Força bruta

- busca exaustiva
- É possível resolver qualquer problema com ela
- Busca entre todas as opções, tenta tudo e avalia uma a uma
- Busca sequencial é uma estratégia de força bruta
- Exemplo: Problema do caixeiro viajante a força testa todos os trajetos
- TESTA UM POR UM
- Serve pra tudo mas é lenta
- Exemplo com vaga de estacionamento (dia a dia): Vamos passar por todas as vagas e escolher a mais próxima da saída. Sempre acha a melhor vaga mas pode demorar.
- Também conhecida como backtracking
- Mais recomendada para saber todas as soluções possíveis não otimização
- Usada também em casos que temos uma restrição, bounding fuction (função delimitadora) é possui a restrição

N-queens problem

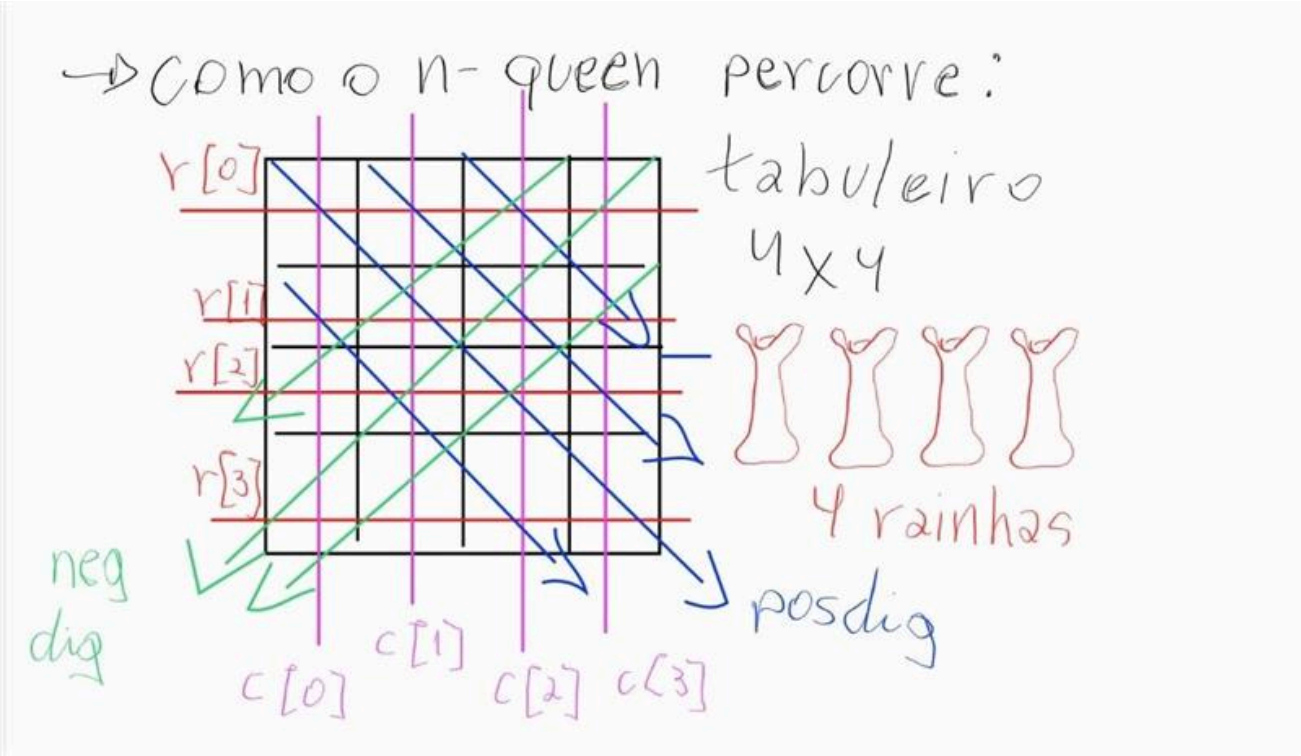
- Basicamente é um problema sobre xadrez, temos um tabuleiro n x n e n rainhas, devemos coloca-las no tabuleiro de um modo que elas não fiquem sobre ataque (não pode estar na mesma coluna, mesma reta e nem mesma diagonal) (vamos considerar n = 4)
- Possui mais de uma solução e queremos todas elas
- As rainhas devem ser colocadas em linhas diferentes, colunas diferentes e diagonais diferentes
- Diagonais negativas: Da esquerda para a direita
  - A principal diagonal negativa resulta em zero a seguinte conta: linha - coluna
- Diagonais positivas: Da direita para a esquerda
  - A principal diagonal positiva resulta em 3 para: linha + coluna
- chat
- Backtracking: Pensar, tentar, desfazer e tentar outra coisa

```
#include <iostream>
using namespace std;

const int N = 4;
char board[N][N]; // tabuleiro
bool col[N]; // colunas ocupadas
```

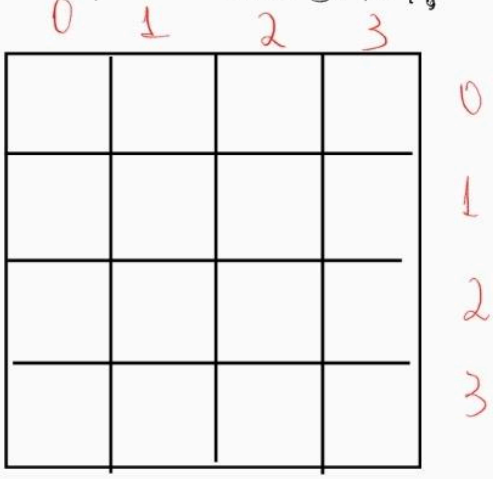
```
bool posDiag[2*N]; // diagonais ↘ (índice r+c)
bool negDiag[2*N]; // diagonais ↙ (índice r-c + (N-1))

void printBoard() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cout << board[i][j] << " ";
        }
    cout << endl;
    }
    cout << "--------------" << endl;

}

void backtrack(int r) { //backstrack é o nome da função, r representa a
linha atual
    if (r == N) { // se colocou rainha em todas as linhas mostrar o
tabuleiro
        printBoard();
        return; // sai da função depois que imprimi
    }
    for (int c = 0; c < N; c++) { //testando as colunas, vai testar todas
        if (col[c] || posDiag[r+c] || negDiag[r-c+N-1])
        //col[c] verifica a coluna atual que o loop for tá passando
        //posDiag verifica a diagonal positiva (indice diagonal = linha +
coluna)
        //negDiag verifica a diagonal negativa (linha-coluna+4-1)
        //se algum desses for true o local analisado não pode ser ocupado
            continue; //faz continuar para a proxima coluna, sai do if

        col[c] = true; //marca a coluna c como ocupada
        posDiag[r+c] = true; // marca a diagonal r+c como ocupada
        negDiag[r-c+N-1] = true; // marca diagonal como ocupada
        board[r][c] = 'Q'; // marca como Q o lugar com as coordenadas (r,c)

        backtrack(r+1); // tenta a próxima linha, estavamos na linha 0 agora
na 1
        col[c] = false; //
        posDiag[r+c] = false;
        negDiag[r-c+N-1] = false;
        board[r][c] = '.';
    }
}
int main() {
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
```

```
            board[i][j] = '.';
    for (int i = 0; i < N; i++) col[i] = false;
    for (int i = 0; i < 2*N; i++) posDiag[i] = negDiag[i] = false;
backtrack(0); // começa da linha 0

return 0;

}
```

- É um problema clássico de backtracking porque precisamos tentar uma posição, ver se funciona, e se não funcionar, desfazer e tentar outra
- Esse problema é um dos que ilustra melhor como a força bruta funciona, ele testa todos os casos em cada linha, coluna e diagonal
- Exemplo desenhado:

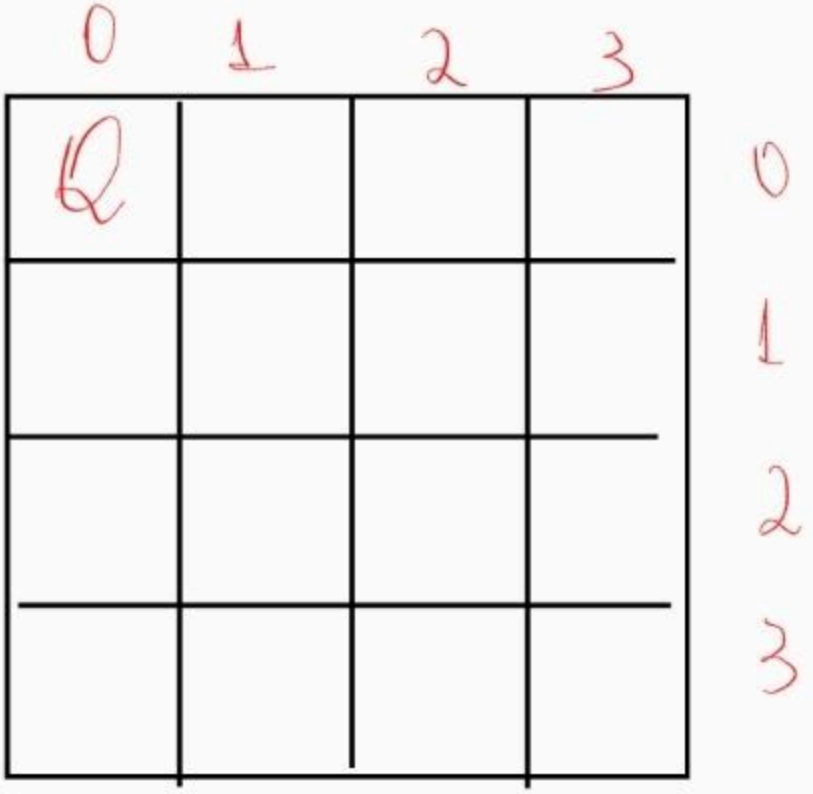


→ começa assim:



- Backtrace (0) → linha 0 (false: Não tem rainha)
  - tenta coluna 0 (col[0] = false)
  - posdig = [0] → false
  - negdig = [3] → false

Todas falsas → coloca a rainha em board [0] [0]



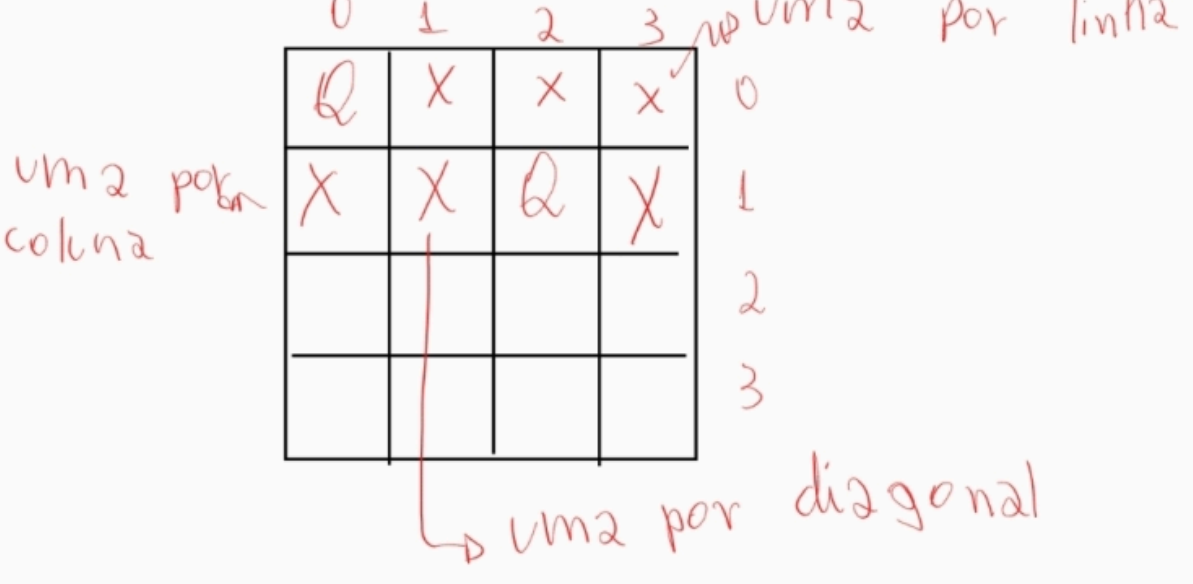
- Agora vamos para o backtrace(1)

linha 1:

- tenta coluna 0, ela está true (tem uma rainha em board[0][0], a rainha não pode ser colocada em
- tenta coluna 1, ela está em false
- tenta posdiag[0], ela está true, a rainha não pode ser colocada em Board[1][1]
- tenta coluna 2, ela está em false
  - posdiag = false
  - negdiag = false

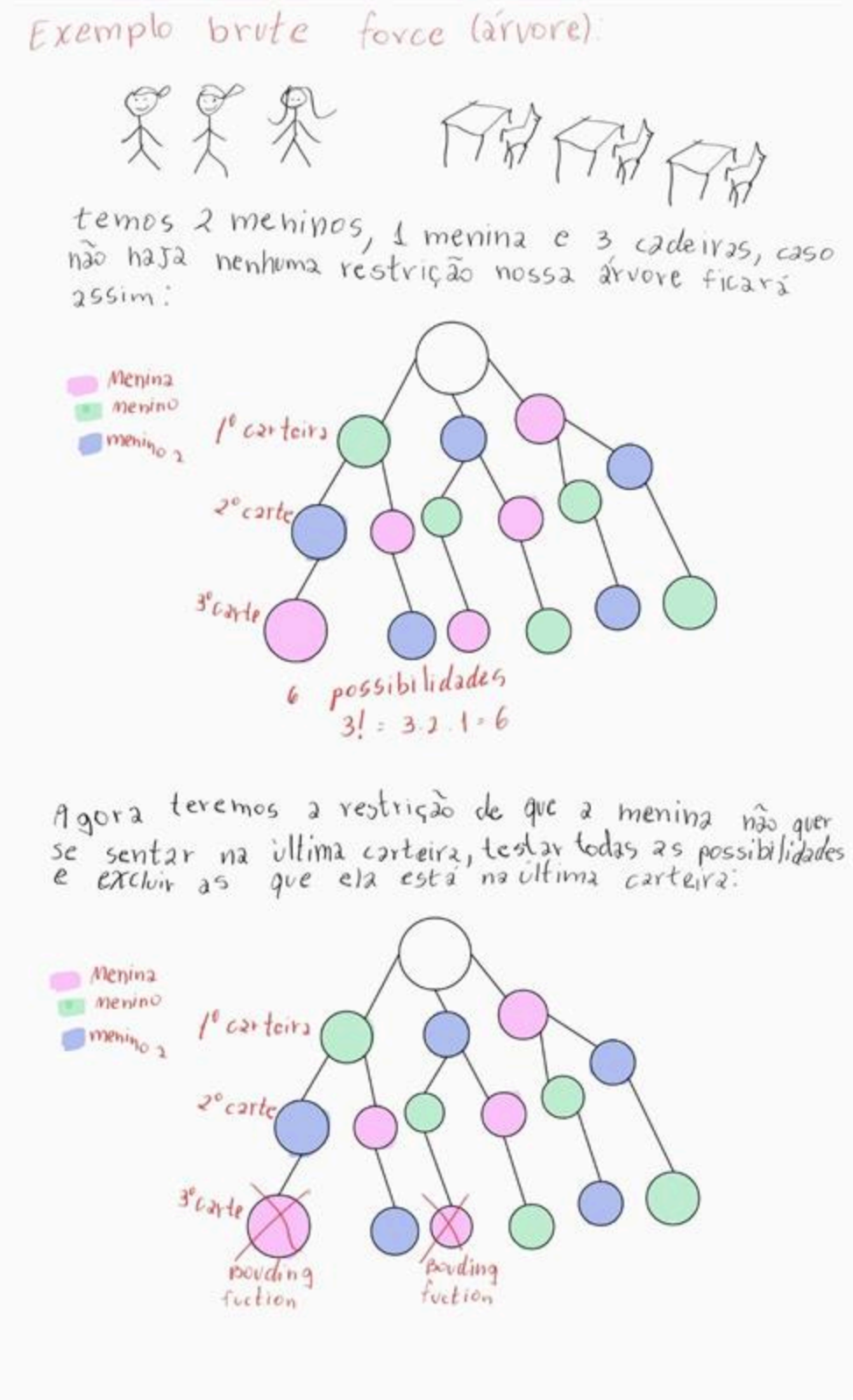
→ todas falsas coloca rainha em board[1][2]

→ todas falsas coloca rainha em board[1][2]



- E por ai vai os testes...

Arvore de decisão da estratégia força bruta



Complexidade

- Para N-Queens: N! (n = quantidade de rainhas)

- Para caixeiro viajante: (n-1)!/2 (numero de cidades - 1 fatorial)/2
- Mochila: 2ˆn (n = numero de itens)