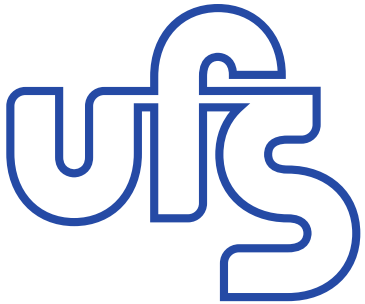
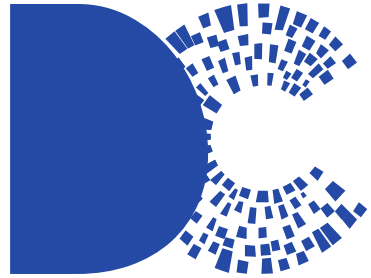


UNIVERSIDADE FEDERAL DE SERGIPE



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
  - Neste paradigma, as soluções candidatas são geradas 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
  - Neste paradigma, as soluções candidatas são geradas 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

Introdução

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

Introdução

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

Introdução

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

Introdução

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

Introdução

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

Introdução

- 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

Introdução

- 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

Introdução

- 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

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

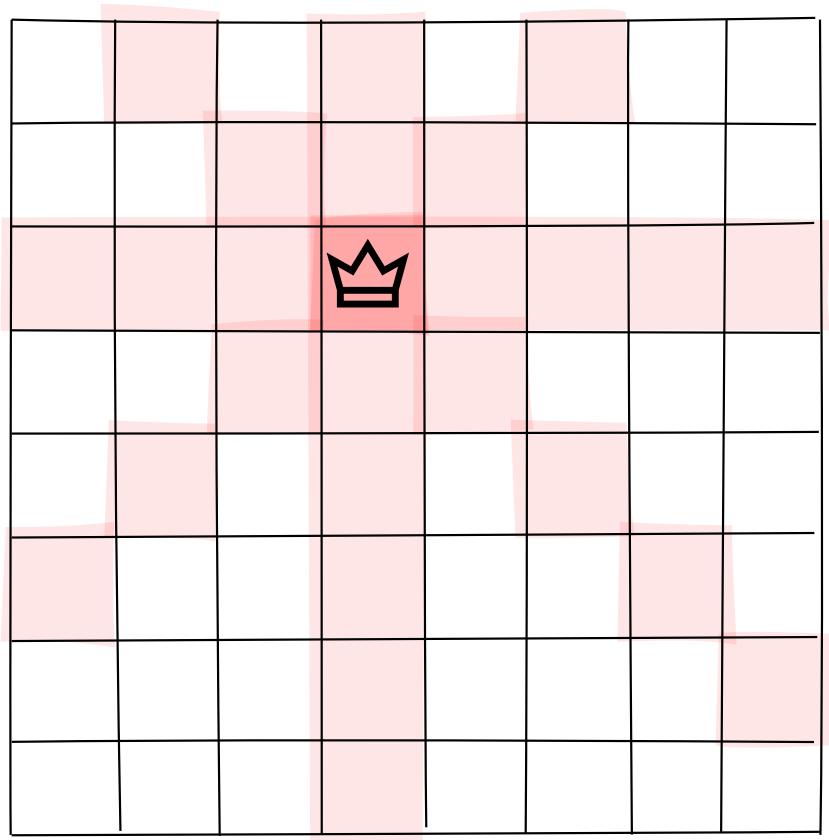
- Busca em profundidade
  - As soluções candidatas ou promissoras que atendem as restrições são exploradas primeiro
  - É feito o incremento de solução, sempre atendendo as regras impostas pelo problema
  - Este processo pode levar a uma solução completa, mas não existe uma garantia

As soluções parciais que
não atendem as restrições
são desprezadas

Espaço de busca reduzido

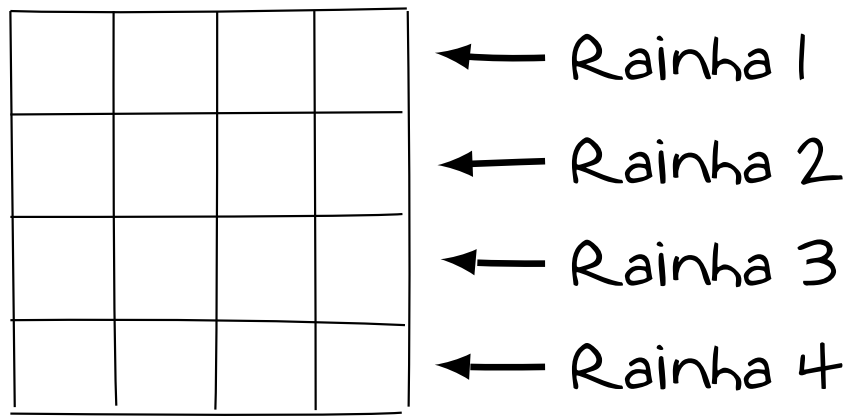
Backtracking

- 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



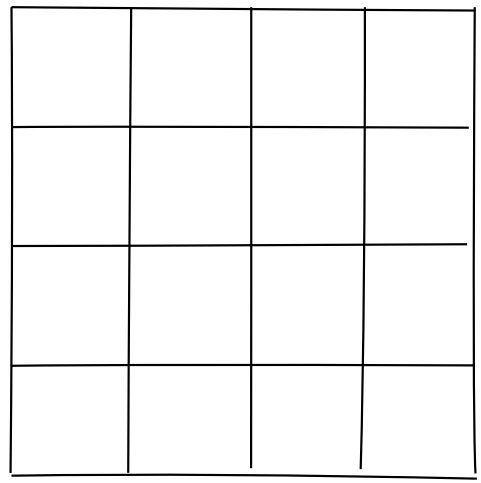
Backtracking

- 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

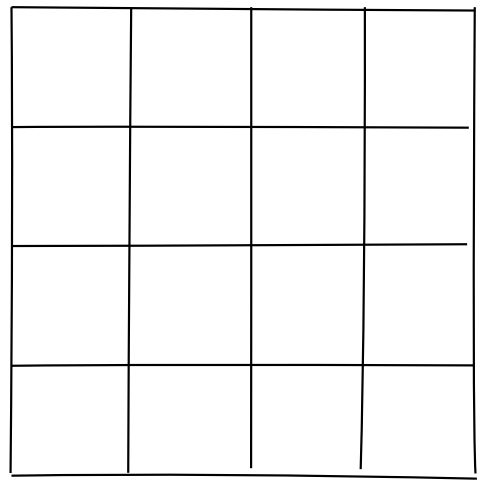


Solução 0

Nenhuma rainha posicionada

Backtracking

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



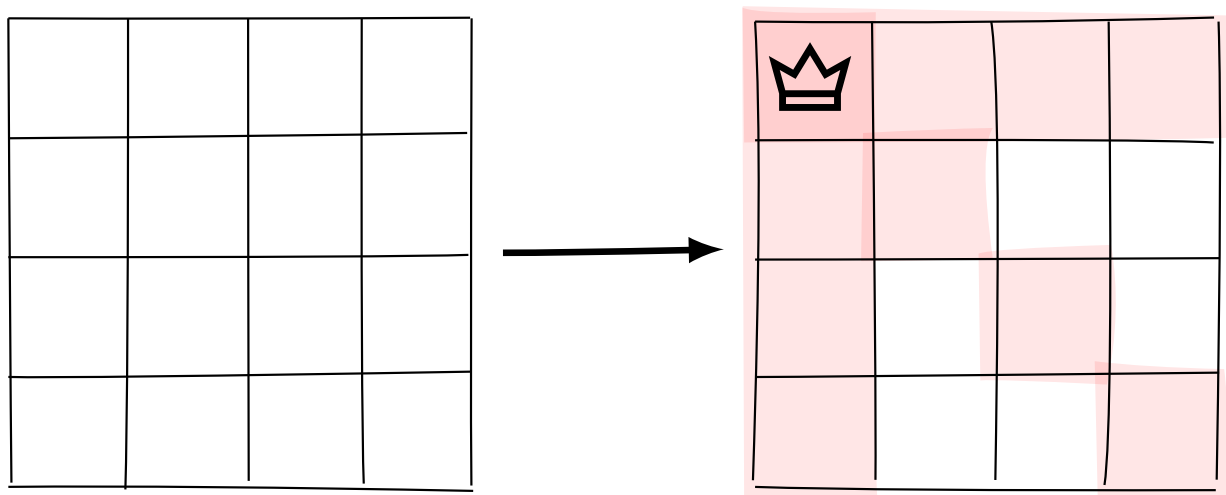
Solução 0

Nenhuma rainha posicionada

Sem violação de ataque

Backtracking

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

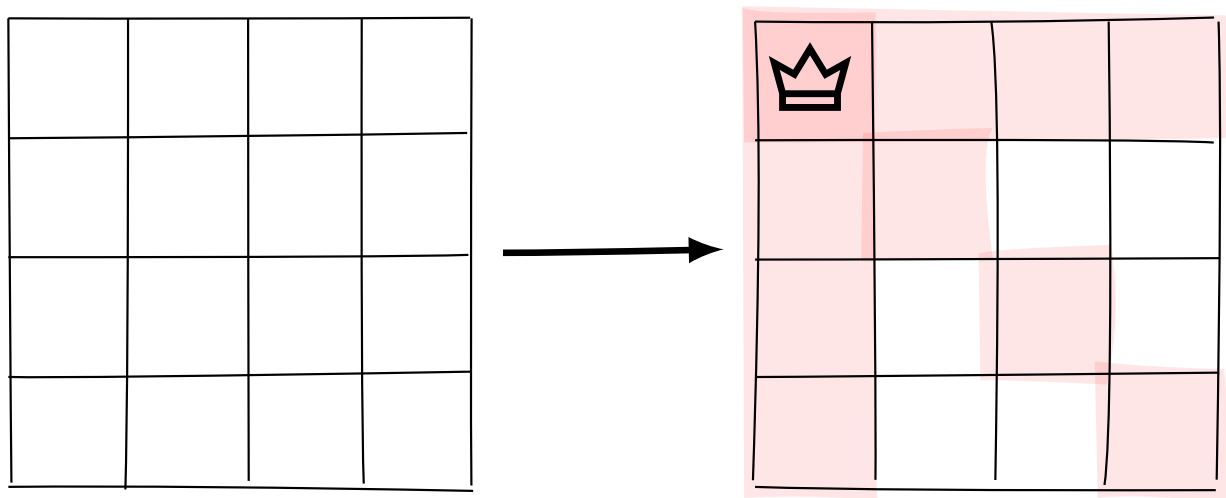


Solução 0 Solução 1

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 0

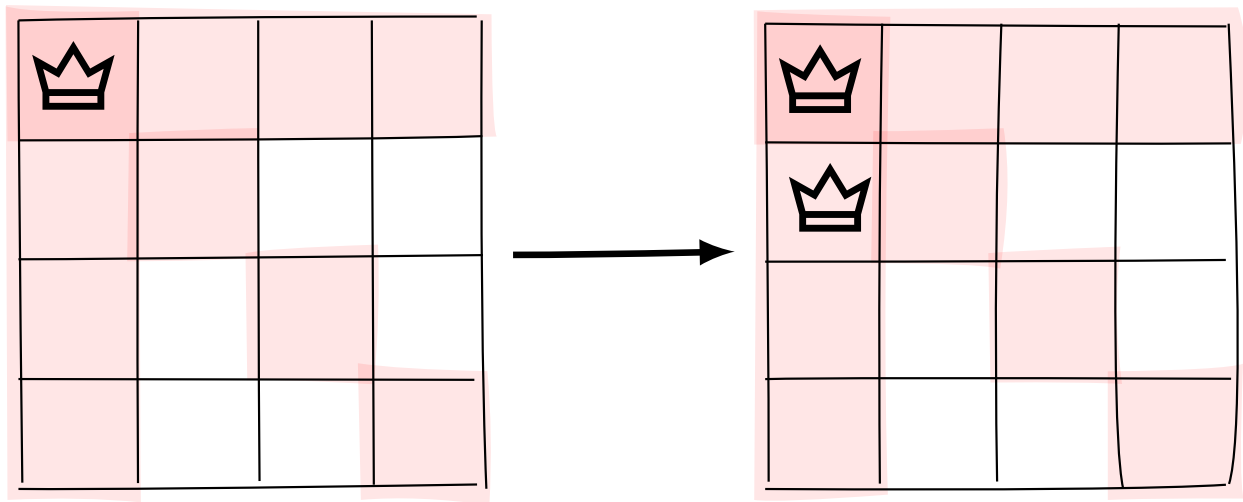
Solução 1

A rainha 1 foi posicionada em (0, 0)

Sem violação de ataque

Backtracking

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



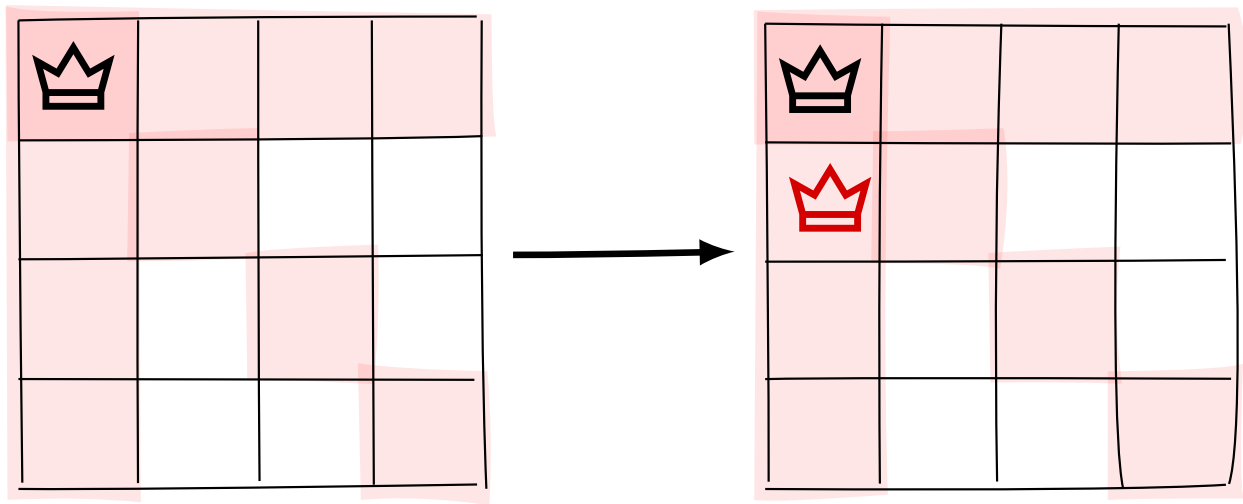
Solução 1

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

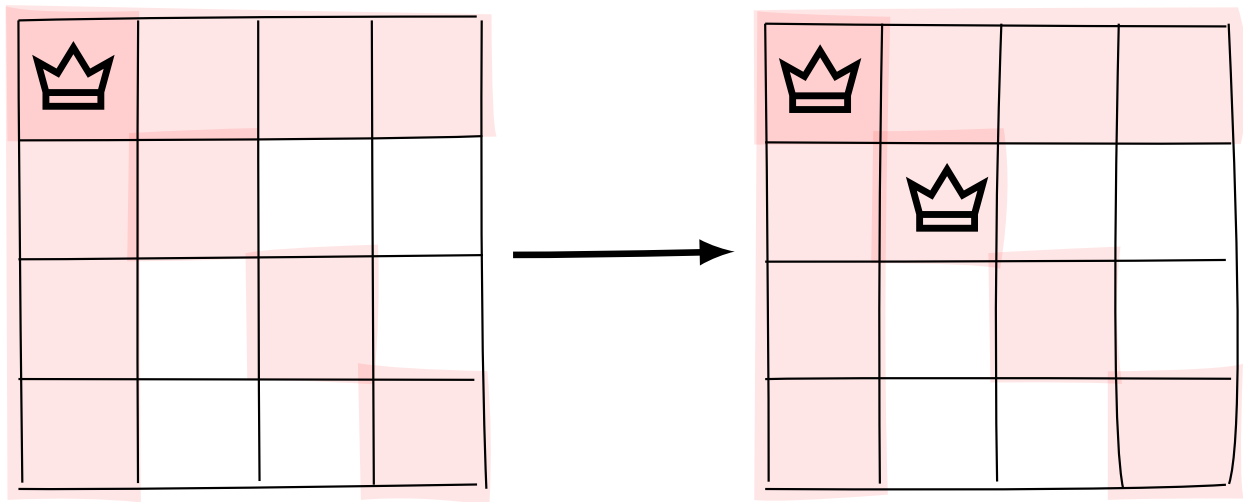
Solução X

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



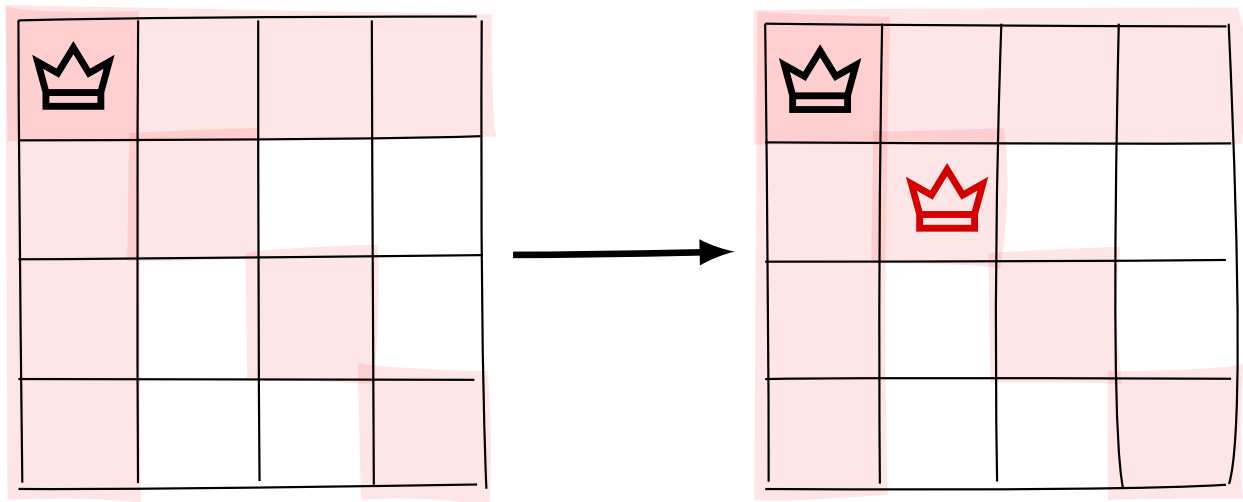
Solução 1

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



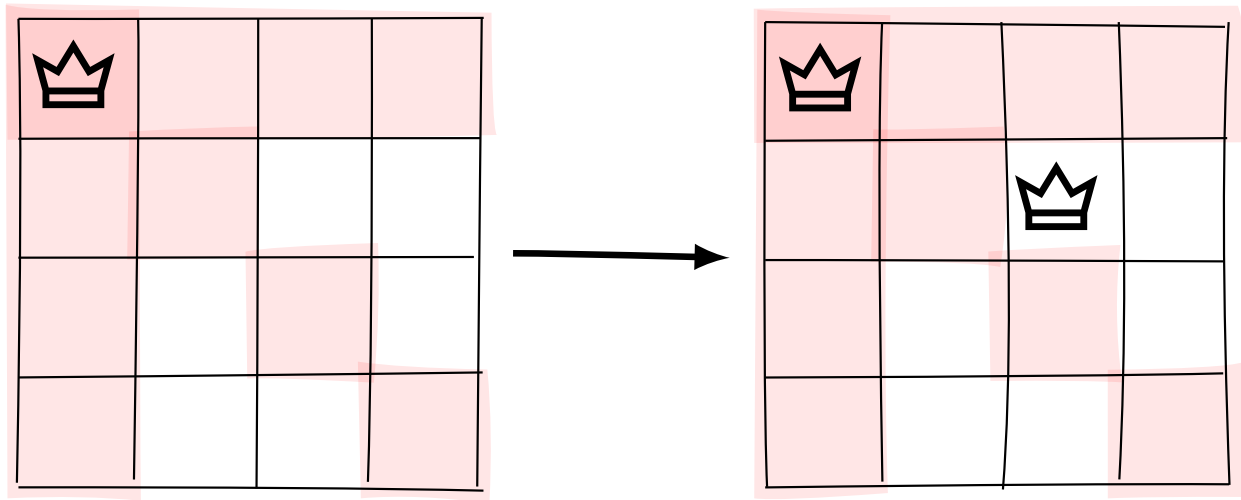
Solução 1 Solução X

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

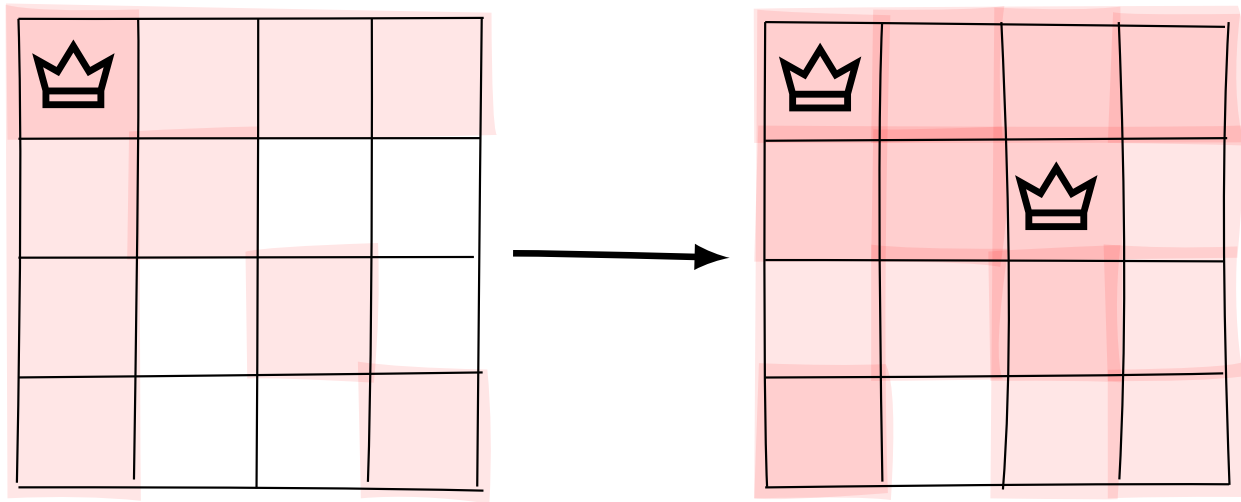


Solução 1 Solução 2

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



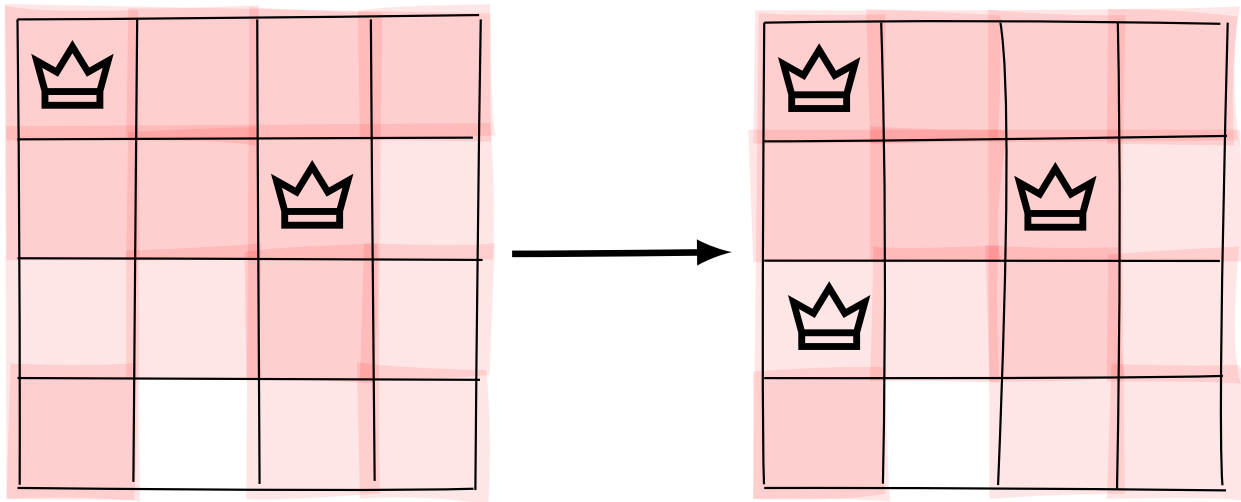
Solução 1 Solução 2

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



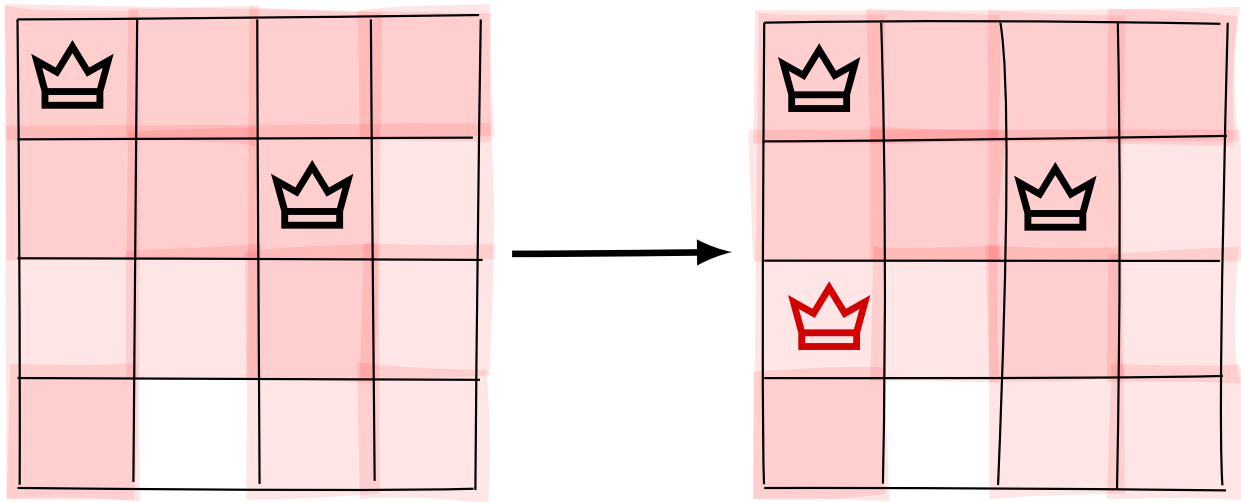
Solução 2

Solução Y

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



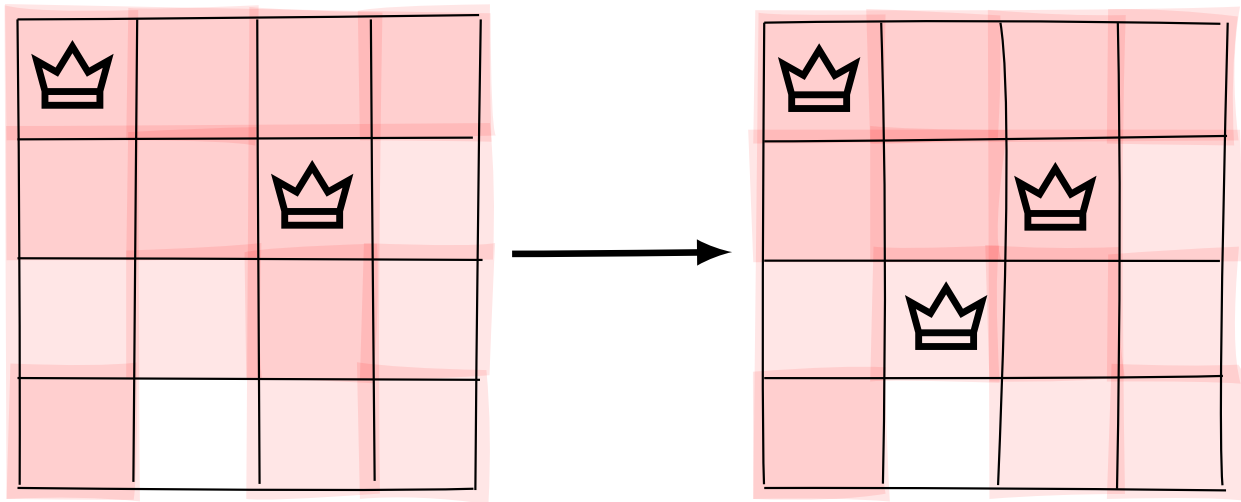
Solução 2 → Solução Y

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

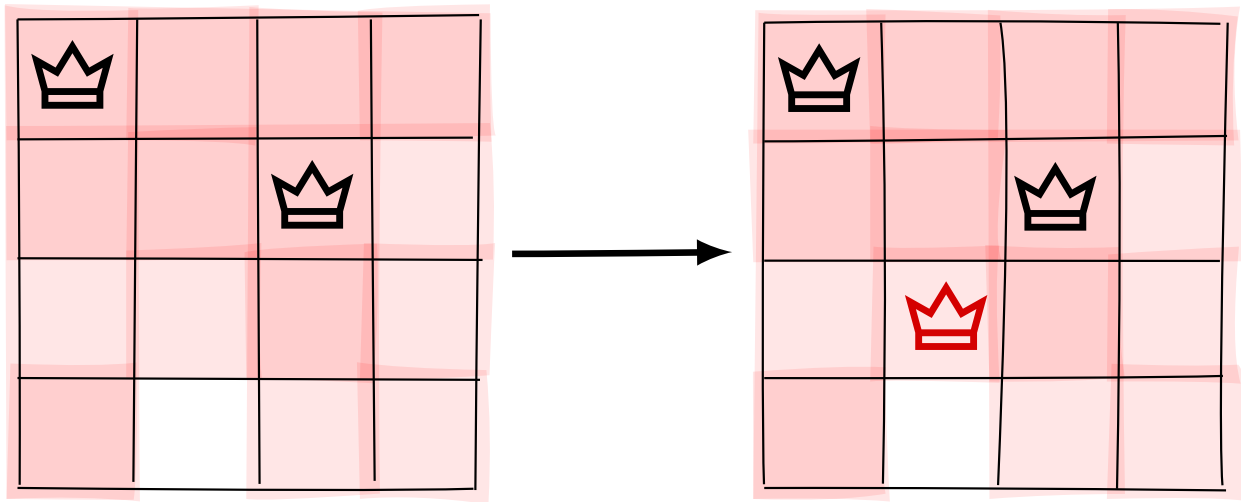


Solução 2 → Solução Y

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



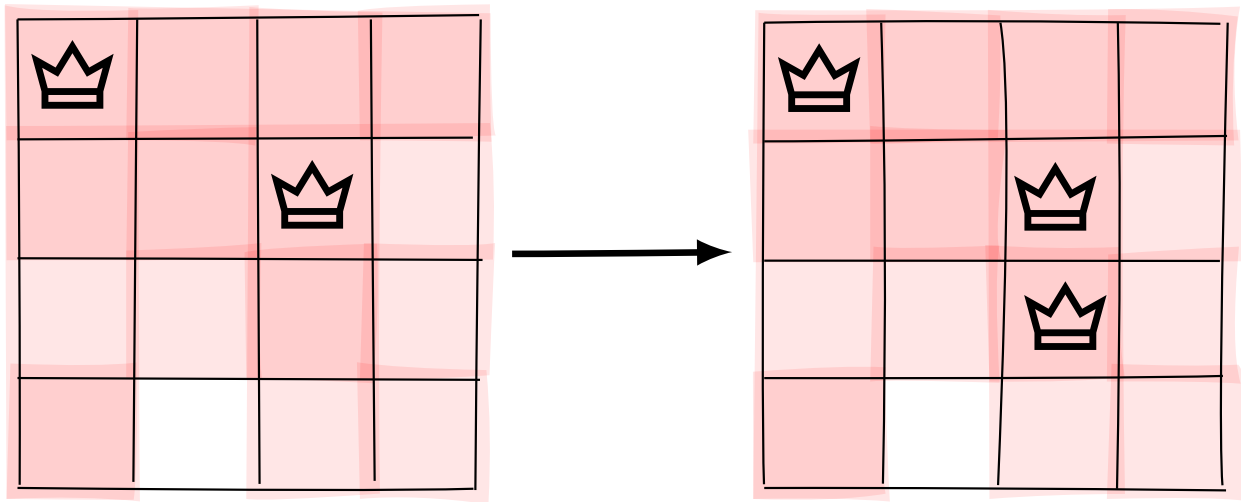
Solução 2 → Solução Y

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



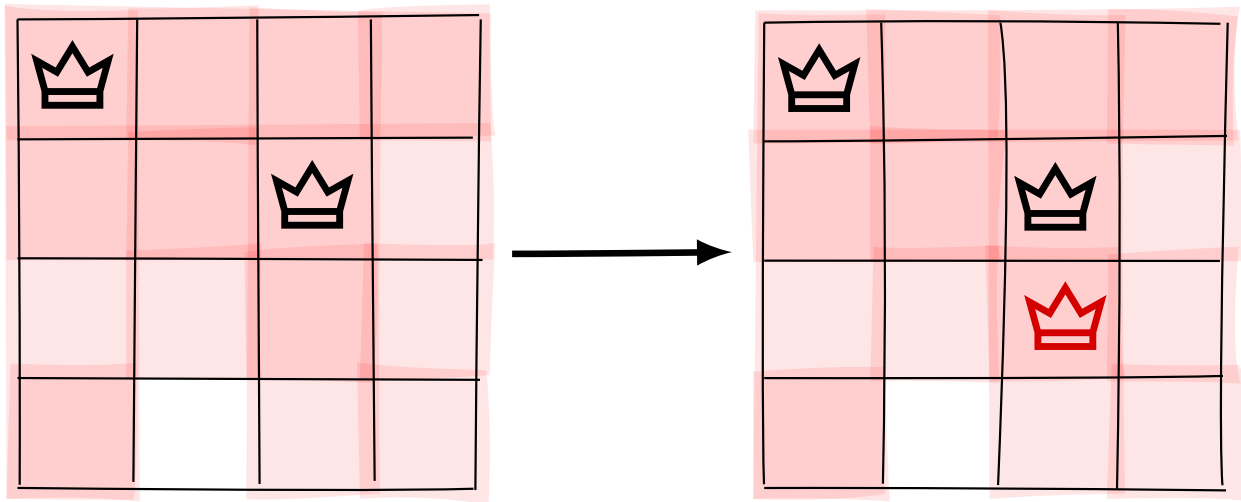
Solução 2

Solução Y

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



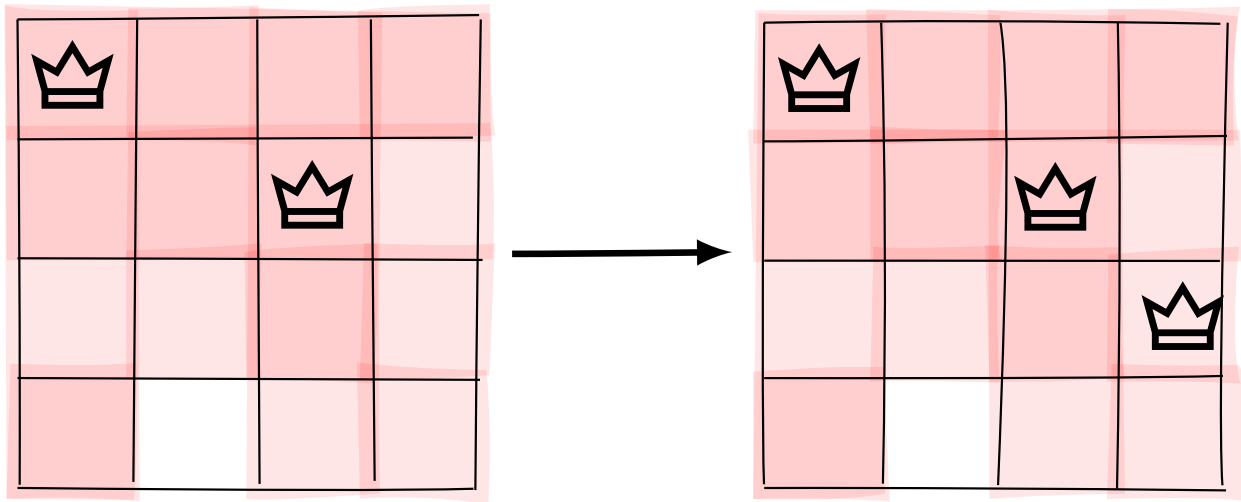
Solução 2 → Solução Y

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



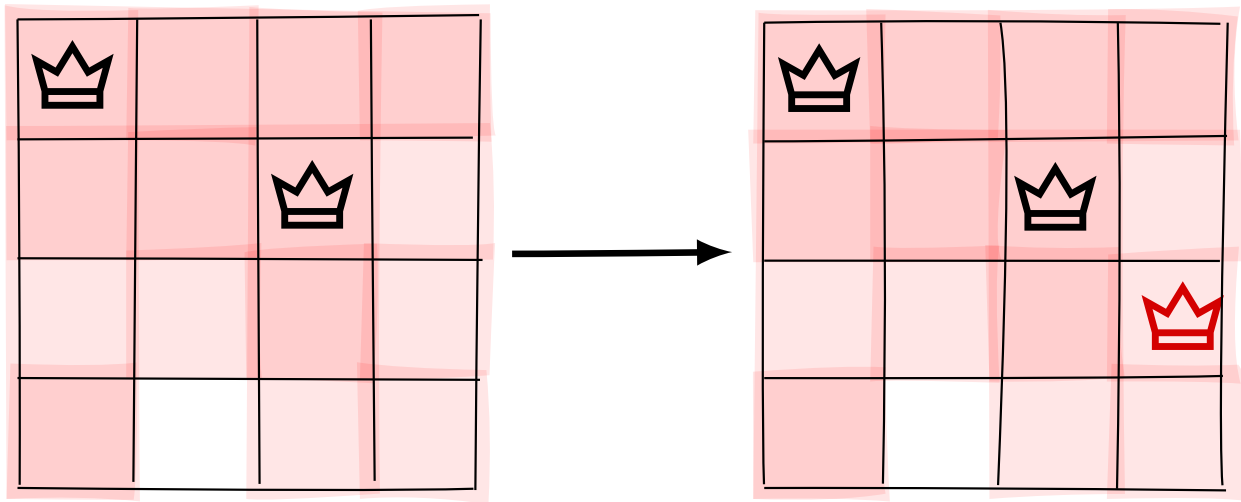
Solução 2

Solução Y

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

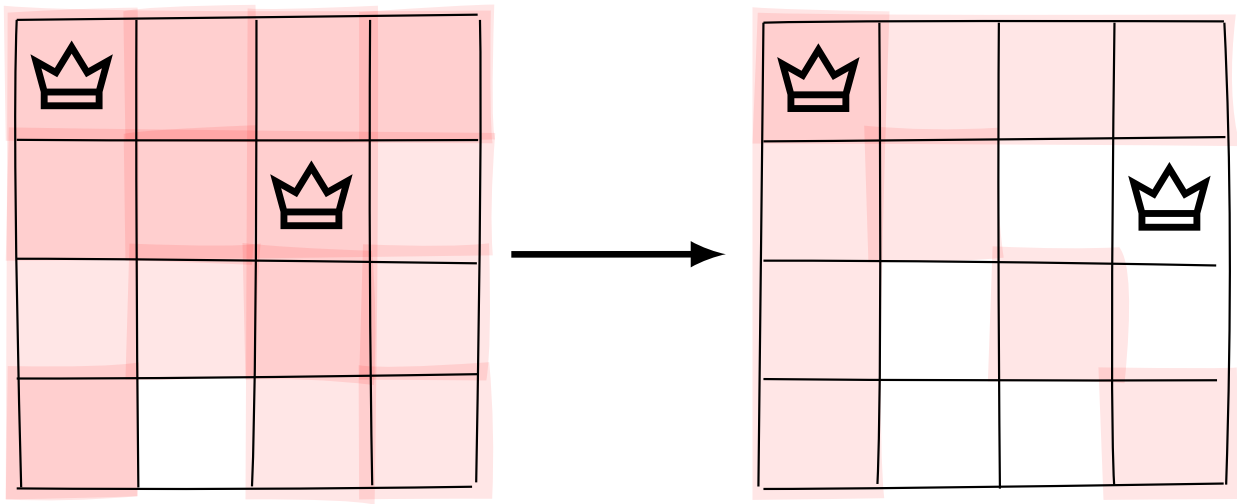
Solução Y

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

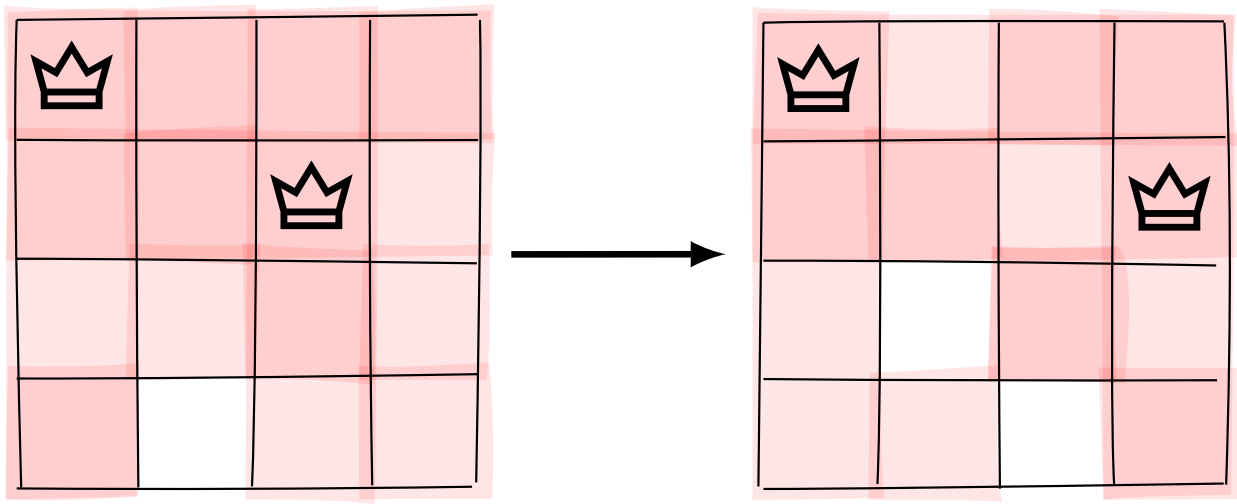


Solução 2 Solução 3

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



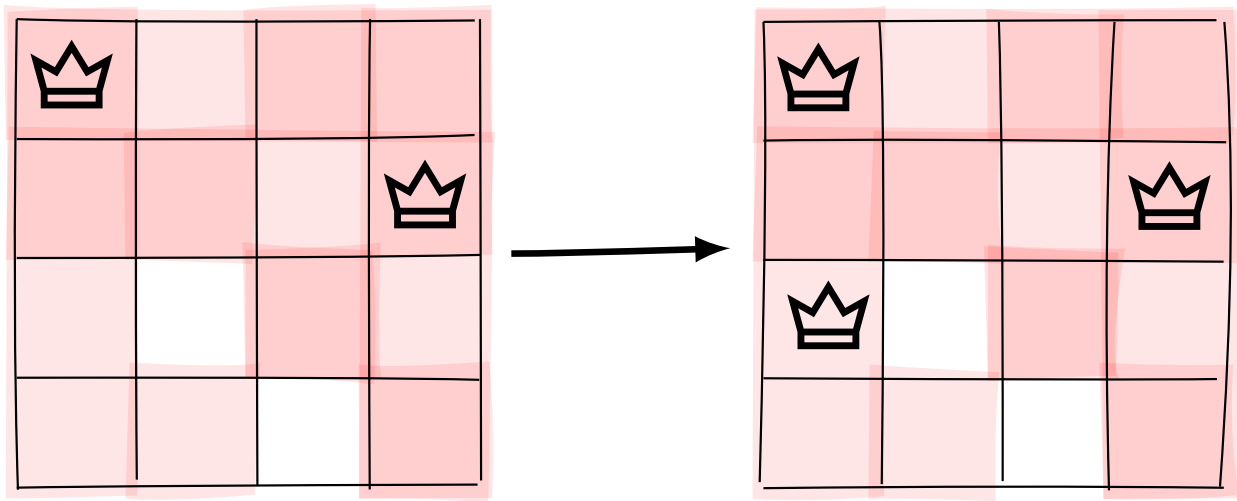
Solução 2 Solução 3

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

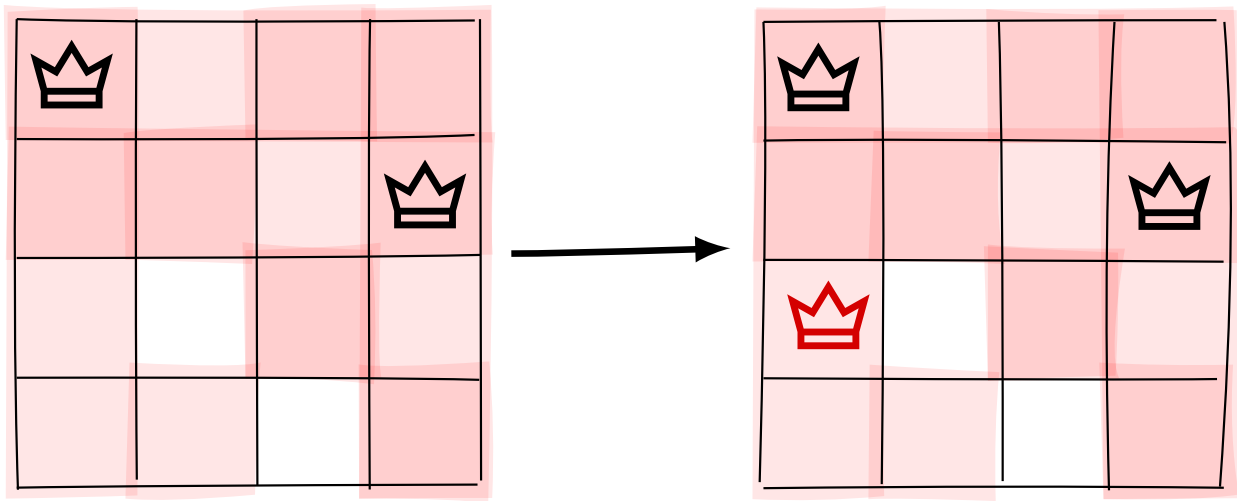


Solução 3 Solução Y

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



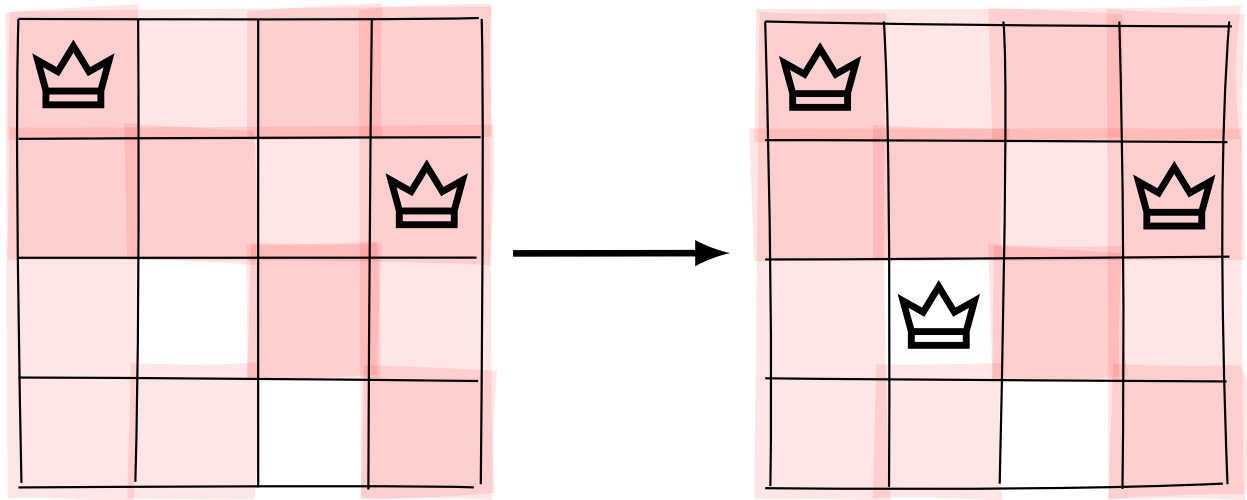
Solução 3 → Solução X

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

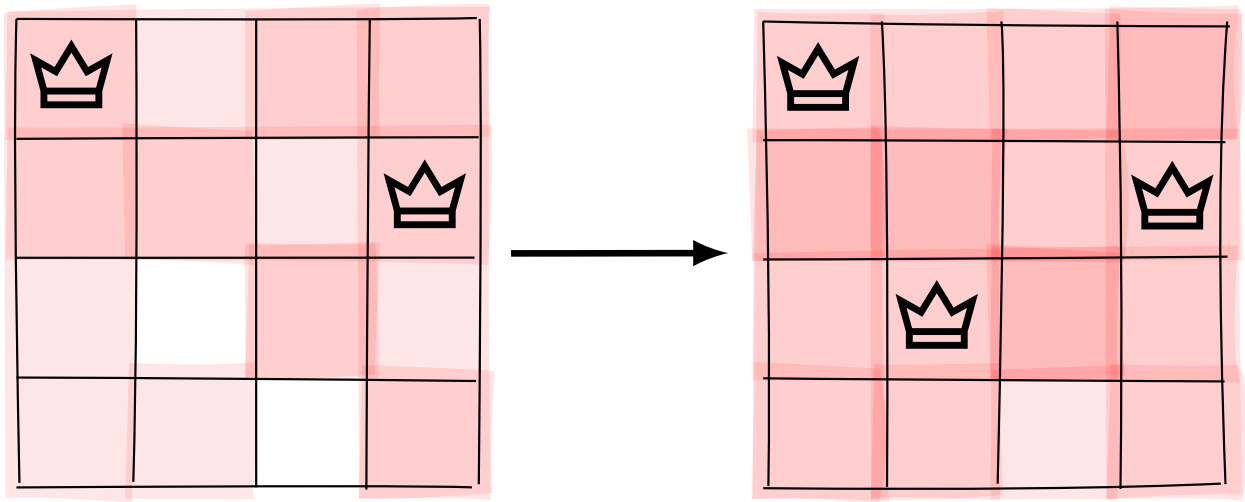


Solução 3 Solução 4

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



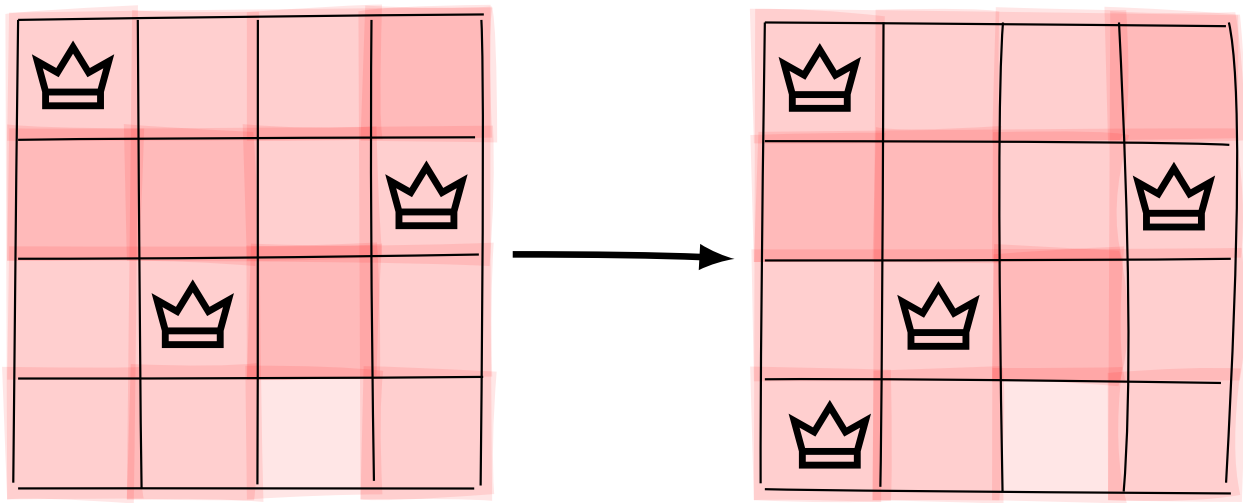
Solução 3 Solução 4

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



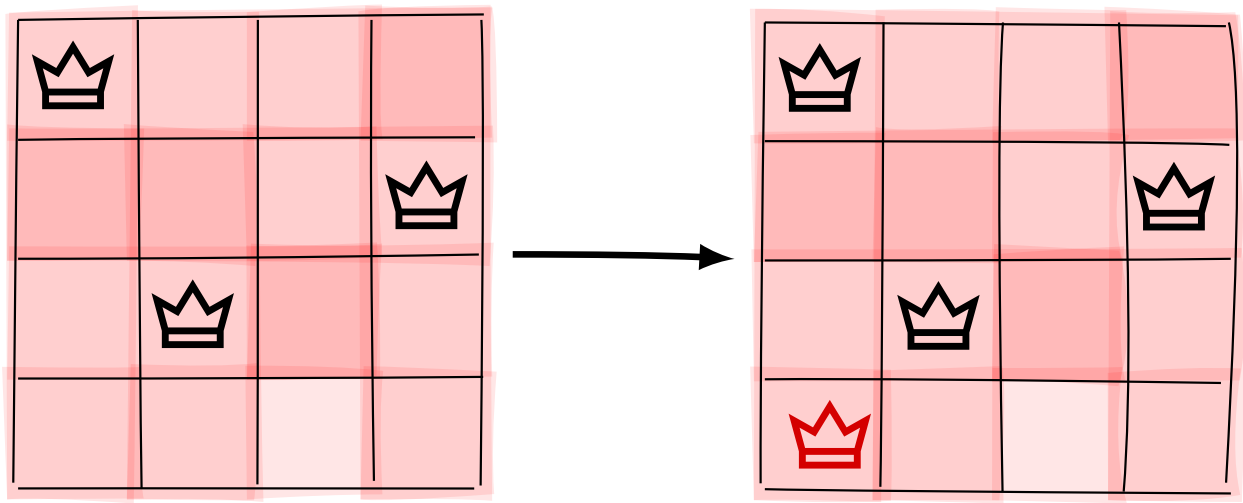
Solução 4

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



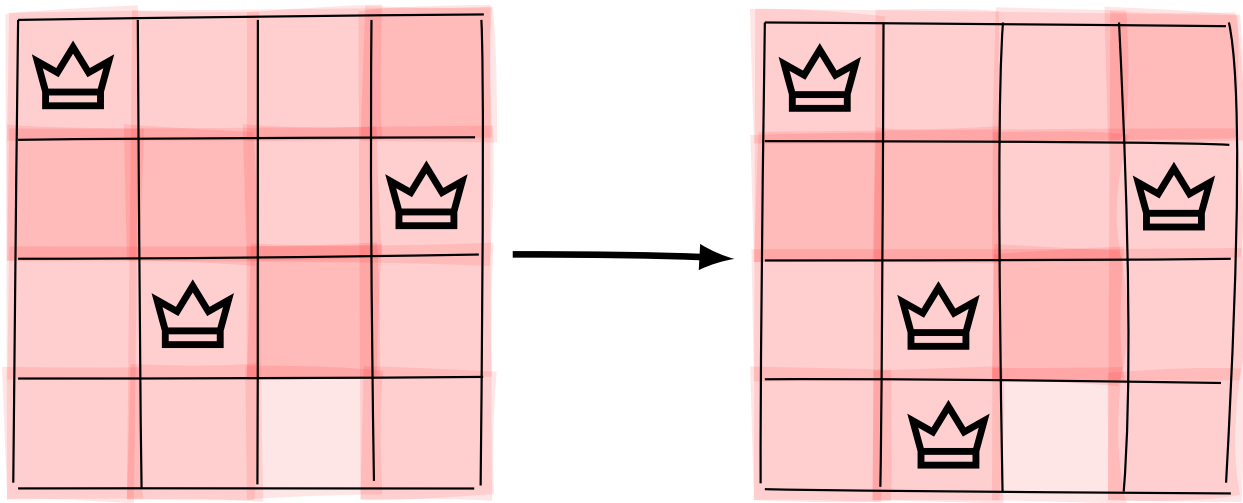
Solução 4 → Solução X

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



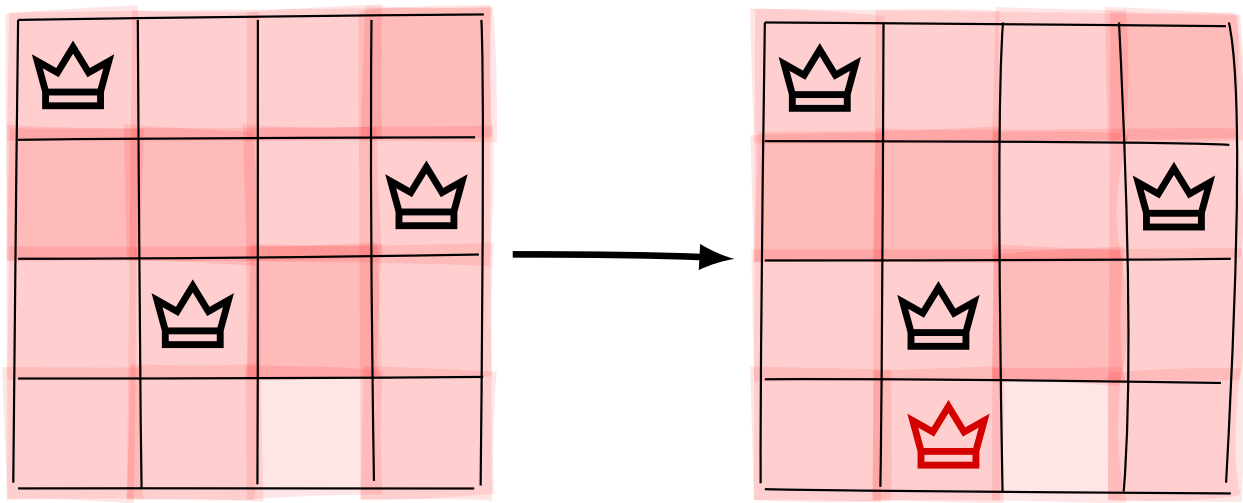
Solução 4

Solução Y

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

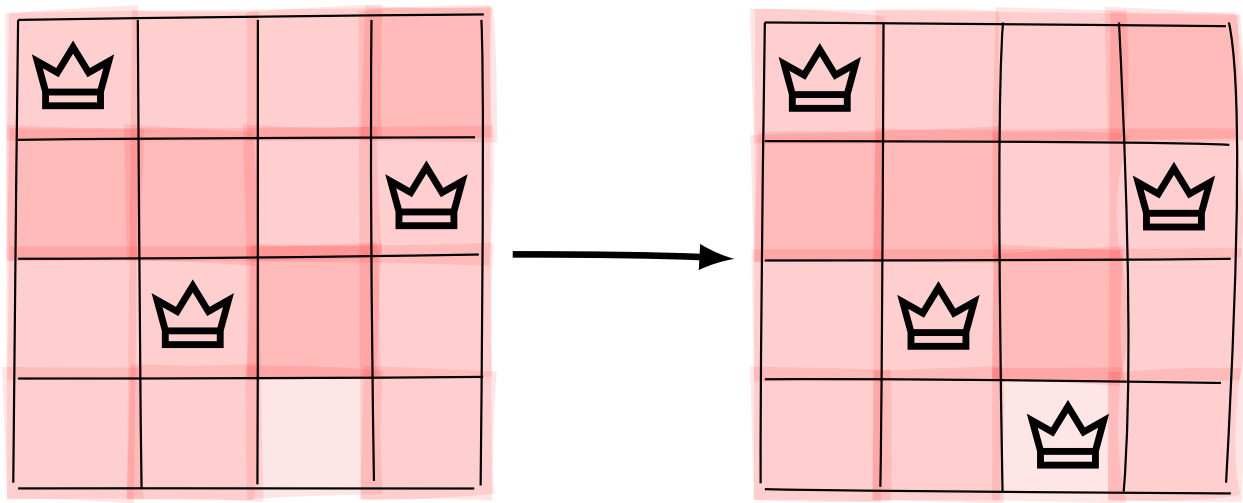
Solução X

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

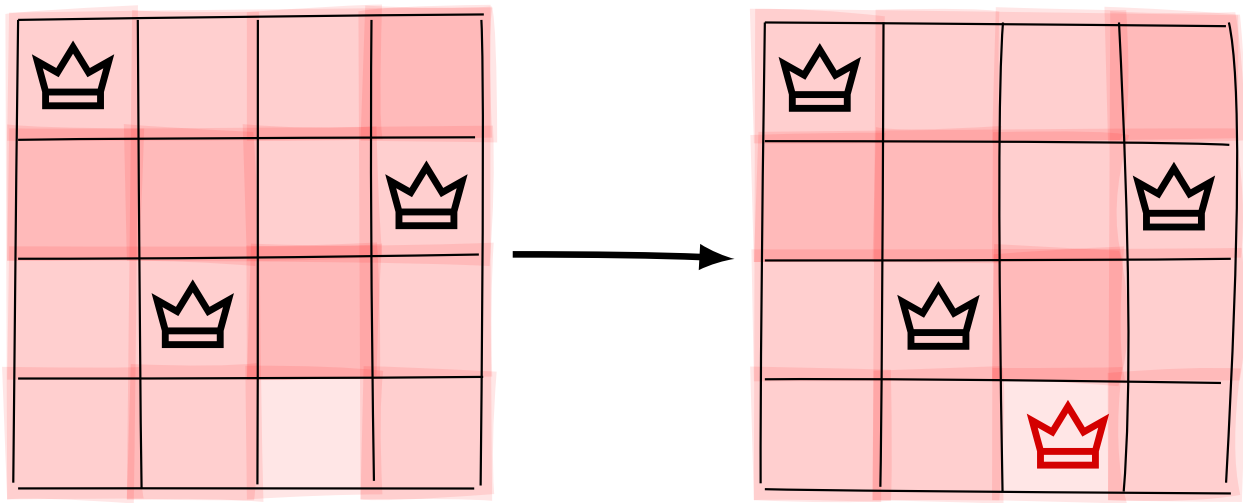


Solução 4 → Solução X

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



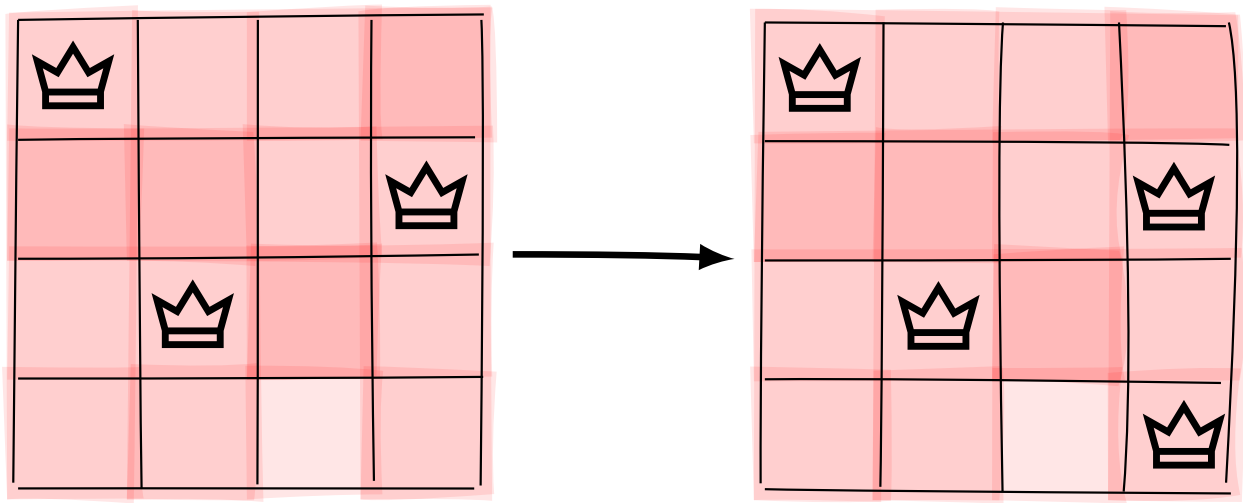
Solução 4 → Solução X

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

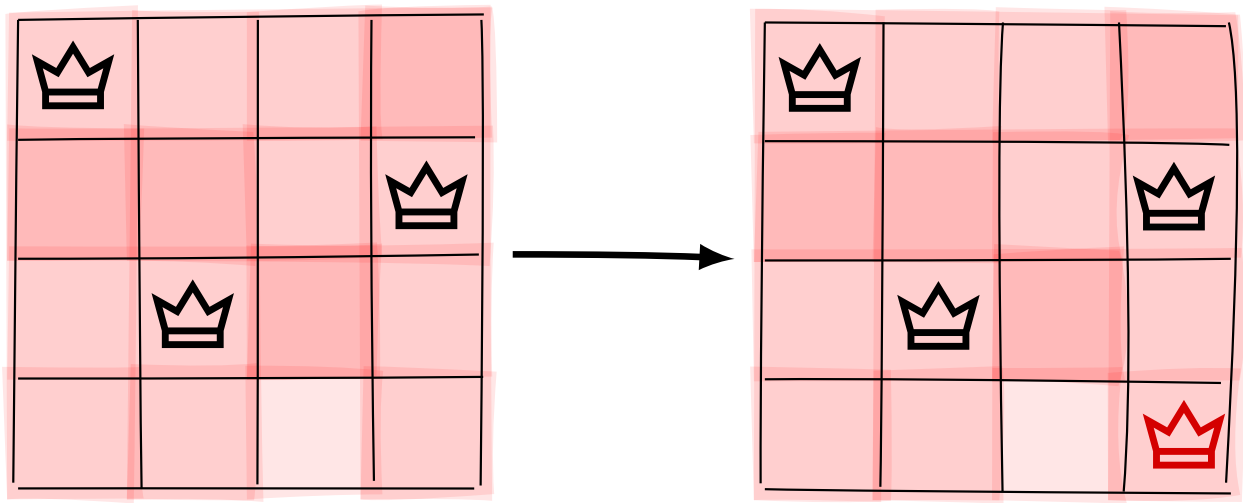


Solução 4 → Solução X

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



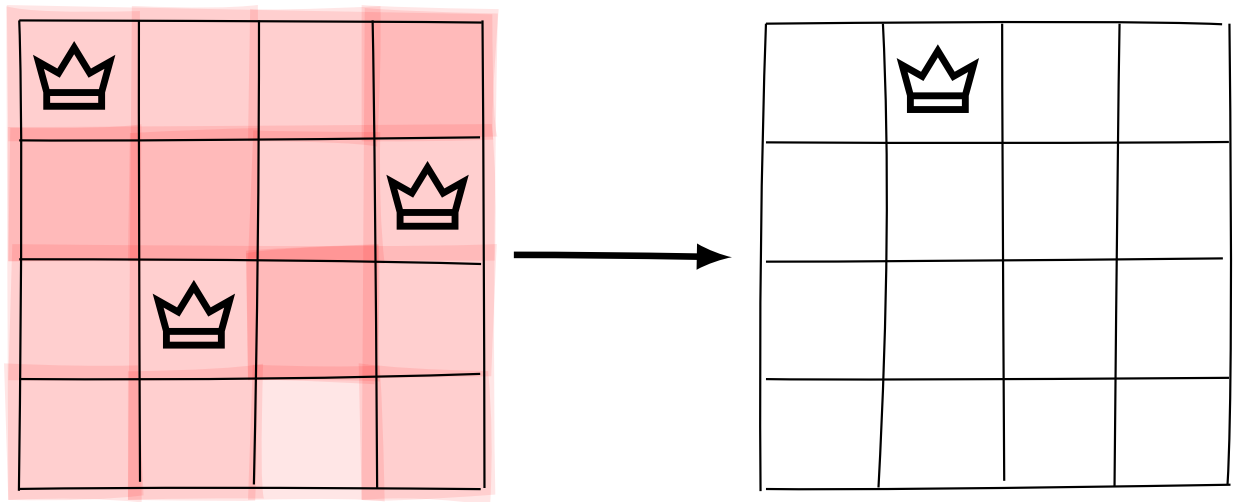
Solução 4 → Solução X

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

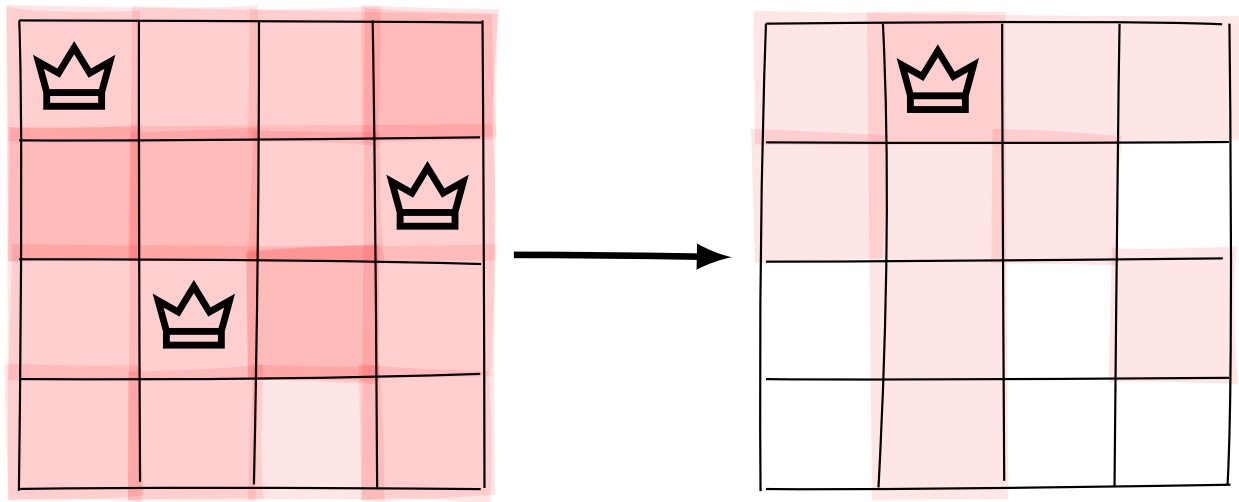


Solução 4 Solução 5

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



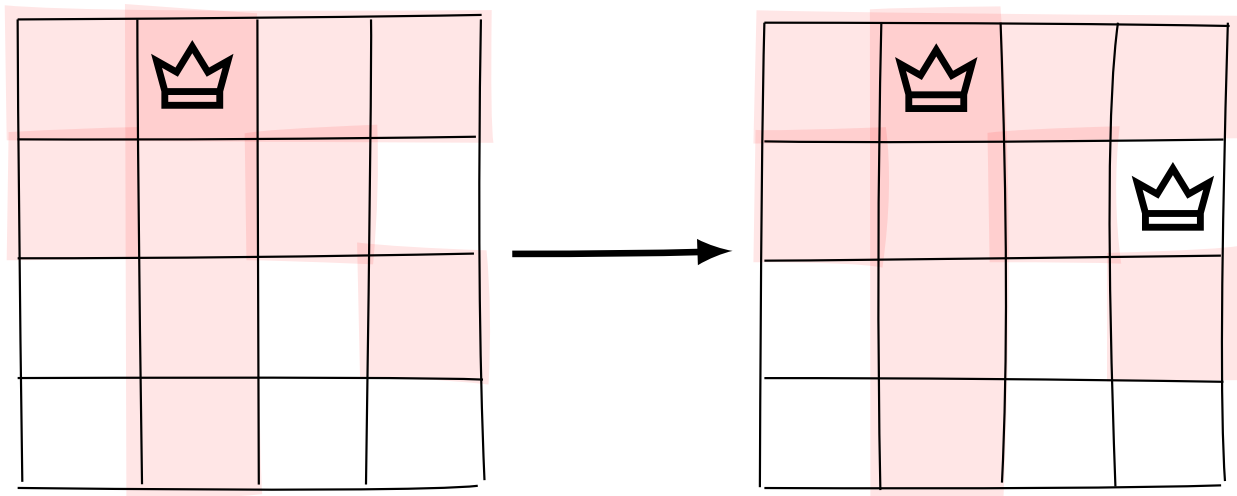
Solução 4 Solução 5

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

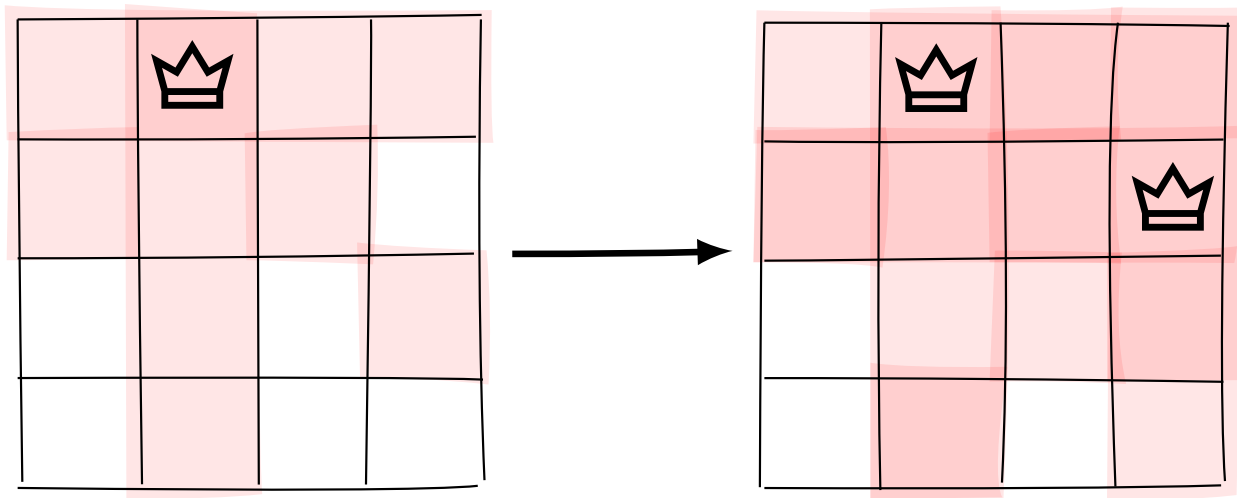


Solução 5 Solução 6

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



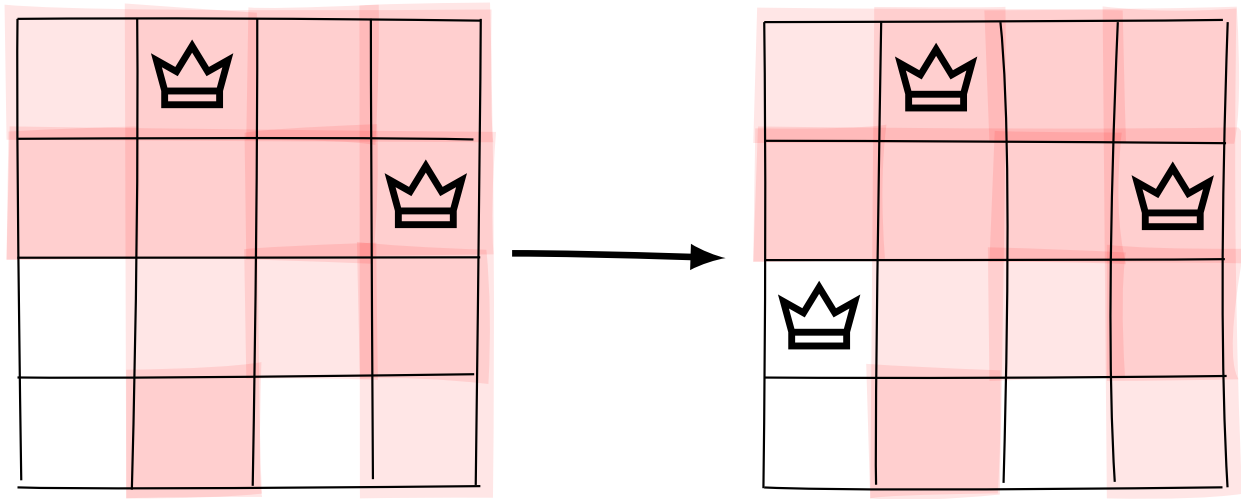
Solução 5 Solução 6

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

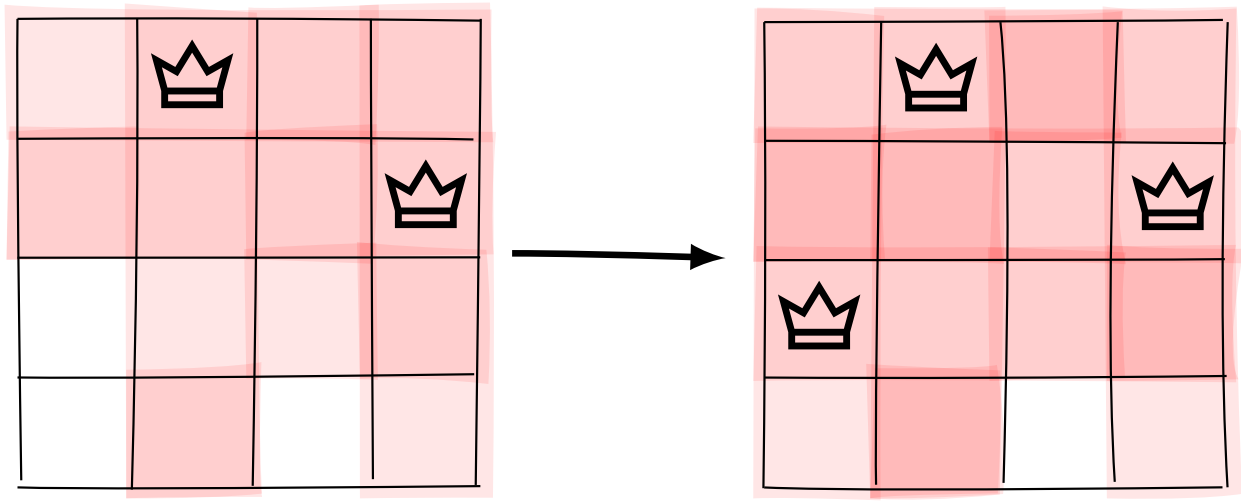


Solução 6 Solução 7

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



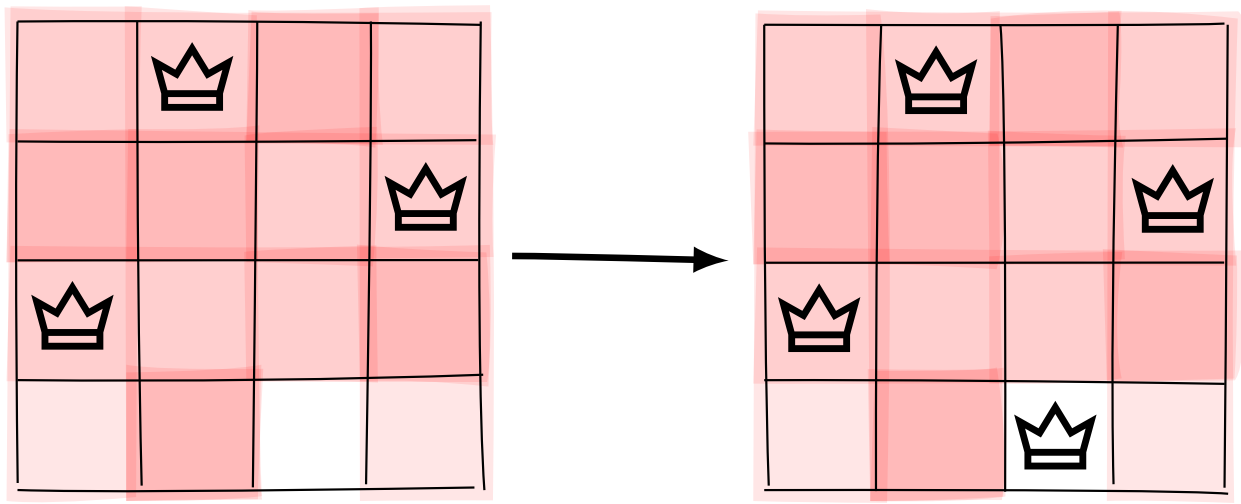
Solução 6 Solução 7

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

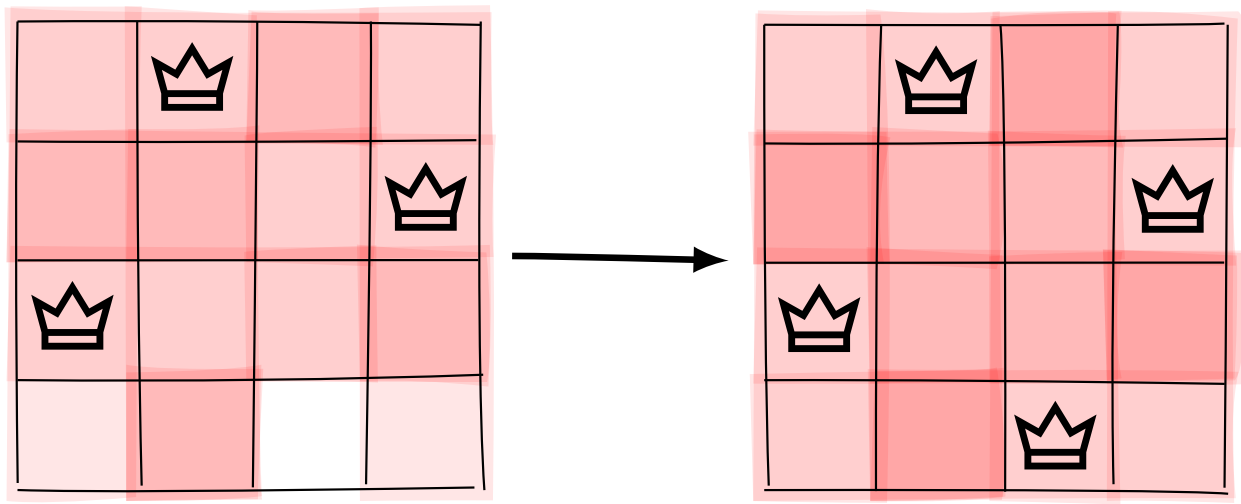


Solução 7 Solução 8

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



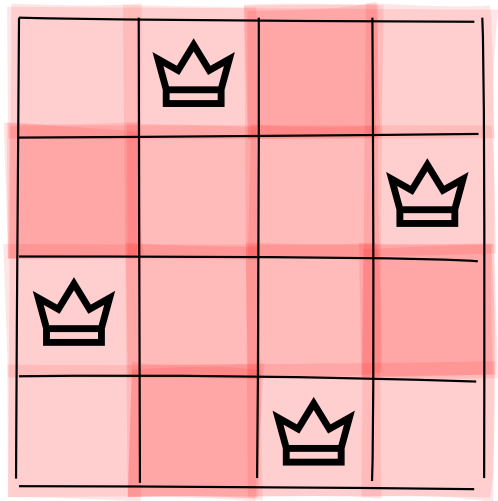
Solução 7 Solução 8

A rainha 4 foi posicionada em (3, 2)

Sem violação de ataque

Backtracking

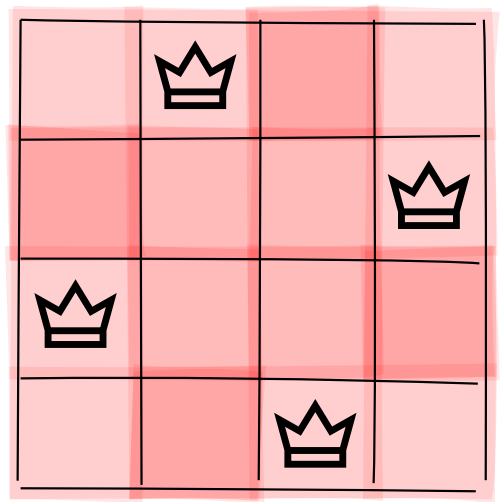
- Problema das 4-rainhas: em um tabuleiro 4×4, cada rainha é posicionada em linhas distintas



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

- 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

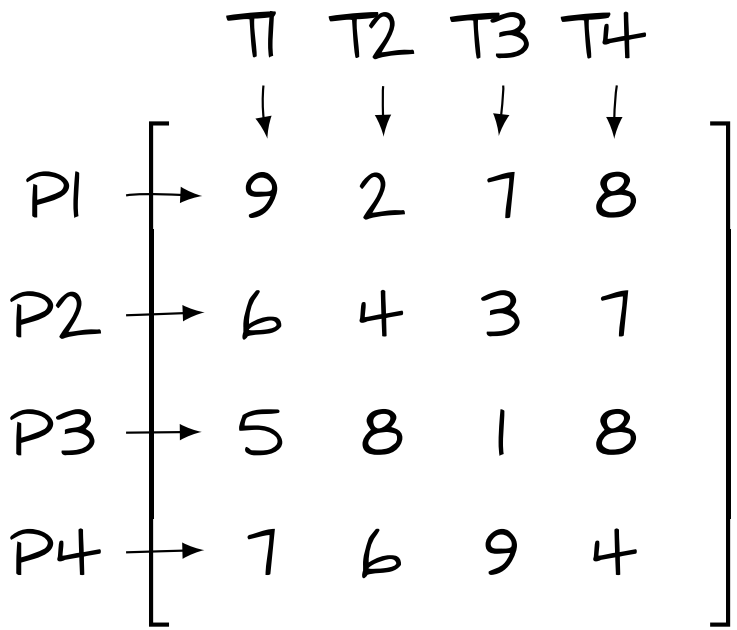
Branch-and-bound

- 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

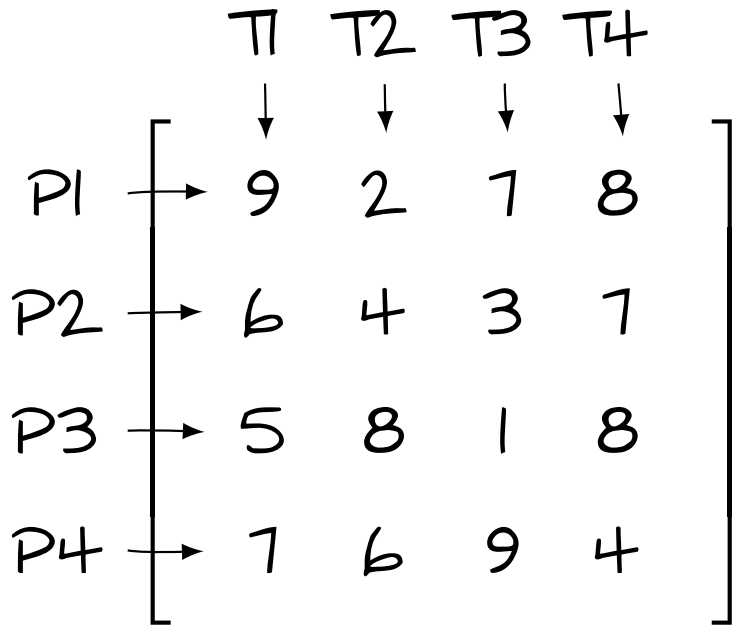
- Problema de alocar *n* pessoas para *n* trabalhos
  - Cada pessoa pode realizar somente um trabalho
  - A matriz armazena o valor que cada pessoa (linhas) recebe para realizar um trabalho (colunas)



Minimização do custo total
para realização dos trabalhos

Branch-and-bound

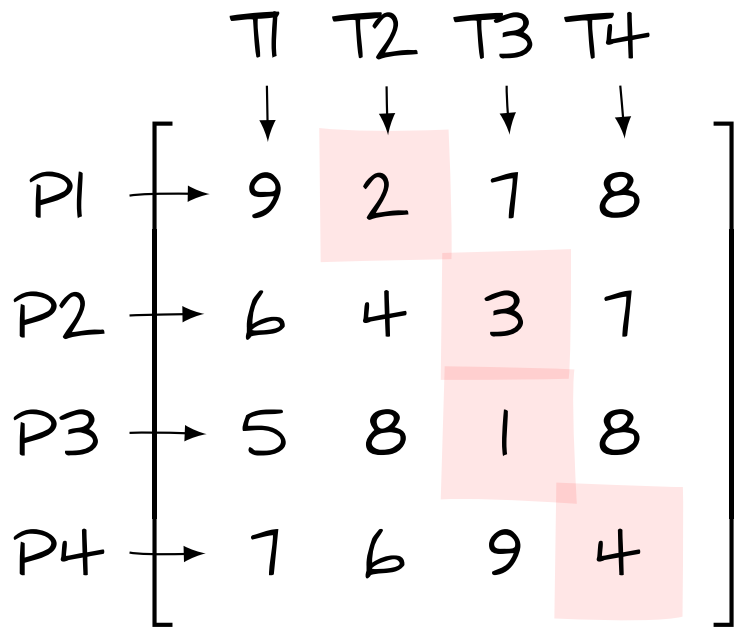
- Problema de alocar *n* pessoas para *n* trabalhos
  - Como parte da estratégia de minimização, o menor custo de cada pessoa (linha) para um trabalho (coluna) é selecionado para obter o limite inferior



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

Branch-and-bound

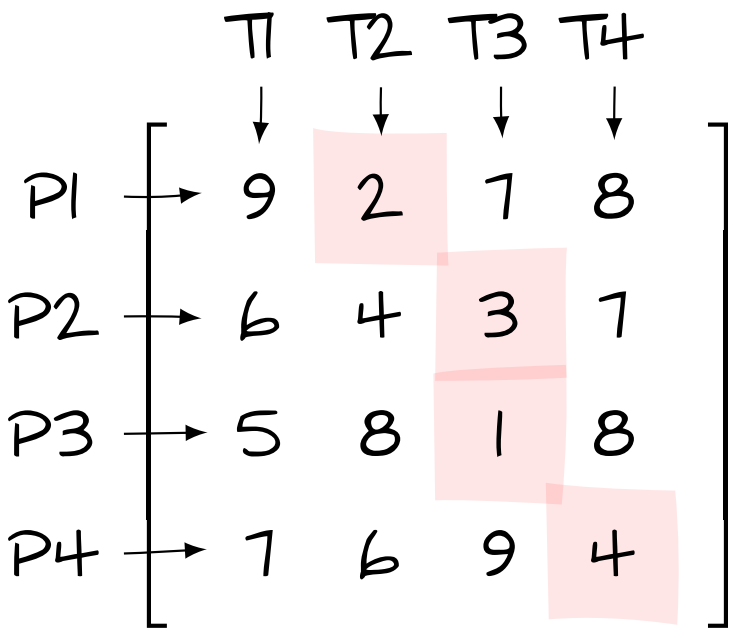
- Problema de alocar *n* pessoas para *n* trabalhos
  - Como parte da estratégia de minimização, o menor custo de cada pessoa (linha) para um trabalho (coluna) é selecionado para obter o limite inferior

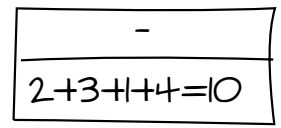


O limite inferior é 2 + 3 + 1 + 4 = 10

Branch-and-bound

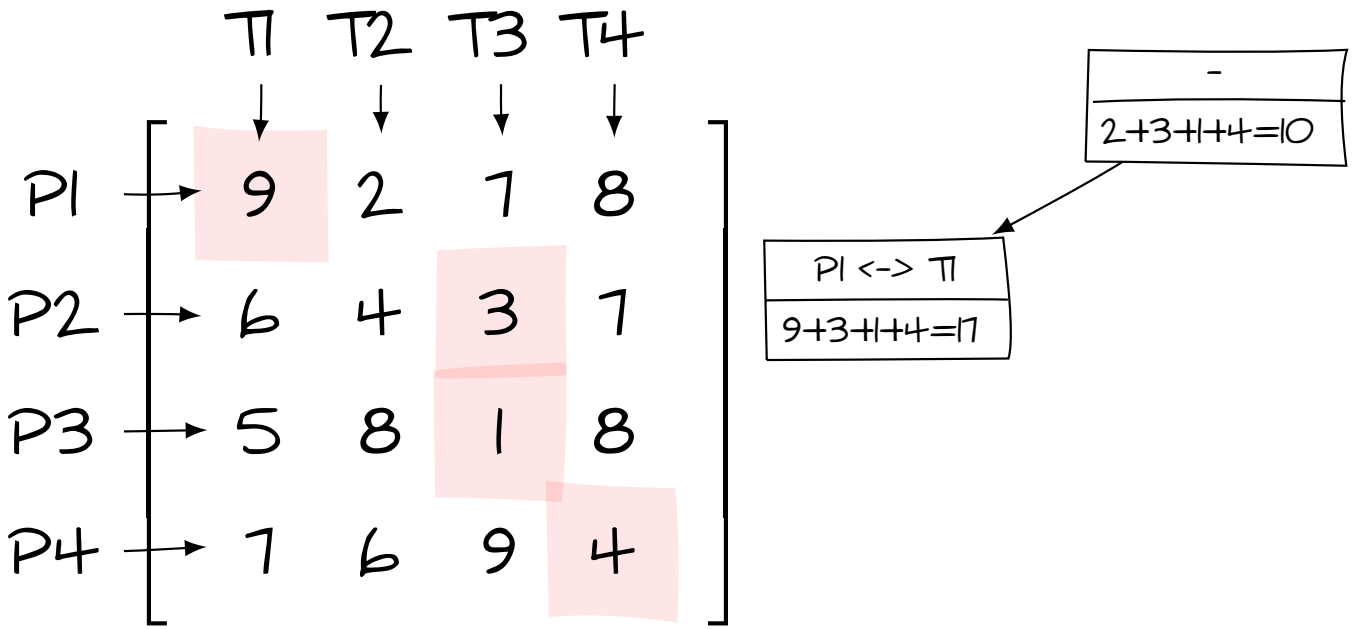
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1





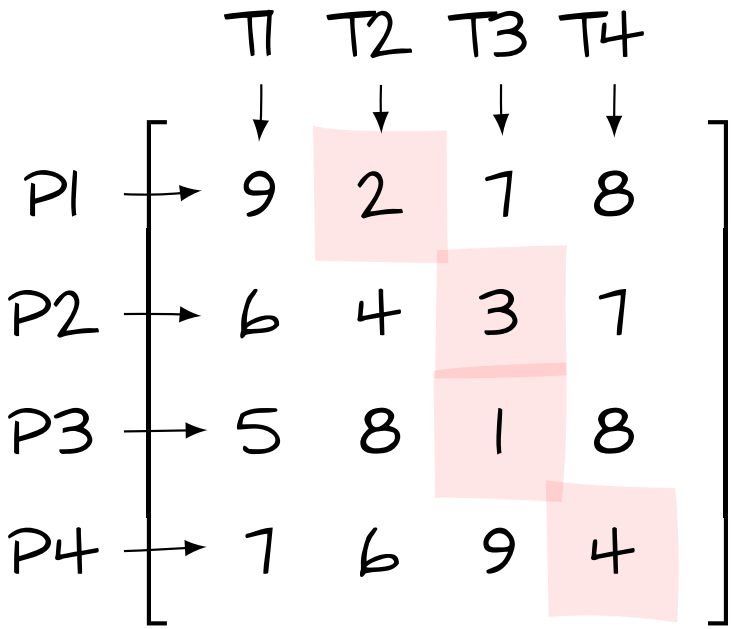
Branch-and-bound

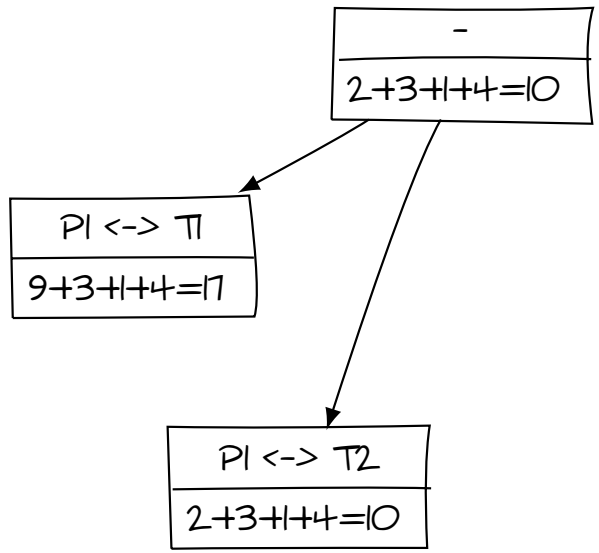
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1



Branch-and-bound

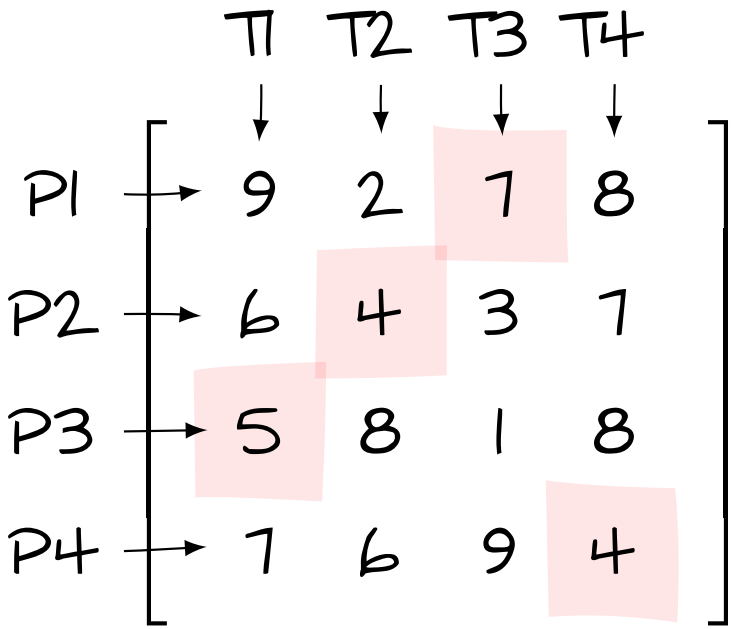
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

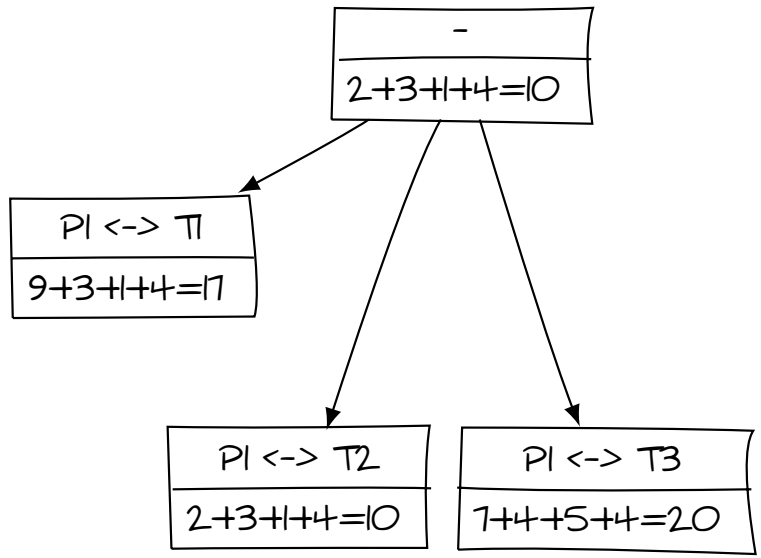




Branch-and-bound

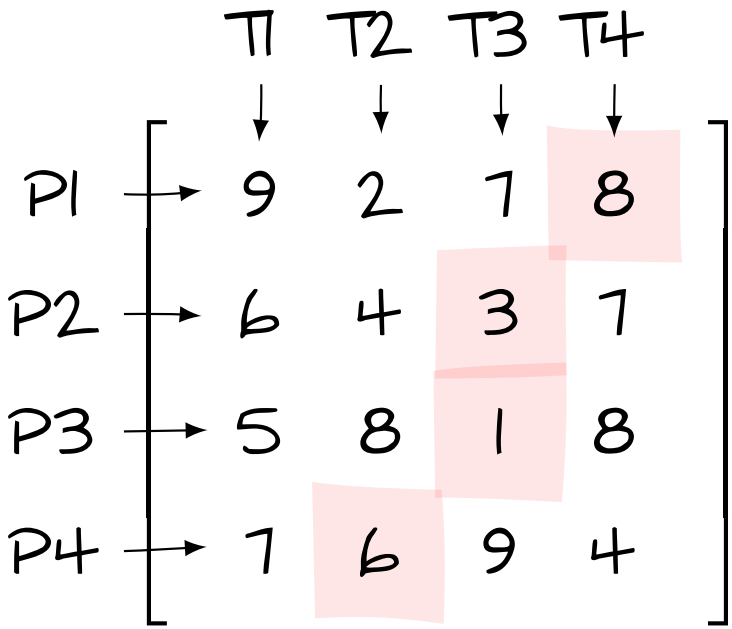
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

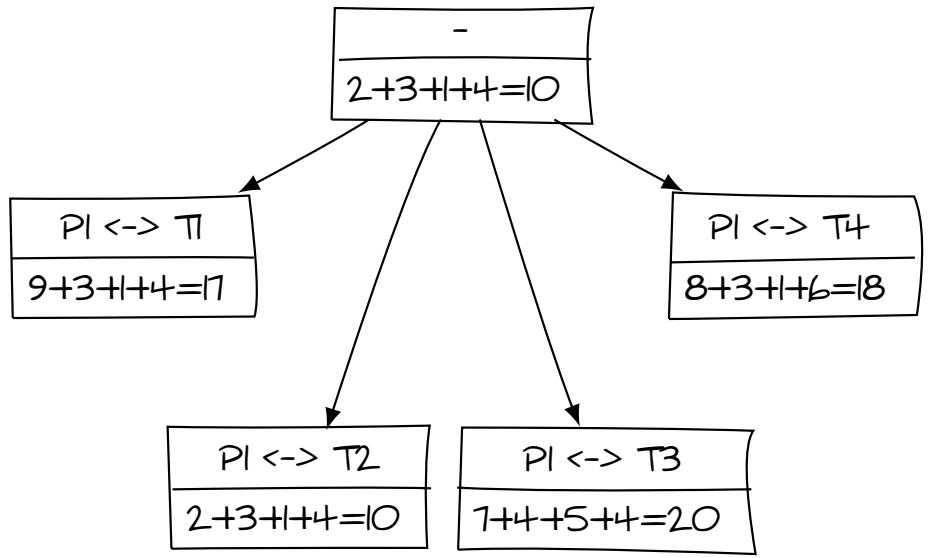




Branch-and-bound

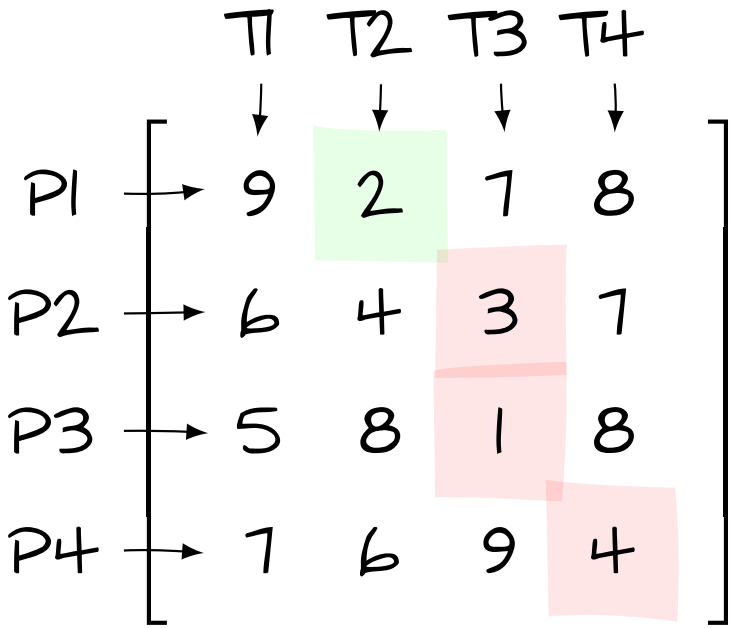
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*1

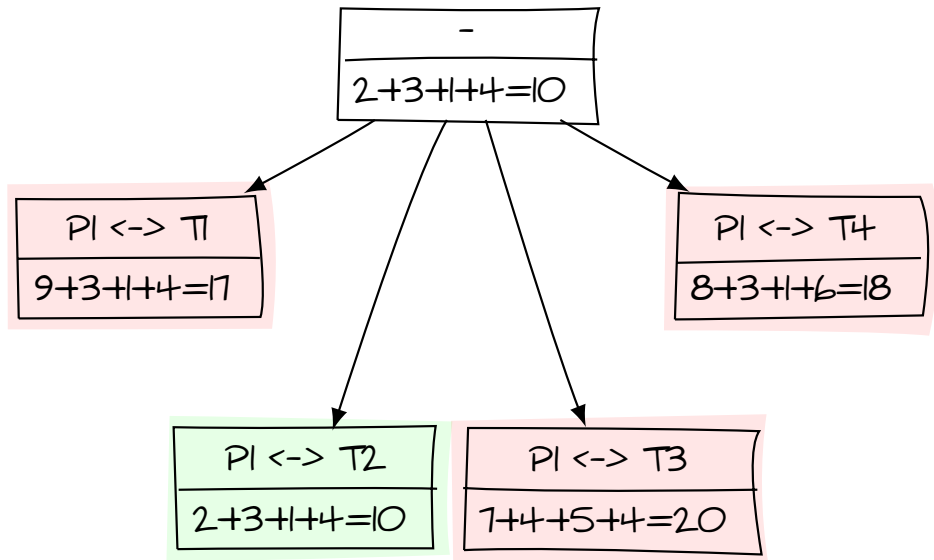




Branch-and-bound

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

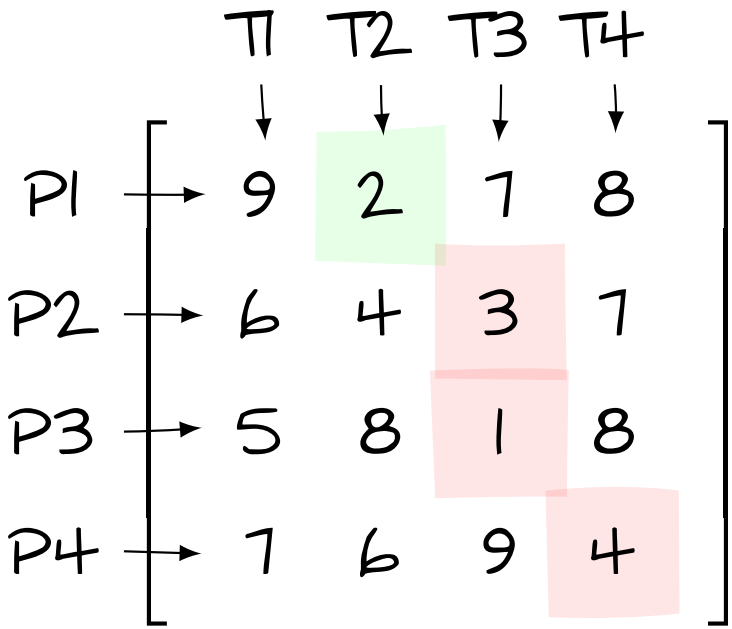




A solução parcial mais promissora aloca a pessoa P1 para o trabalho T2

Branch-and-bound

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

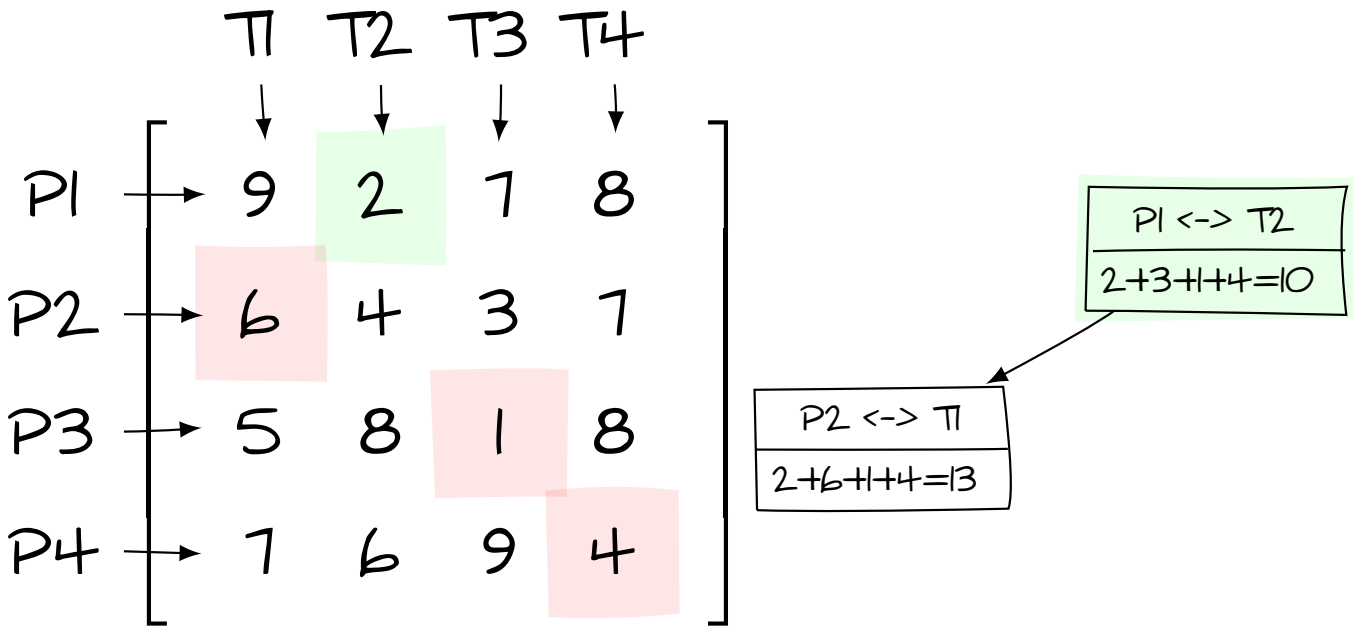


P1 <-> T2

2+3+1+4=10

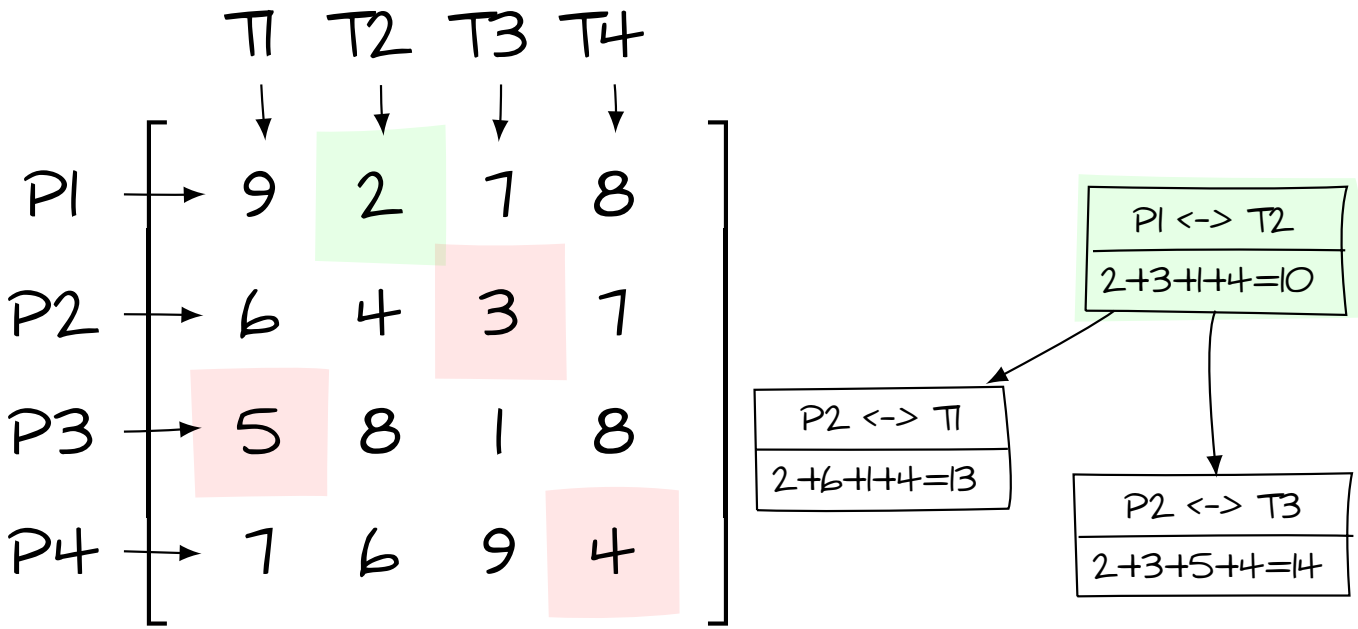
Branch-and-bound

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



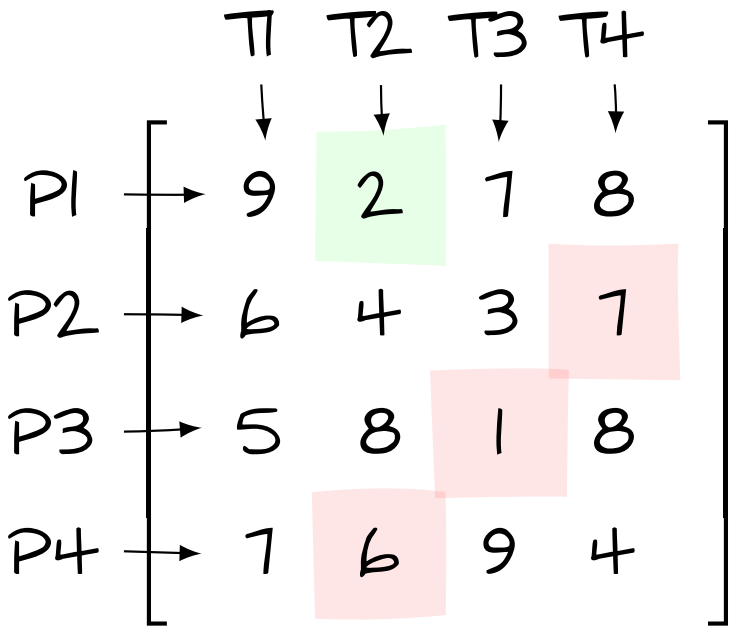
Branch-and-bound

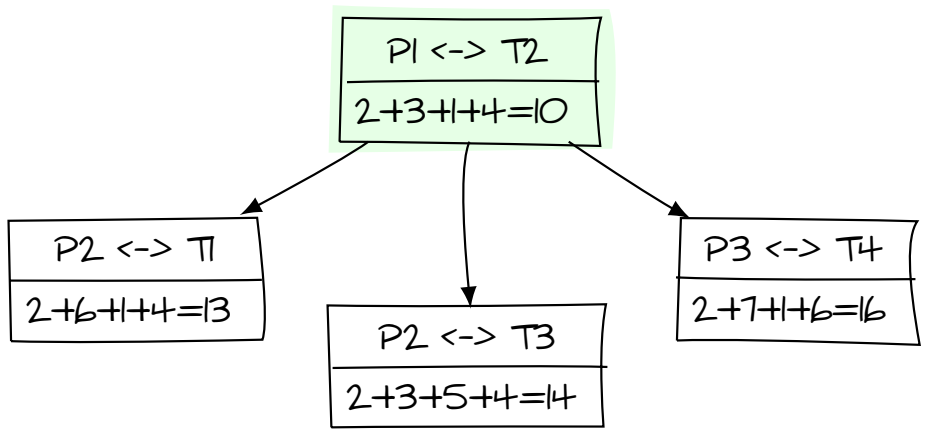
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P2*



Branch-and-bound

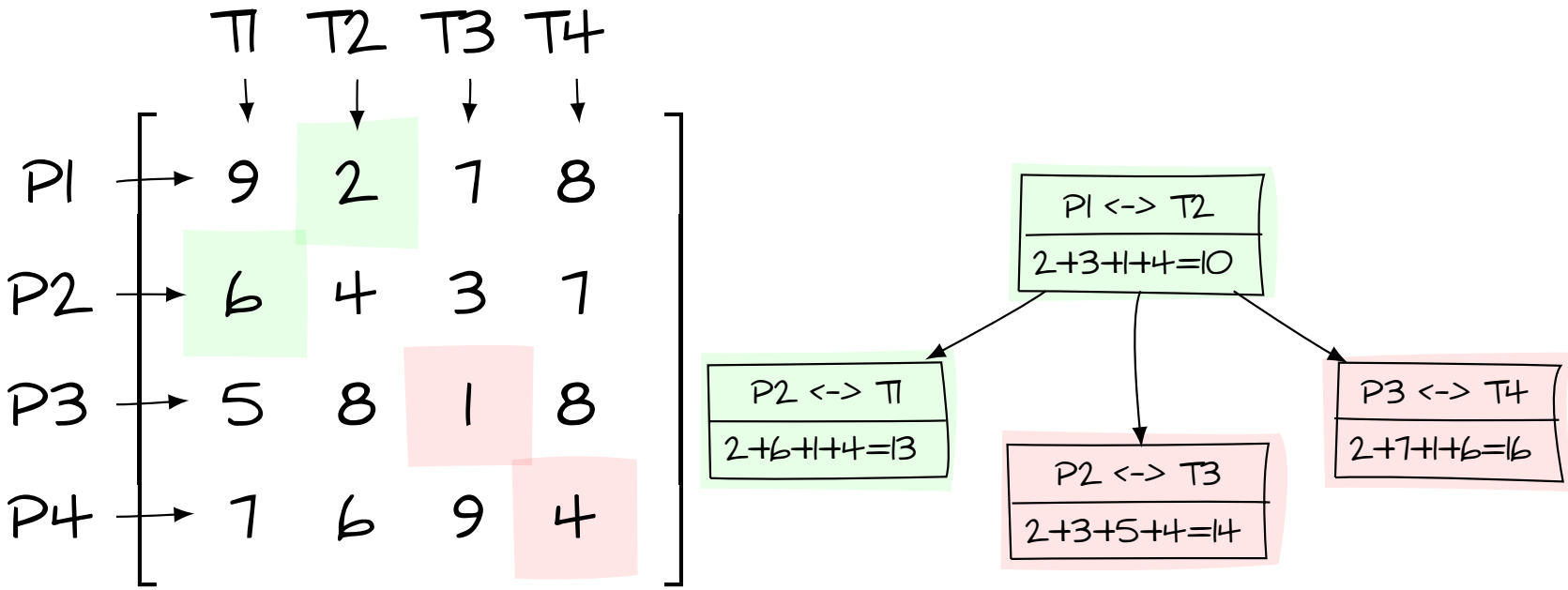
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P2*





Branch-and-bound

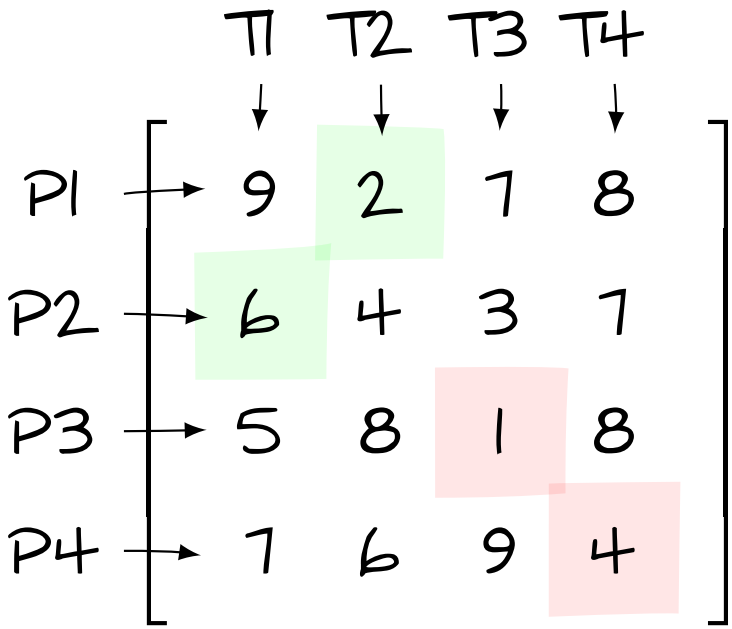
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P2*



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

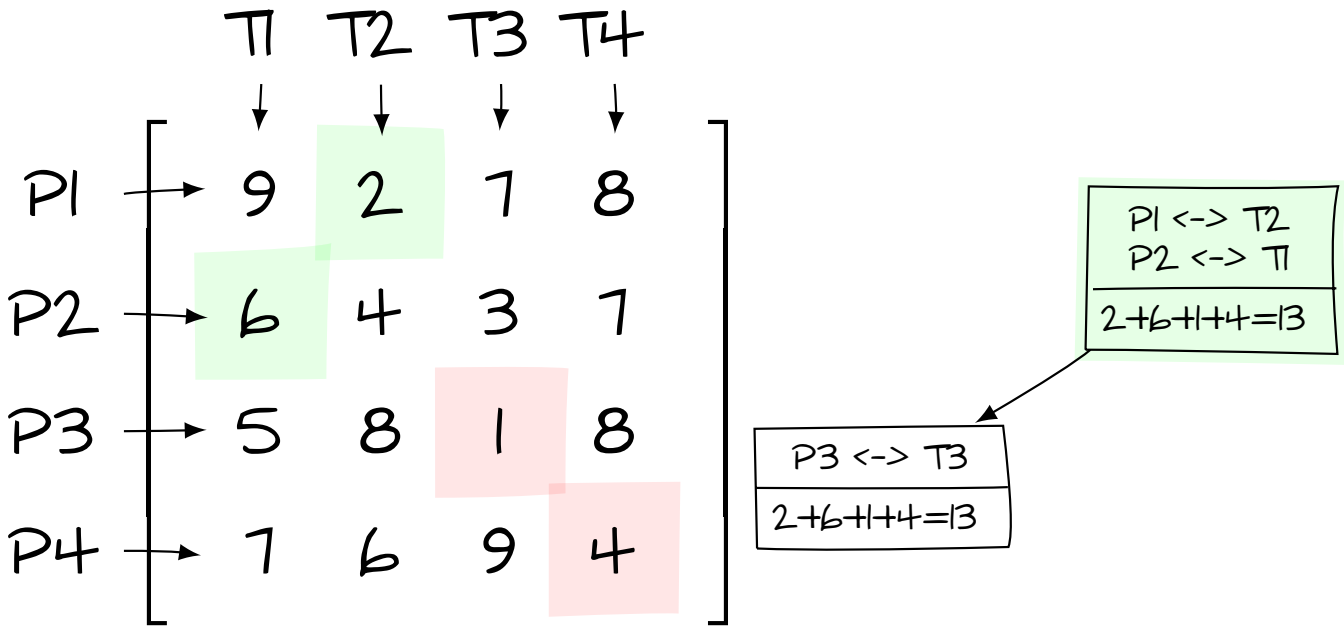


P1 <-> T2
P2 <-> T1

2+6+1+4=13

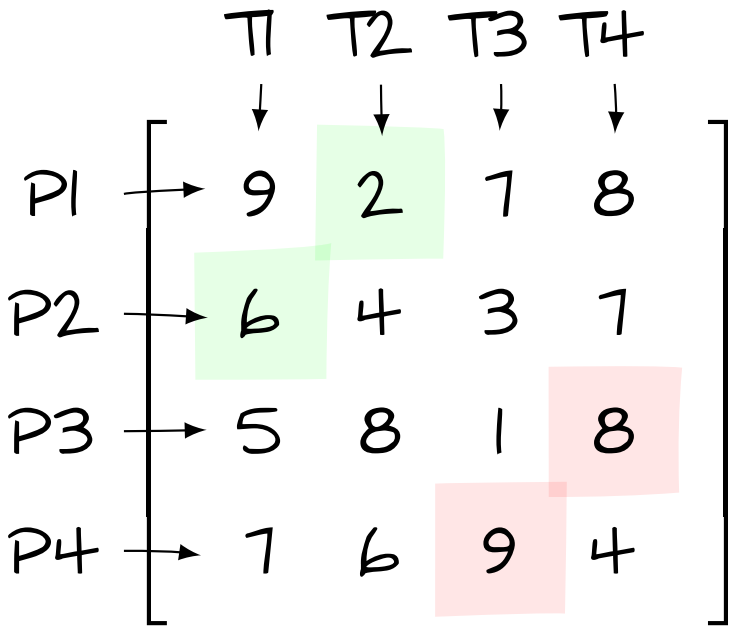
Branch-and-bound

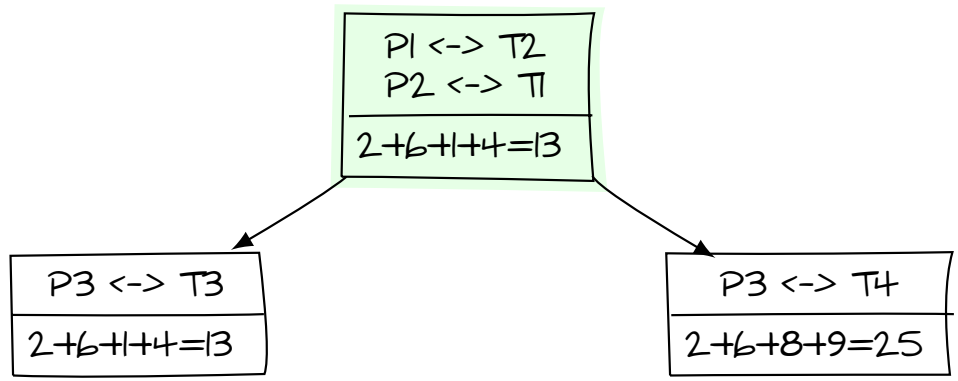
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*3



Branch-and-bound

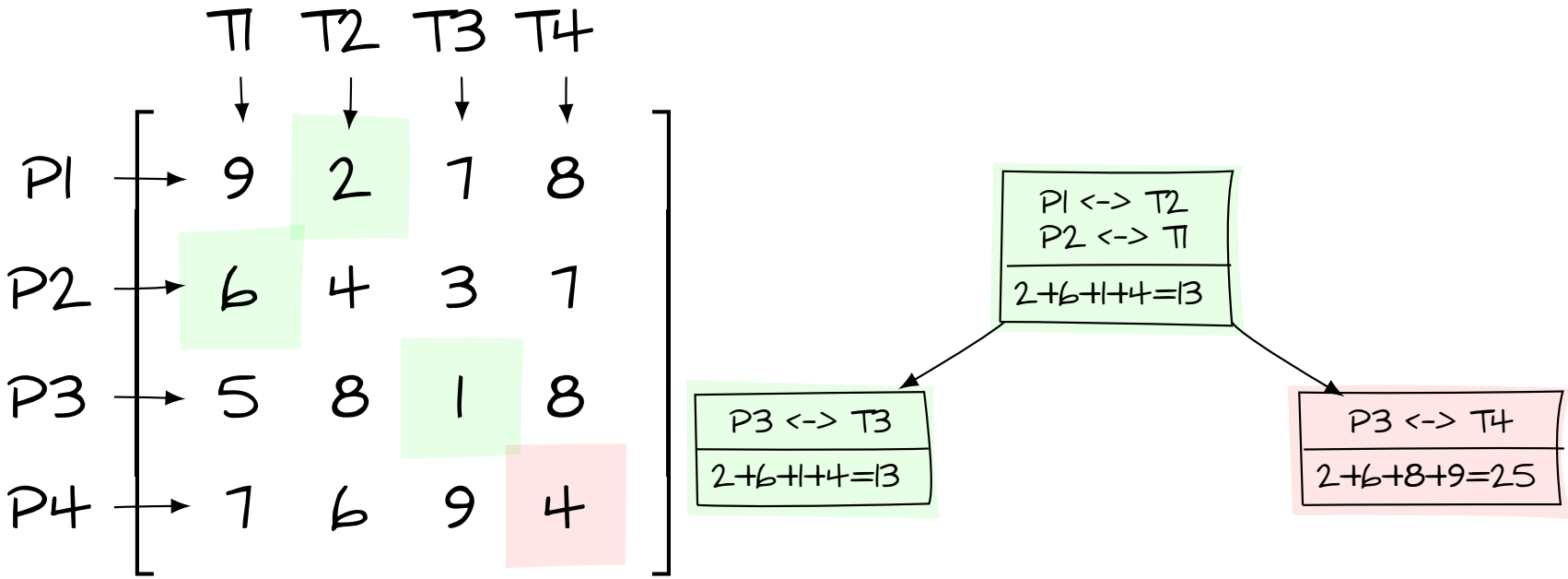
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P*3





Branch-and-bound

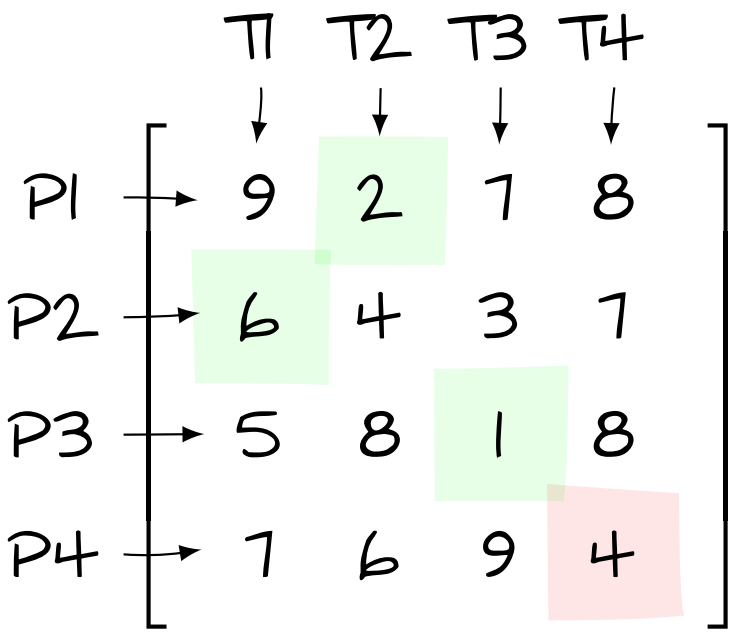
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P3*



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 *P4*

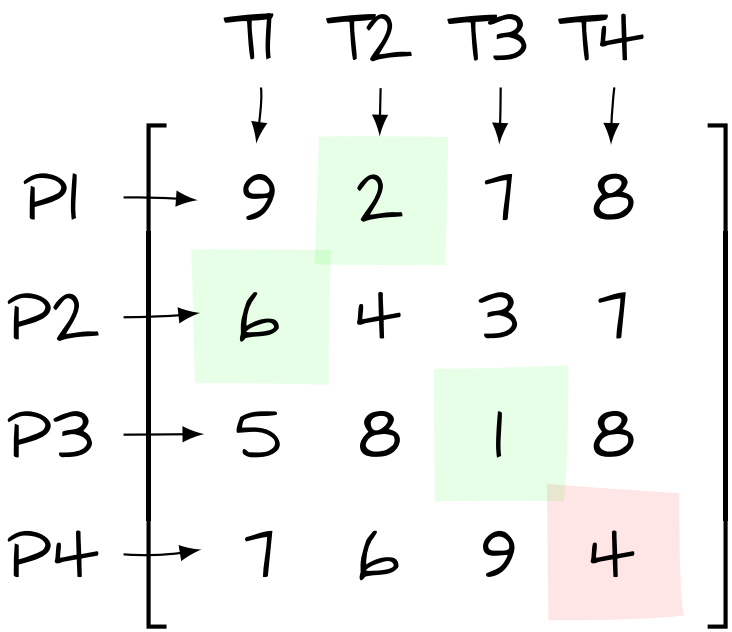


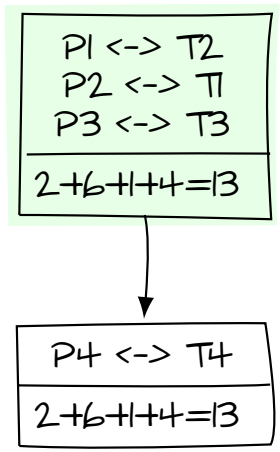
P1 <-> T2
P2 <-> T1
P3 <-> T3

2+6+1+4=13

Branch-and-bound

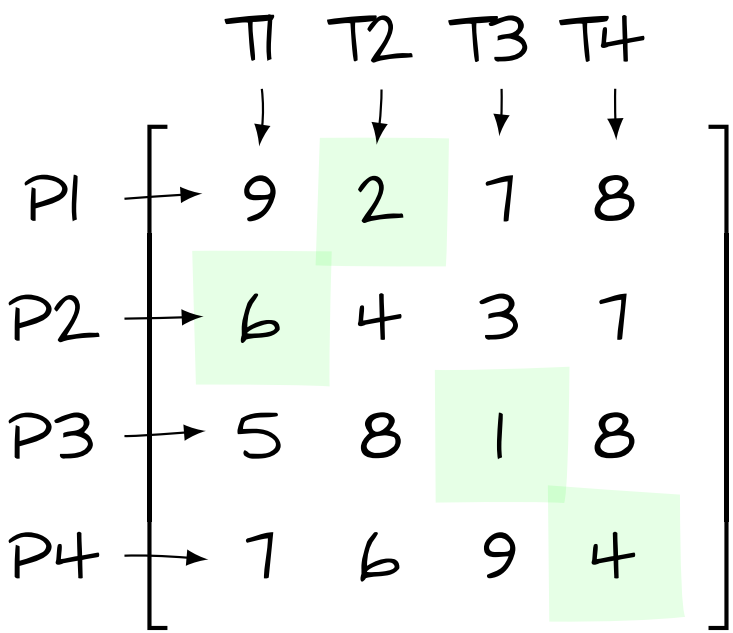
- Problema de alocar *n* pessoas para *n* trabalhos
  - É feita a alocação da pessoa *P4*

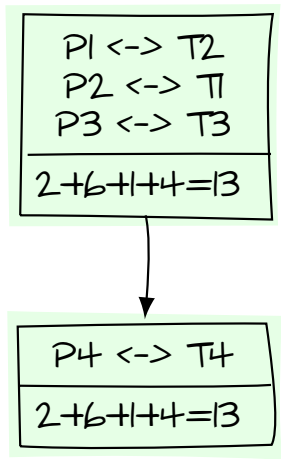




Branch-and-bound

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

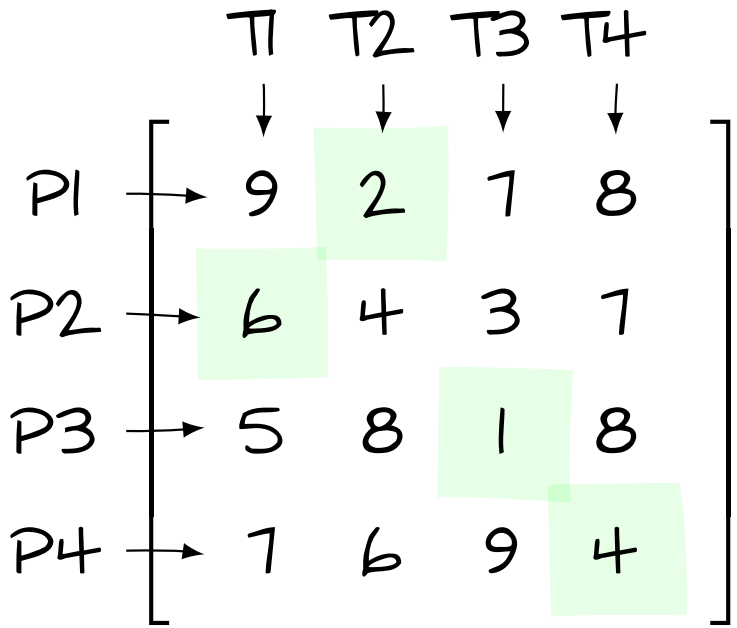




A solução completa mais promissora aloca a pessoa P4 para o trabalho T4

Branch-and-bound

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



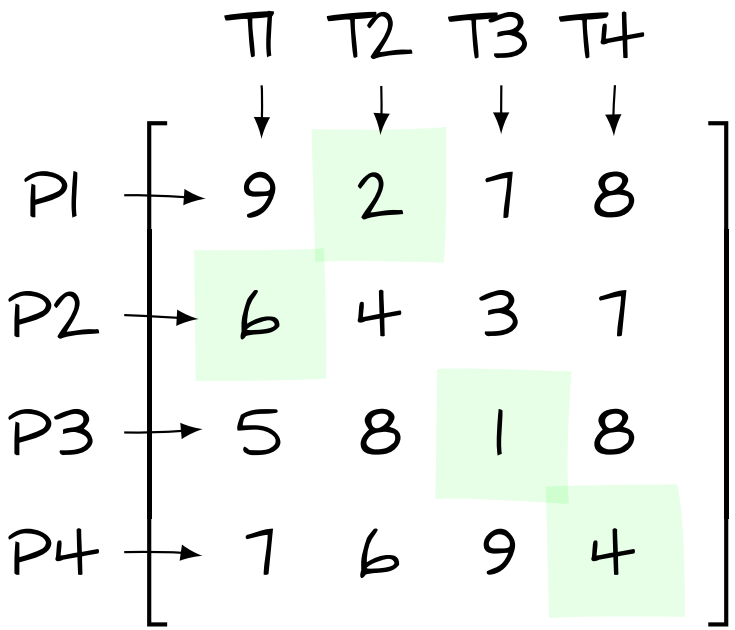
P1 <-> T2
P2 <-> T1
P3 <-> T3
P4 <-> T4

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

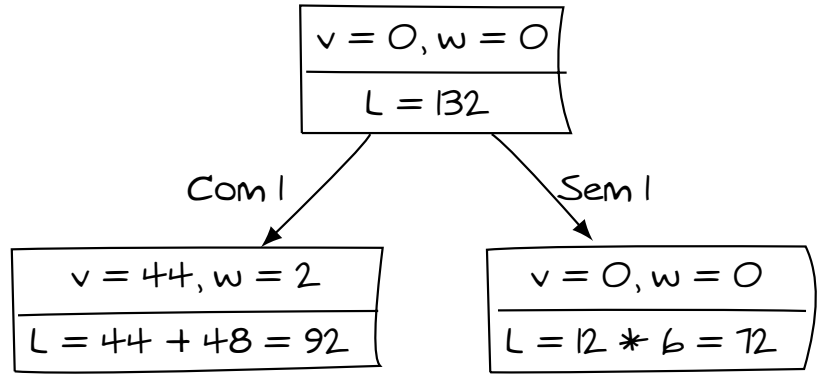
$v = 0, w = 0$

$L = 132$

Branch-and-bound

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

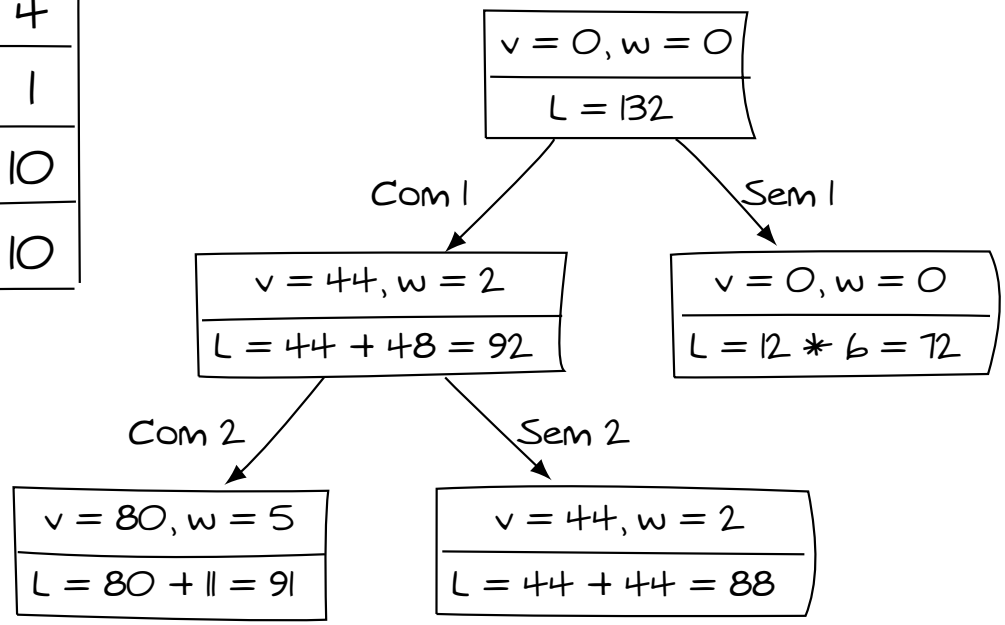
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



Branch-and-bound

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

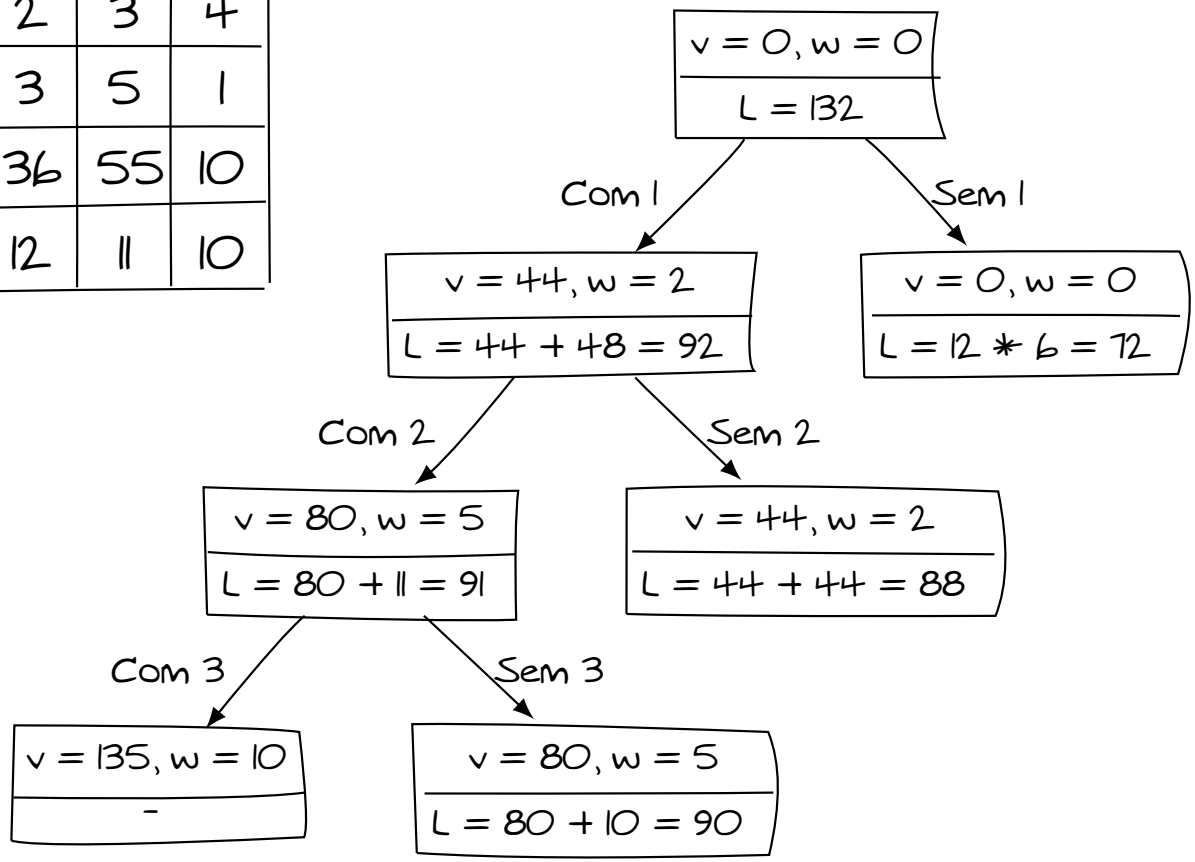
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



Branch-and-bound

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

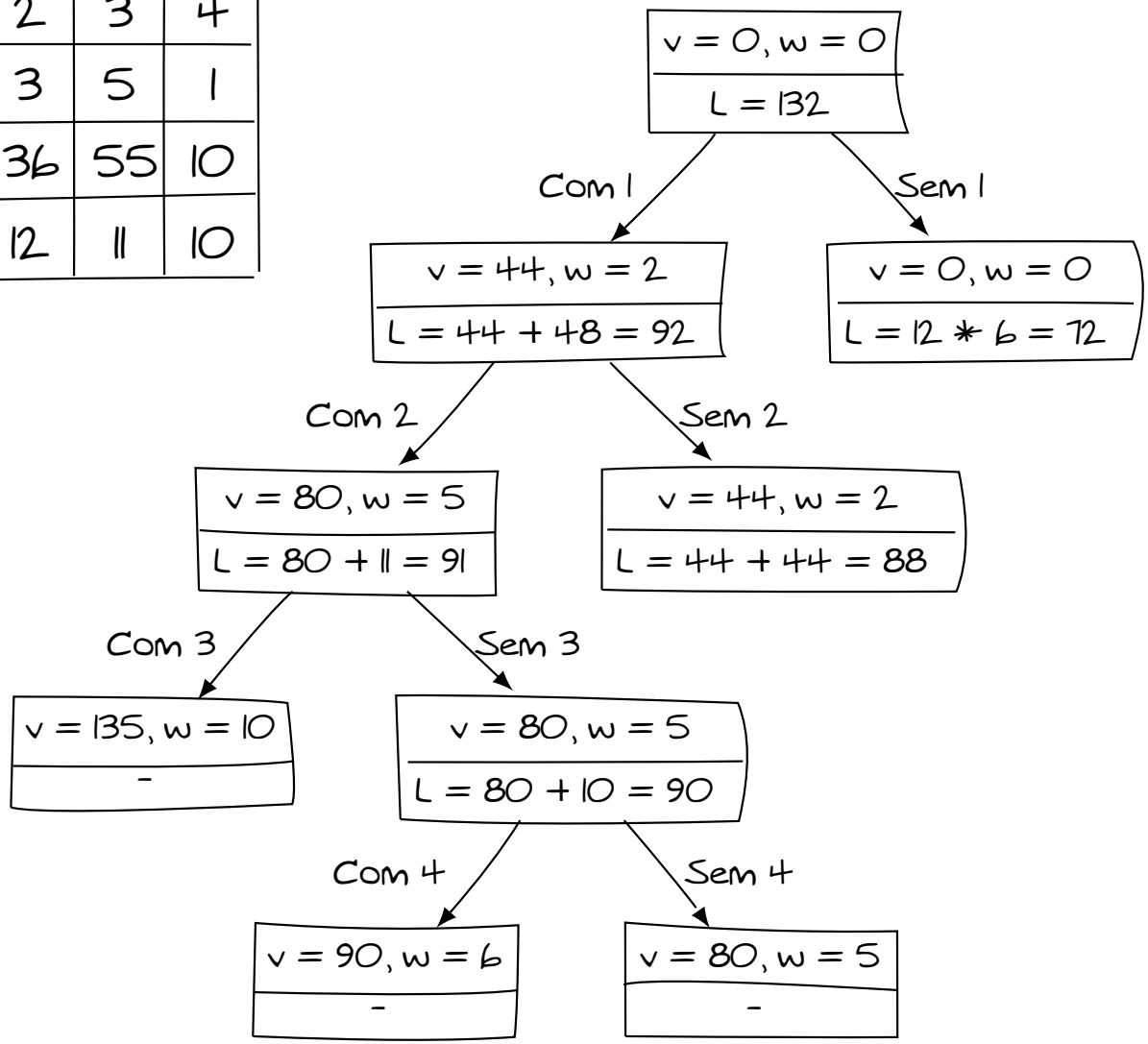
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



Branch-and-bound

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

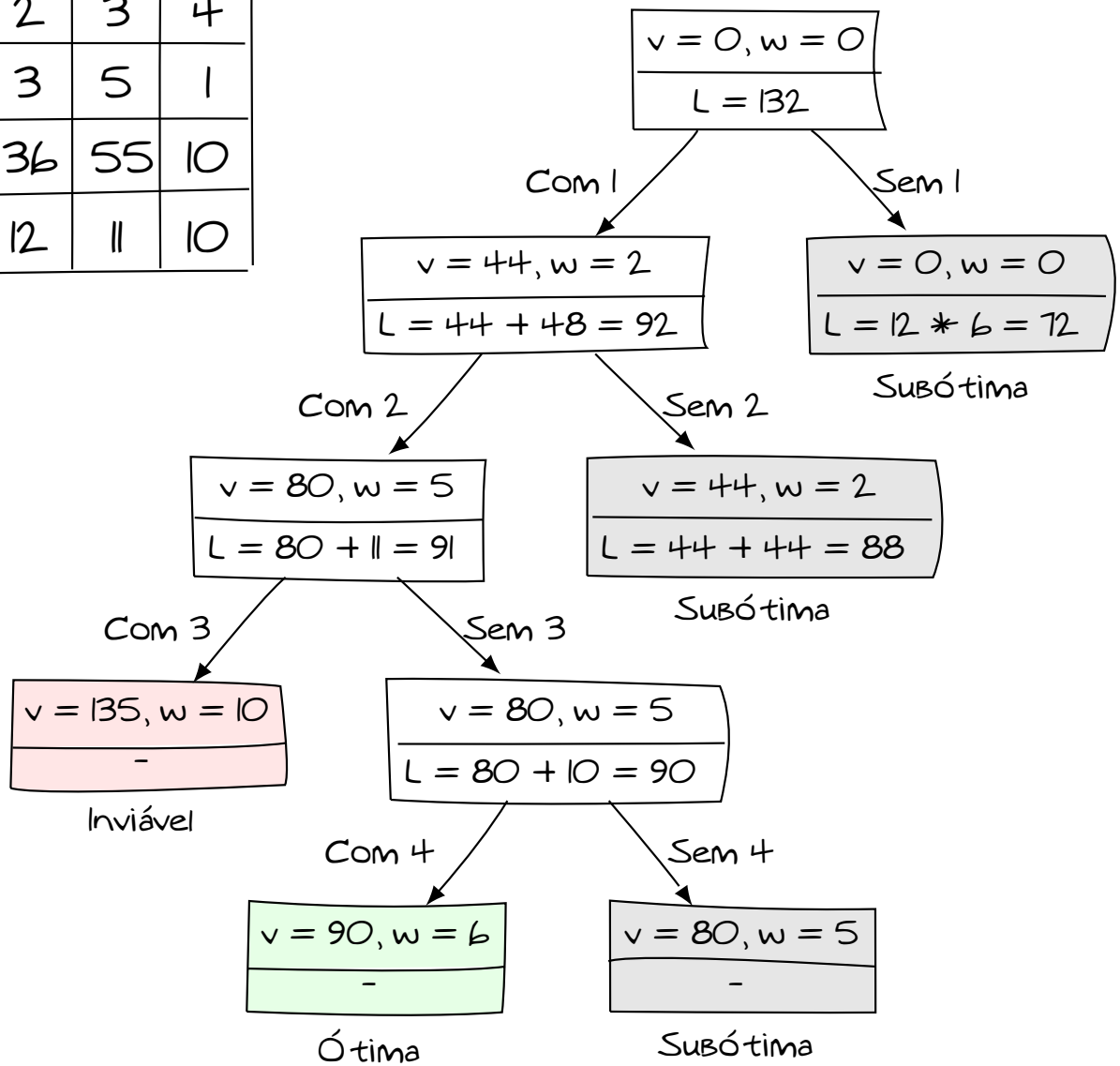
i	1	2	3	4
w_i	2	3	5	1
v_i	44	36	55	10
v_i/w_i	22	12	11	10



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 (finalizando 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 quando existe uma saída, é sempre um espaço livre na borda do labirinto que não é o ponto de partida

Exercício

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

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

```
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:
2  INICIO@2,2
3  F@2,2->1,2
4  D@1,2->1,3
5  BT@1,2<-1,3
6  E@1,2->1,1
7  T@1,1->2,1
8  BT@1,1<-2,1
9  BT@1,2<-1,1
10 BT@2,2<-1,2
11 T@2,2->3,2
12 SAIDA@3,2
13 L1:
14 INICIO@1,1
15 T@1,1->2,1
16 BT@1,1<-2,1
17 SEM_SAIDA
```