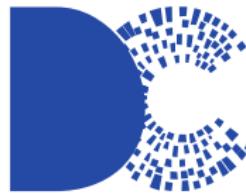




UNIVERSIDADE
FEDERAL DE
SERGIPE



DEPARTAMENTO
DE COMPUTAÇÃO

Backtracking e branch-and-bound

Projeto e Análise de Algoritmos

Bruno Prado

Departamento de Computação / UFS

Introdução

- ▶ Classe de problemas difíceis para algoritmos
 - ▶ Soluções com complexidade exponencial

Introdução

- ▶ Classe de problemas difíceis para algoritmos
 - ▶ Soluções com complexidade exponencial
 - ▶ Demandam tempo de resposta aceitável

Introdução

- ▶ Classe de problemas difíceis para algoritmos
 - ▶ Soluções com complexidade exponencial
 - ▶ Demandam tempo de resposta aceitável
 - ▶ São uma classe de problemas muito relevantes

Introdução

- ▶ Classe de problemas difíceis para algoritmos
 - ▶ Soluções com complexidade exponencial
 - ▶ Demandam tempo de resposta aceitável
 - ▶ São uma classe de problemas muito relevantes

Busca exaustiva \longleftrightarrow *Espaço exponencial*

Introdução

- ▶ Como evitar a busca exaustiva por soluções?
 - ▶ As técnicas de *backtracking* e *branch-and-bound* podem limitar ou reduzir este espaço de soluções

Introdução

- ▶ Como evitar a busca exaustiva por soluções?
 - ▶ As técnicas de *backtracking* e *branch-and-bound* podem limitar ou reduzir este espaço de soluções
 - ▶ São geradas soluções candidatas de forma incremental, buscando atender as restrições do problema e gerar novas soluções baseadas nestas soluções promissoras

Introdução

- ▶ Como evitar a busca exaustiva por soluções?
 - ▶ As técnicas de *backtracking* e *branch-and-bound* podem limitar ou reduzir este espaço de soluções
 - ▶ São geradas soluções candidatas de forma incremental, buscando atender as restrições do problema e gerar novas soluções baseadas nestas soluções promissoras
 - ▶ Apesar de bons resultados práticos, em uma análise de pior caso, estas abordagens podem recair no mesmo desempenho da busca exaustiva

Backtracking

- ▶ O que é *backtracking*?
 - ▶ *Back* = voltar + *tracking* = encaminhamento

Backtracking

- ▶ O que é *backtracking*?
 - ▶ *Back* = voltar + *tracking* = encaminhamento
 - ▶ O algoritmo constrói um conjunto de soluções parciais, sempre avaliando se as regras ou restrições impostas pelo problema são satisfeitas

Backtracking

- ▶ O que é *backtracking*?
 - ▶ *Back* = voltar + *tracking* = encaminhamento
 - ▶ O algoritmo constrói um conjunto de soluções parciais, sempre avaliando se as regras ou restrições impostas pelo problema são satisfeitas
 - ▶ Quando uma solução parcial parece promissora, ou seja, atende as regras ou restrições, são geradas novas soluções parciais a partir desta solução

Backtracking

- ▶ O que é *backtracking*?
 - ▶ *Back* = voltar + *tracking* = encaminhamento
 - ▶ O algoritmo constrói um conjunto de soluções parciais, sempre avaliando se as regras ou restrições impostas pelo problema são satisfeitas
 - ▶ Quando uma solução parcial parece promissora, ou seja, atende as regras ou restrições, são geradas novas soluções parciais a partir desta solução
 - ▶ Se nenhuma solução obtida atende as regras ou restrições, o algoritmo deve retroceder e avaliar a próxima solução parcial ainda não explorada, caso exista

Backtracking

- ▶ Busca em profundidade
 - ▶ As soluções candidatas ou promissoras que atendem as restrições são exploradas primeiro

Backtracking

- ▶ Busca em profundidade
 - ▶ As soluções candidatas ou promissoras que atendem as restrições são exploradas primeiro
 - ▶ É feito o incremento de solução, sempre atendendo as regras impostas pelo problema

Backtracking

- ▶ Busca em profundidade
 - ▶ As soluções candidatas ou promissoras que atendem as restrições são exploradas primeiro
 - ▶ É feito o incremento de solução, sempre atendendo as regras impostas pelo problema
 - ▶ Este processo pode levar a uma solução completa, mas não existe uma garantia

Backtracking

- ▶ Busca em profundidade
 - ▶ As soluções candidatas ou promissoras que atendem as restrições são exploradas primeiro
 - ▶ É feito o incremento de solução, sempre atendendo as regras impostas pelo problema
 - ▶ Este processo pode levar a uma solução completa, mas não existe uma garantia

AS SOLUÇÕES PARCIAIS QUE
NÃO ATENDEM AS RESTRIÇÕES
SÃO DESPREZADAS

Backtracking

- ▶ Busca em profundidade
 - ▶ As soluções candidatas ou promissoras que atendem as restrições são exploradas primeiro
 - ▶ É feito o incremento de solução, sempre atendendo as regras impostas pelo problema
 - ▶ Este processo pode levar a uma solução completa, mas não existe uma garantia

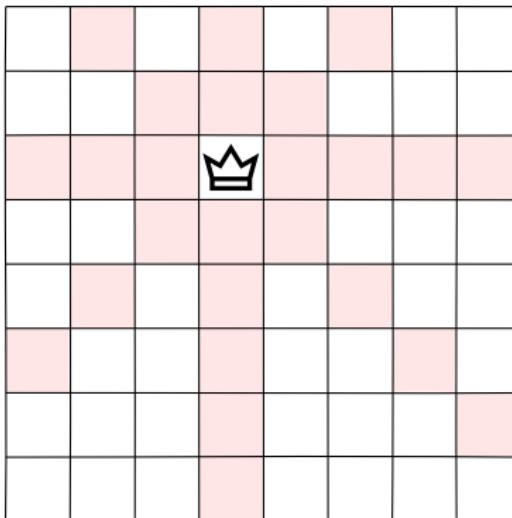
AS SOLUÇÕES PARCIAIS QUE
NÃO ATENDEM AS RESTRIÇÕES
SÃO DESPREZADAS



ESPAÇO DE BUSCA REDUZIDO

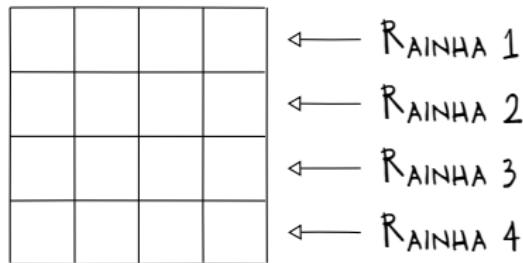
Backtracking

- ▶ Problema das n -rainhas: colocar n rainhas em um tabuleiro de dimensões $n \times n$, de forma que nenhuma das rainhas esteja em linhas de ataque diagonais, horizontais e verticais, sem limite de alcance



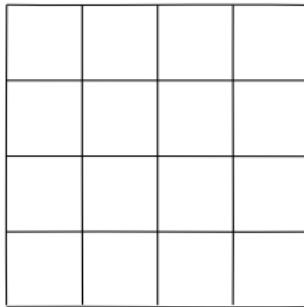
Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas

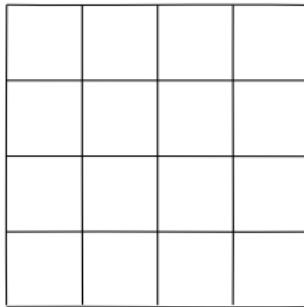


SOLUÇÃO \emptyset

NENHUMA RAINHA POSICIONADA

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



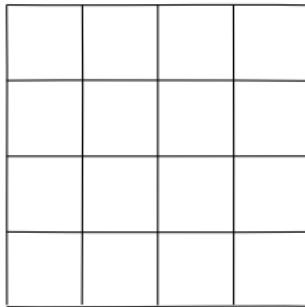
SOLUÇÃO \emptyset

NENHUMA RAINHA POSICIONADA

SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



QUEEN			

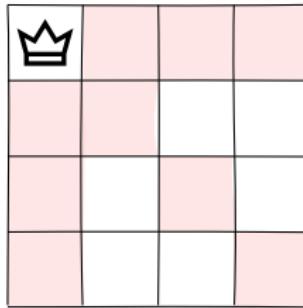
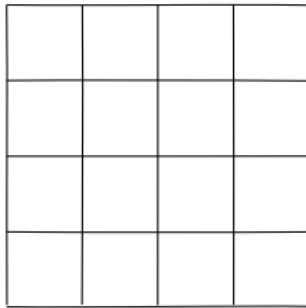
SOLUÇÃO \emptyset

SOLUÇÃO 1

A RAINHA 1 FOI POSICIONADA EM (\emptyset, \emptyset)

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO \emptyset

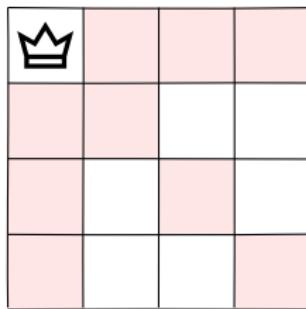
SOLUÇÃO 1

A RAINHA 1 FOI POSICIONADA EM (\emptyset, \emptyset)

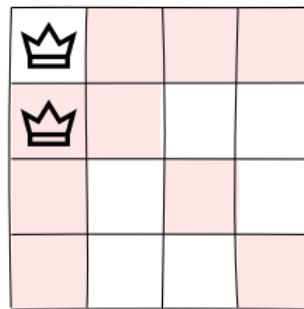
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 1

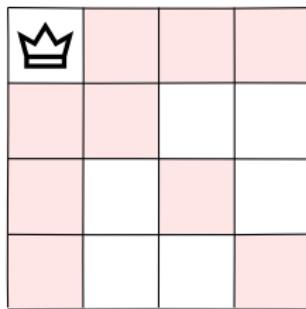


SOLUÇÃO X

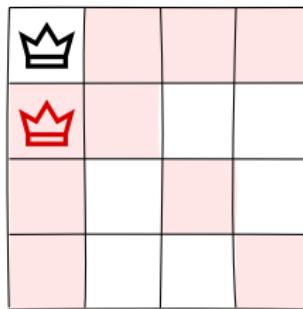
A RAINHA 2 FOI POSICIONADA EM $(1, \emptyset)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 1



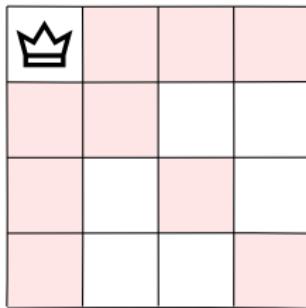
SOLUÇÃO X

A RAINHA 2 FOI POSICIONADA EM $(1, \emptyset)$

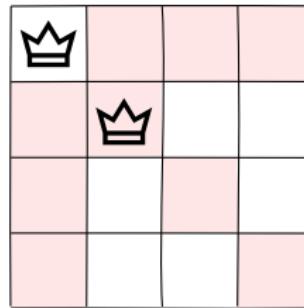
EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 1

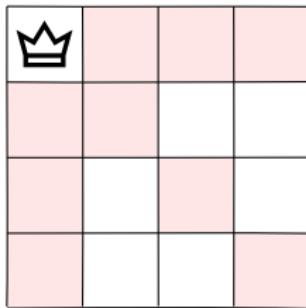


SOLUÇÃO X

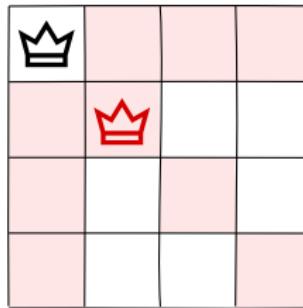
A RAINHA 2 FOI POSICIONADA EM $(1, 1)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 1



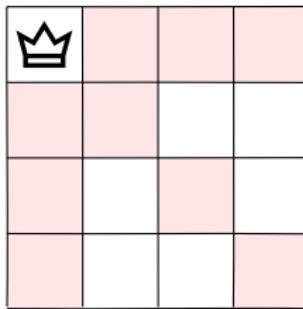
SOLUÇÃO X

A RAINHA 2 FOI POSICIONADA EM $(1, 1)$

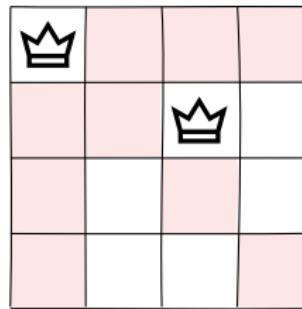
EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 1

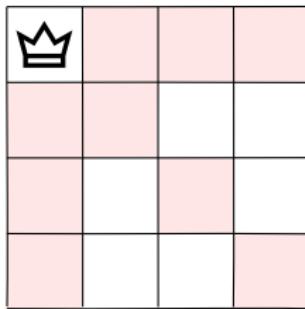


SOLUÇÃO 2

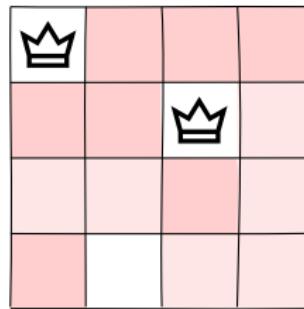
A RAINHA 2 FOI POSICIONADA EM (1, 2)

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 1



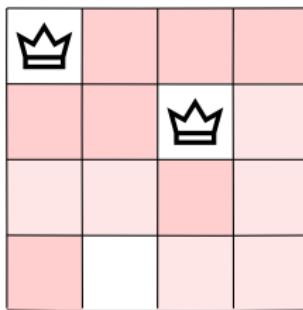
SOLUÇÃO 2

A RAINHA 2 FOI POSICIONADA EM (1, 2)

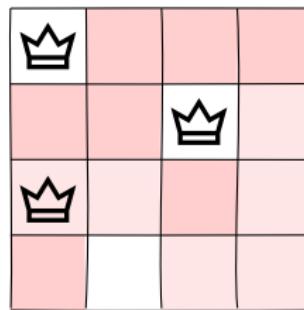
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2

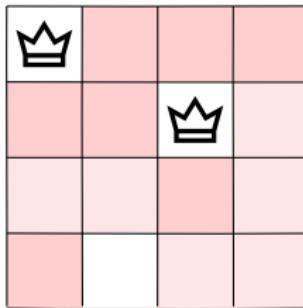


SOLUÇÃO X

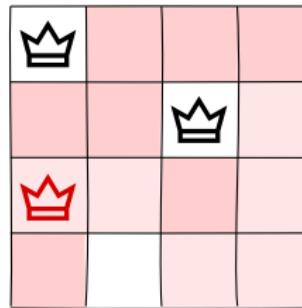
A RAINHA 3 FOI POSICIONADA EM $(2, \emptyset)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2



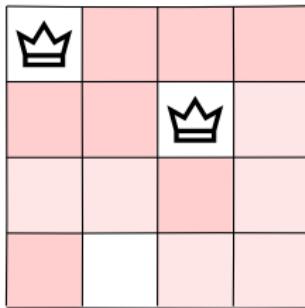
SOLUÇÃO X

A RAINHA 3 FOI POSICIONADA EM $(2, \emptyset)$

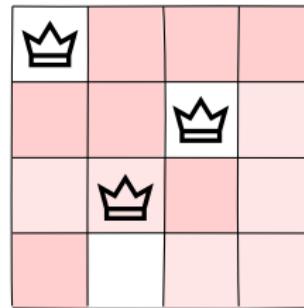
EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2

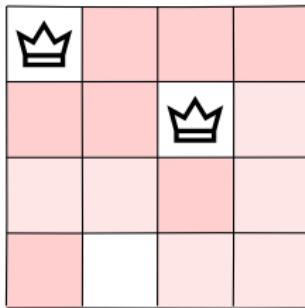


SOLUÇÃO X

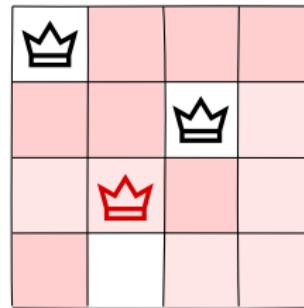
A RAINHA 3 FOI POSICIONADA EM $(2, 1)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2



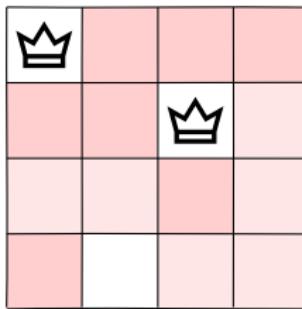
SOLUÇÃO X

A RAINHA 3 FOI POSICIONADA EM $(2, 1)$

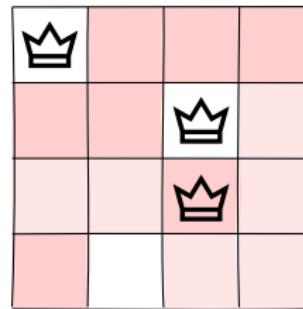
EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2

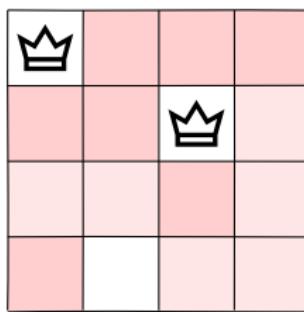


SOLUÇÃO X

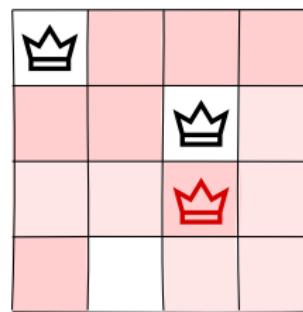
A RAINHA 3 FOI POSICIONADA EM (2, 2)

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2



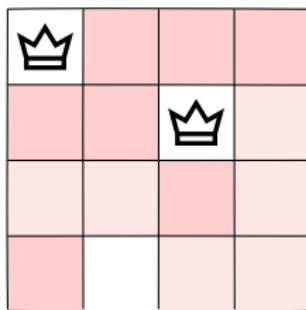
SOLUÇÃO X

A RAINHA 3 FOI POSICIONADA EM $(2, 2)$

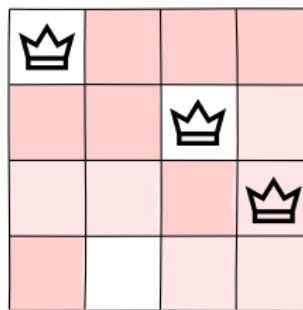
EXISTEM VIOLAÇÕES DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2

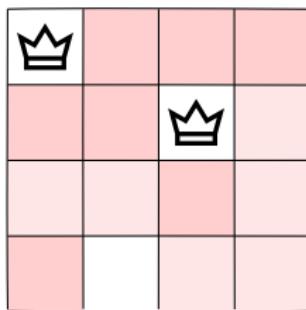


SOLUÇÃO X

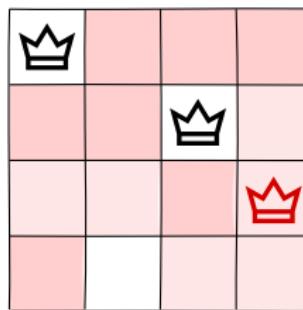
A RAINHA 3 FOI POSICIONADA EM (2, 3)

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2



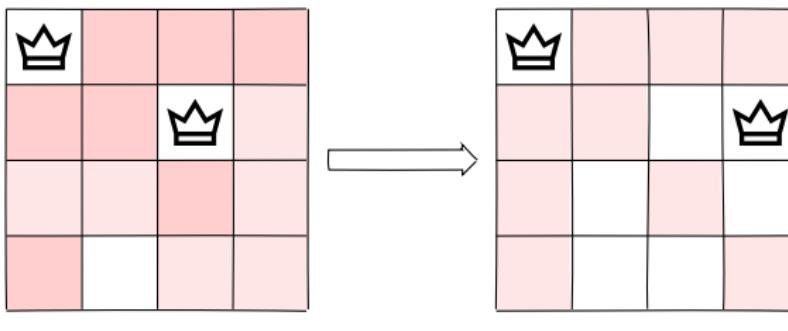
SOLUÇÃO X

A RAINHA 3 FOI POSICIONADA EM (2, 3)

EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

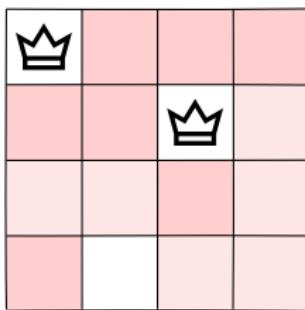
- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



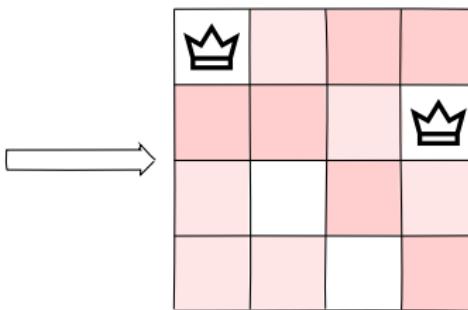
BACKTRACK: A RAINHA 2 FOI REPOSIIONADA EM $(1, 3)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 2



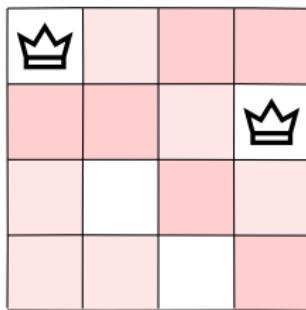
SOLUÇÃO 3

BACKTRACK: A RAINHA 2 FOI REPOSIIONADA EM $(1, 3)$

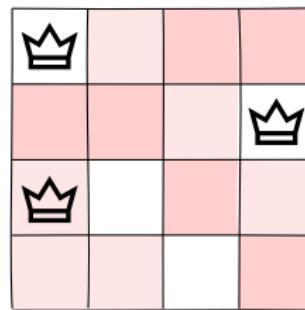
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 3

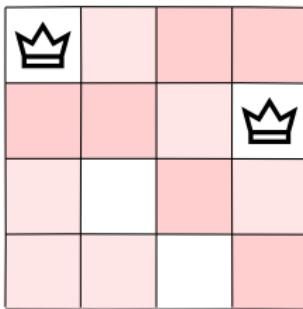


SOLUÇÃO X

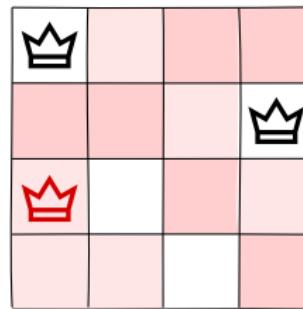
A RAINHA 3 FOI POSICIONADA EM $(2, \emptyset)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 3



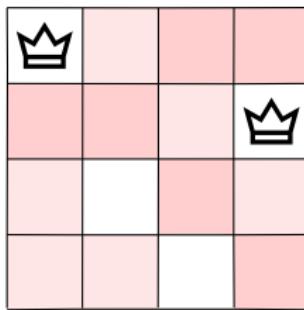
SOLUÇÃO X

A RAINHA 3 FOI POSICIONADA EM $(2, \emptyset)$

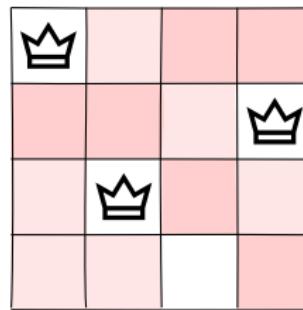
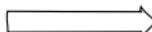
EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 3

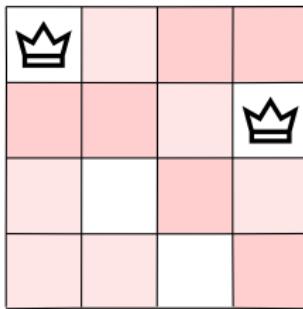


SOLUÇÃO 4

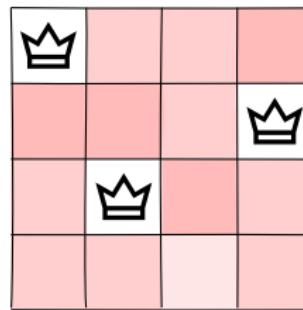
A RAINHA 3 FOI POSICIONADA EM $(2, 1)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 3



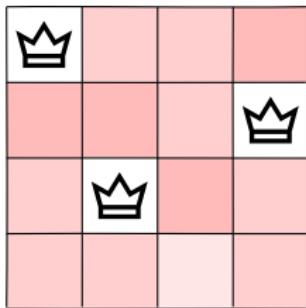
SOLUÇÃO 4

A RAINHA 3 FOI POSICIONADA EM $(2, 1)$

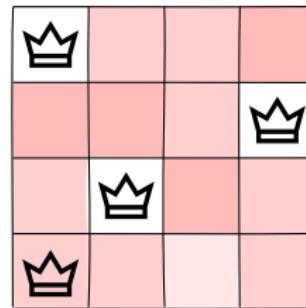
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4

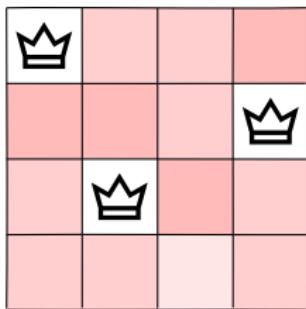


SOLUÇÃO X

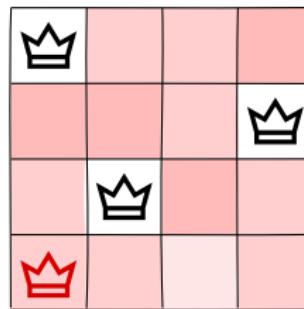
A RAINHA 4 FOI POSICIONADA EM $(3, \emptyset)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4



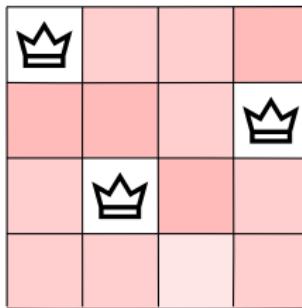
SOLUÇÃO X

A RAINHA 4 FOI POSICIONADA EM $(3, \emptyset)$

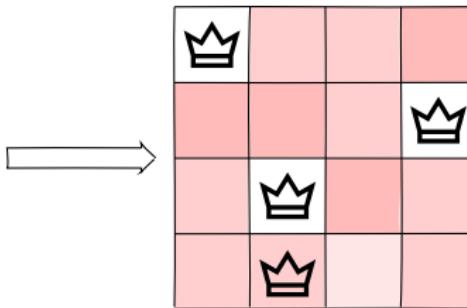
EXISTEM VIOLAÇÕES DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4

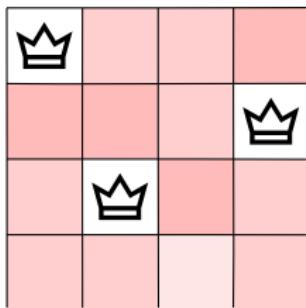


SOLUÇÃO X

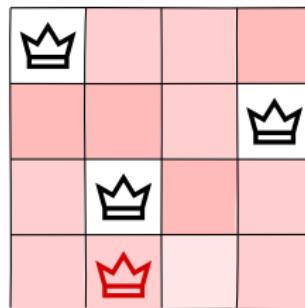
A RAINHA 4 FOI POSICIONADA EM $(3, 1)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4



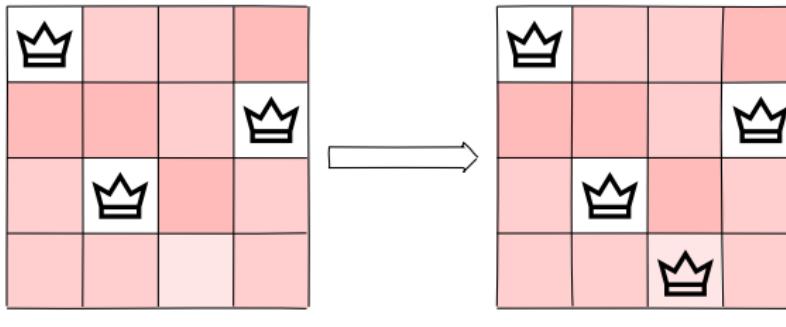
SOLUÇÃO X

A RAINHA 4 FOI POSICIONADA EM $(3, 1)$

EXISTEM VIOLAÇÕES DE ATAQUE

Backtracking

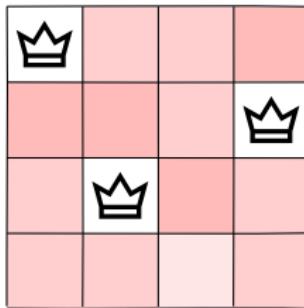
- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



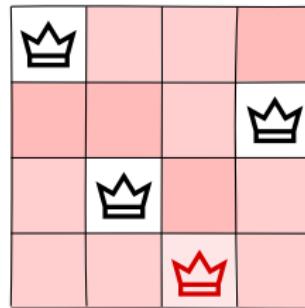
A RAINHA 4 FOI POSICIONADA EM $(3, 2)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4



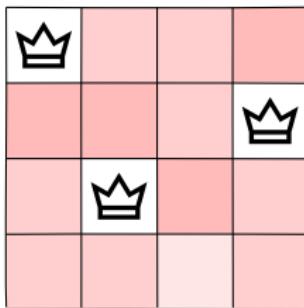
SOLUÇÃO X

A RAINHA 4 FOI POSICIONADA EM $(3, 2)$

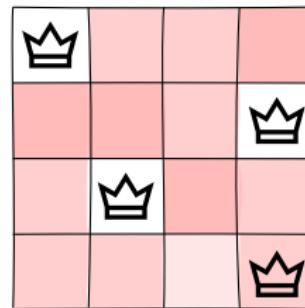
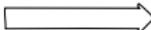
EXISTE VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4

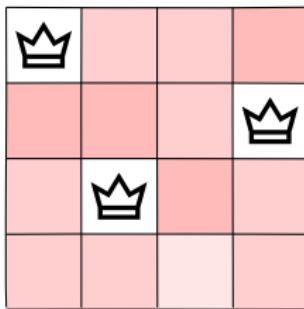


SOLUÇÃO X

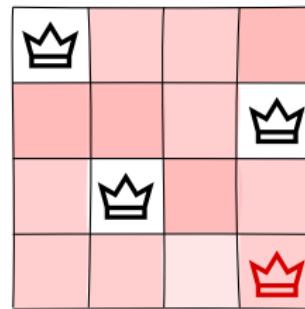
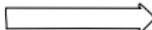
A RAINHA 4 FOI POSICIONADA EM $(3, 3)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4



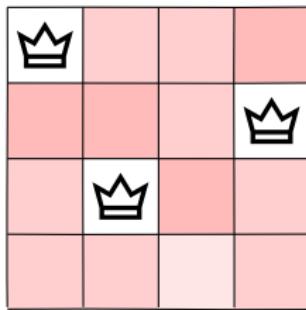
SOLUÇÃO X

A RAINHA 4 FOI POSICIONADA EM $(3, 3)$

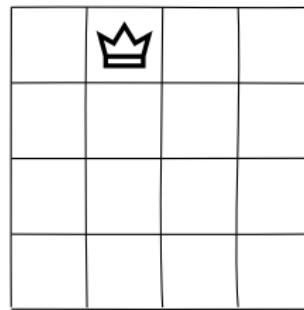
EXISTEM VIOLAÇÕES DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4

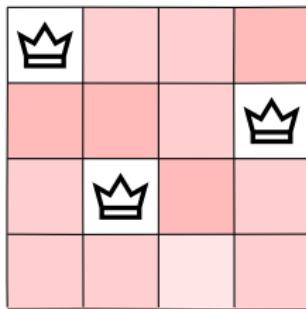


SOLUÇÃO 5

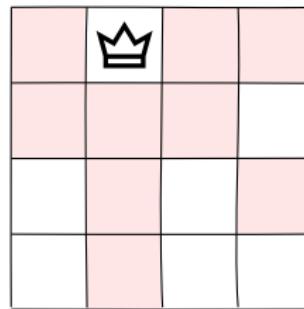
BACKTRACK: A RAINHA 1 FOI REPOSICIONADA EM $(\emptyset, 1)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 4



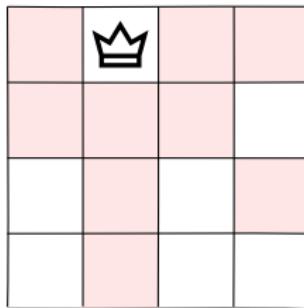
SOLUÇÃO 5

BACKTRACK: A RAINHA 1 FOI REPOSIIONADA EM $(\emptyset, 1)$

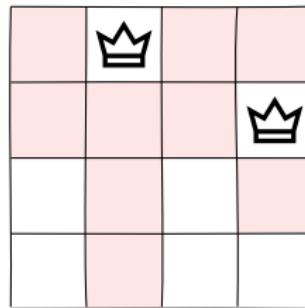
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 5

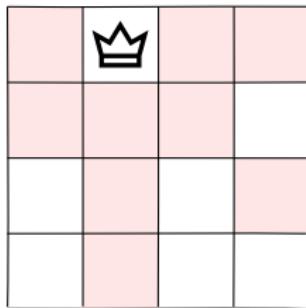


SOLUÇÃO 6

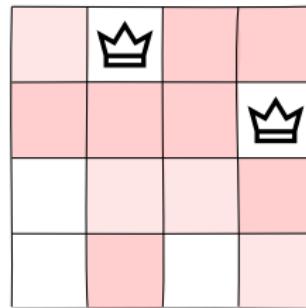
A RAINHA 2 FOI POSICIONADA EM $(1, 3)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 5



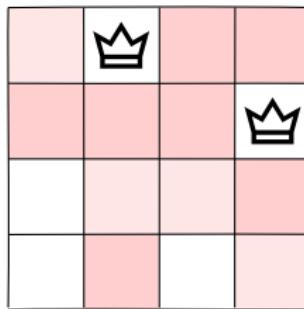
SOLUÇÃO 6

A RAINHA 2 FOI POSICIONADA EM $(1, 3)$

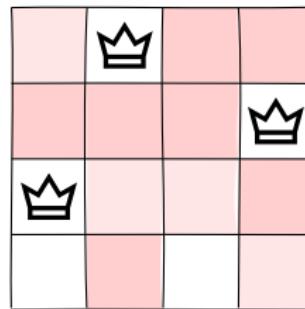
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 6

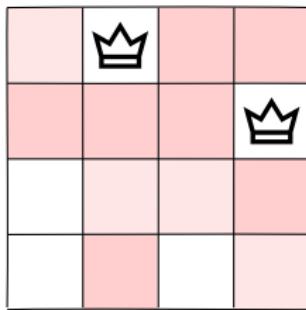


SOLUÇÃO 7

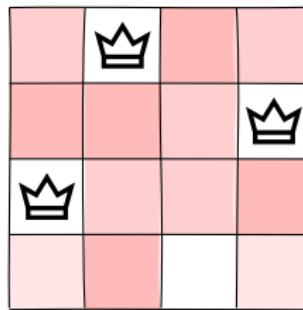
A RAINHA 3 FOI POSICIONADA EM $(2, \emptyset)$

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 6



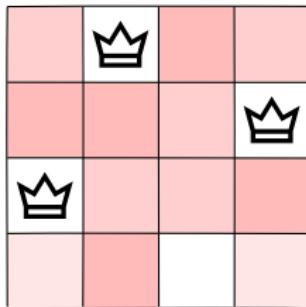
SOLUÇÃO 7

A RAINHA 3 FOI POSICIONADA EM $(2, \emptyset)$

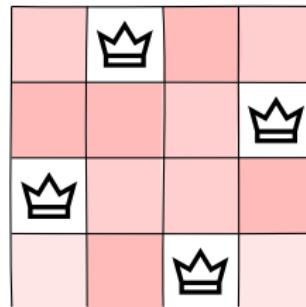
SEM VIOLAÇÃO DE ATAQUE

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 7

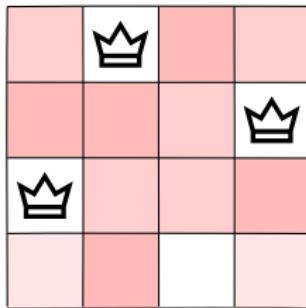


SOLUÇÃO 8

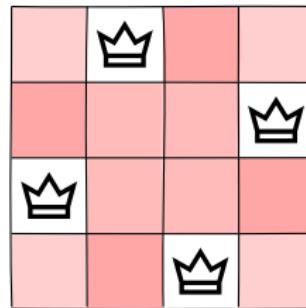
A RAINHA 4 FOI POSICIONADA EM (3, 2)

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



SOLUÇÃO 7



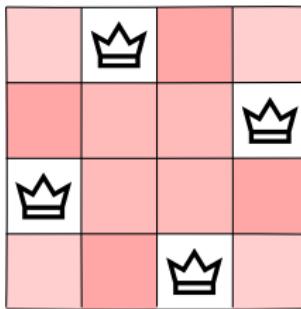
SOLUÇÃO 8

A RAINHA 4 FOI POSICIONADA EM (3, 2)

SEM VIOLAÇÃO DE ATAQUE

Backtracking

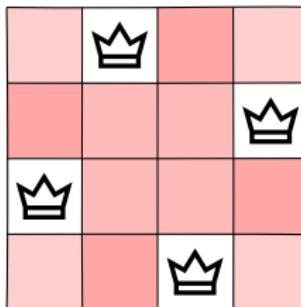
- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



EM UMA BUSCA EXAUSTRIVA, SERIAM NECESSÁRIAS
QUE TODAS AS $N^N = 4^4 = 256$ SOLUÇÕES FOSSEM AVALIADAS
PARA VERIFICAR QUAIS ATENDEM ÀS REGRAS DO PROBLEMA

Backtracking

- ▶ Problema das 4-rainhas: em um tabuleiro 4×4 , cada rainha é posicionada em linhas distintas



APLICANDO A TÉCNICA DE BACKTRACKING,
FORAM EXPLORADAS APENAS 8 SOLUÇÕES
(CERCA DE 3% DO ESPAÇO TOTAL)

Branch-and-bound

- ▶ O que é *branch-and-bound*?
 - ▶ *Branch* = desviar + *bound* = limitar

Branch-and-bound

- ▶ O que é *branch-and-bound*?
 - ▶ *Branch* = desviar + *bound* = limitar
 - ▶ Em problemas de otimização, as soluções geradas procuram minimizar ou maximizar alguma métrica do problema, atendendo as restrições do problema

Branch-and-bound

- ▶ O que é *branch-and-bound*?
 - ▶ *Branch* = desviar + *bound* = limitar
 - ▶ Em problemas de otimização, as soluções geradas procuram minimizar ou maximizar alguma métrica do problema, atendendo as restrições do problema
 - ▶ As soluções parciais geradas são avaliadas e os cenários inválidos são descartados

Branch-and-bound

- ▶ O que é *branch-and-bound*?
 - ▶ *Branch* = desviar + *bound* = limitar
 - ▶ Em problemas de otimização, as soluções geradas procuram minimizar ou maximizar alguma métrica do problema, atendendo as restrições do problema
 - ▶ As soluções parciais geradas são avaliadas e os cenários inválidos são descartados
 - ▶ O valor da melhor solução obtida até o momento armazenada, para que seja verificado se as próximas soluções geradas são melhores

Branch-and-bound

- ▶ Redução do espaço de soluções
 - ▶ Esta técnica procura deduzir quais caminhos não irão conduzir o algoritmo para encontrar uma solução

Branch-and-bound

- ▶ Redução do espaço de soluções
 - ▶ Esta técnica procura deduzir quais caminhos não irão conduzir o algoritmo para encontrar uma solução
 - ▶ Não existe garantia de se obter o melhor resultado possível (ótimo), entretanto, a solução gerada atende a todas as restrições definidas pelo problema

Branch-and-bound

- ▶ Redução do espaço de soluções
 - ▶ Esta técnica procura deduzir quais caminhos não irão conduzir o algoritmo para encontrar uma solução
 - ▶ Não existe garantia de se obter o melhor resultado possível (ótimo), entretanto, a solução gerada atende a todas as restrições definidas pelo problema

DEFINIÇÃO DE LIMITANTES
INFERIORES OU SUPERIORES

Branch-and-bound

- ▶ Redução do espaço de soluções
 - ▶ Esta técnica procura deduzir quais caminhos não irão conduzir o algoritmo para encontrar uma solução
 - ▶ Não existe garantia de se obter o melhor resultado possível (ótimo), entretanto, a solução gerada atende a todas as restrições definidas pelo problema

DEFINIÇÃO DE LIMITANTES
INFERIORES OU SUPERIORES

+

ARMAZENAMENTO DO VALOR
DA MELHOR SOLUÇÃO OBTIDA

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ Cada pessoa pode realizar somente um trabalho
 - ▶ A matriz armazena o valor que cada pessoa (linhas) recebe para realizar um trabalho (colunas)

$$\left[\begin{array}{cccc} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{array} \right]$$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ Cada pessoa pode realizar somente um trabalho
 - ▶ A matriz armazena o valor que cada pessoa (linhas) recebe para realizar um trabalho (colunas)

	T1	T2	T3	T4
P1	↓	↓	↓	↓
P2	→ 9	2	7	8
P3	→ 6	4	3	7
P4	→ 5	8	1	8
	→ 7	6	9	4

MINIMIZAÇÃO DO CUSTO TOTAL
PARA REALIZAÇÃO DOS TRABALHOS

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ Como parte da estratégia de minimização, o menor custo de cada pessoa (linha) para um trabalho (coluna) é selecionado para obter o limite inferior

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

NO CÁLCULO DESTE LIMITANTE, UMA PESSOA PODE REALIZAR MÚLTIPLOS TRABALHOS, POIS NÃO É BUSCADA UMA SOLUÇÃO

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ Como parte da estratégia de minimização, o menor custo de cada pessoa (linha) para um trabalho (coluna) é selecionado para obter o limite inferior

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

O LIMITE INFERIOR É $2 + 3 + 1 + 4 = 10$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_1

	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

↓ ↓ ↓ ↓

$\begin{array}{l} - \\ 2+3+1+4=10 \end{array}$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_1

	T1	T2	T3	T4	
P1	9	2	7	8	
P2	6	4	3	7	
P3	5	8	1	8	
P4	7	6	9	4	

Diagram illustrating the branch-and-bound process for the assignment problem:

- A bracket groups the columns T1 through T4.
- An arrow points from the value 9 in the P1-T1 cell to a box containing $P_1 \leftrightarrow T_1$ and $9+3+1+4=17$.
- An arrow points from the sum $2+3+1+4=10$ in the top-right corner to the same box.

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_1

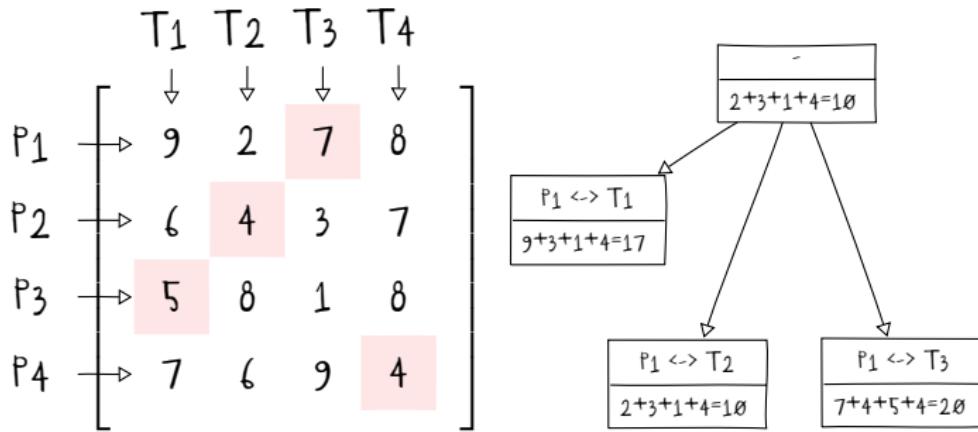
	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

Diagram illustrating the Branch-and-Bound process for the assignment problem:

- The matrix shows the cost of assigning person P_i to task T_j .
- Arrows point from each row P_i to its corresponding column T_j .
- Cells containing values 2, 3, 1, and 4 are highlighted in red.
- A bracket groups the columns T_1, T_2, T_3, T_4 .
- Three boxes represent the search space at different stages:
 - Top box: $-$, $2+3+1+4=10$
 - Middle box: $P_1 \leftrightarrow T_1$, $9+3+1+4=17$
 - Bottom box: $P_1 \leftrightarrow T_2$, $2+3+1+4=10$Arrows show the transition from the top box to the middle one, and from the middle one to the bottom one.

Branch-and-bound

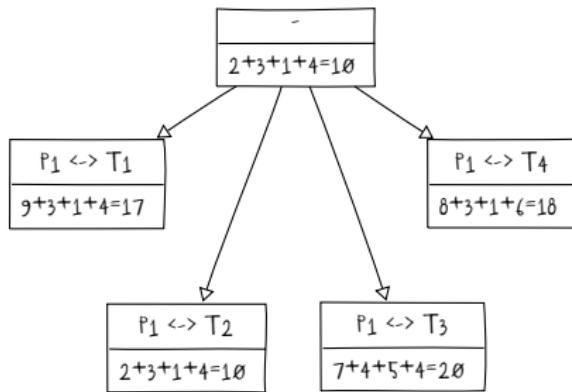
- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_1



Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_1

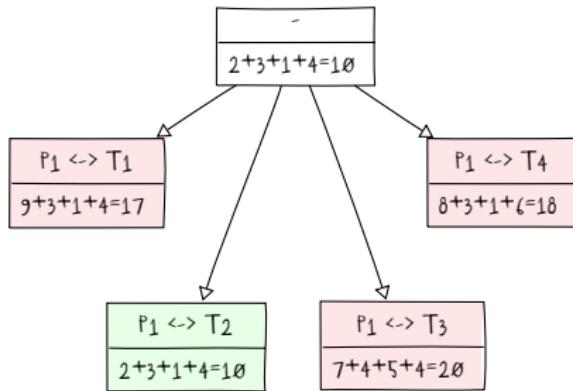
	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4



Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa $P1$

	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4



A SOLUÇÃO PARCIAL MAIS PROMISSORA
ALOCA A PESSOA $P1$ PARA O TRABALHO $T2$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_2

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

↓ ↓ ↓ ↓

$P_1 \leftrightarrow T_2$
 $2+3+1+4=10$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_2

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

Diagram illustrating the branch-and-bound search for the assignment problem:

- The matrix shows the cost of assigning person P_i to task T_j .
- Row P_1 : Costs [9, 2, 7, 8]. The cell (P_1, T_2) is highlighted green.
- Row P_2 : Costs [6, 4, 3, 7]. The cell (P_2, T_1) is highlighted red.
- Row P_3 : Costs [5, 8, 1, 8]. The cell (P_3, T_3) is highlighted red.
- Row P_4 : Costs [7, 6, 9, 4]. The cell (P_4, T_4) is highlighted red.

Two boxes show partial assignments and their total costs:

- A box above the matrix contains $P_1 \leftrightarrow T_2$ and $2+3+1+4=10$.
- A box below the matrix contains $P_2 \leftrightarrow T_1$ and $2+6+1+4=13$.

An arrow points from the bottom box to the red cell (P_2, T_1) in the matrix.

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_2

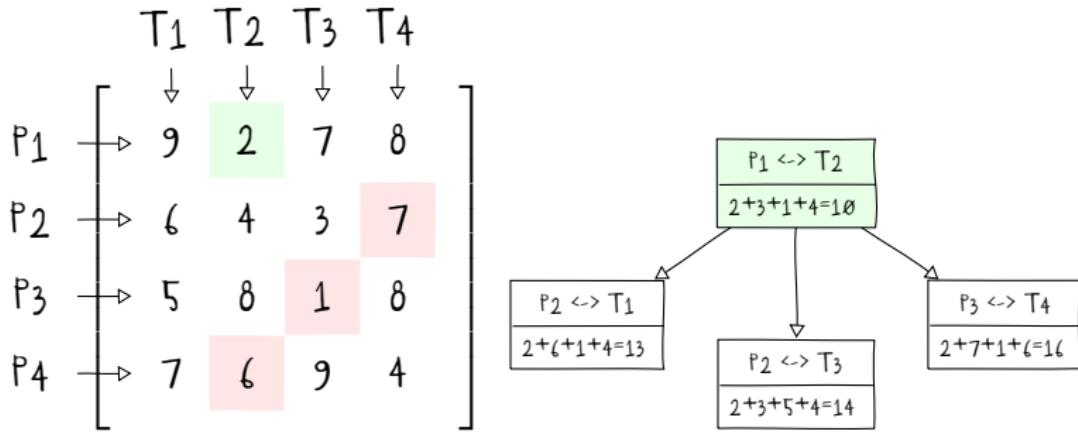
	T ₁	T ₂	T ₃	T ₄
P ₁	9	2	7	8
P ₂	6	4	3	7
P ₃	5	8	1	8
P ₄	7	6	9	4

Diagram illustrating the Branch-and-Bound process for the second person (P₂) allocation:

- The current state is shown in the table above.
- The green box highlights the allocation for P₂ to T₂ (value 2).
- Three boxes represent the branches from this state:
 - Top box: P₁ ↔ T₂, 2 + 3 + 1 + 4 = 10
 - Middle box: P₂ ↔ T₁, 2 + 6 + 1 + 4 = 13
 - Bottom box: P₂ ↔ T₃, 2 + 3 + 5 + 4 = 14
- Arrows indicate the search space: an upward arrow from the middle box to the top box, and a downward arrow from the top box to the bottom box.

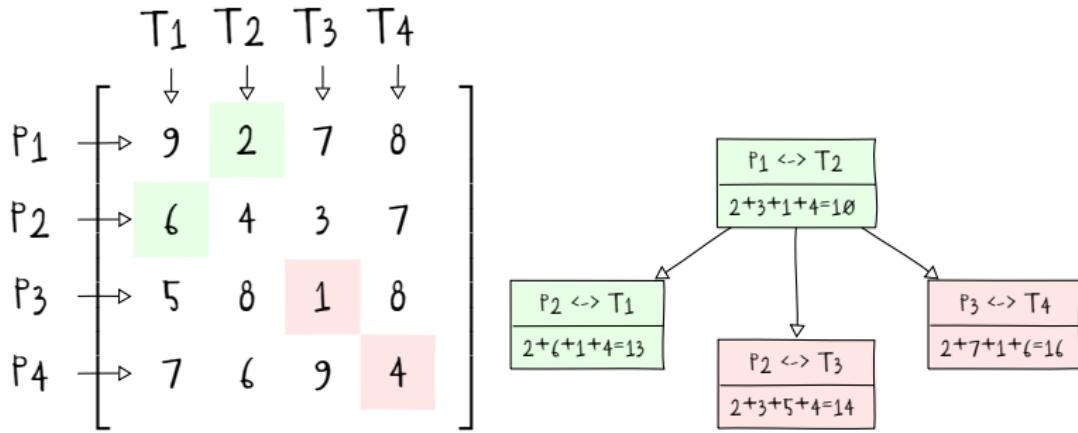
Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa $P2$



Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_2



A SOLUÇÃO PARCIAL MAIS PROMISSORA
ALOCA A PESSOA P₂ PARA O TRABALHO T₁

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_3

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

$P_1 \leftrightarrow T_2$
$P_2 \leftrightarrow T_1$
$2+6+1+4=13$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_3

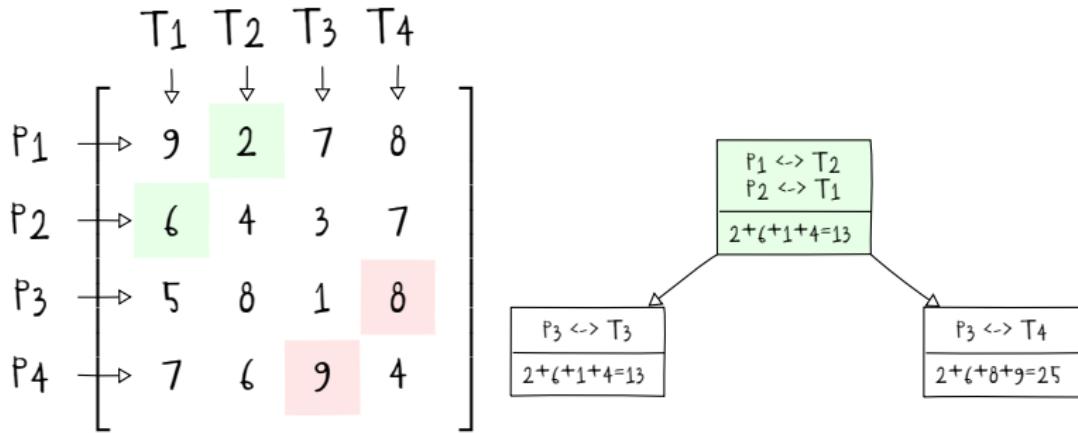
	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

Diagram illustrating the branch-and-bound process for the assignment problem:

- The matrix shows the cost of assigning person P_i to task T_j . The best assignment found so far is $P_1 \leftrightarrow T_2$ and $P_2 \leftrightarrow T_1$, with a total cost of 13.
- An arrow points from the assignment $P_3 \leftrightarrow T_3$ to its corresponding cost calculation box.
- A bracket on the right side groups the last three rows of the matrix.

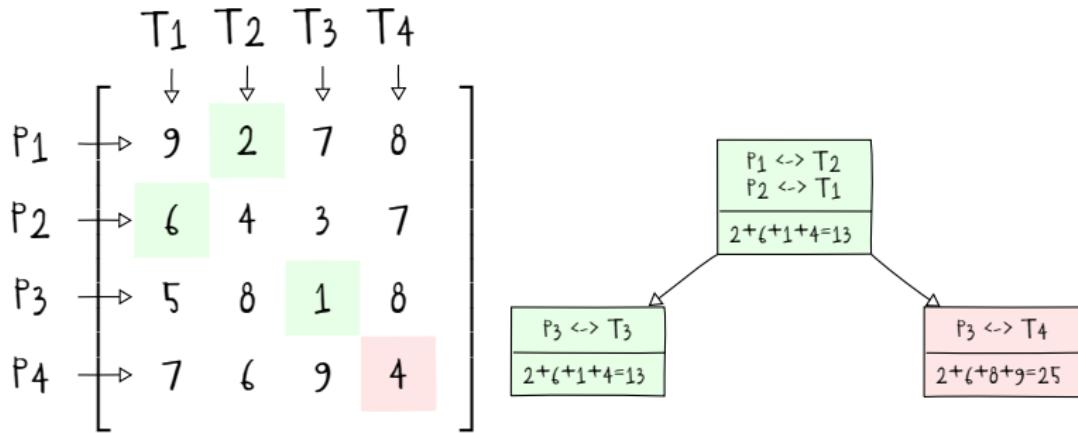
Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_3



Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_3



A SOLUÇÃO PARCIAL MAIS PROMISSORA
ALOCA A PESSOA P_3 PARA O TRABALHO T_3

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_4

	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

$P_1 \leftrightarrow T_2$
 $P_2 \leftrightarrow T_1$
 $P_3 \leftrightarrow T_3$

$$2+6+1+4=13$$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa P_4

	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

$P_1 \leftrightarrow T_2$
 $P_2 \leftrightarrow T_1$
 $P_3 \leftrightarrow T_3$

 $2+6+1+4=13$

$P_4 \leftrightarrow T_4$

 $2+6+1+4=13$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos
 - ▶ É feita a alocação da pessoa $P4$

	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

$P_1 \leftrightarrow T_2$
 $P_2 \leftrightarrow T_1$
 $P_3 \leftrightarrow T_3$

 $2+6+1+4=13$

$P_4 \leftrightarrow T_4$

 $2+6+1+4=13$

A SOLUÇÃO COMPLETA MAIS PROMISSORA
ALOCA A PESSOA $P4$ PARA O TRABALHO $T4$

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos

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

$P_1 \leftrightarrow T_2$
 $P_2 \leftrightarrow T_1$
 $P_3 \leftrightarrow T_3$
 $P_4 \leftrightarrow T_4$

$2 + 6 + 1 + 4 = 13$

NA BUSCA EXAUATIVA, EM UM CENÁRIO DE PIOR CASO,
SERIAM GERADAS ATÉ $N! = 4! = 24$ SOLUÇÕES VÁLIDAS

Branch-and-bound

- ▶ Problema de alocar n pessoas para n trabalhos

	T1	T2	T3	T4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

Diagram illustrating the Branch-and-Bound search space for the assignment problem. The matrix shows costs for assigning people P1-P4 to tasks T1-T4. The solution path is highlighted in green boxes:

- P1 is assigned to T2 (cost 2)
- P2 is assigned to T1 (cost 6)
- P3 is assigned to T3 (cost 1)
- P4 is assigned to T4 (cost 4)

The total cost of the solution is 13.

LIMITANDO O ESPAÇO DE BUSCA COM BRANCH-AND-BOUND
É GERADA APENAS 1 SOLUÇÃO (CERCA DE 4% DAS SOLUÇÕES)

Branch-and-bound

- ▶ Problema da mochila com *branch-and-bound*
 - ▶ O limitante superior é definido pela utilização da capacidade $W = 6$ que maximiza o valor total
 - ▶ É feito o cálculo da relação entre o valor e o peso $\frac{v_i}{w_i}$ de cada item de forma ordenada na tabela

i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10

Branch-and-bound

- ▶ Problema da mochila com *branch-and-bound*
 - ▶ O limitante superior é definido pela utilização da capacidade $W = 6$ que maximiza o valor total
 - ▶ É feito o cálculo da relação entre o valor e o peso $\frac{v_i}{w_i}$ de cada item de forma ordenada na tabela

i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10

$$L = \max(W \times \frac{v_i}{w_i}) = 132$$

Branch-and-bound

- Problema da mochila com *branch-and-bound*

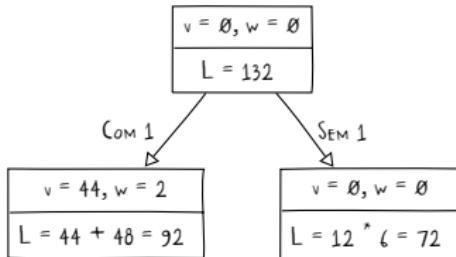
I	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10

$v = \emptyset, w = \emptyset$
$L = 132$

Branch-and-bound

- Problema da mochila com *branch-and-bound*

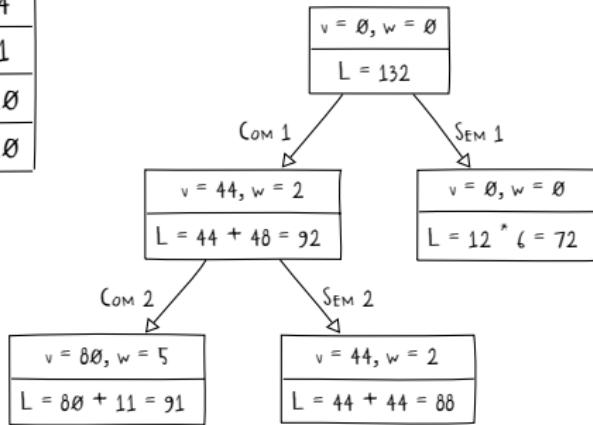
I	1	2	3	4
w _i	2	3	5	1
v _i	44	36	55	10
v _i /w _i	22	12	11	10



Branch-and-bound

- ▶ Problema da mochila com *branch-and-bound*

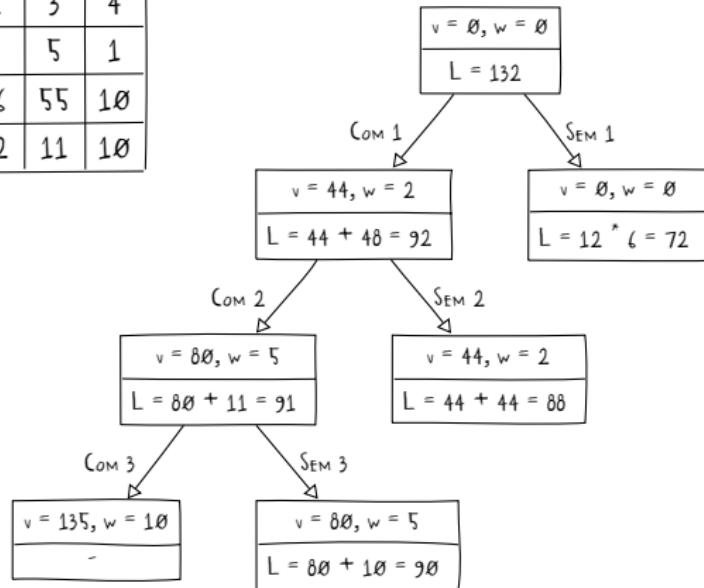
I	1	2	3	4
w _i	2	3	5	1
v _i	44	36	55	10
v _i /w _i	22	12	11	10



Branch-and-bound

► Problema da mochila com *branch-and-bound*

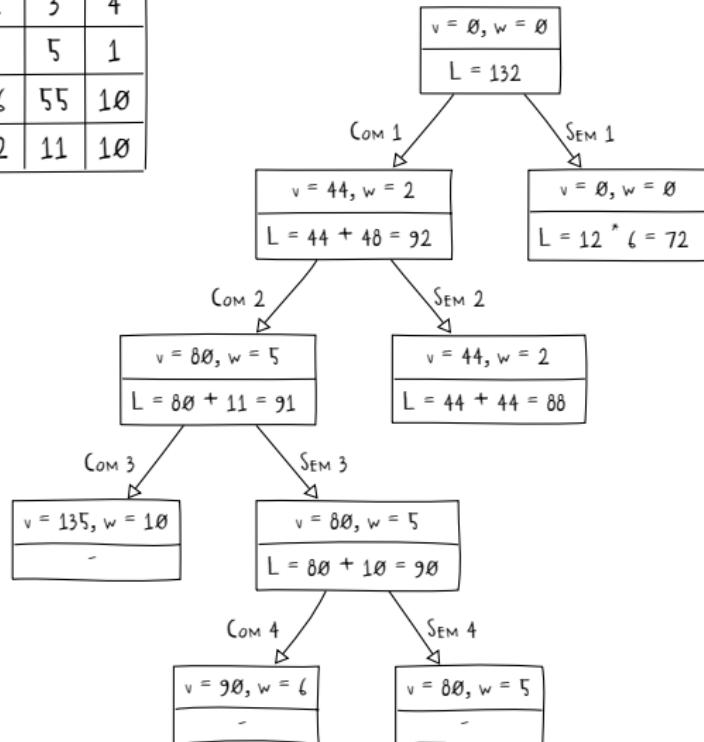
I	1	2	3	4
w _i	2	3	5	1
v _i	44	36	55	10
v _i /w _i	22	12	11	10



Branch-and-bound

► Problema da mochila com *branch-and-bound*

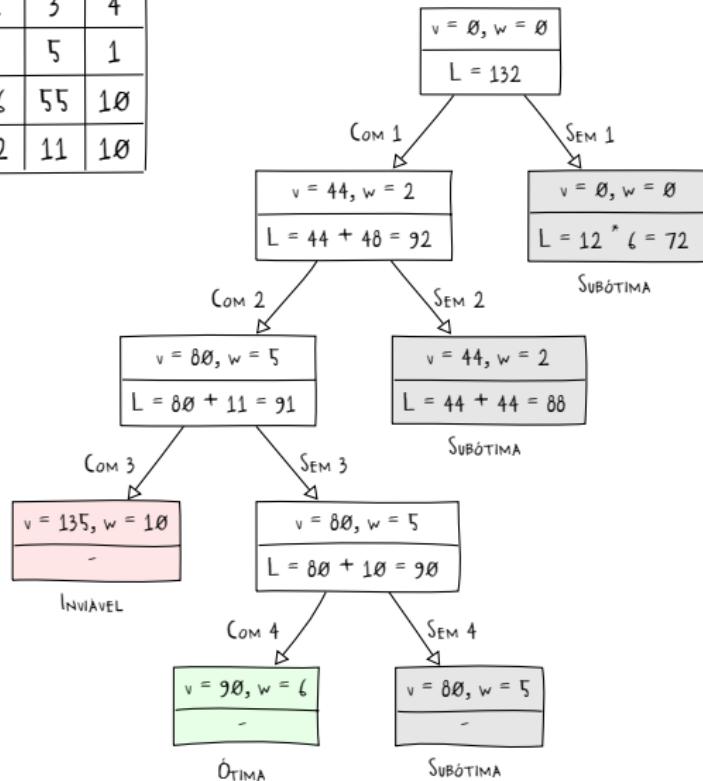
1	1	2	3	4
w _i	2	3	5	1
v _i	44	36	55	10
v _i /w _i	22	12	11	10



Branch-and-bound

► Problema da mochila com *branch-and-bound*

I	1	2	3	4
w _i	2	3	5	1
v _i	44	36	55	10
v _i /w _i	22	12	11	10



Exercício

- ▶ A empresa de tecnologia Poxim Tech está desenvolvendo um robô humanoide que é capaz de se deslocar de forma totalmente autônoma e sem precisar do conhecimento prévio do ambiente físico no qual está localizado
 - ▶ Durante o seu deslocamento, que é feito um passo por vez, podem ser realizadas as seguintes operações, listadas em ordem de prioridade
 - ▶ Direita (D)
 - ▶ Frente (F)
 - ▶ Esquerda (E)
 - ▶ Trás (T)

Exercício

- ▶ A medida que vai explorando o ambiente, o robô cria uma mapa interno para as rotas exploradas
 - ▶ Caso uma rota não gere uma solução, outro caminho deve ser escolhido para ser explorado até que a solução seja obtida ou que não existam mais opções, ou seja, a busca é finalizada no ponto de partida
 - ▶ Para demonstrar suas habilidades exploratórias, são criados labirintos com exatamente 1 entrada e até 1 saída, com tamanho máximo de 100 por 100 posições
 - ▶ É possível que nenhuma rota seja possível para atravessar o labirinto criado, mas se existir uma saída, é sempre um espaço na borda do labirinto que não é o ponto de partida

Exercício

- ▶ Formato do arquivo de entrada

- ▶ $\#NL$

- ▶ $[Largura] \times [Altura]$

- ▶ $M_{x,y} = 0$ (espaço), 1 (parede), X (partida)

$$M_{0,0} \quad \cdots \quad M_{0,L-1}$$

$$\vdots \quad \ddots \quad \vdots$$

$$M_{A-1,0} \quad \cdots \quad M_{A-1,L-1}$$

```
1 2
2 5 4
3 1 1 1 1 1
4 1 0 0 0 1
5 1 0 X 1 1
6 1 1 0 1 1
7 3 4
8 1 1 1
9 1 X 1
10 1 0 1
11 1 1 1
```

Exercício

- ▶ Formato do arquivo de saída
 - ▶ A rota é descrita pelas coordenadas visitadas

```
1 L0:INI@2,2|F->1,2|D->1,3|BT@1,3->1,2|E->1,1|T->2,1|BT@2  
    ,1->1,1|BT@1,1->1,2|BT@1,2->2,2|T->3,2|FIM@3,2  
2 L1:INI@1,1|T->2,1|BT@2,1->1,1|FIM@-, -
```