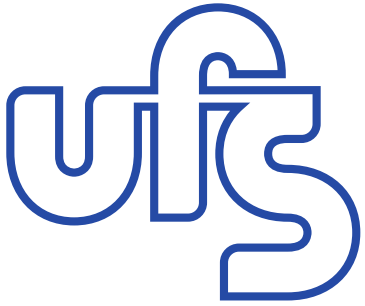
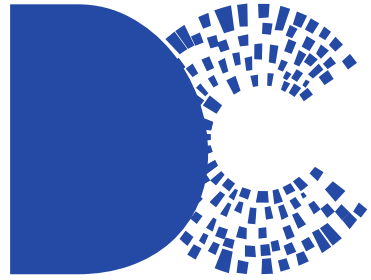




Backtracking e *branch-and-bound*

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

$$\textit{Busca exaustiva} \longleftrightarrow \textit{Espaço exponencial}$$

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

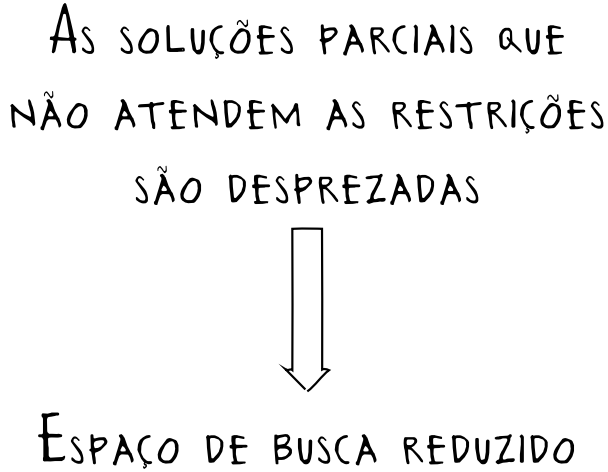
Backtracking

- 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

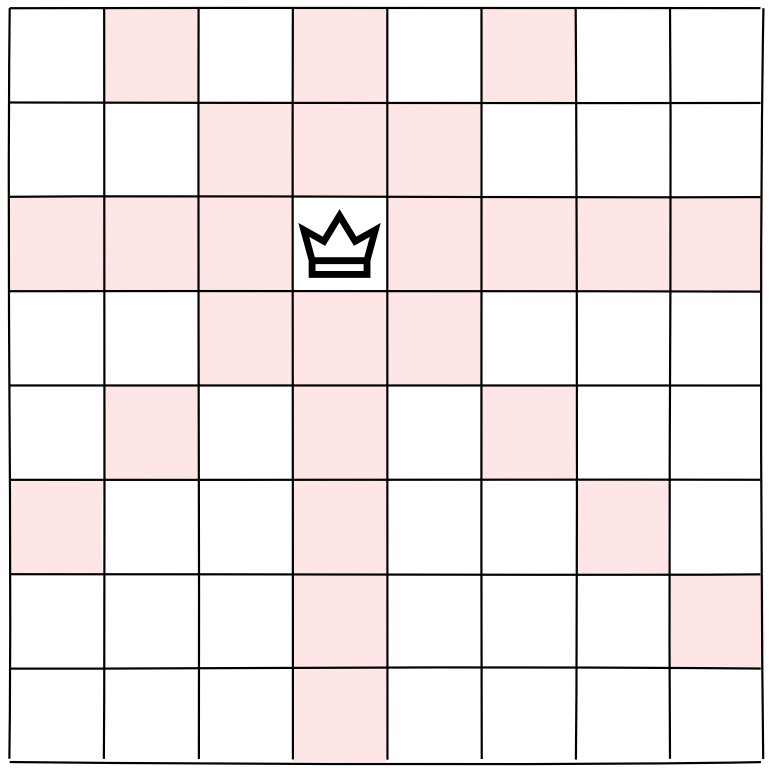
Backtracking

- 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



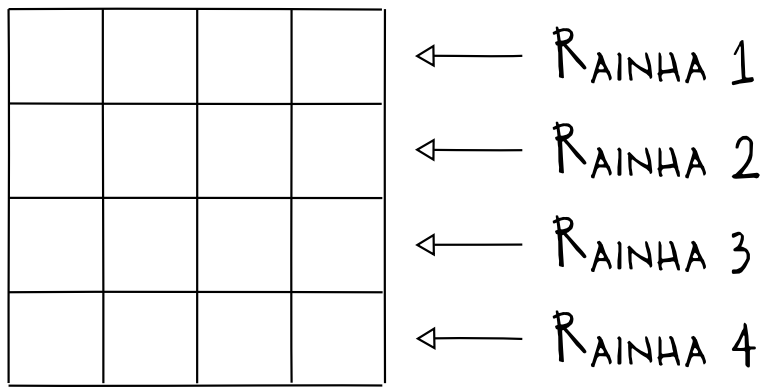
Backtracking

- 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



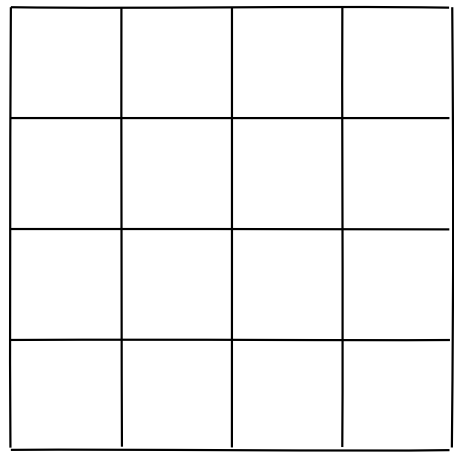
Backtracking

- 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

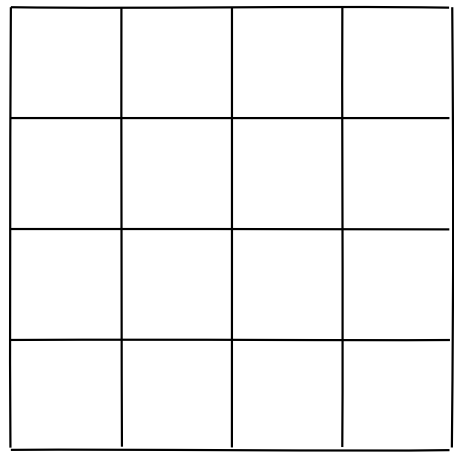


Solução Ø

Nenhuma rainha posicionada

Backtracking

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



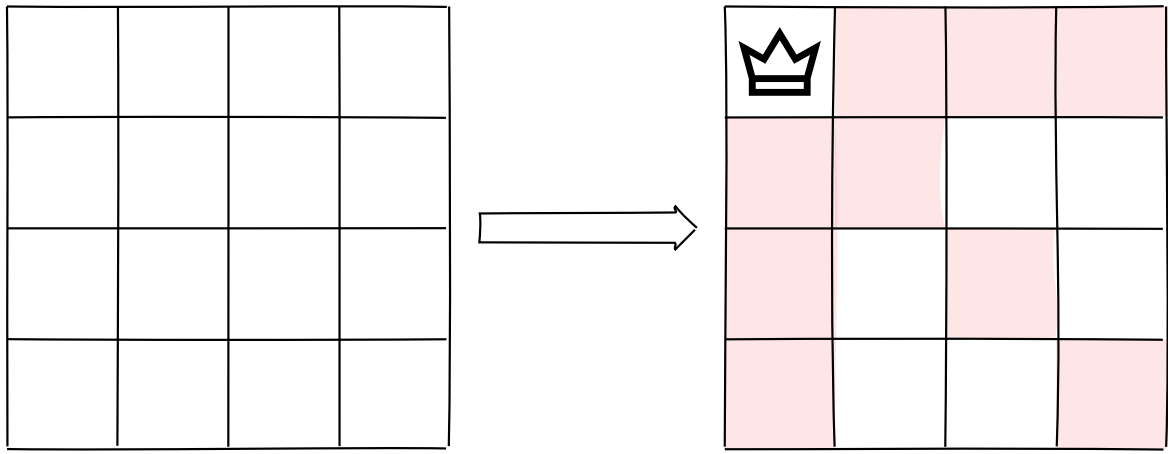
Solução Ø

Nenhuma rainha posicionada

Sem violação de ataque

Backtracking

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



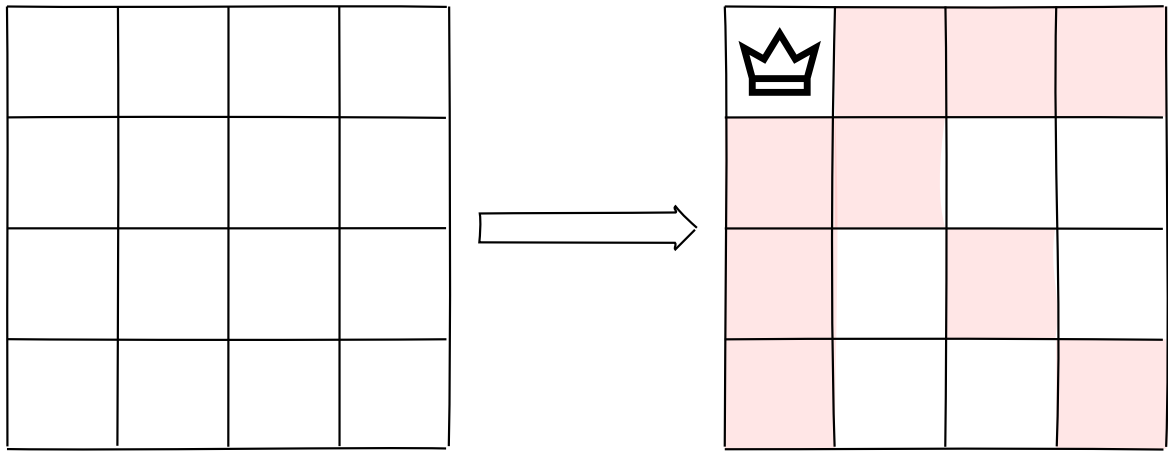
Solução 0

Solução 1

A rainha 1 foi posicionada em (0, 0)

Backtracking

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



Solução Ø

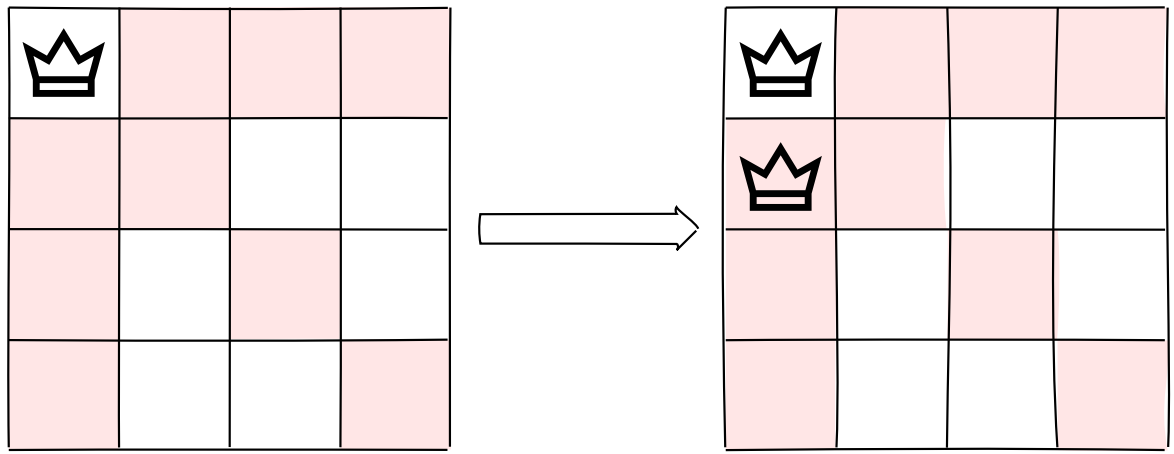
Solução 1

A rainha 1 foi posicionada em (Ø, Ø)

Sem violação de ataque

Backtracking

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



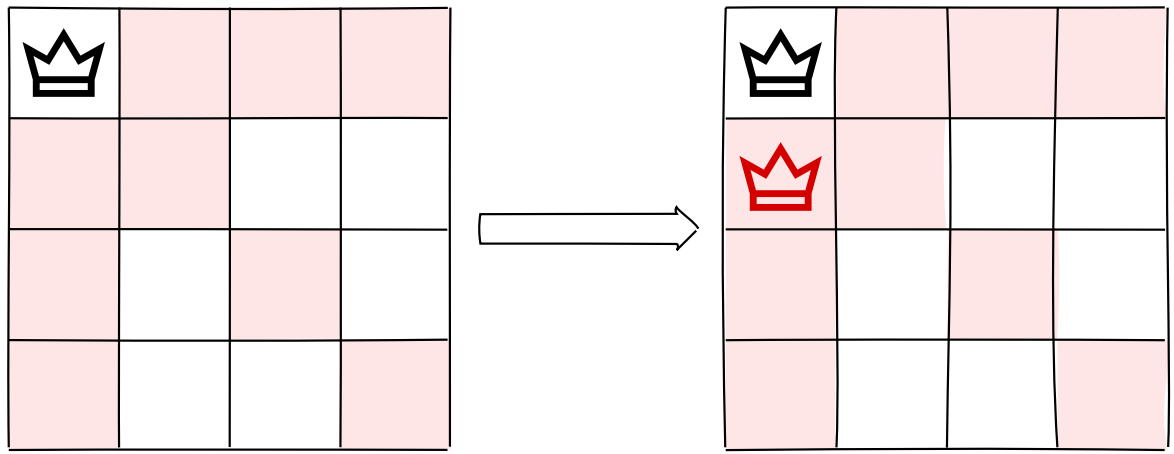
Solução 1

Solução X

A rainha 2 foi posicionada em (1, 0)

Backtracking

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



Solução 1

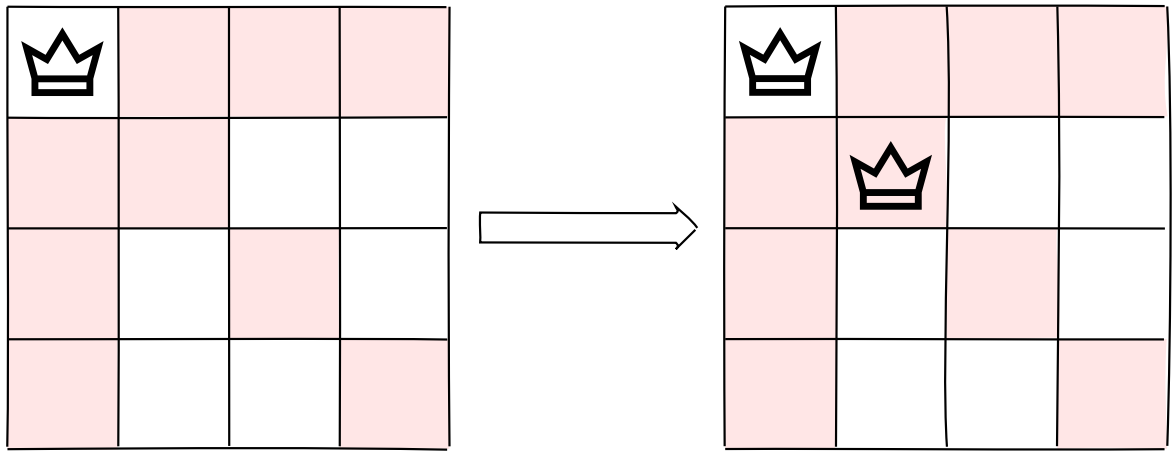
Solução X

A rainha 2 foi posicionada em (1, 0)

Existe violação de ataque

Backtracking

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



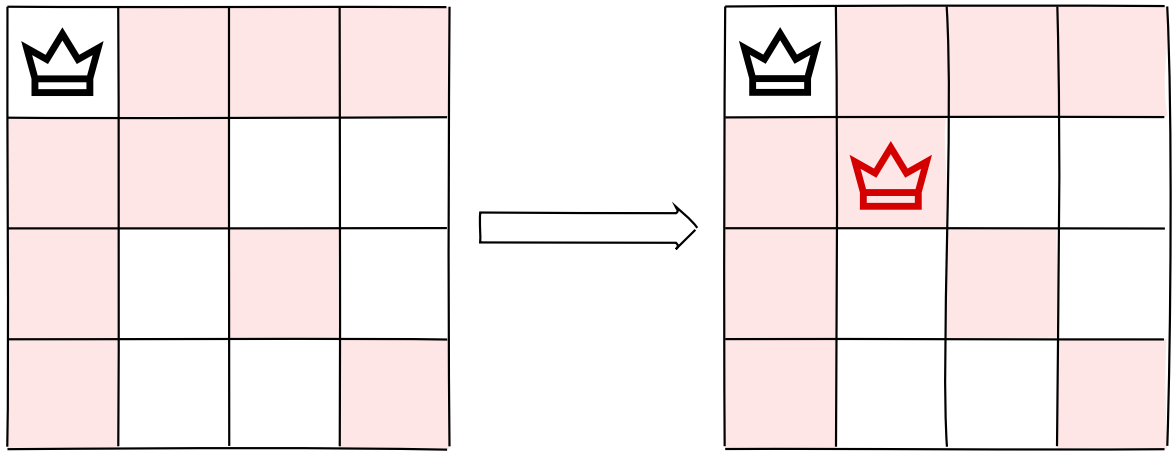
Solução 1

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

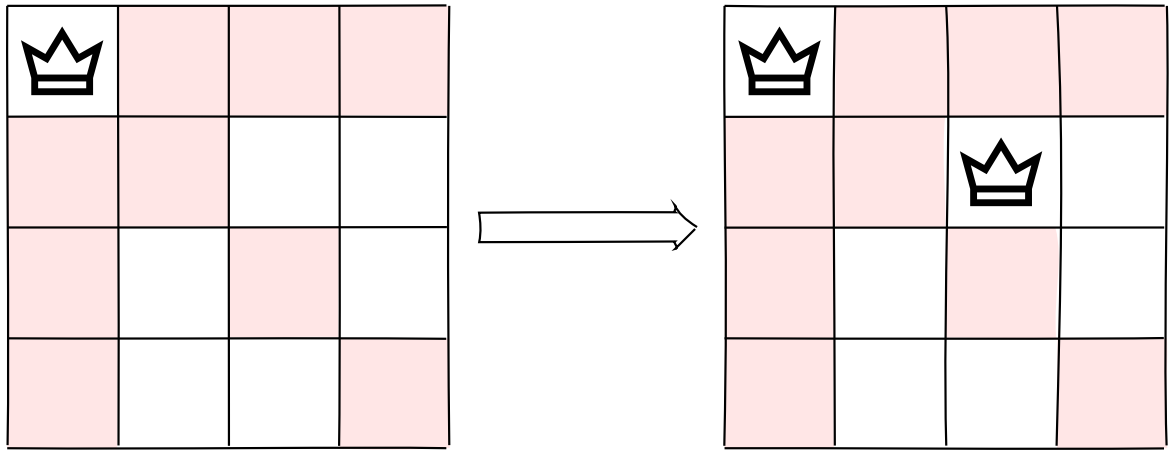
Solução X

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



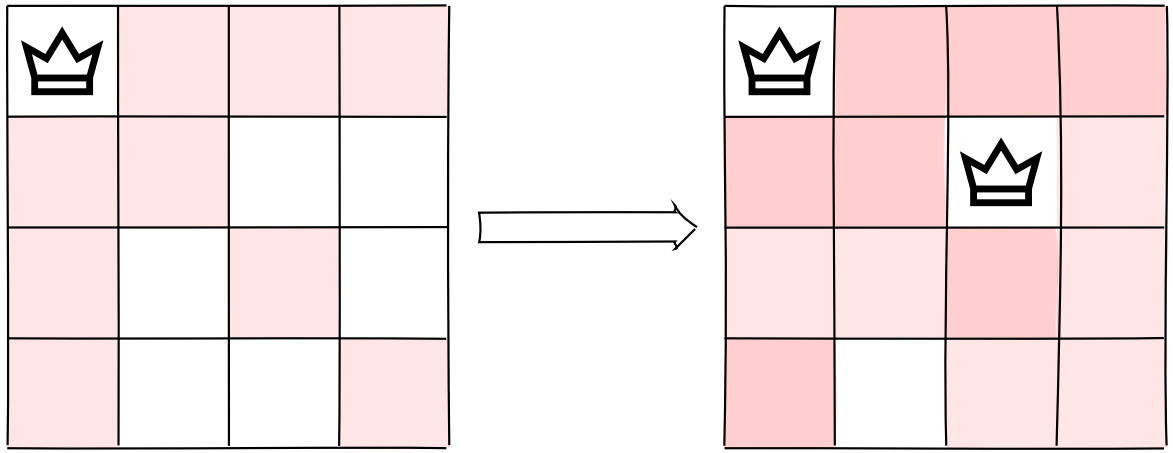
Solução 1

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

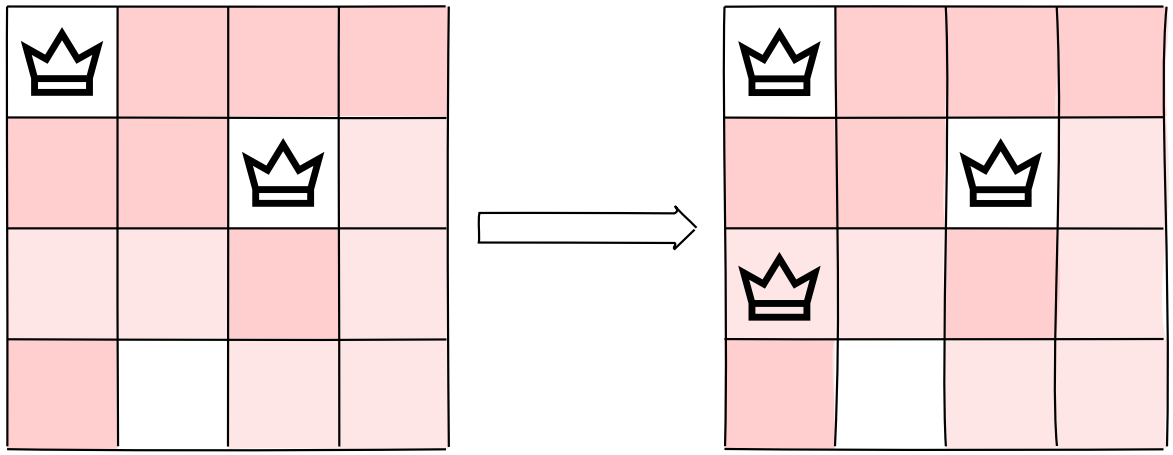
Solução 2

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



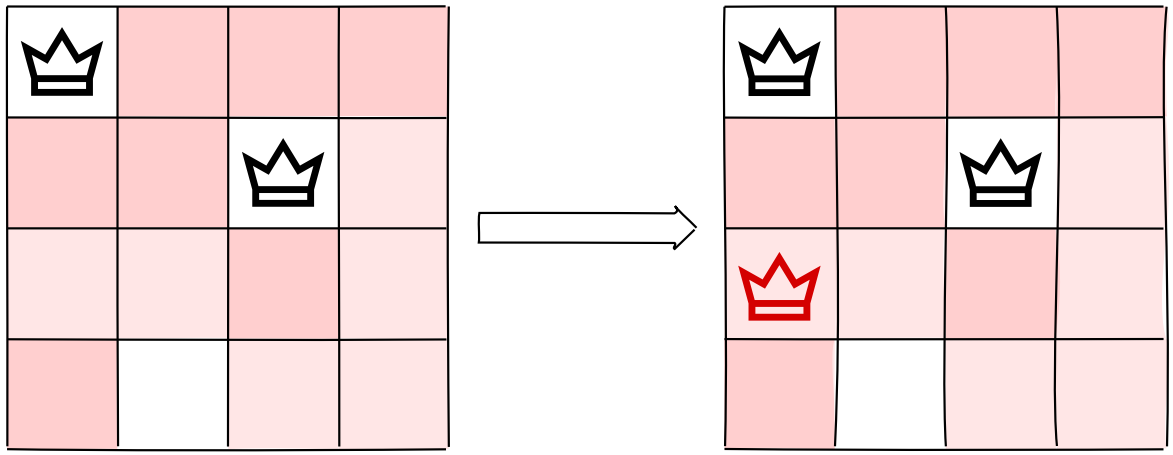
Solução 2

Solução X

A rainha 3 foi posicionada em (2, 0)

Backtracking

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



Solução 2

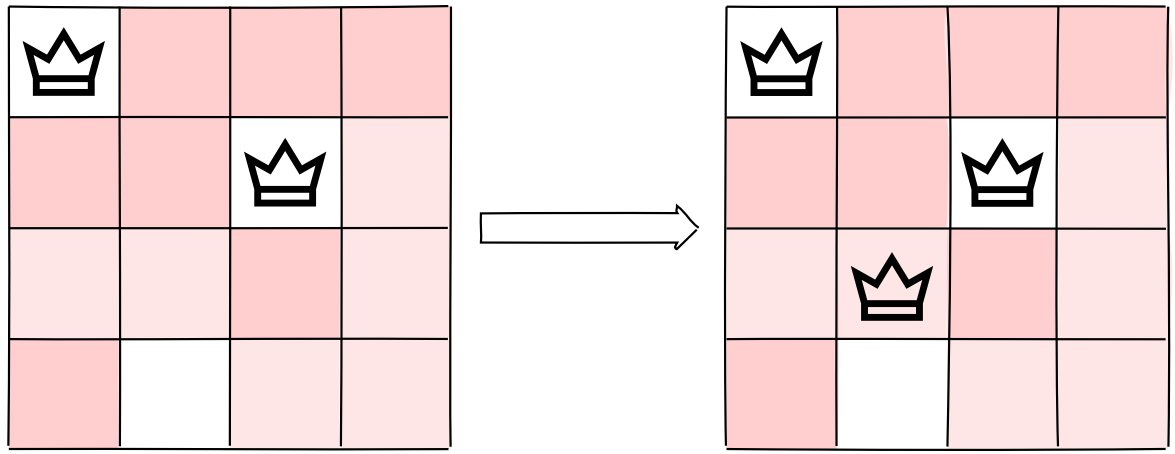
Solução X

A rainha 3 foi posicionada em (2, 0)

Existe violação de ataque

Backtracking

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



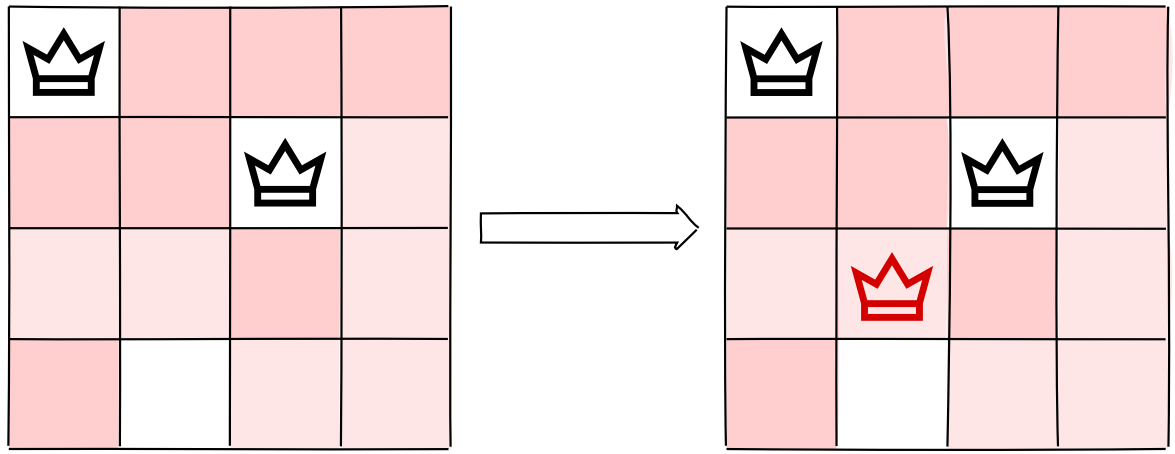
Solução 2

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

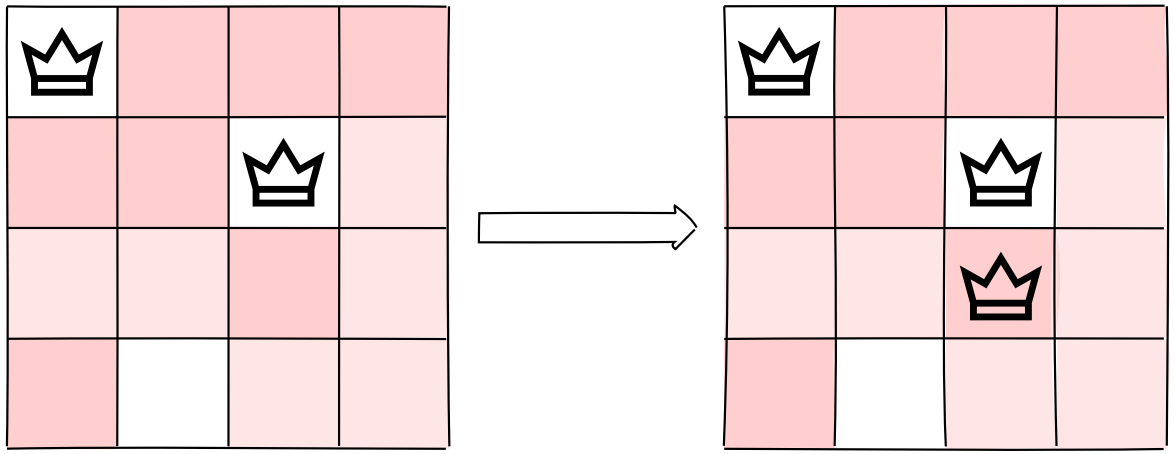
Solução X

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



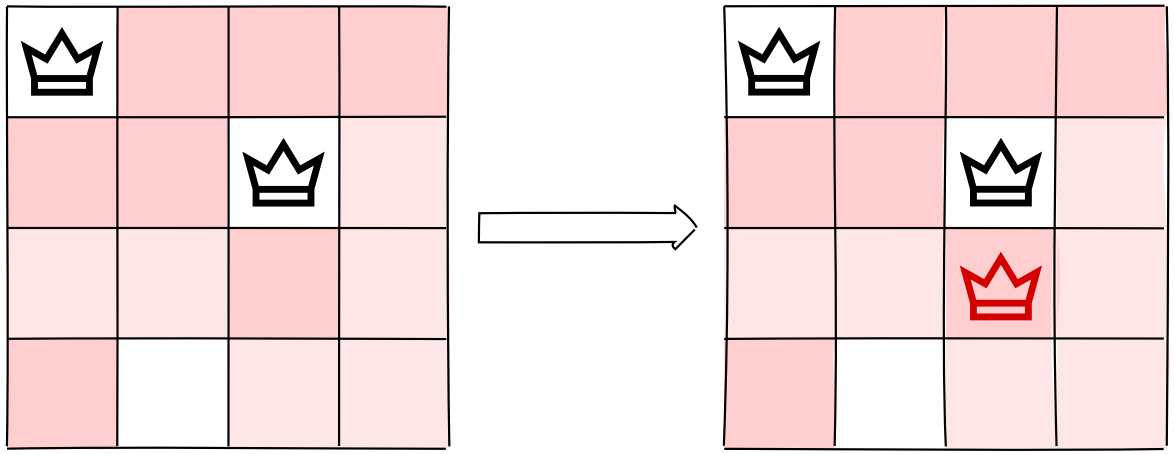
Solução 2

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

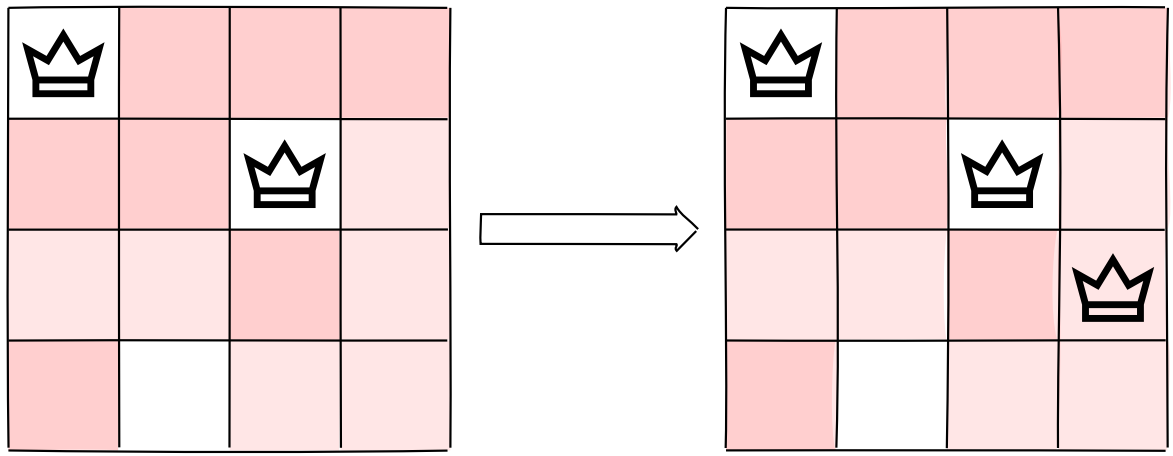
Solução X

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



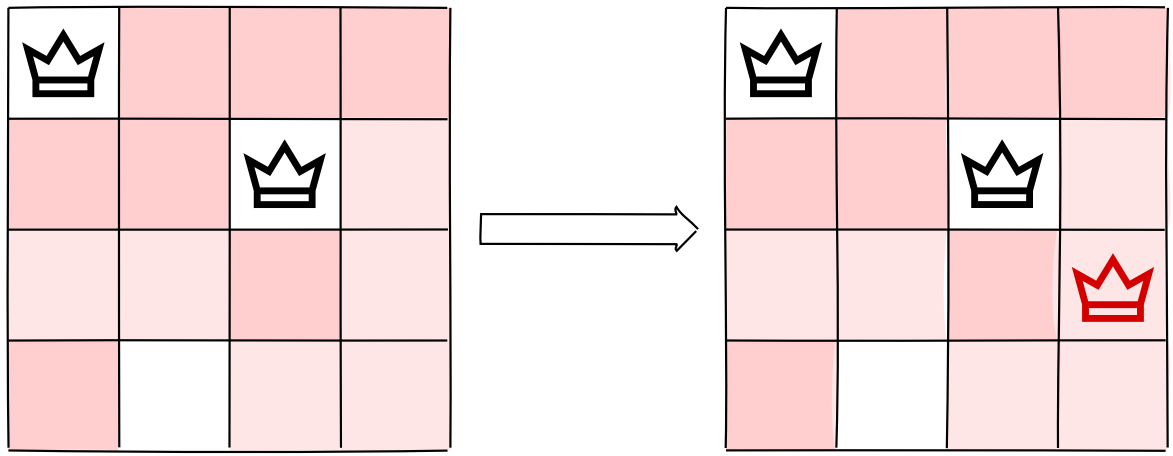
Solução 2

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

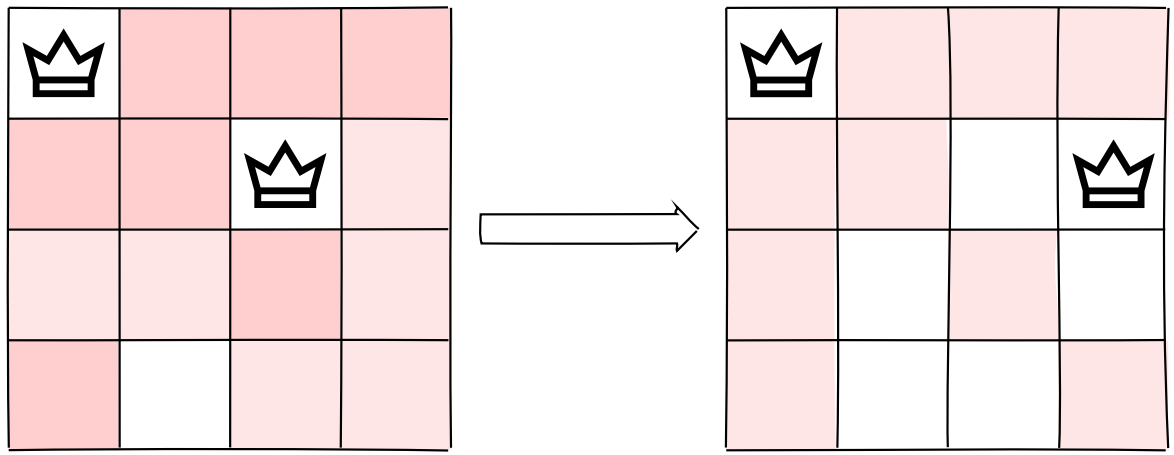
Solução X

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

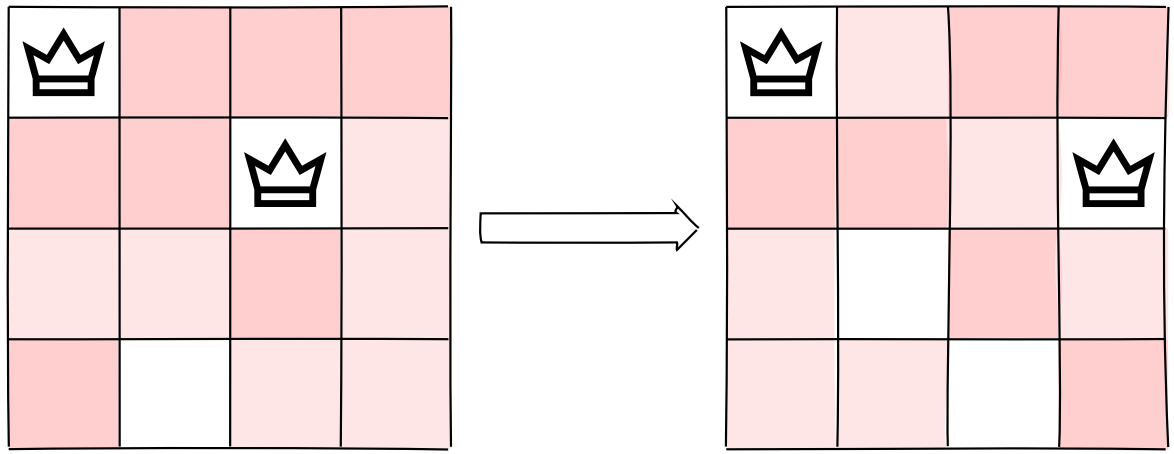


Solução 2 Solução 3

Backtrack: a rainha 2 foi reposicionada em (1, 3)

Backtracking

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



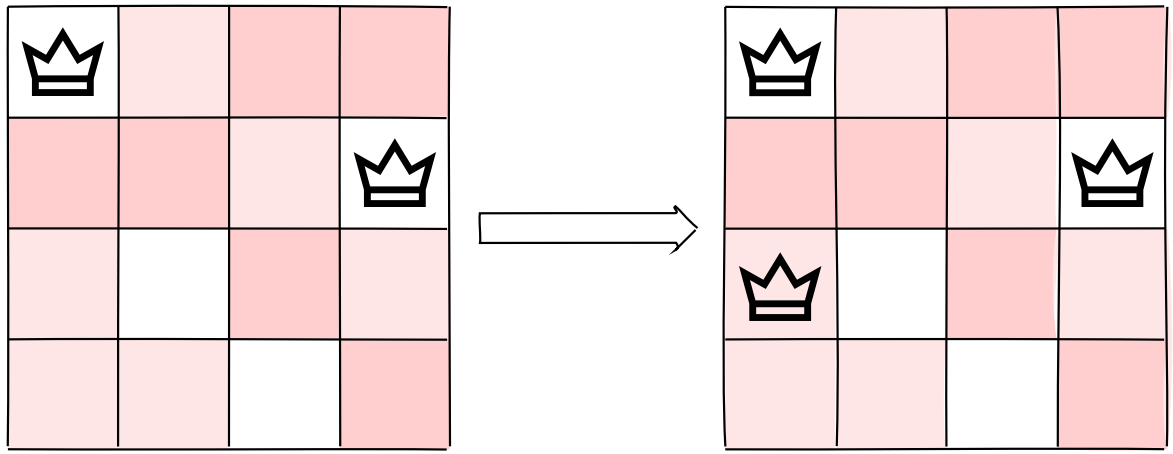
Solução 2 Solução 3

Backtrack: a rainha 2 foi reposicionada em (1, 3)

Sem violação de ataque

Backtracking

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



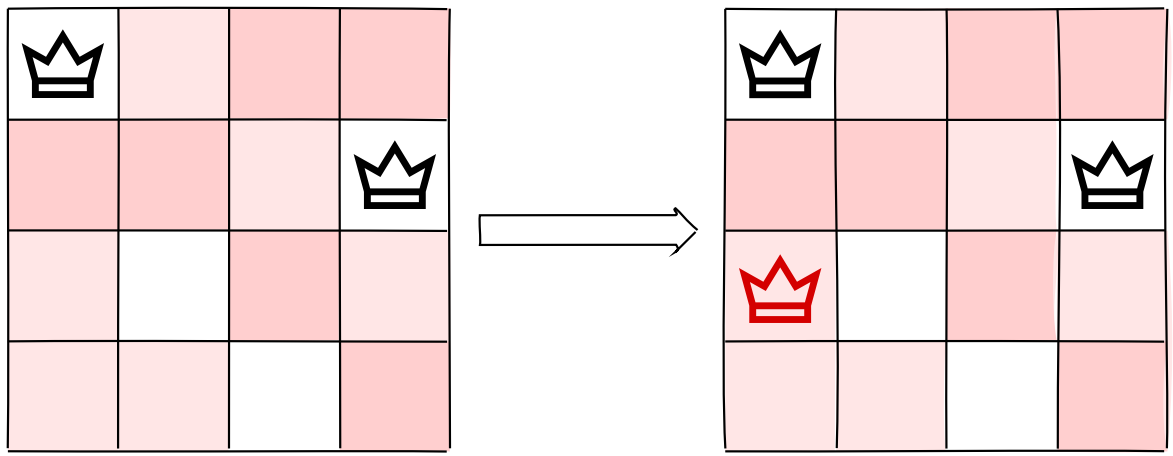
Solução 3

Solução X

A rainha 3 foi posicionada em (2, 0)

Backtracking

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



Solução 3

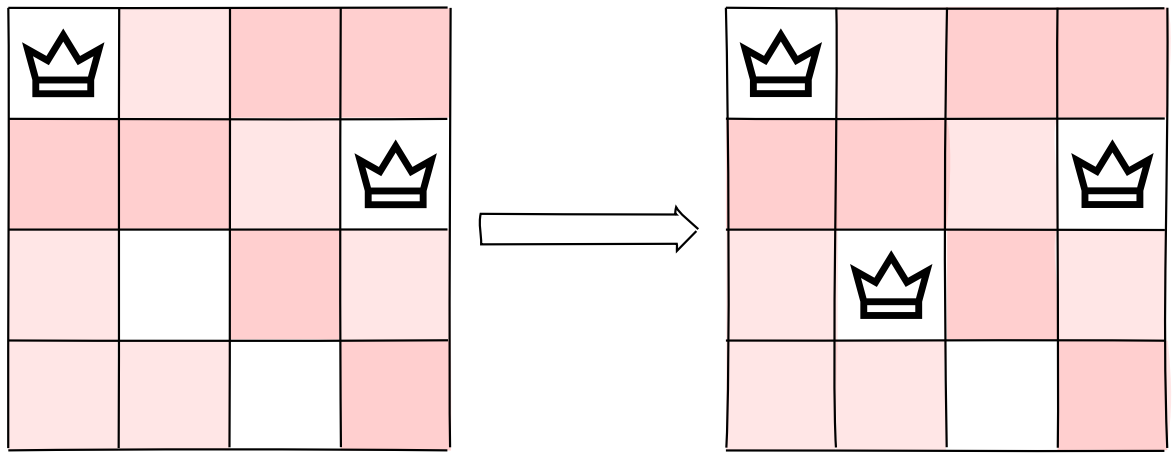
Solução X

A rainha 3 foi posicionada em (2, 0)

Existe violação de ataque

Backtracking

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

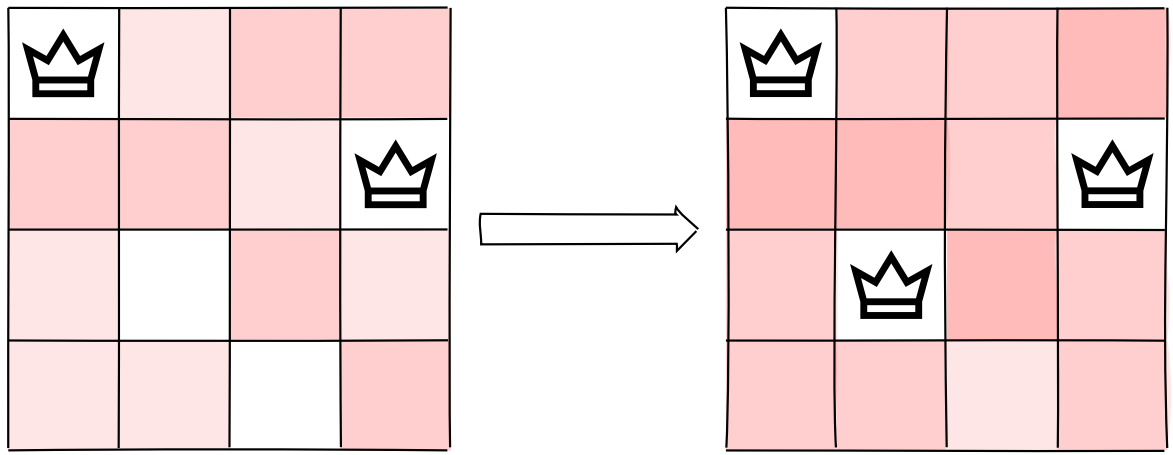


Solução 3 Solução 4

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



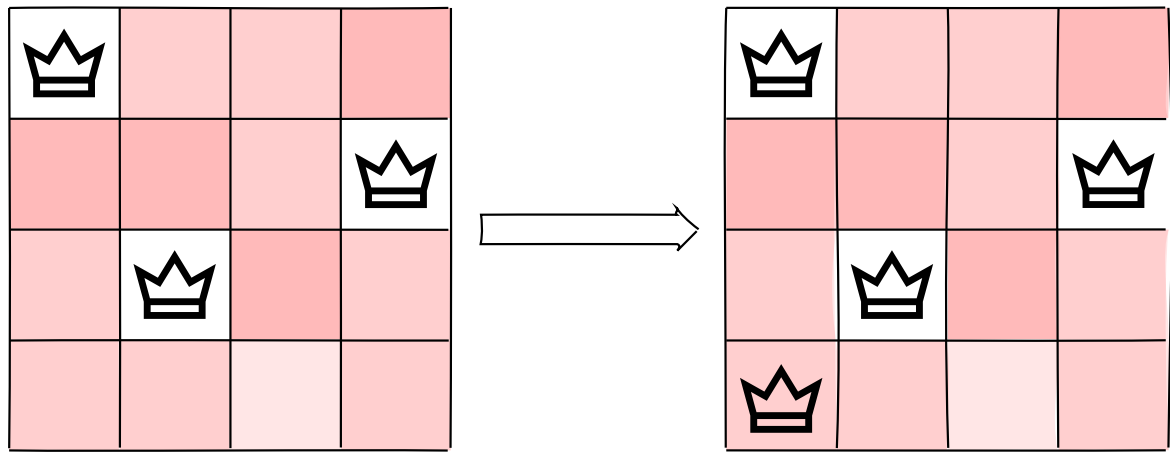
Solução 3 Solução 4

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



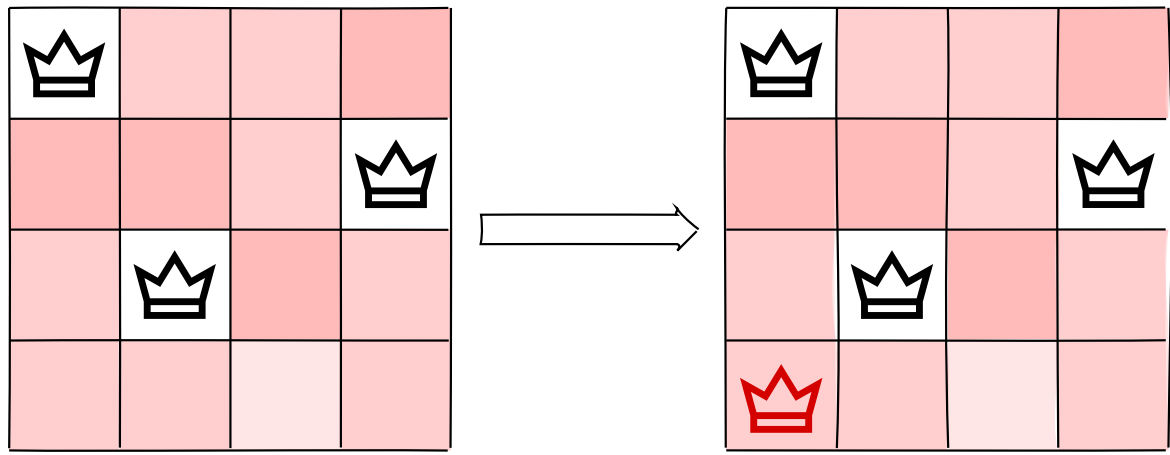
Solução 4

Solução X

A rainha 4 foi posicionada em (3, 0)

Backtracking

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



Solução 4

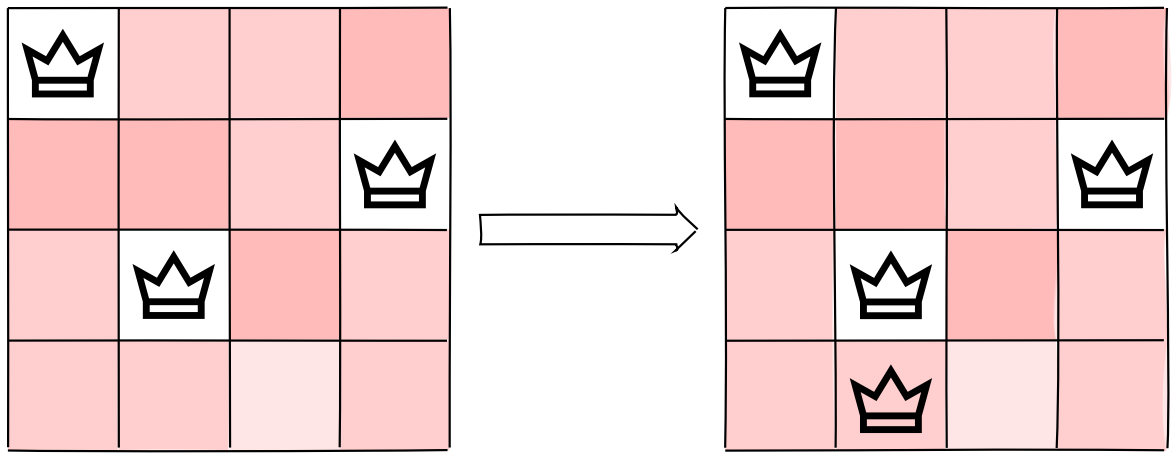
Solução X

A rainha 4 foi posicionada em (3, 0)

Existem violações de ataque

Backtracking

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



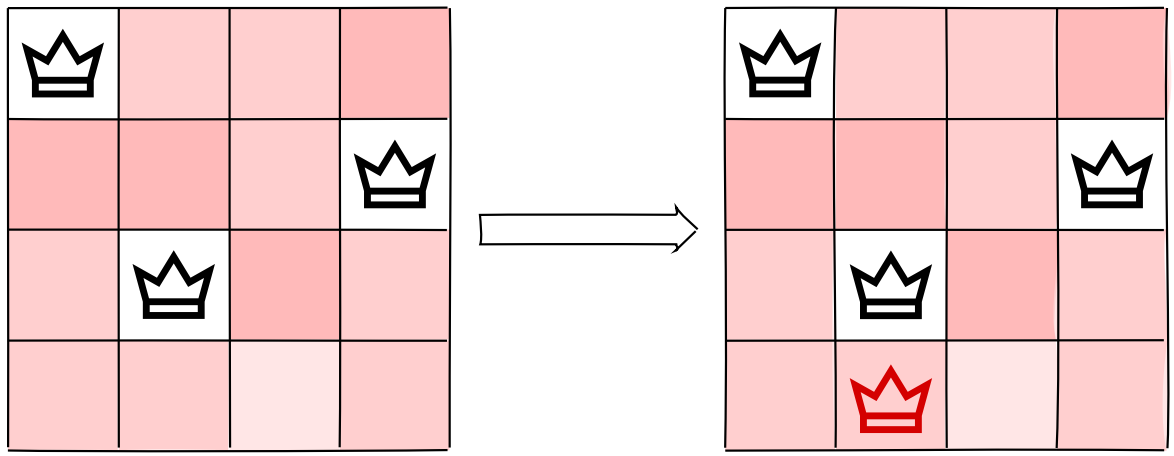
Solução 4

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

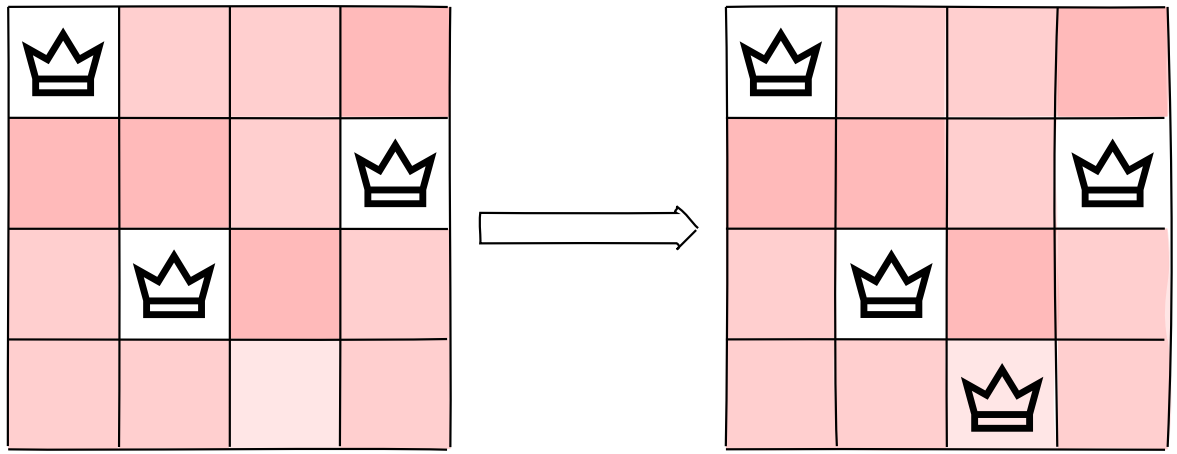
Solução X

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



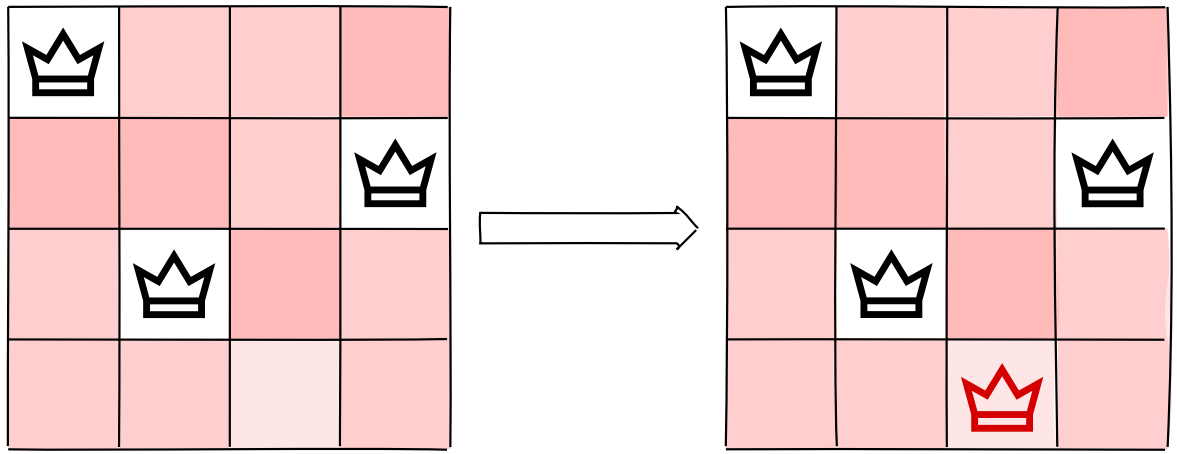
Solução 4

Solução X

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

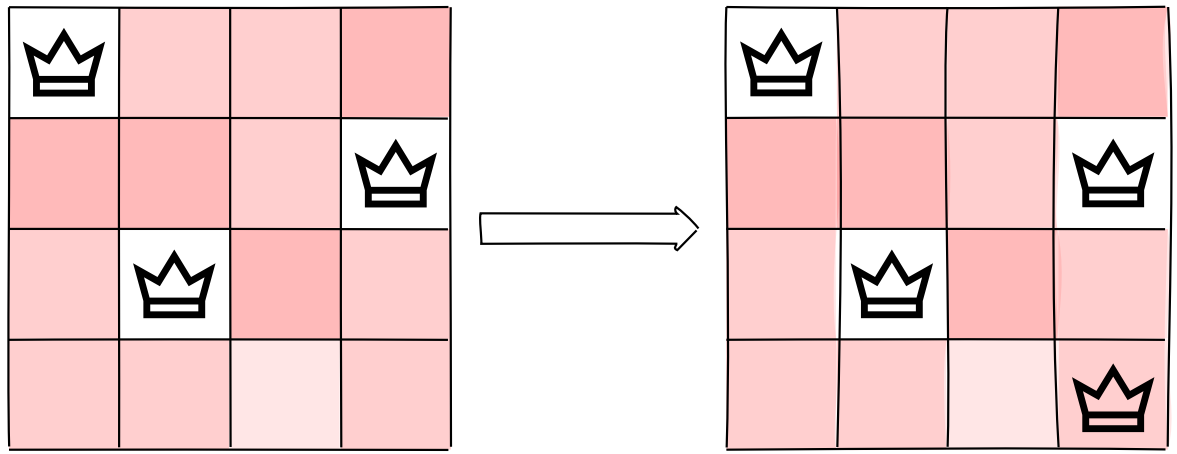
Solução X

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



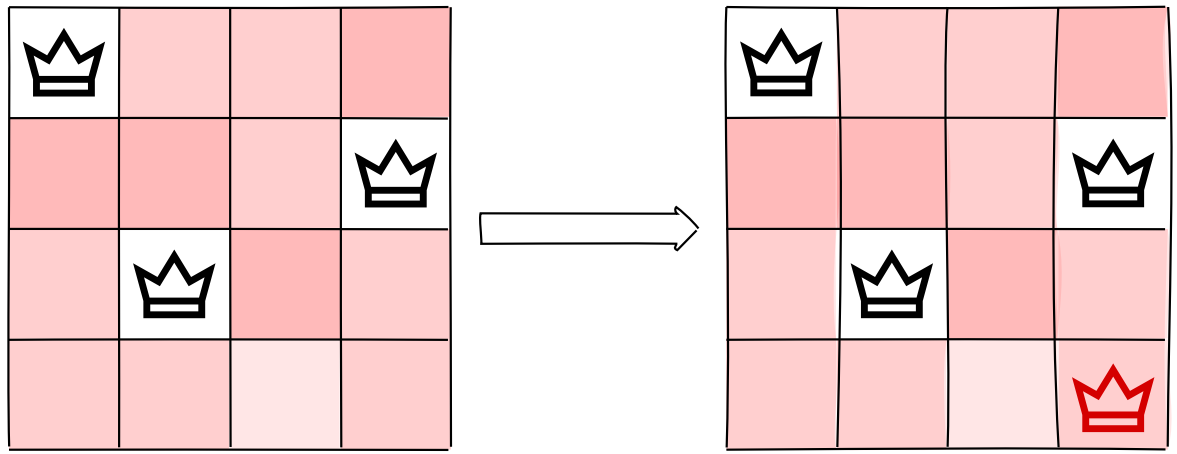
Solução 4

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

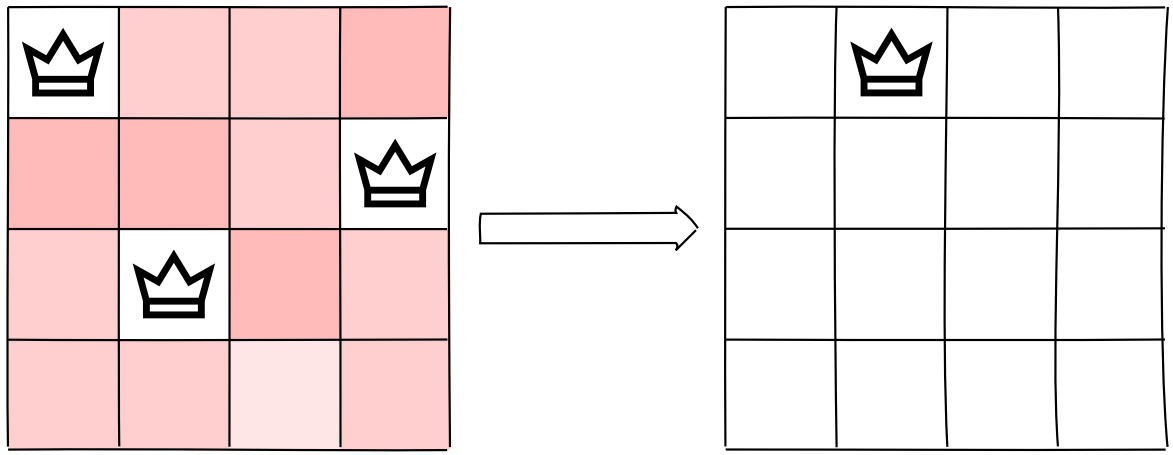
Solução X

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

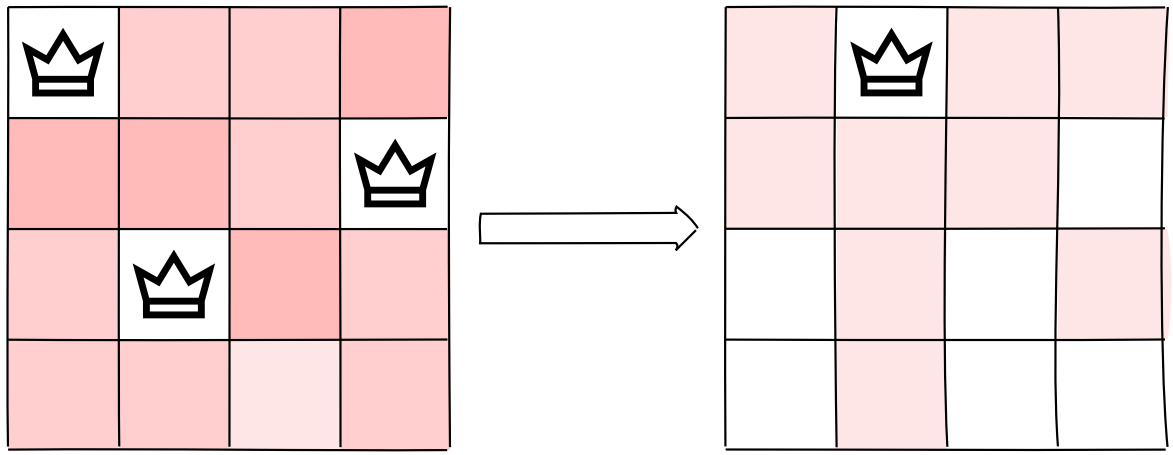


Solução 4 Solução 5

Backtrack: a rainha 1 foi reposicionada em (0, 1)

Backtracking

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



Solução 4

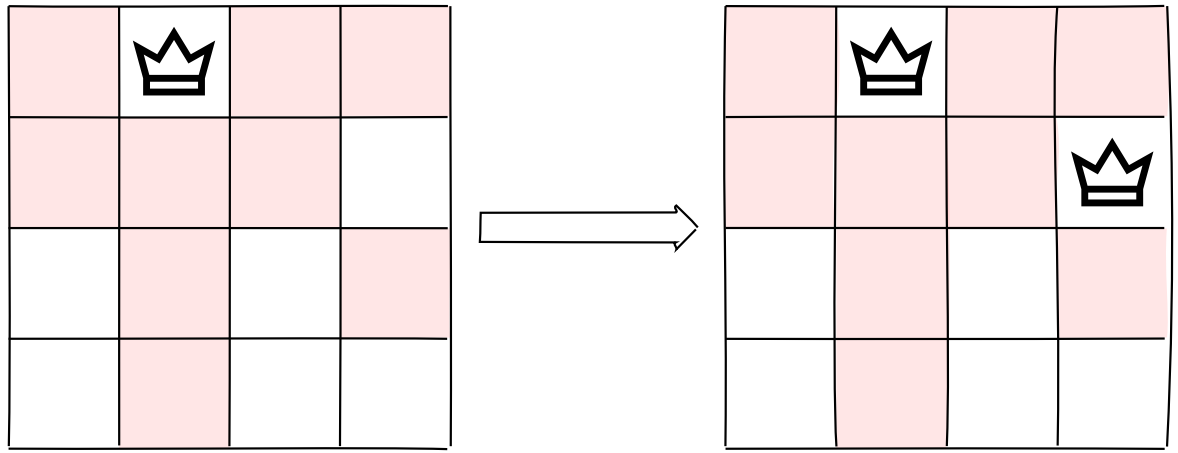
Solução 5

Backtrack: a rainha 1 foi reposicionada em (0, 1)

Sem violação de ataque

Backtracking

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



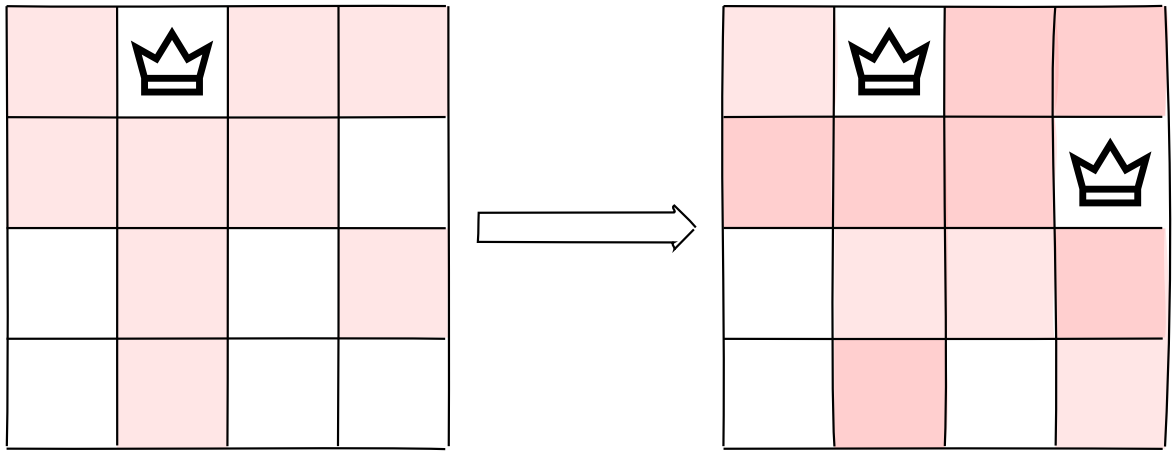
Solução 5

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

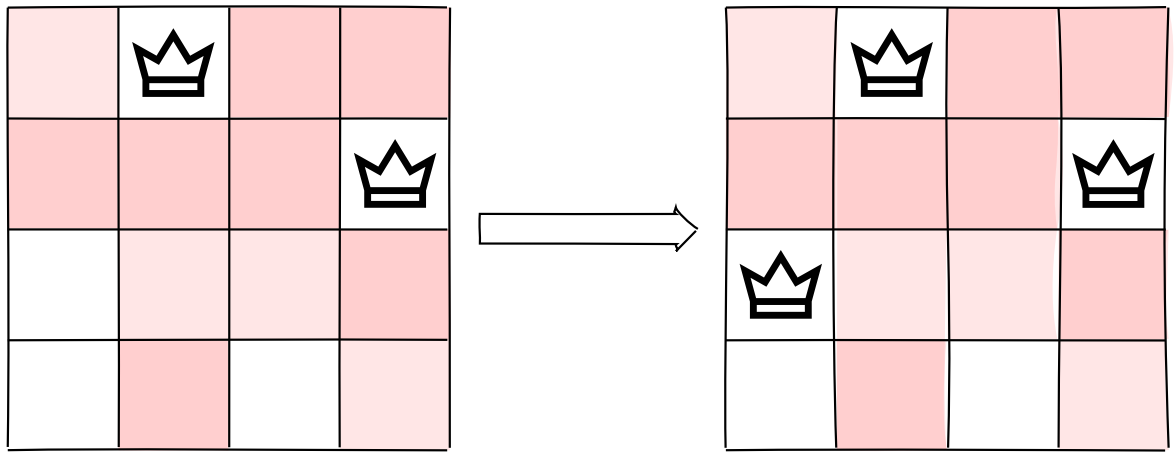
Solução 6

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



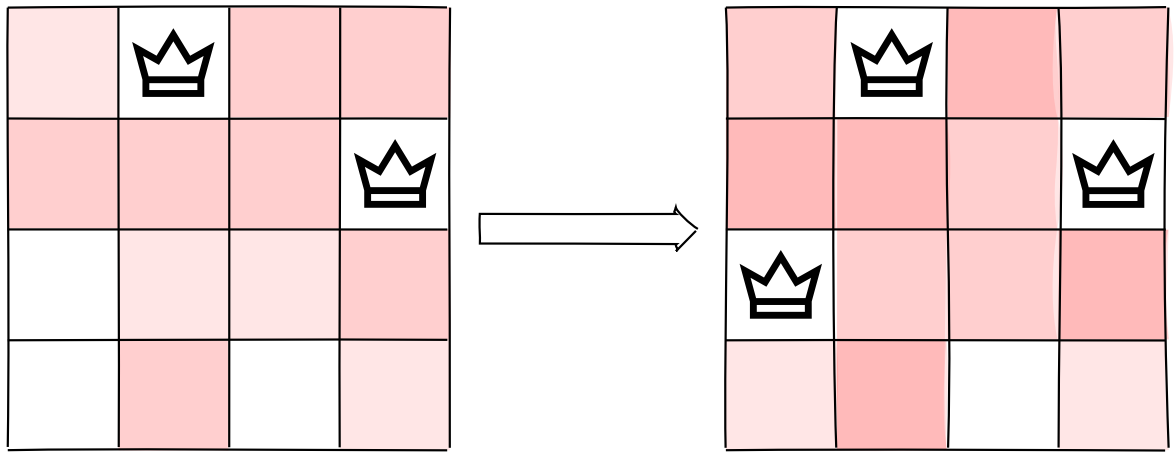
Solução 6

Solução 7

A rainha 3 foi posicionada em (2, 0)

Backtracking

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



Solução 6

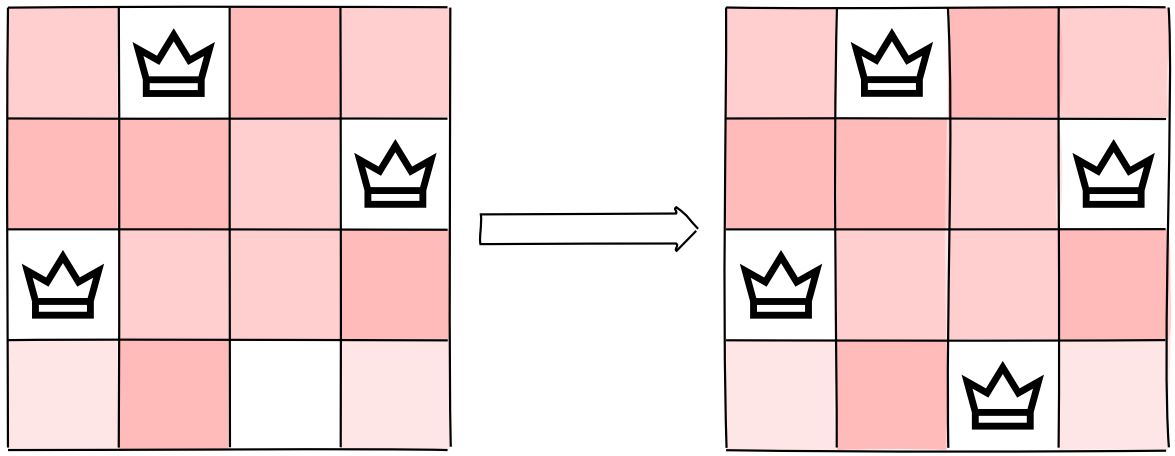
Solução 7

A rainha 3 foi posicionada em (2, 0)

Sem violação de ataque

Backtracking

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



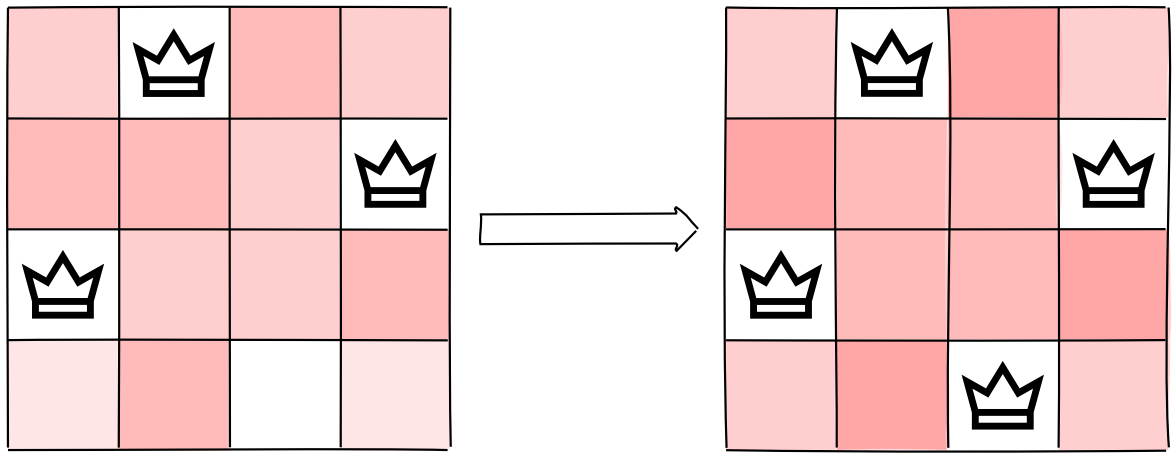
Solução 7

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

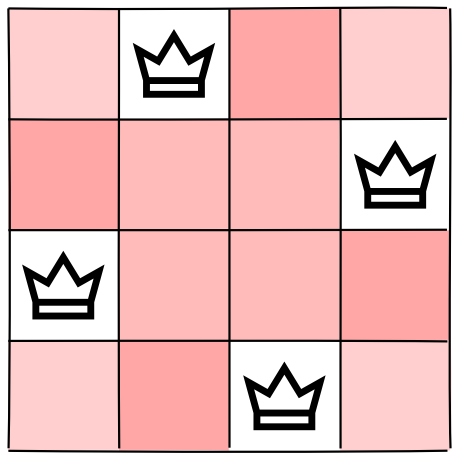
Solução 8

A rainha 4 foi posicionada em (3, 2)

Sem violação de ataque

Backtracking

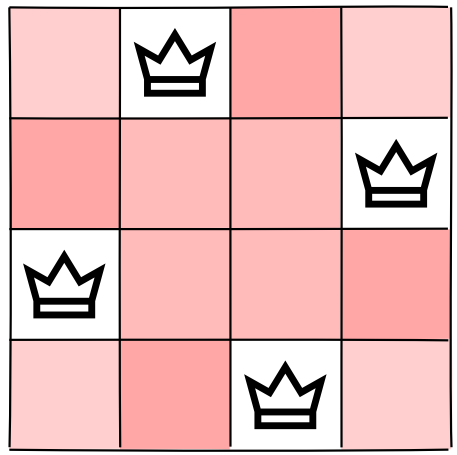
- Problema das 4-rainhas: em um tabuleiro 4×4, cada rainha é posicionada em linhas distintas



Em uma busca exaustiva, 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

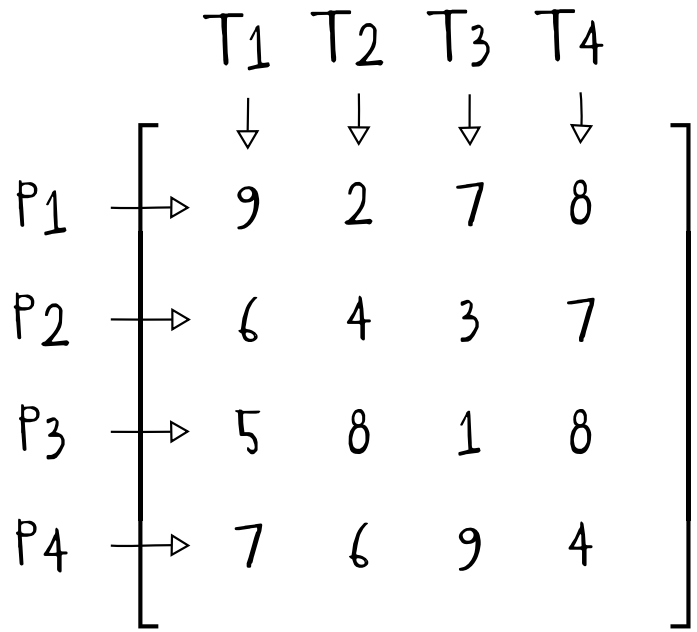
Branch-and-bound

- 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)

$$\begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

Branch-and-bound

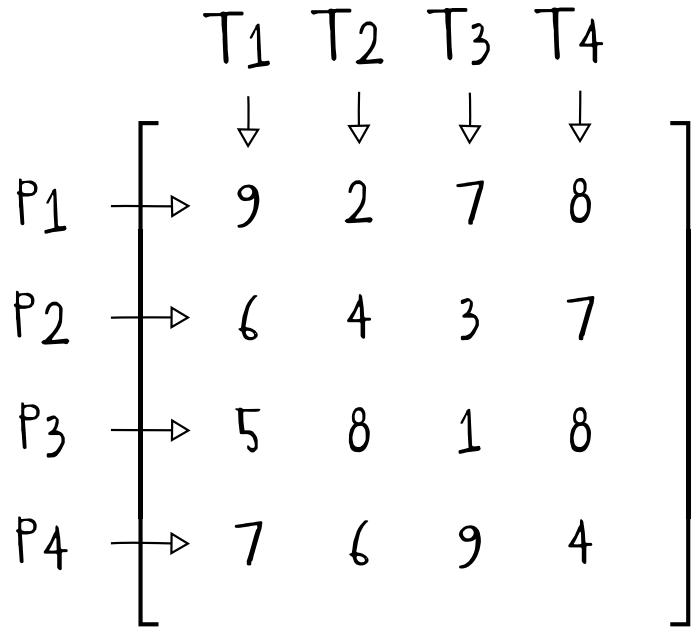
- Problema de alocar *n* pessoas para *n* trabalhos
  - Cada pessoa pode realizar somente um trabalho
  - A matriz armazena o valor que cada pessoa (linhas) recebe para realizar um trabalho (colunas)



MINIMIZAÇÃO DO CUSTO TOTAL
PARA REALIZAÇÃO DOS TRABALHOS

Branch-and-bound

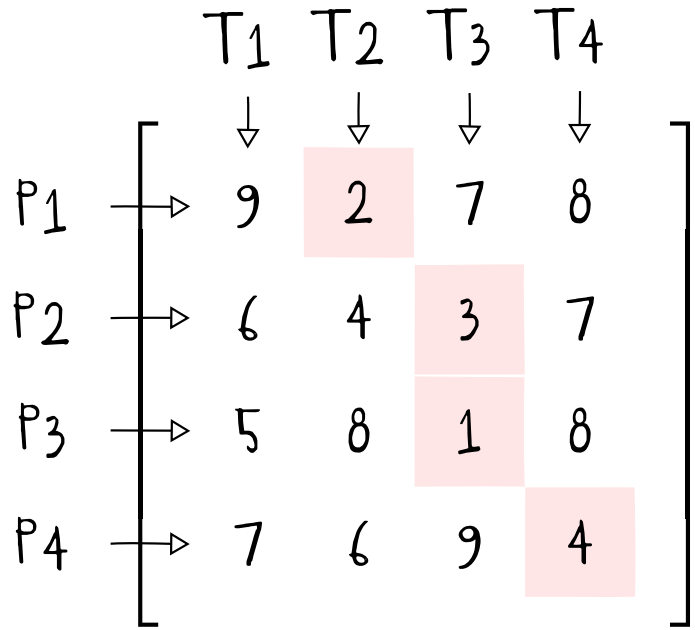
- Problema de alocar *n* pessoas para *n* trabalhos
  - Como parte da estratégia de minimização, o menor custo de cada pessoa (linha) para um trabalho (coluna) é selecionado para obter o limite inferior



No cálculo deste limitante, uma pessoa pode realizar múltiplos trabalhos, pois não é buscada uma solução

Branch-and-bound

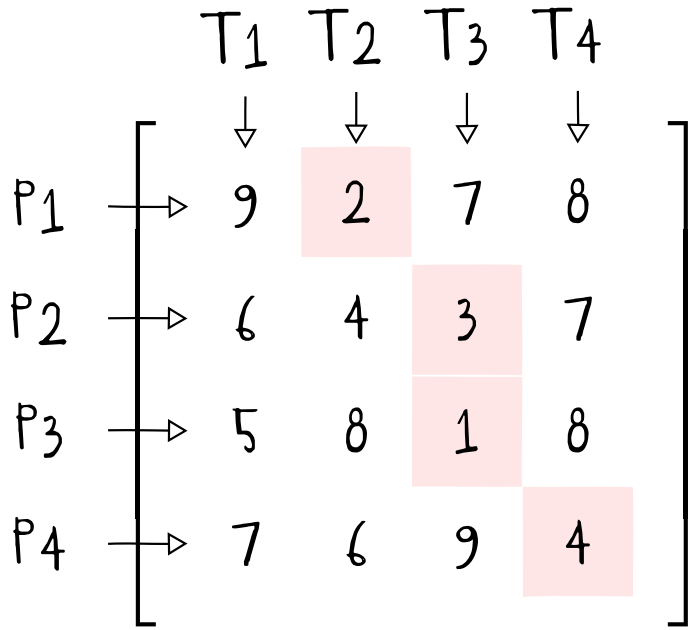
- Problema de alocar *n* pessoas para *n* trabalhos
  - Como parte da estratégia de minimização, o menor custo de cada pessoa (linha) para um trabalho (coluna) é selecionado para obter o limite inferior

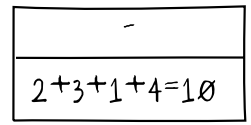


O LIMITE INFERIOR É 2 + 3 + 1 + 4 = 10

Branch-and-bound

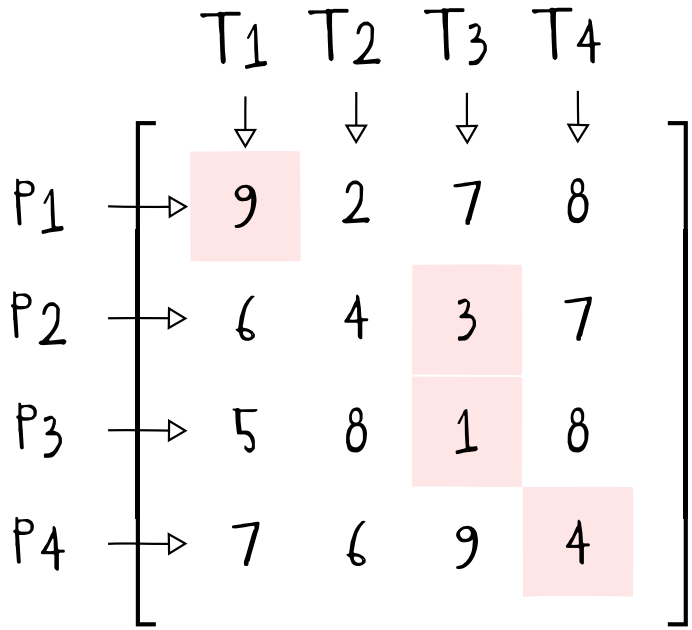
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

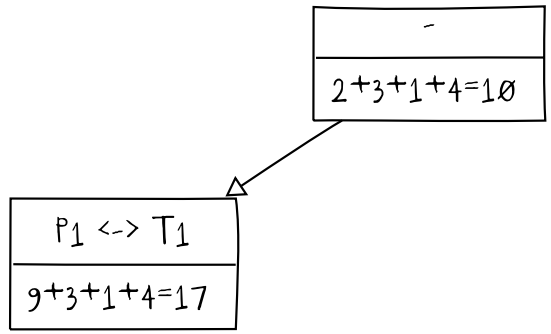




Branch-and-bound

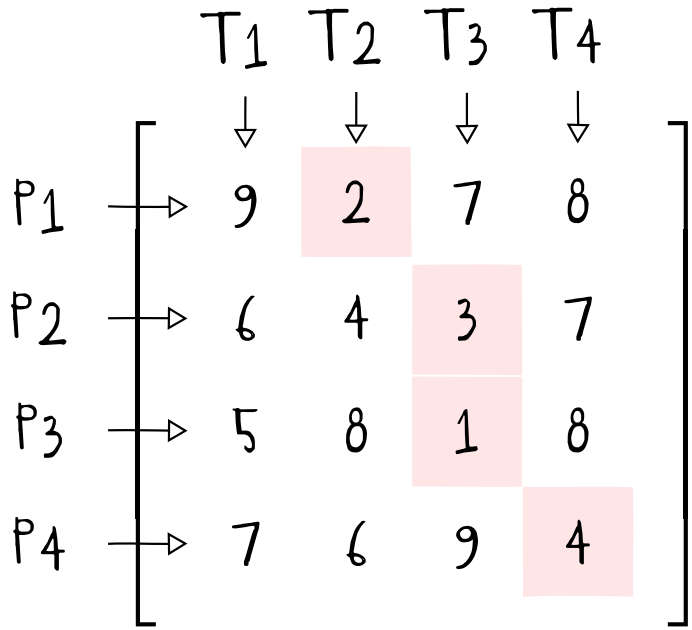
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

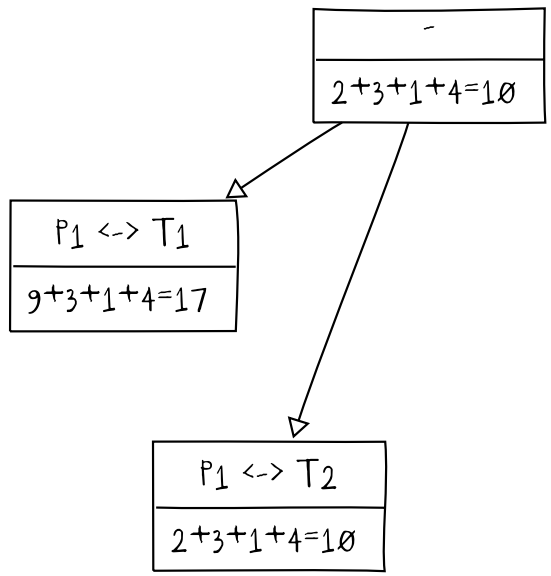




Branch-and-bound

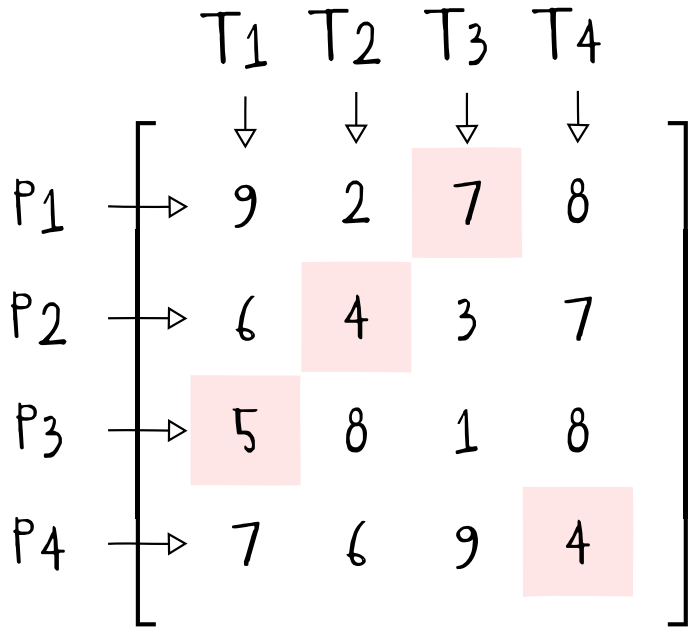
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

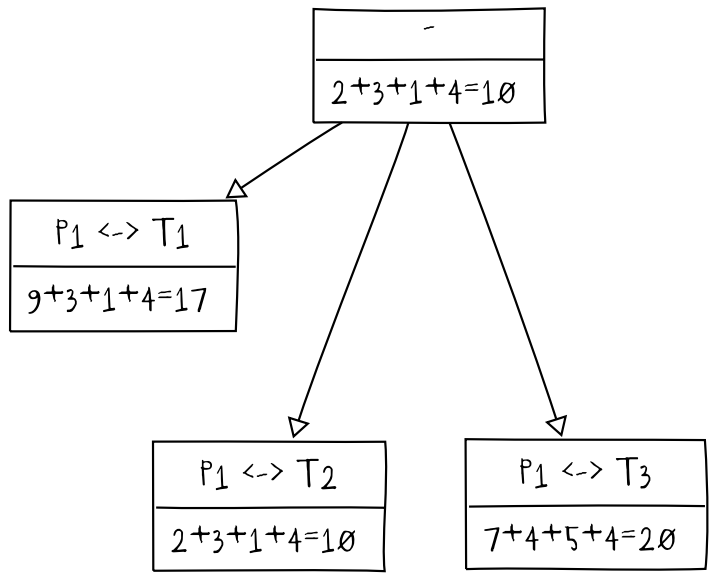




Branch-and-bound

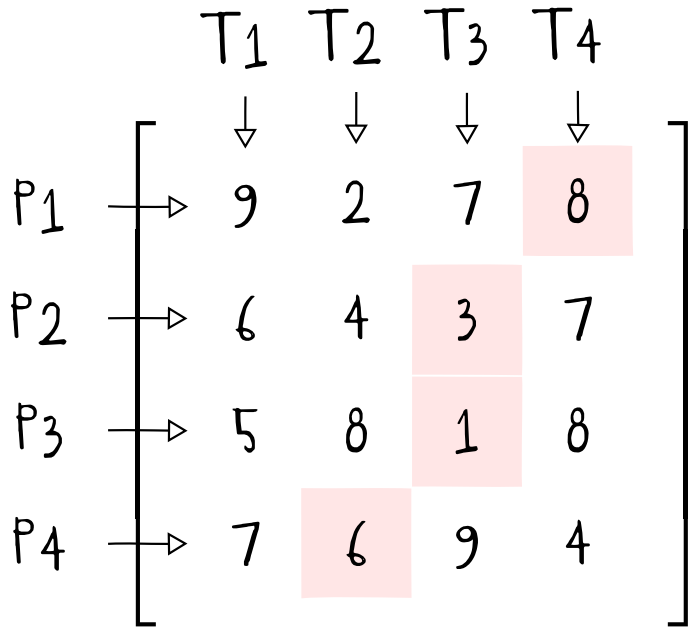
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

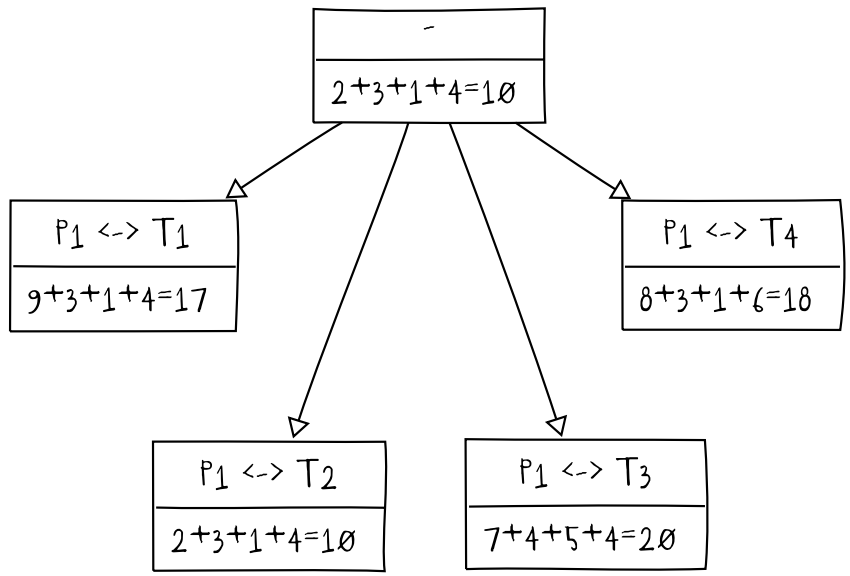




Branch-and-bound

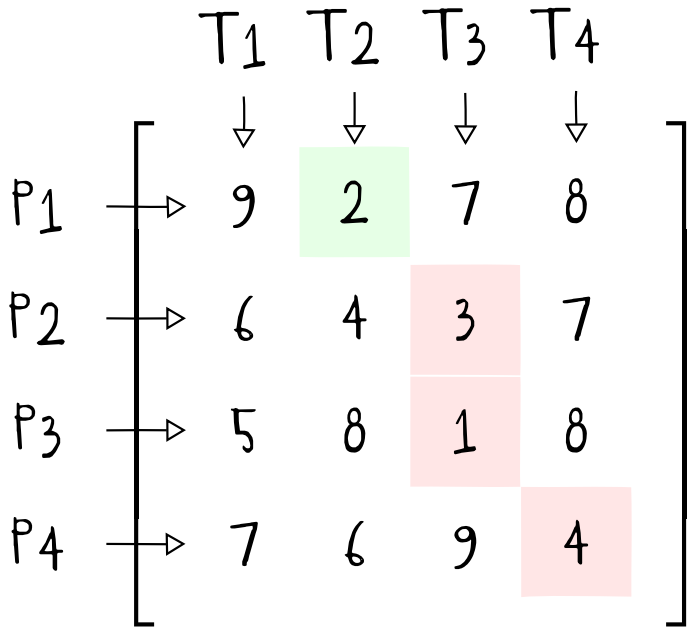
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

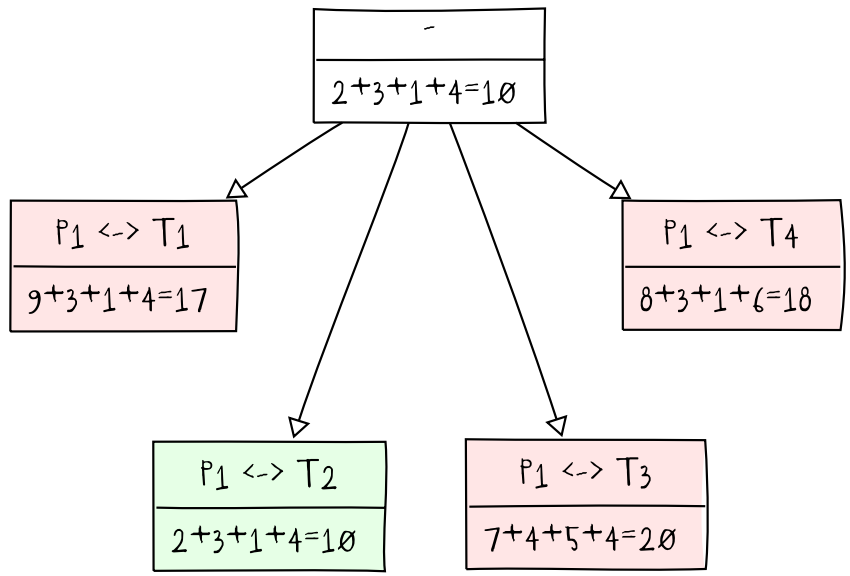




Branch-and-bound

- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

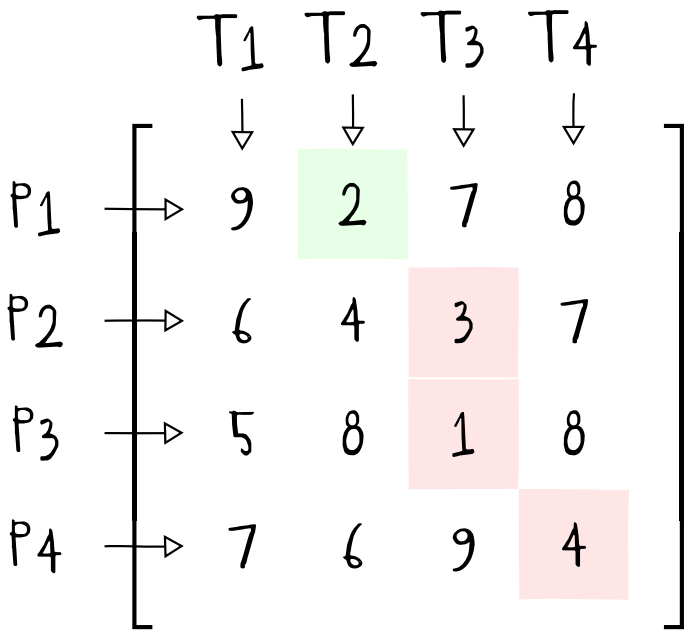




A SOLUÇÃO PARCIAL MAIS PROMISSORA
ALOCA A PESSOA P1 PARA O TRABALHO T2

Branch-and-bound

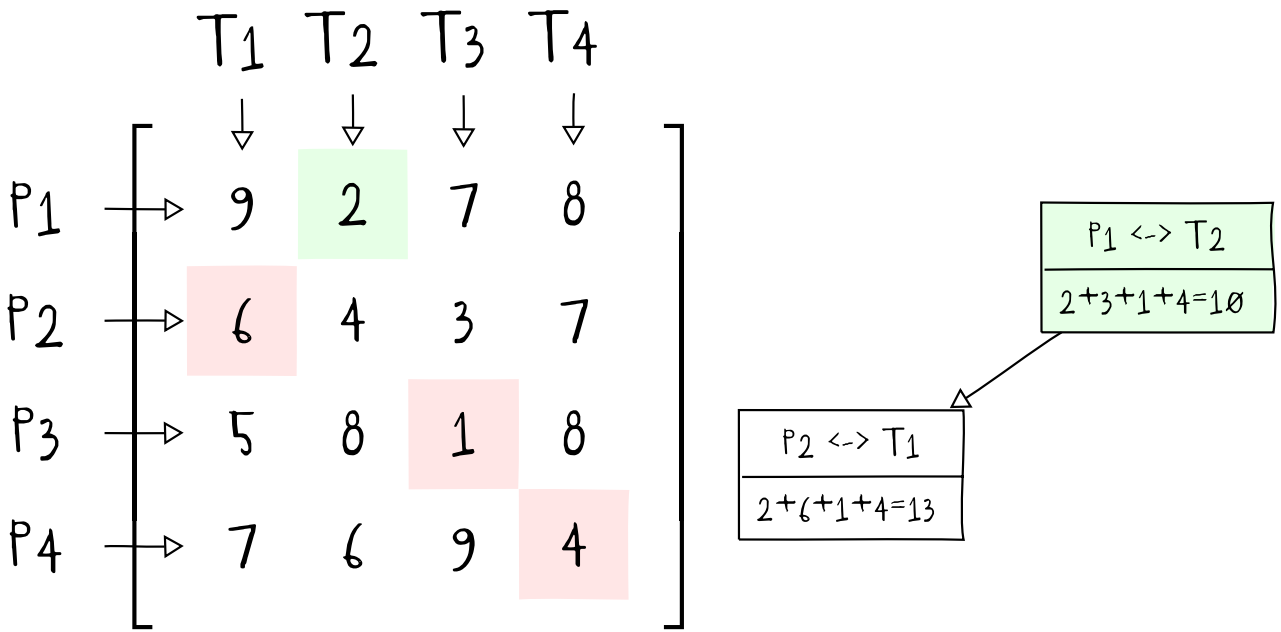
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*2



P1 <-> T2
2+3+1+4=10

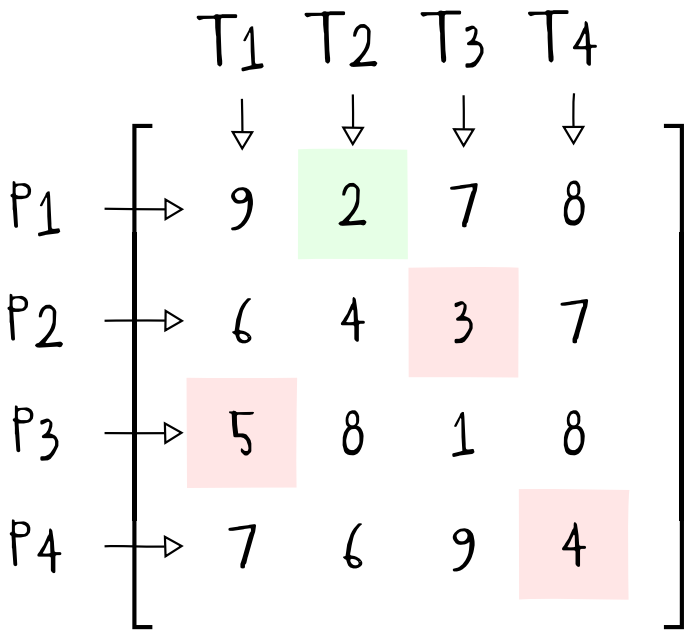
Branch-and-bound

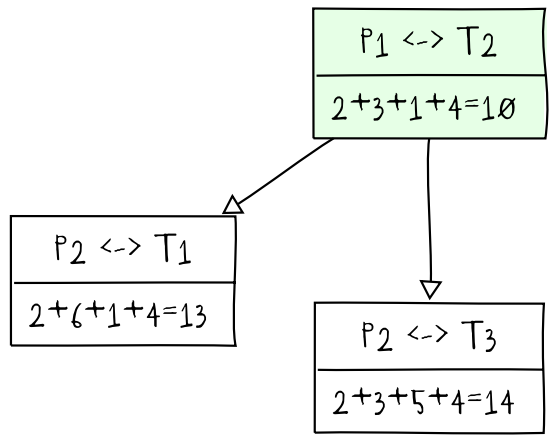
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*2



Branch-and-bound

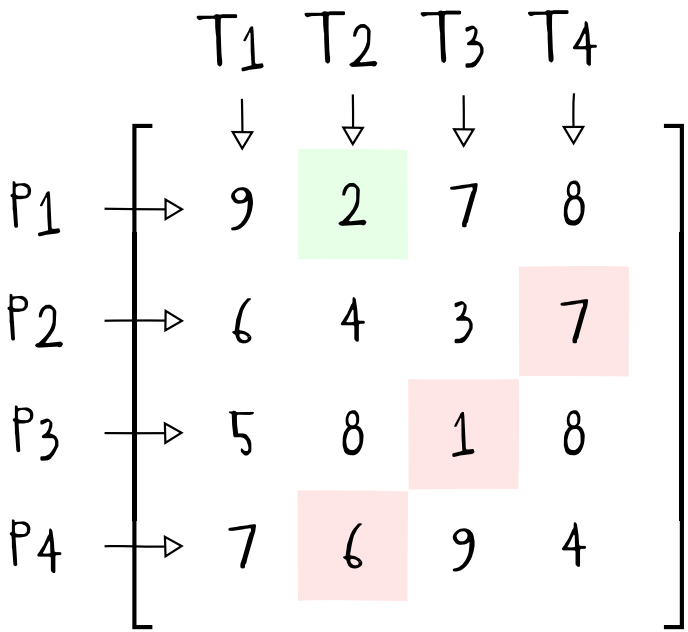
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*2

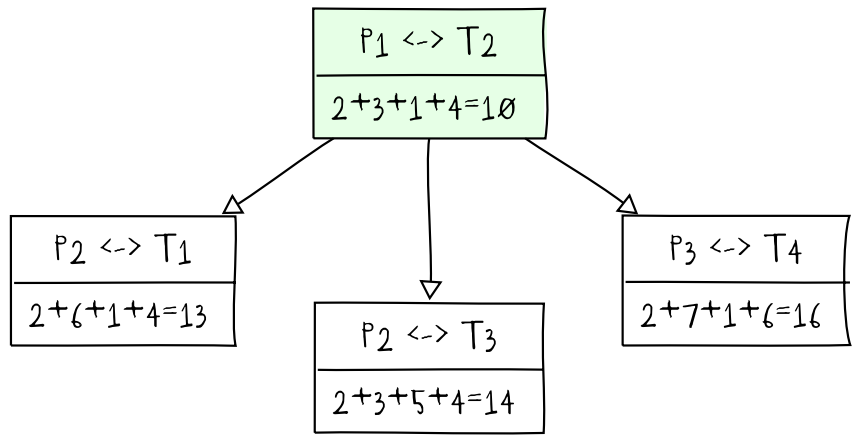




Branch-and-bound

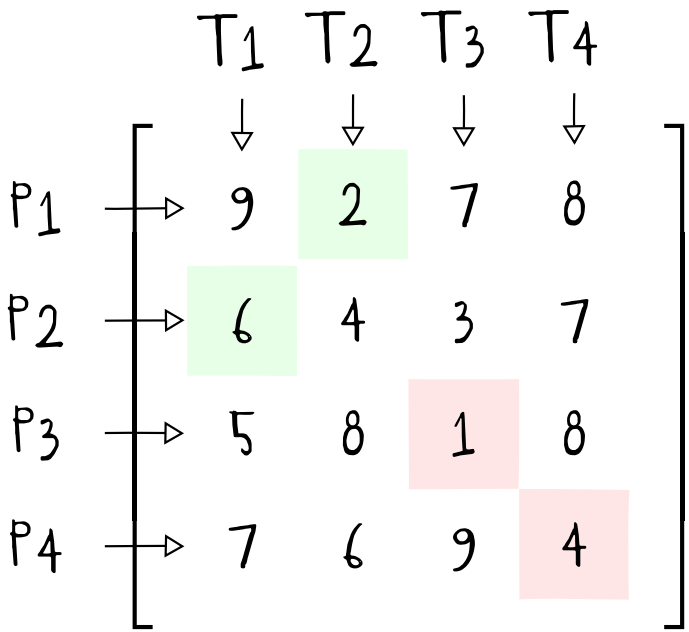
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*2

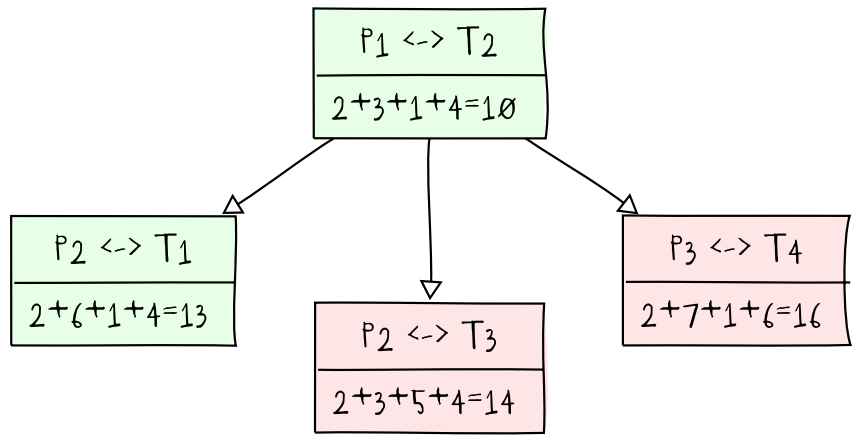




Branch-and-bound

- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*2

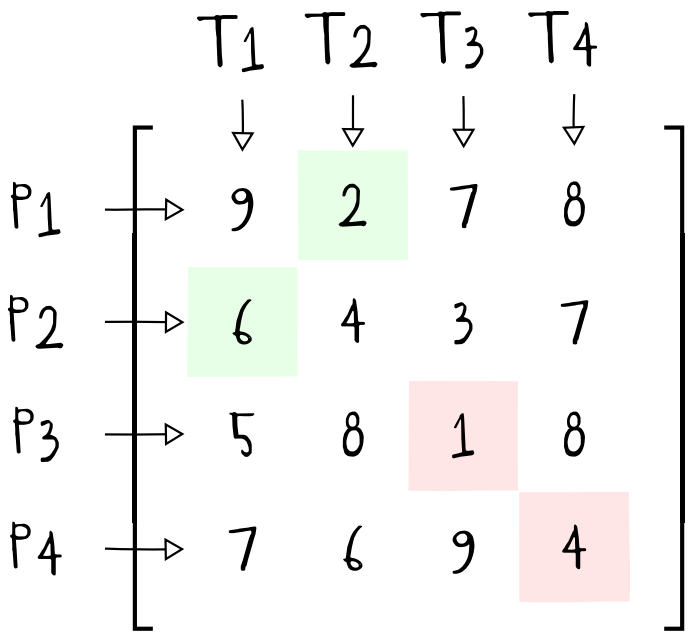




A SOLUÇÃO PARCIAL MAIS PROMISSORA
ALOCA A PESSOA P2 PARA O TRABALHO T1

Branch-and-bound

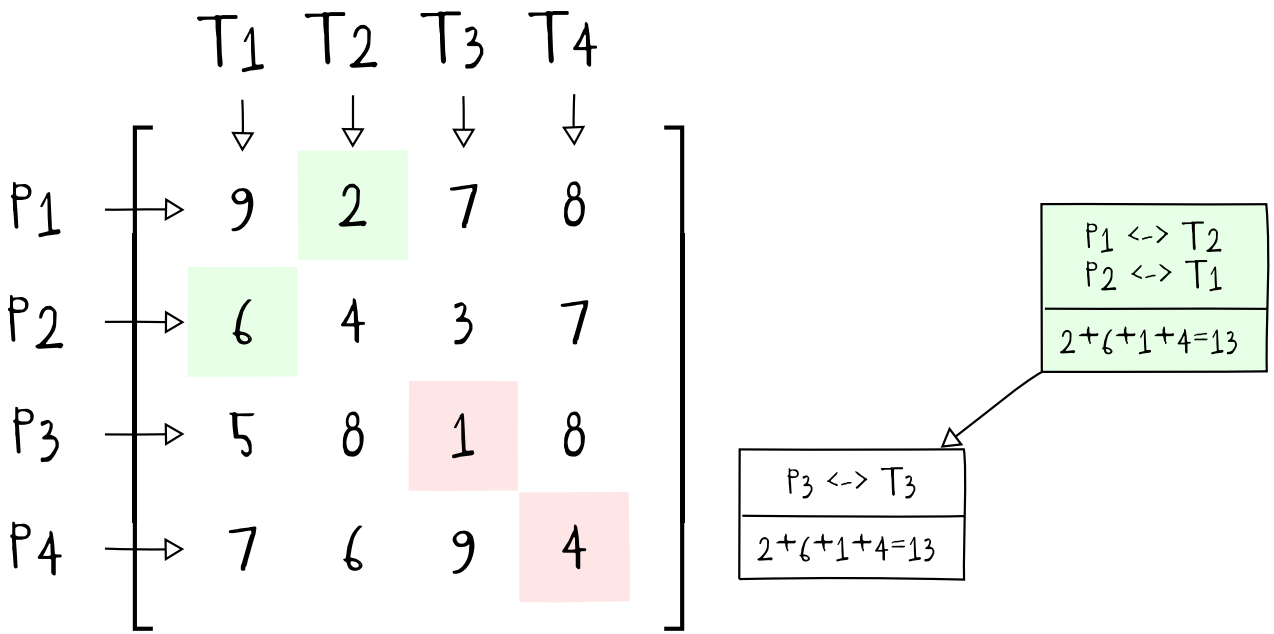
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*3



P1 <-> T2 P2 <-> T1
2+6+1+4=13

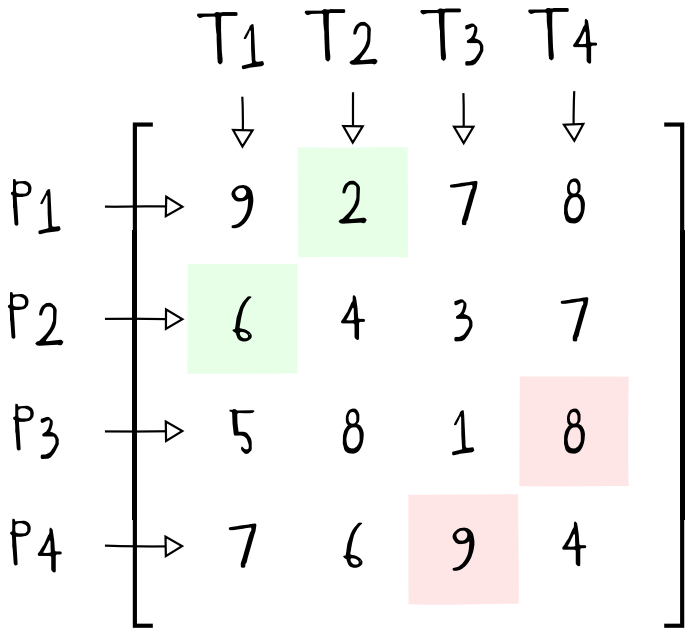
Branch-and-bound

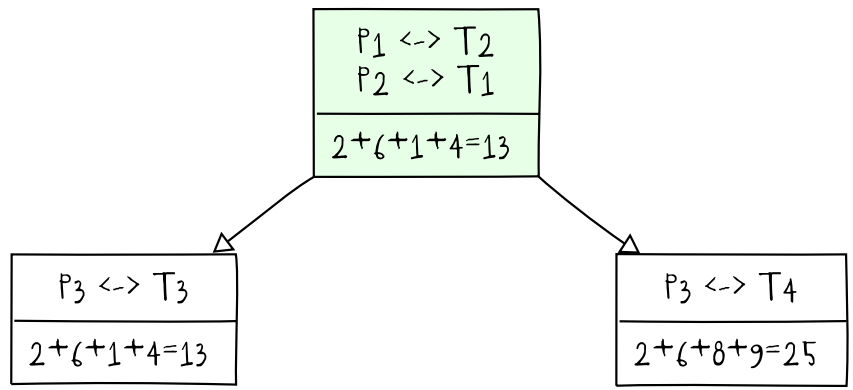
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*3



Branch-and-bound

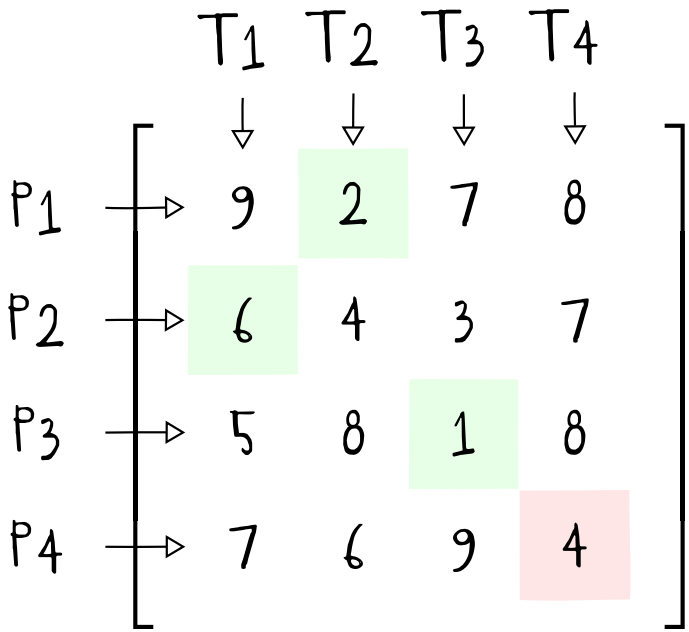
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*3

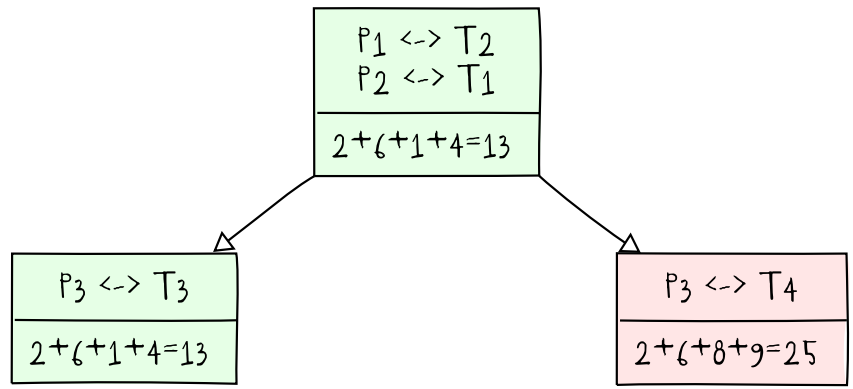




Branch-and-bound

- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*3

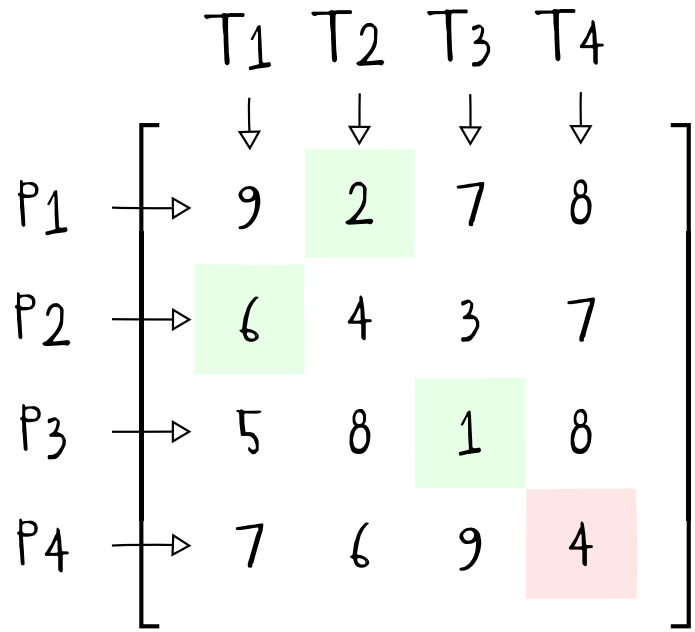




A SOLUÇÃO PARCIAL MAIS PROMISSORA
ALOCA A PESSOA P3 PARA O TRABALHO T3

Branch-and-bound

- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*4

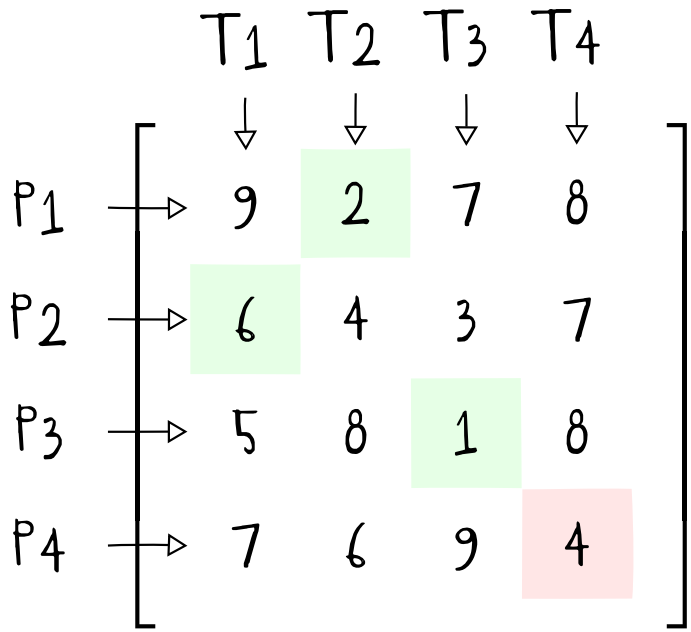


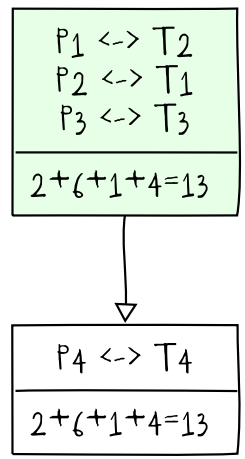
P1 <-> T2
P2 <-> T1
P3 <-> T3

2+6+1+4=13

Branch-and-bound

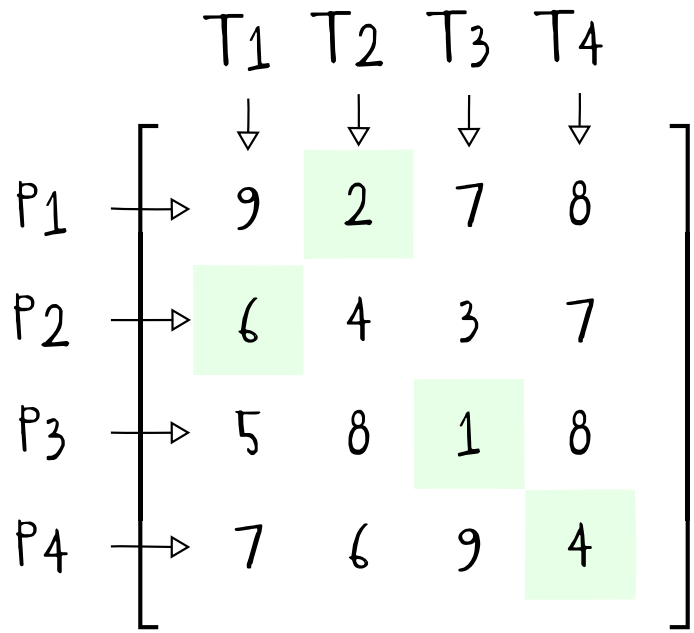
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*4

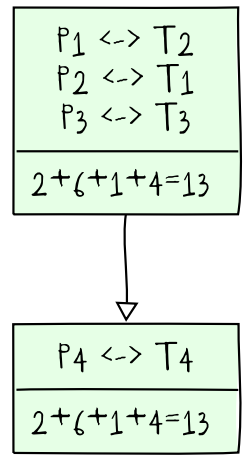




Branch-and-bound

- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*4

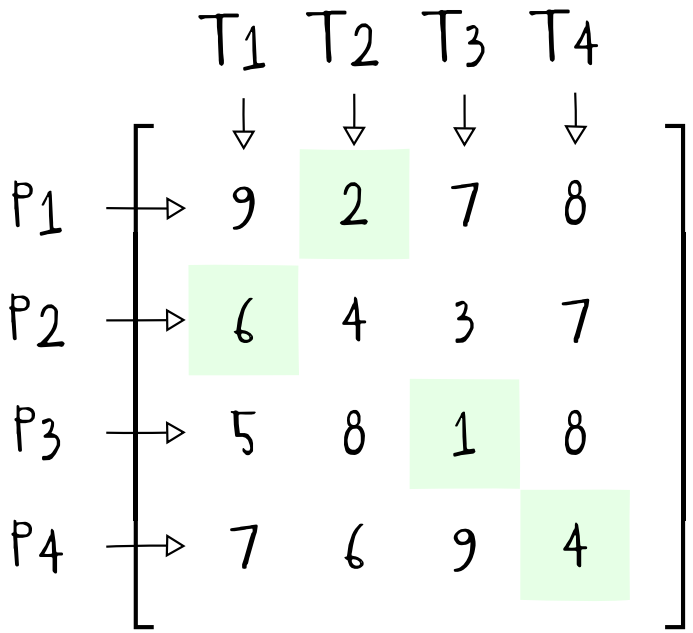




A SOLUÇÃO COMPLETA MAIS PROMISSORA ALOCA A PESSOA P4 PARA O TRABALHO T4

Branch-and-bound

- Problema de alocar *n* pessoas para *n* trabalhos



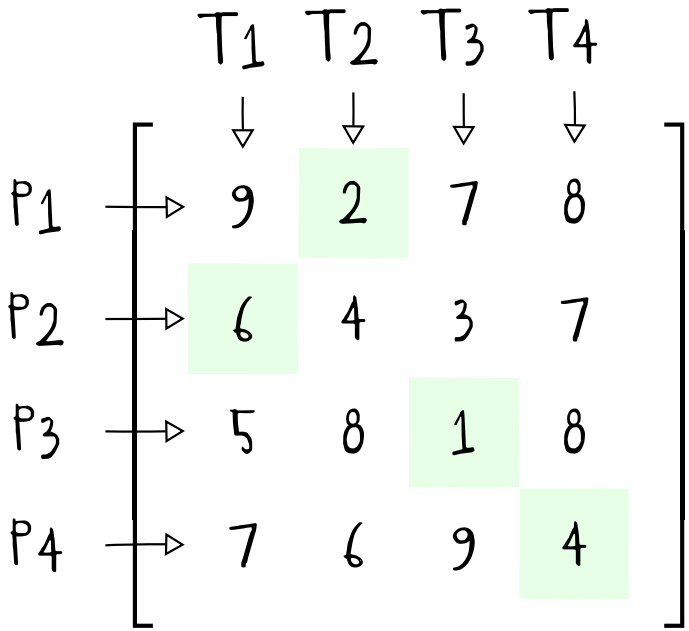
P1 <-> T2
P2 <-> T1
P3 <-> T3
P4 <-> T4

2+6+1+4=13

NA BUSCA EXAUSTIVA, 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



P1 <-> T2
P2 <-> T1
P3 <-> T3
P4 <-> T4

2+6+1+4=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*

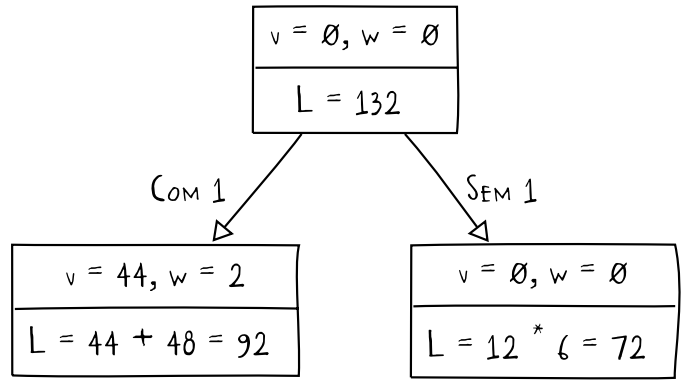
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10

$v = 0, w = 0$
$L = 132$

Branch-and-bound

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

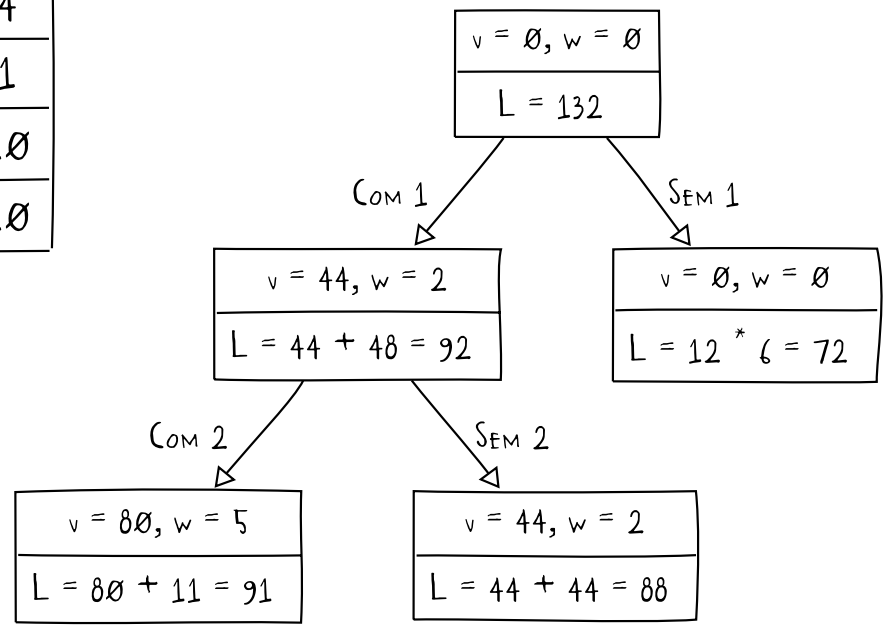
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



Branch-and-bound

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

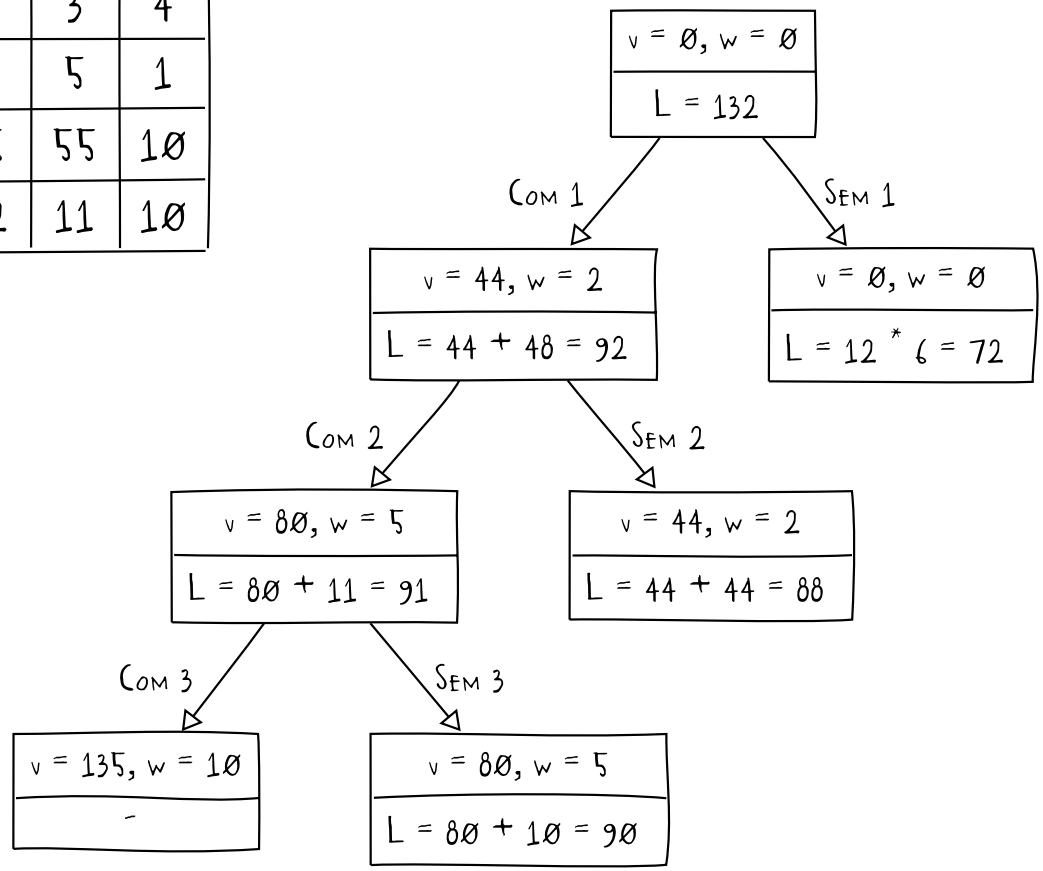
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



Branch-and-bound

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

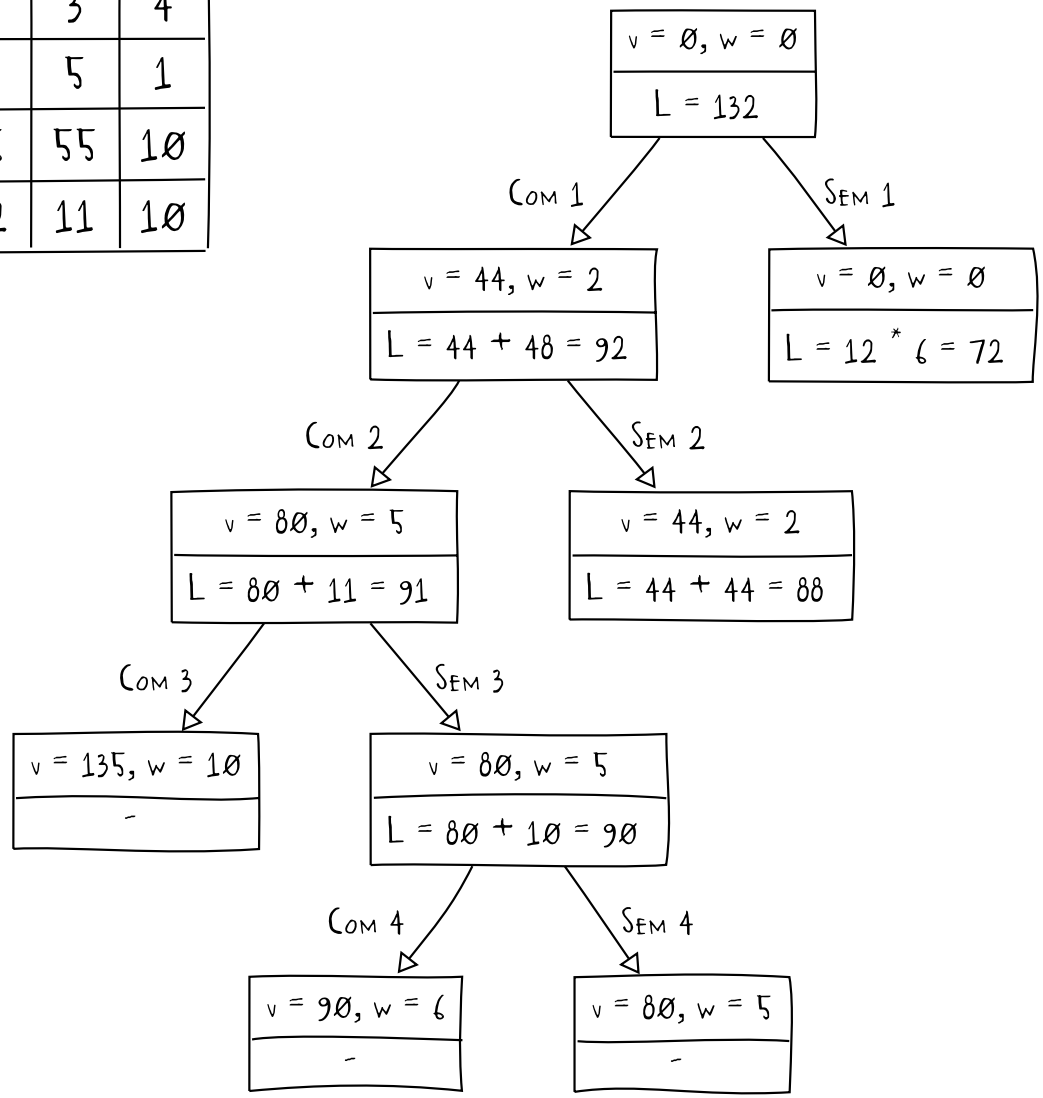
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



Branch-and-bound

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

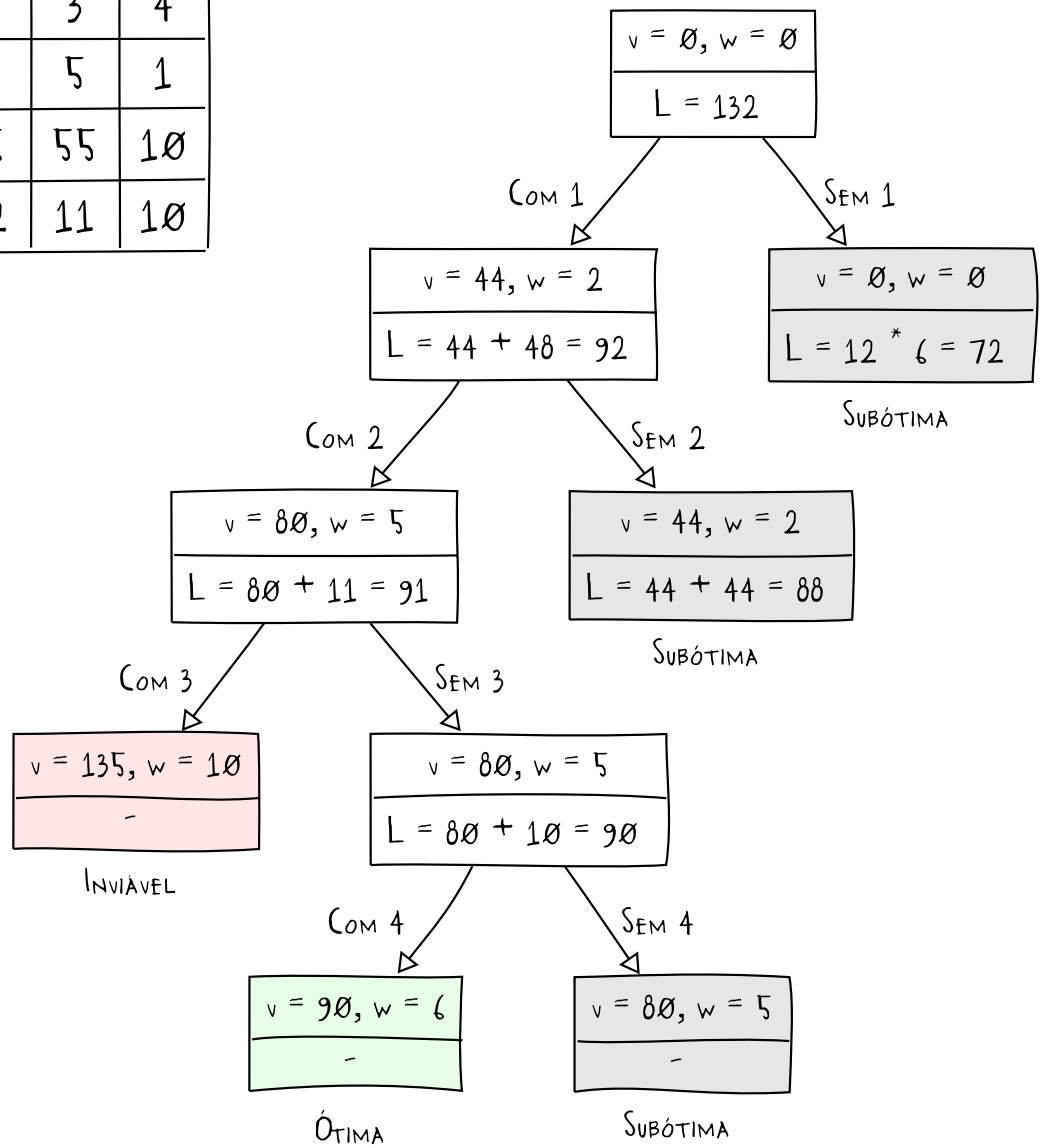
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



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



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
  - $\#NL$
  - $[Largura] \times [Altura]$
  - $M_{x,y} = 0\,(espaço)\,, 1\,(parede)\,, X\,(partida)$

$$\begin{matrix} M_{0,0} & \cdots & M_{0,L-1} \\ \vdots & \ddots & \vdots \\ M_{A-1,0} & \cdots & M_{A-1,L-1} \end{matrix}$$

```
2
5 4
1 1 1 1 1
1 0 0 0 1
1 0 X 1 1
1 1 0 1 1
3 4
1 1 1
1 X 1
1 0 1
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@-,-
```