	1	1	0
8	0	1	1	1	0	1	0	1

- Existem um conjunto C de 5 pessoas escolhidas de P tal que qualquer par de pessoas de C se conhecem?

Força Bruta

Força Bruta

- Existem 56 combinações simples de 5 elementos escolhidos dentre um conjunto de 8 elementos:

Força Bruta

- Existem 56 combinações simples de 5 elementos escolhidos dentre um conjunto de 8 elementos:

1 2 3 4 5	1 2 4 6 8	1 3 5 7 8	2 3 5 6 8
1 2 3 4 6	1 2 4 7 8	1 3 6 7 8	2 3 5 7 8
1 2 3 4 7	1 2 5 6 7	1 4 5 6 7	2 3 6 7 8
1 2 3 4 8	1 2 5 6 8	1 4 5 6 8	2 4 5 6 7
1 2 3 5 6	1 2 5 7 8	1 4 5 7 8	2 4 5 6 8
1 2 3 5 7	1 2 6 7 8	1 4 6 7 8	2 4 5 7 8
1 2 3 5 8	1 3 4 5 6	1 5 6 7 8	2 4 6 7 8
1 2 3 6 7	1 3 4 5 7	2 3 4 5 6	2 5 6 7 8
1 2 3 6 8	1 3 4 5 8	2 3 4 5 7	3 4 5 6 7
1 2 3 7 8	1 3 4 6 7	2 3 4 5 8	3 4 5 6 8
1 2 4 5 6	1 3 4 6 8	2 3 4 6 7	3 4 5 7 8
1 2 4 5 7	1 3 4 7 8	2 3 4 6 8	3 4 6 7 8
1 2 4 5 8	1 3 5 6 7	2 3 4 7 8	3 5 6 7 8
1 2 4 6 7	1 3 5 6 8	2 3 5 6 7	4 5 6 7 8

Força Bruta

- Existem 56 combinações simples de 5 elementos escolhidos dentre um conjunto de 8 elementos:

1 2 3 4 5	1 2 4 6 8	1 3 5 7 8	2 3 5 6 8
1 2 3 4 6	1 2 4 7 8	1 3 6 7 8	2 3 5 7 8
1 2 3 4 7	1 2 5 6 7	1 4 5 6 7	2 3 6 7 8
1 2 3 4 8	1 2 5 6 8	1 4 5 6 8	2 4 5 6 7
1 2 3 5 6	1 2 5 7 8	1 4 5 7 8	2 4 5 6 8
1 2 3 5 7	1 2 6 7 8	1 4 6 7 8	2 4 5 7 8
1 2 3 5 8	1 3 4 5 6	1 5 6 7 8	2 4 6 7 8
1 2 3 6 7	1 3 4 5 7	2 3 4 5 6	2 5 6 7 8
1 2 3 6 8	1 3 4 5 8	2 3 4 5 7	3 4 5 6 7
1 2 3 7 8	1 3 4 6 7	2 3 4 5 8	3 4 5 6 8
1 2 4 5 6	1 3 4 6 8	2 3 4 6 7	3 4 5 7 8
1 2 4 5 7	1 3 4 7 8	2 3 4 6 8	3 4 6 7 8
1 2 4 5 8	1 3 5 6 7	2 3 4 7 8	3 5 6 7 8
1 2 4 6 7	1 3 5 6 8	2 3 5 6 7	4 5 6 7 8

Força Bruta

Força Bruta

- Note que todos os pares de pessoas do subconjunto $C = \{1, 3, 4, 6, 7\}$ se conhecem:

Força Bruta

- Note que todos os pares de pessoas do subconjunto $C = \{1, 3, 4, 6, 7\}$ se conhecem:

x	1	3	4	6	7
1	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1

Força Bruta

- Note que todos os pares de pessoas do subconjunto $C = \{1, 3, 4, 6, 7\}$ se conhecem:

x	1	3	4	6	7
1	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1

- Como enumerar todas as combinações simples de r elementos de um conjunto de tamanho n ?

Força Bruta

- Note que todos os pares de pessoas do subconjunto $C = \{1, 3, 4, 6, 7\}$ se conhecem:

x	1	3	4	6	7
1	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1

- Como enumerar todas as combinações simples de r elementos de um conjunto de tamanho n ?



Backtracking

Introdução

Introdução

- É um refinamento da estratégia dos algoritmos de força bruta.

Introdução

- É um refinamento da estratégia dos algoritmos de força bruta.
- Parte das possíveis soluções que podem ser eliminadas sem que sejam explicitamente executadas.

Introdução

- É um refinamento da estratégia dos algoritmos de força bruta.
- Parte das possíveis soluções que podem ser eliminadas sem que sejam explicitamente executadas.
- Problemas cujas soluções podem ser definidas através de uma sequência de decisões.

Introdução

- É um refinamento da estratégia dos algoritmos de força bruta.
- Parte das possíveis soluções que podem ser eliminadas sem que sejam explicitamente executadas.
- Problemas cujas soluções podem ser definidas através de uma sequência de decisões.
- Os problemas podem ser modelados por uma árvore de decisão que representa todas as possíveis sequências de decisão.

Introdução

Introdução

- Se houver mais de uma decisão disponível para cada uma das n decisões, a busca exaustiva (força bruta) será exponencial.

Introdução

- Se houver mais de uma decisão disponível para cada uma das n decisões, a busca exaustiva (força bruta) será exponencial.
- A eficiência da estratégia depende da possibilidade de limitar a busca.

Introdução

- Se houver mais de uma decisão disponível para cada uma das n decisões, a busca exaustiva (força bruta) será exponencial.
- A eficiência da estratégia depende da possibilidade de limitar a busca.
- Necessário definir um espaço de solução para o problema:

Introdução

- Se houver mais de uma decisão disponível para cada uma das n decisões, a busca exaustiva (força bruta) será exponencial.
- A eficiência da estratégia depende da possibilidade de limitar a busca.
- Necessário definir um espaço de solução para o problema:
 - Que inclua a solução ótima.

Introdução

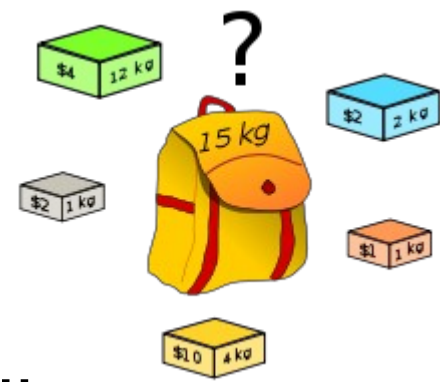
- Se houver mais de uma decisão disponível para cada uma das n decisões, a busca exaustiva (força bruta) será exponencial.
- A eficiência da estratégia depende da possibilidade de limitar a busca.
- Necessário definir um espaço de solução para o problema:
 - Que inclua a solução ótima.
 - Que possa ser pesquisada de forma organizada.

Exemplos

Problema da Mochila

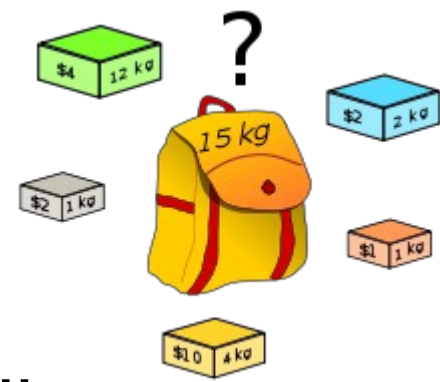


Problema da Mochila

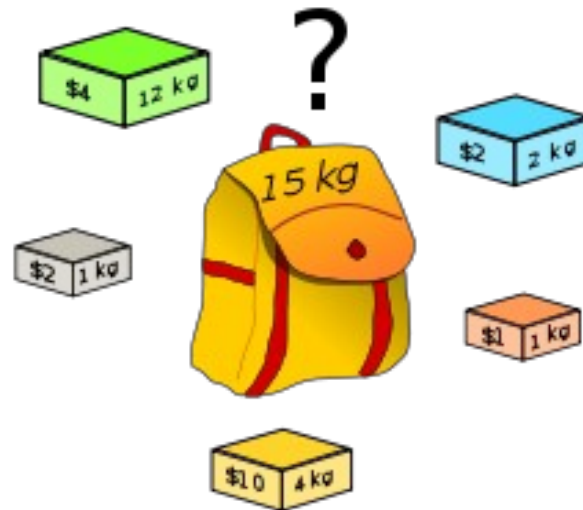


- Deve-se preencher uma mochila com diversos itens com pesos e/ou valores diferentes.

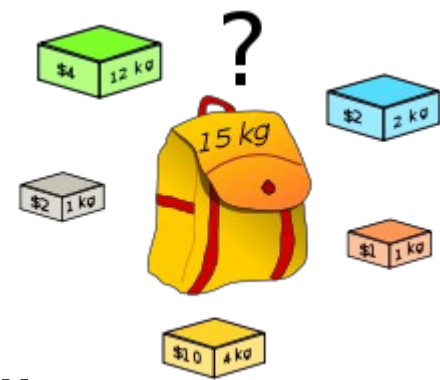
Problema da Mochila



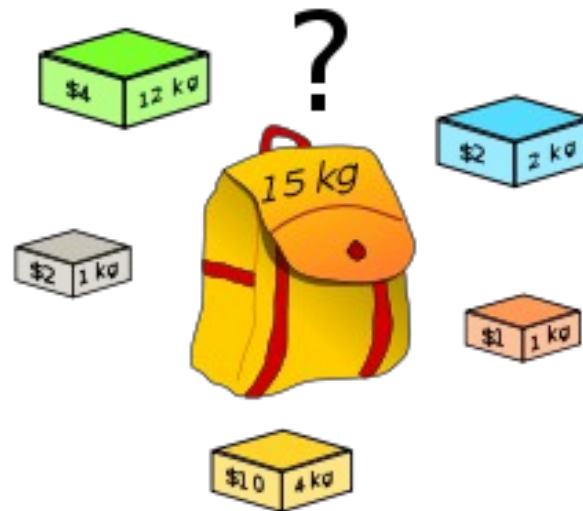
- Deve-se preencher uma mochila com diversos itens com pesos e/ou valores diferentes.



Problema da Mochila



- Deve-se preencher uma mochila com diversos itens com pesos e/ou valores diferentes.



- O objetivo é preencher a mochila com o maior valor possível, não ultrapassando o peso máximo suportado pela mochila.

Problema da Mochila

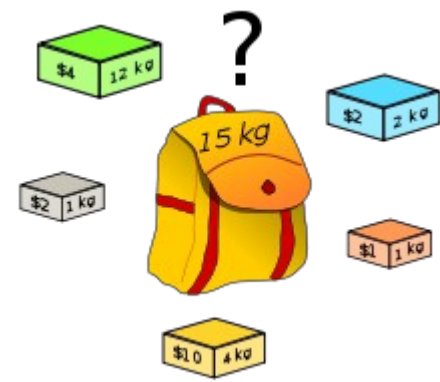


Problema da Mochila



- Entrada

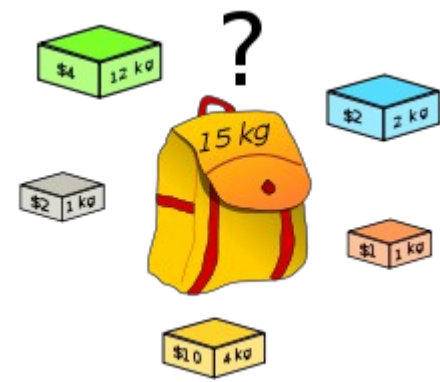
Problema da Mochila



- Entrada

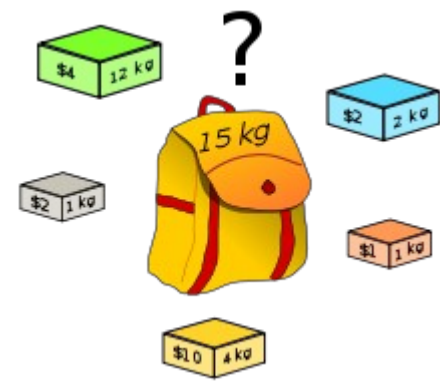
- **N** itens com pesos p_i e valores v_i , onde a capacidade da mochila é **K**.

Problema da Mochila



- Entrada
 - **N** itens com pesos p_i e valores v_i , onde a capacidade da mochila é **K**.
- O objetivo é obter um conjunto **S** de itens, tais que:

Problema da Mochila



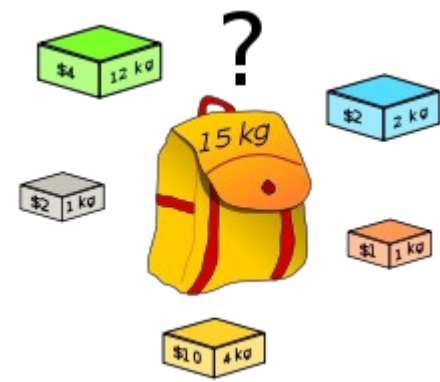
- Entrada
 - N itens com pesos p_i e valores v_i , onde a capacidade da mochila é K .
- O objetivo é obter um conjunto S de itens, tais que:
 - A soma dos pesos dos itens $S \leq K$ e a soma dos valores dos itens em S seja a maior possível.

Problema da Mochila

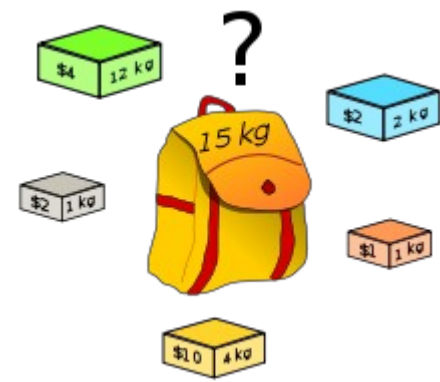


Problema da Mochila

- Qual item escolher primeiro?



Problema da Mochila



- Qual item escolher primeiro?
 - Maior valor?

Problema da Mochila

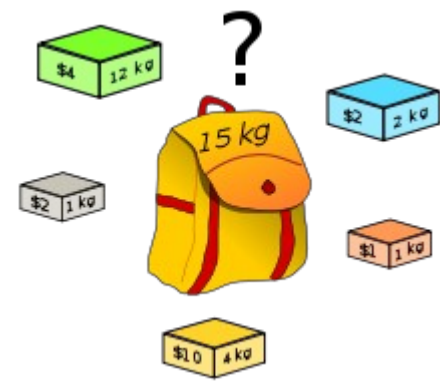


- Qual item escolher primeiro?
 - Maior valor?
 - Menor peso?

Problema da Mochila



Problema da Mochila



- Solução por força bruta

Problema da Mochila



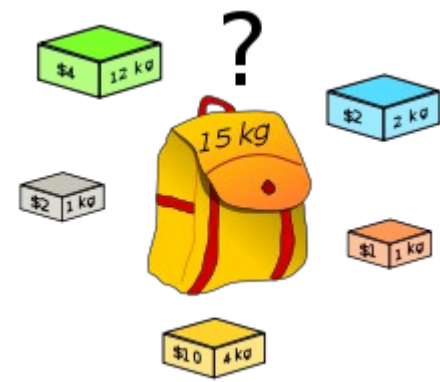
- Solução por força bruta
 - Gerar todas as possíveis combinações.

Problema da Mochila



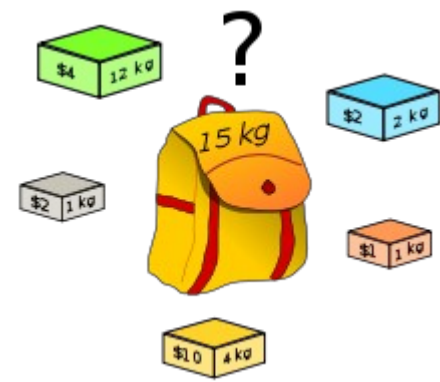
- Solução por força bruta
 - Gerar todas as possíveis combinações.
 - Com n itens, existem 2^n soluções.

Problema da Mochila



- Solução por força bruta
 - Gerar todas as possíveis combinações.
 - Com n itens, existem 2^n soluções.
 - Checar se cada solução satisfaz limite de peso.

Problema da Mochila



- Solução por força bruta
 - Gerar todas as possíveis combinações.
 - Com n itens, existem 2^n soluções.
 - Checar se cada solução satisfaz limite de peso.
 - Salvar a condição que melhor representa a solução.

Problema da Mochila



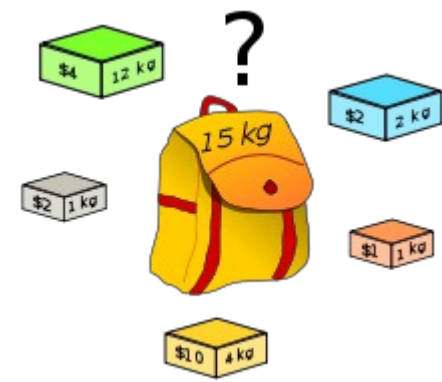
- Solução por força bruta
 - Gerar todas as possíveis combinações.
 - Com n itens, existem 2^n soluções.
 - Checar se cada solução satisfaz limite de peso.
 - Salvar a condição que melhor representa a solução.
 - Pode ser representada como uma árvore.

Problema da Mochila



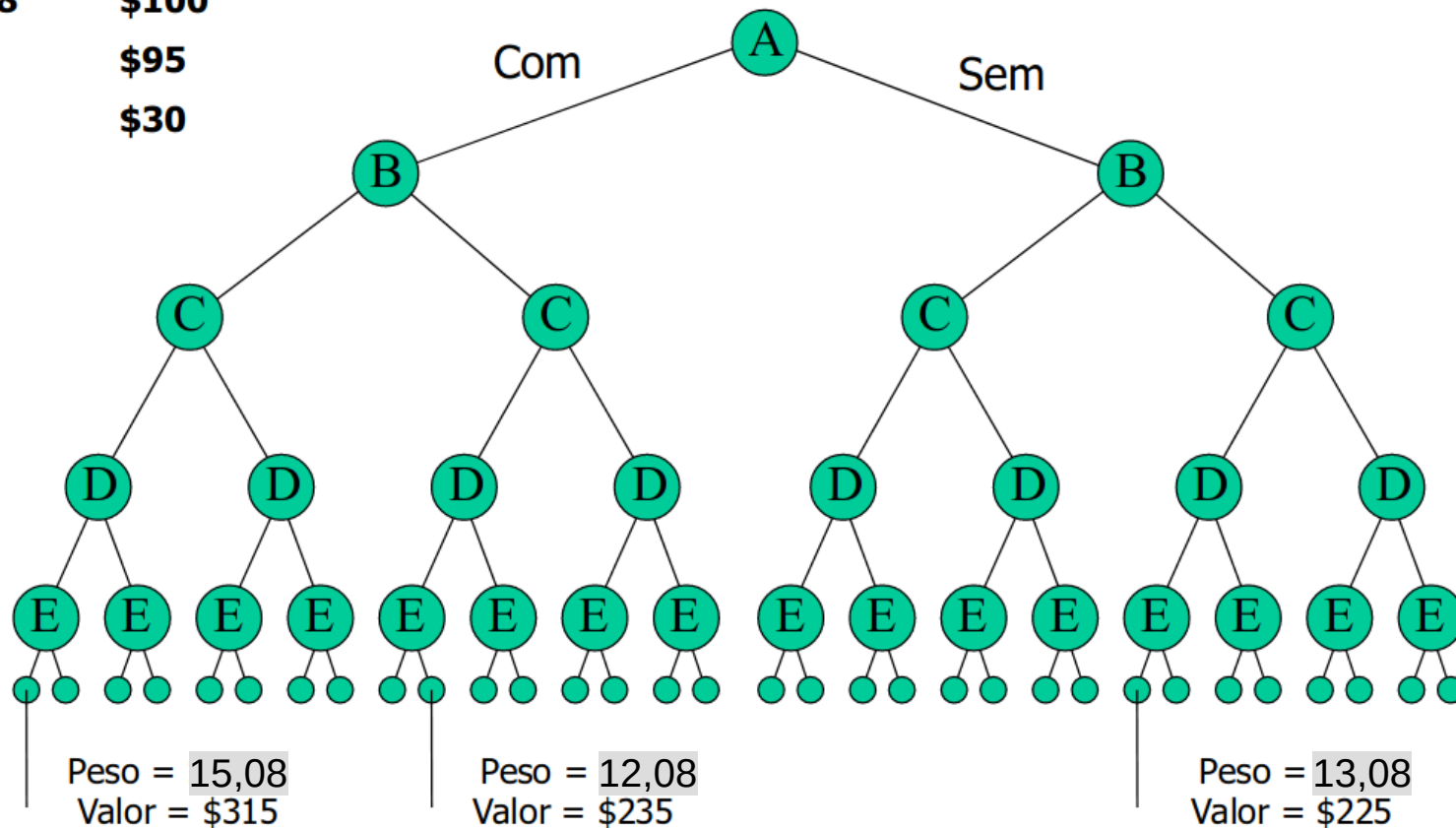
- Exemplo de mochila de 10 kg

Problema da Mochila

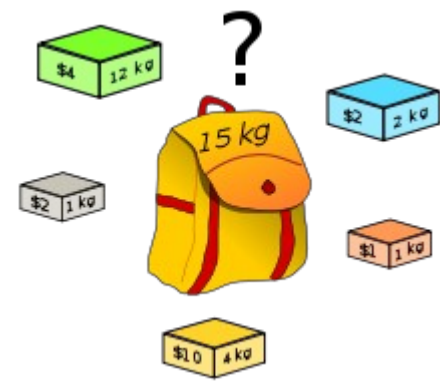


- Exemplo de mochila de 10 kg

A	2	\$40
B	3,1	\$50
C	1,98	\$100
D	5	\$95
E	3	\$30



Problema da Mochila



Problema da Mochila



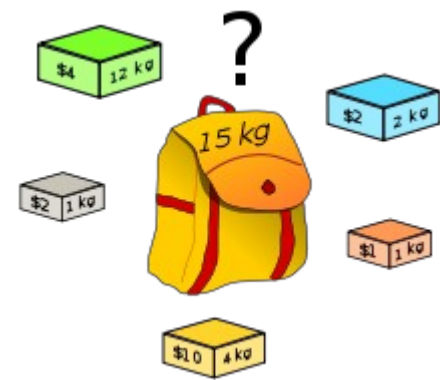
- Se alcançarmos um ponto em que a solução não é mais viável, não precisamos continuar explorando a solução.

Problema da Mochila



- Se alcançarmos um ponto em que a solução não é mais viável, não precisamos continuar explorando a solução.
 - Podemos voltar (*backtrack*) a partir deste ponto.

Problema da Mochila



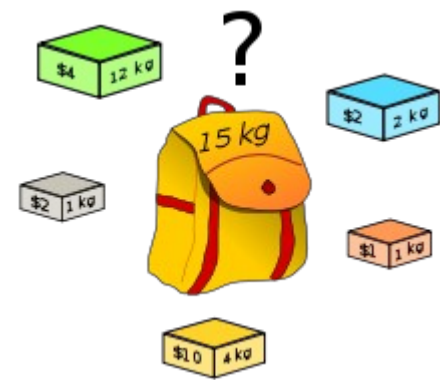
- Se alcançarmos um ponto em que a solução não é mais viável, não precisamos continuar explorando a solução.
 - Podemos voltar (*backtrack*) a partir deste ponto.
- Essa estratégia se torna bastante útil:

Problema da Mochila



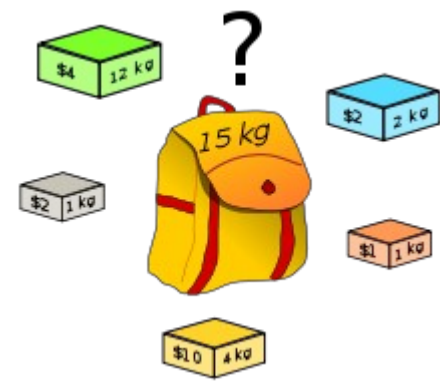
- Se alcançarmos um ponto em que a solução não é mais viável, não precisamos continuar explorando a solução.
 - Podemos voltar (*backtrack*) a partir deste ponto.
- Essa estratégia se torna bastante útil:
 - Na medida em que o número de itens cresce.

Problema da Mochila



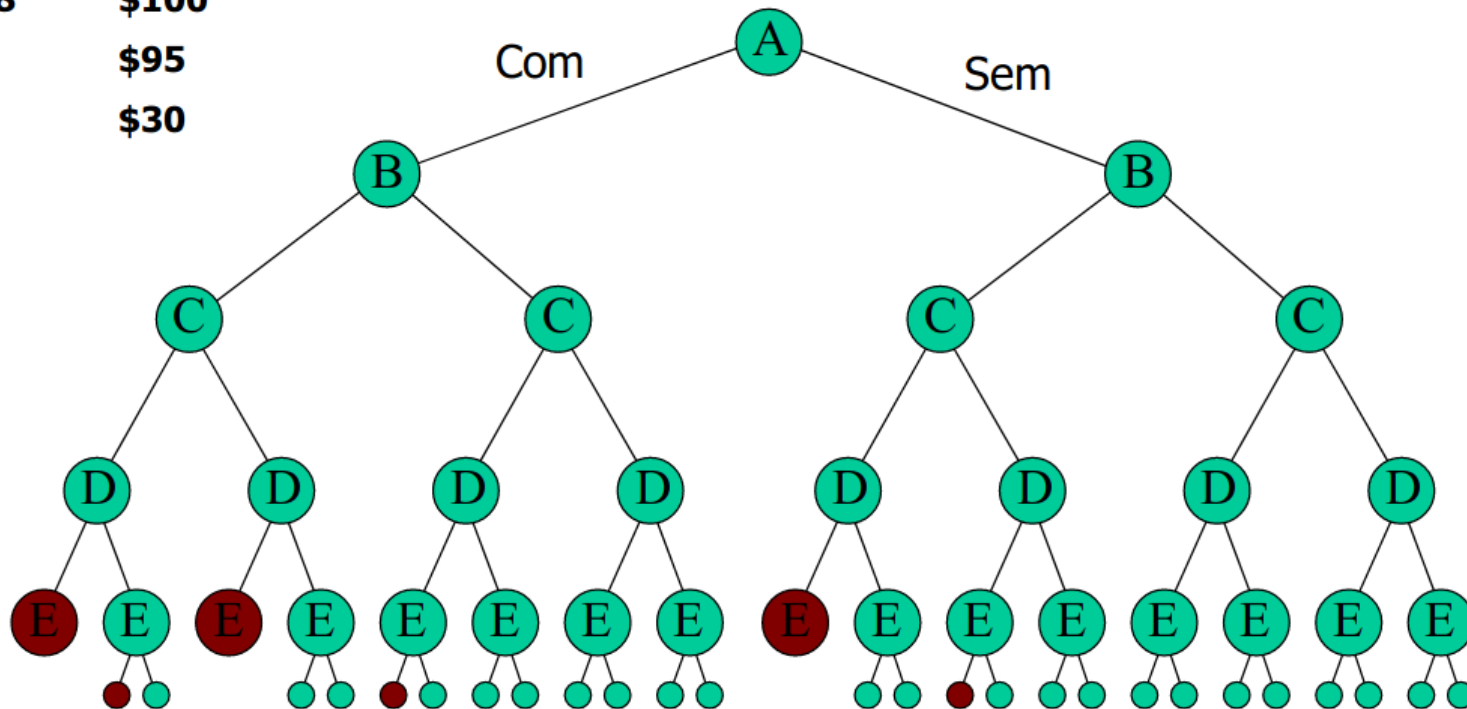
- Se alcançarmos um ponto em que a solução não é mais viável, não precisamos continuar explorando a solução.
 - Podemos voltar (*backtrack*) a partir deste ponto.
- Essa estratégia se torna bastante útil:
 - Na medida em que o número de itens cresce.
 - Na medida em que a capacidade da mochila diminui.

Problema da Mochila



- Backtracking < 10 kg

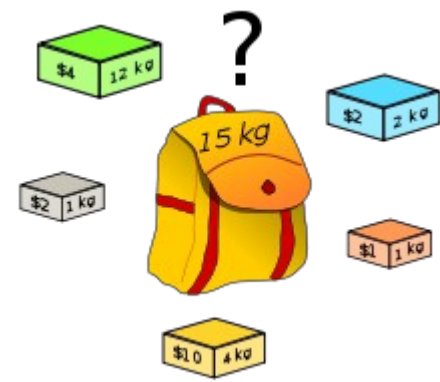
A	2	\$40
B	3,1	\$50
C	1,98	\$100
D	5	\$95
E	3	\$30



Problema da Mochila

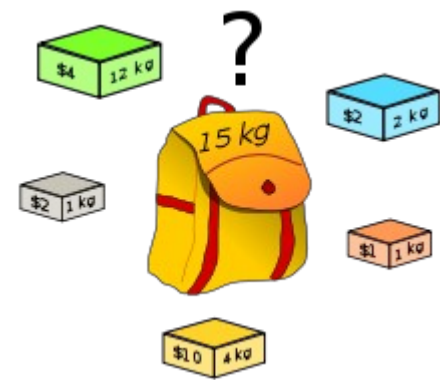


Problema da Mochila



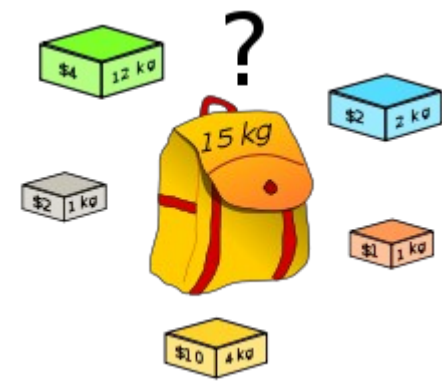
- Pode-se voltar também quando se sabe que a melhor solução da subárvore é pior do que a melhor solução já encontrada.

Problema da Mochila



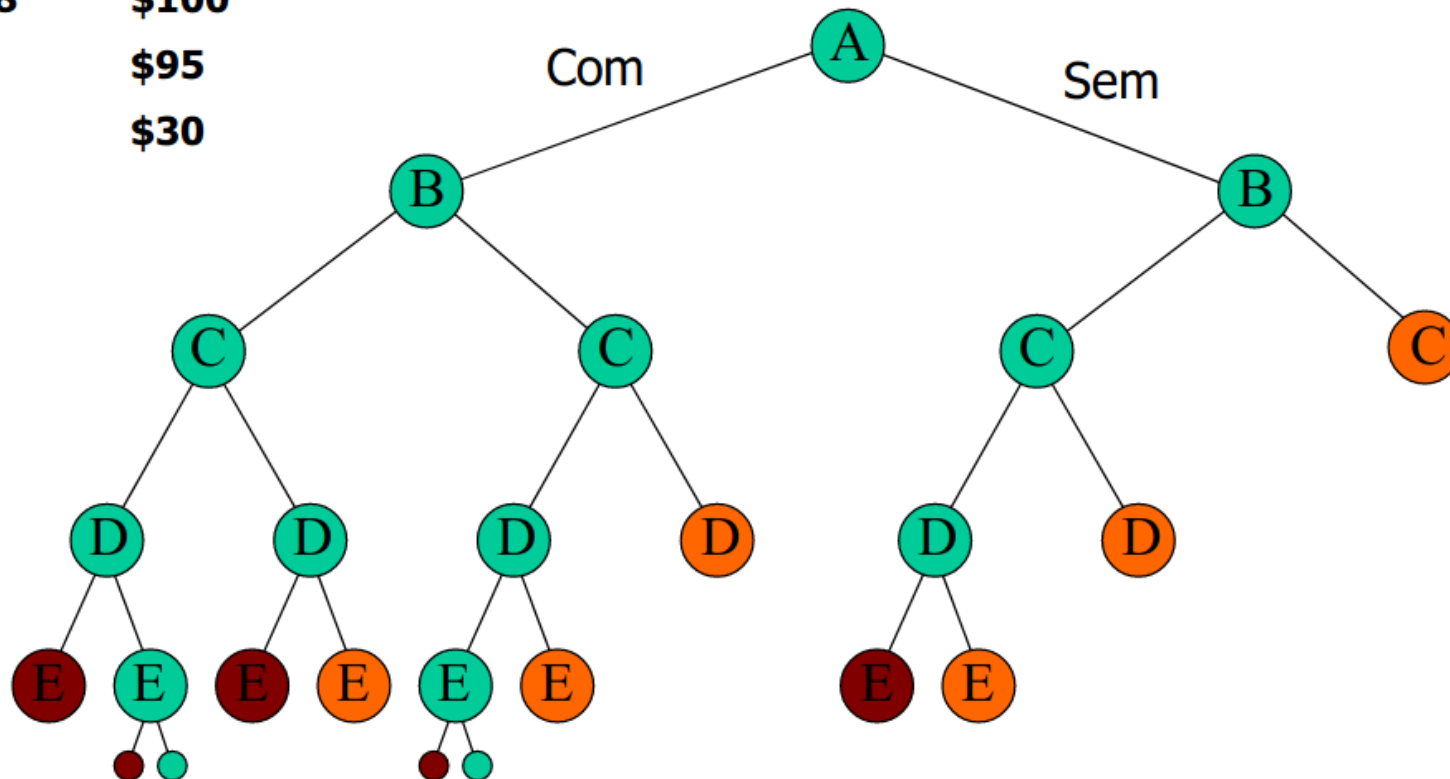
- Pode-se voltar também quando se sabe que a melhor solução da subárvore é pior do que a melhor solução já encontrada.
 - É uma estratégia usada por muitos algoritmos.

Problema da Mochila

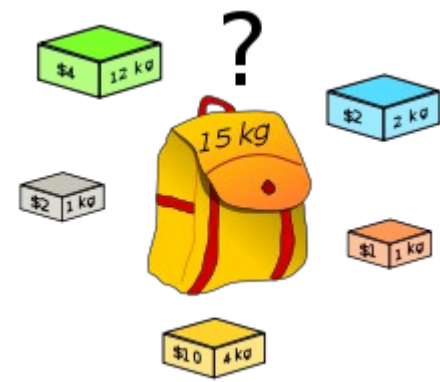


- Backtracking com cortes por qualidade

A	2	\$40
B	3,1	\$50
C	1,98	\$100
D	5	\$95
E	3	\$30



Problema da Mochila



- Solução genérica

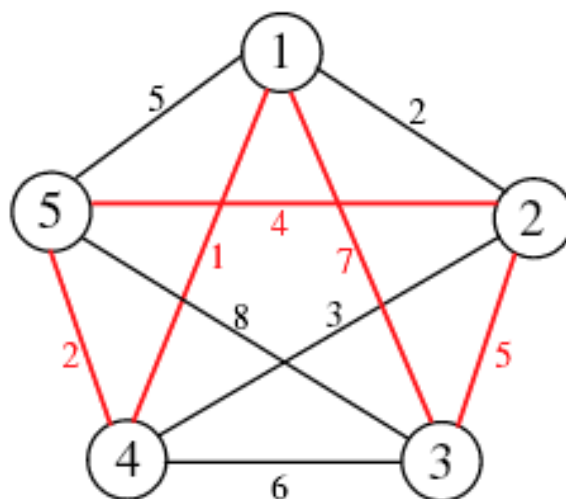
```
def backtrack(v): # v é o nó sendo pesquisado
    if (promissor(v)):
        if (existe_solucacao(v)):
            armazena_solucacao(v)
        else:
            for filho in v:
                backtrack(filho)
    }
```

Problema do Caixeiro Viajante

- O problema consiste em minimizar o custo de um caixeiro viajante que deseja percorrer n cidades, visitando cada cidade apenas uma vez, e retornar para casa.

Problema do Caixeiro Viajante

- O problema consiste em minimizar o custo de um caixeiro viajante que deseja percorrer n cidades, visitando cada cidade apenas uma vez, e retornar para casa.



Problema do Caixeiro Viajante

Problema do Caixeiro Viajante

- Força bruta

Problema do Caixeiro Viajante

- Força bruta
 - Se calcularmos um bilhão de soluções por segundo com um grafo com 30 cidades, demoraria **8 quadrilhões de anos** para achar a melhor solução.

Problema do Caixeiro Viajante

- Força bruta
 - Se calcularmos um bilhão de soluções por segundo com um grafo com 30 cidades, demoraria **8 quadrilhões de anos** para achar a melhor solução.
- Esse problema é **$O(n!)$** .

Sudoku

Sudoku

- Regras:

Sudoku

- Regras:
 - Matriz de 9 x 9.

Sudoku

- Regras:
 - Matriz de 9 x 9.
 - Cada célula pode ter um valor 1 a 9.

Sudoku

- Regras:
 - Matriz de 9 x 9.
 - Cada célula pode ter um valor 1 a 9.
 - Cada linha e coluna possuem sempre números distintos.

Sudoku

- Regras:
 - Matriz de 9 x 9.
 - Cada célula pode ter um valor 1 a 9.
 - Cada linha e coluna possuem sempre números distintos.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku

5	3	1	2	7	6	8	9	4
6	2	4	1	9	5	2		
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9