Klotisover

Versão

Thiago Tosto, Jose Nominato, Filipe Rangel, João Gabriel

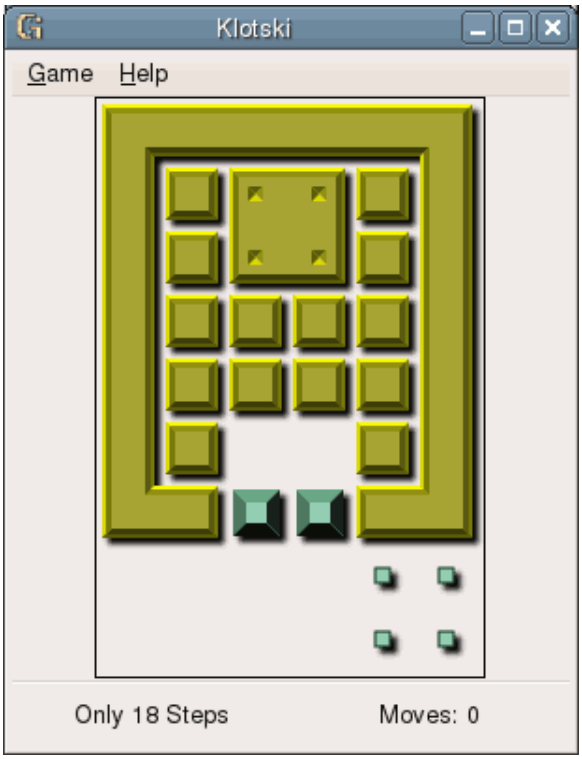
22 jun, 2017

Conteúdo

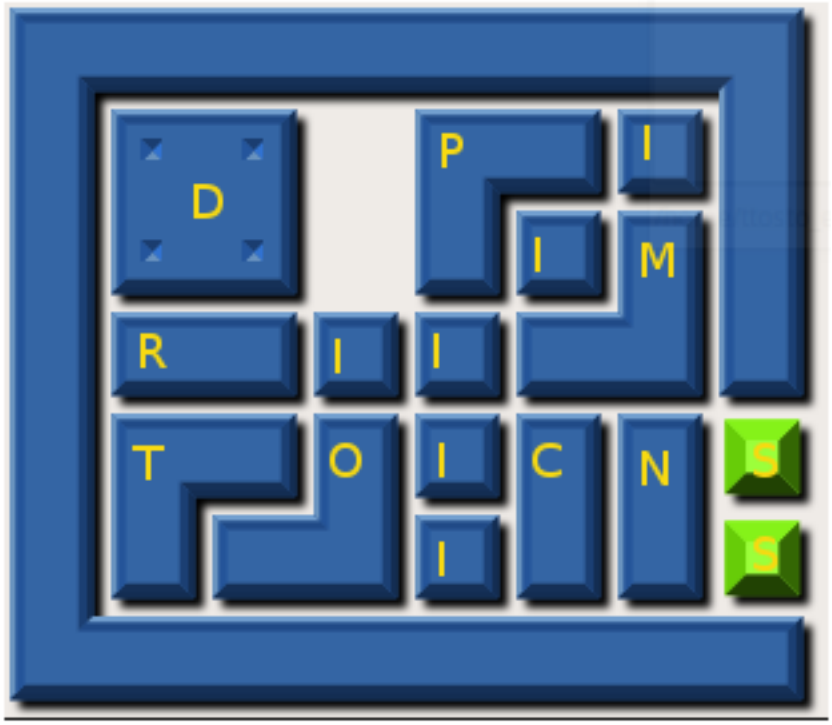
CAPÍTULO 1

Introdução

Klotski é um jogo famoso de quebra-cabeça, onde o objetivo é retirar uma peça principal de dentro de um confinamento em meio a obstaculos móveis.



O jogo tem uma certa complexidade e o desafio proposto por esse trabalho é desenvolver um software capaz de solucionar um caso particular desse jogo.

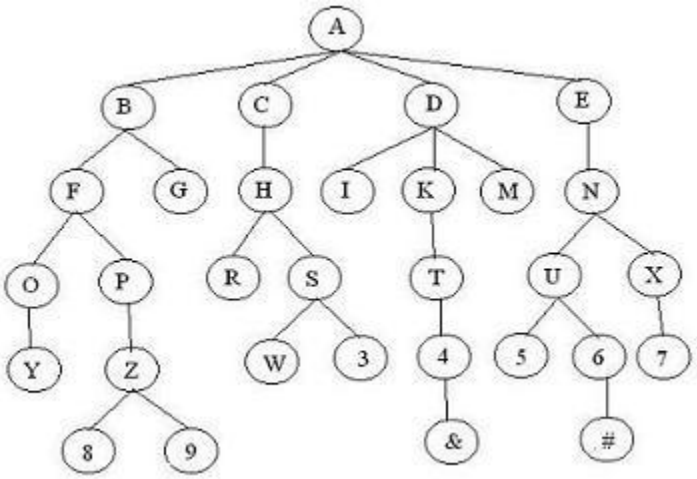


CAPÍTULO 2

Algoritmo

2.1 Estrutura de dados

A estrutura de dados escolhida para solução do problema é uma árvore genérica.



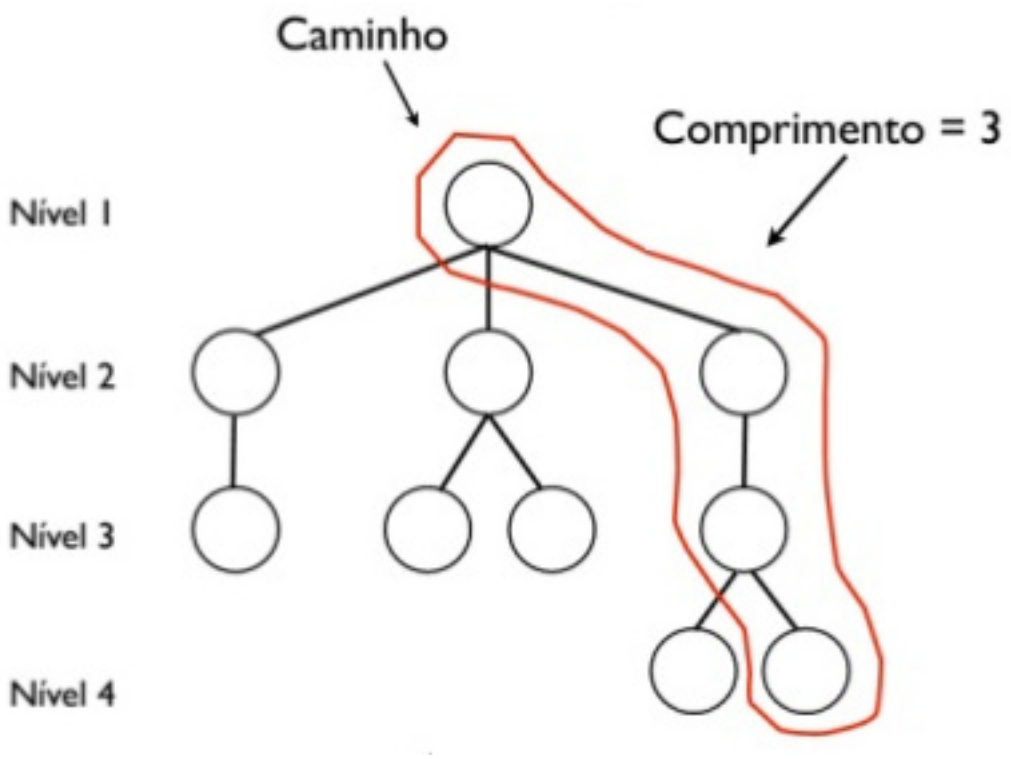
Cada nó desta árvore corresponde a um estado da matriz que representa o jogo, aonde o nó raíz será o estado inicial da matriz.

A partir do estado inicial, é possivel enxergar 8 novos candidatos a novos estados. Cada estado representa um nó filho de seu nó pai.

Recursivamente, temos o mesmo raciocínio para cada nó, até que se chegue no estado desejado que é com a dama em sua saída.
É de suma importância que uma geração de um novo filho não gere um estado que já se encontre na árvore para que se evite loops eternos ou que

cheguemos em uma configuração na qual já se tem o caminho.

O caminho do nó raíz até o nó folha que é o estado final buscado, representa a solução do problema.



O comprimento do caminho representa a quantidade de passos feitos para chegar na solução.

2.2 Modelagem do Klotski

A modelagem do Jogo foi feita através de uma matriz onde letras representam as peças diferentes e 0 os espaços em branco:

A letra “D” representa a **dama** que é a peça que temos como objetivo tirar do quebra cabeça.


[['D','D','0','P','P','G'],
['D','D','0','P','H','M'],
['R','R','I','J','M','M'],
['T','T','O','K','C','N'],
['T','O','O','L','C','N']];

CAPÍTULO 3

Pseudo Código

3.1 Raíz

3.1.1 Gerar Árvore (principal)

- Gera próximo estado (a)
- **8 vezes:**
    - Testa se é solução (b)
    - **(Não) Testa se já existe (c)**
        - (Não) Gera filho e chama recursividade
    - (Sim) Retorna até pai guardando estados no vetor solução

3.2 2º nível de abstração

3.2.1 Gerar próximo estado (a)

- Achar lacuna
- Achar periféricos
- **Para cada periférico:**
    - Tentar mover periférico e guardar estado no vetor estados
- Retorna estados

3.2.2 Solução teste (b)

- compara matriz (i)

3.2.3 Testa se já existe(busca na árvore) (c)

Parâmetros: nó raiz; nó buscado(estado)

- **Para cada filho:**
  - **Testa se filho é igual a nó buscado(compara matriz):**
    - (sim) retorna verdadeiro
    - (não) Chama recursividade passando filho como raíz
- Retorna falso

3.2.4 Gera filho

Parâmetros: pai; estado;

- «Instancia» filho.
- Setar pai (filho.pai = pai).
- Setar estado (filho.estado = estado).
- Anula filhos (filhos.filhos[]).
- Retorna filho

3.3 3º nível de abstração

3.3.1 Compara matriz (i)

Parâmetros: matriz 1; matriz 2;

- **Testa se matriz1 e matriz2 são nulas:**
  - (não) compara
  - (sim) retorna falso