Využití rekurze pro řešení problémů

doc. Mgr. Jiří Dvorský, Ph.D.

Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – TU Ostrava

Prezentace ke dni 12. února 2020



Osnova přednášky

Využití rekurze pro řešení problémů

- Backtracking
- Jezdcova procházka

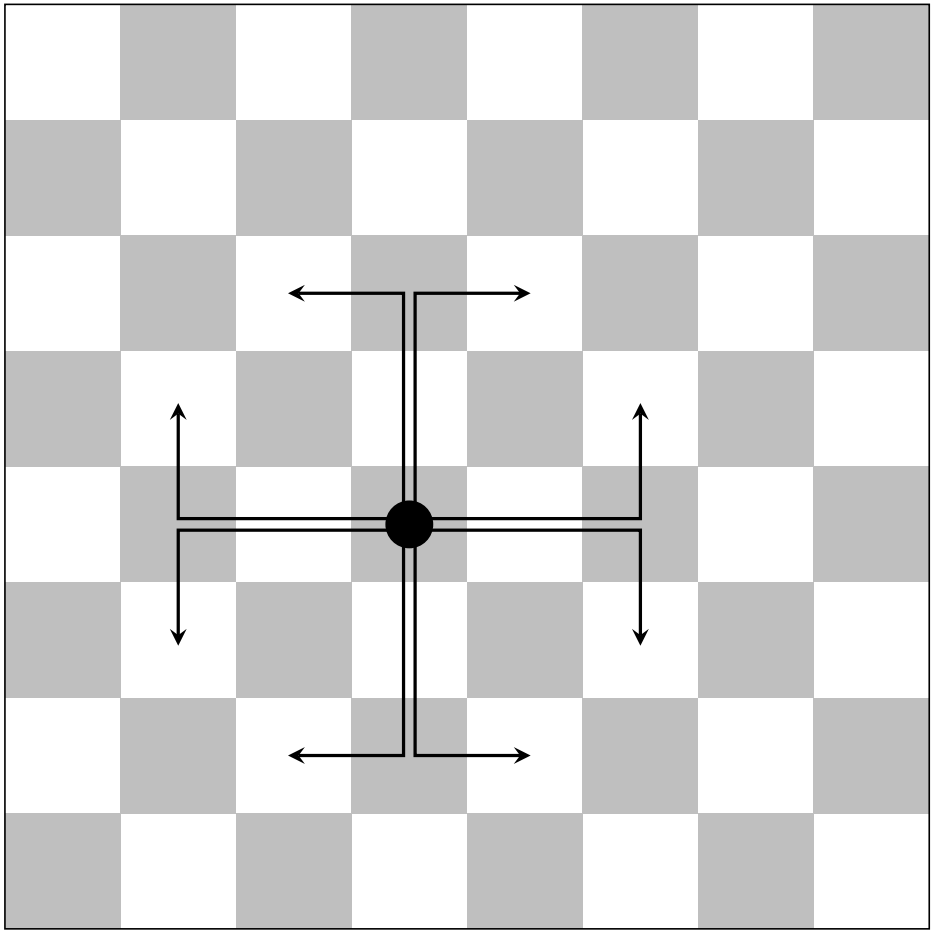
Backtracking

- zpětné vyhledávání, metoda pokusů a oprav, metoda zpětného sledování, metoda prohledávání do hloubky,
- způsob řešení algoritmických problémů
- založený na prohledávání stavového stromu problému
- možné použít pro řešení velkého množství problémů
- obecně exponenciální časová složitost
- jen pro data malé velikosti, popřípadě pro něj existuje dobrá heuristika.

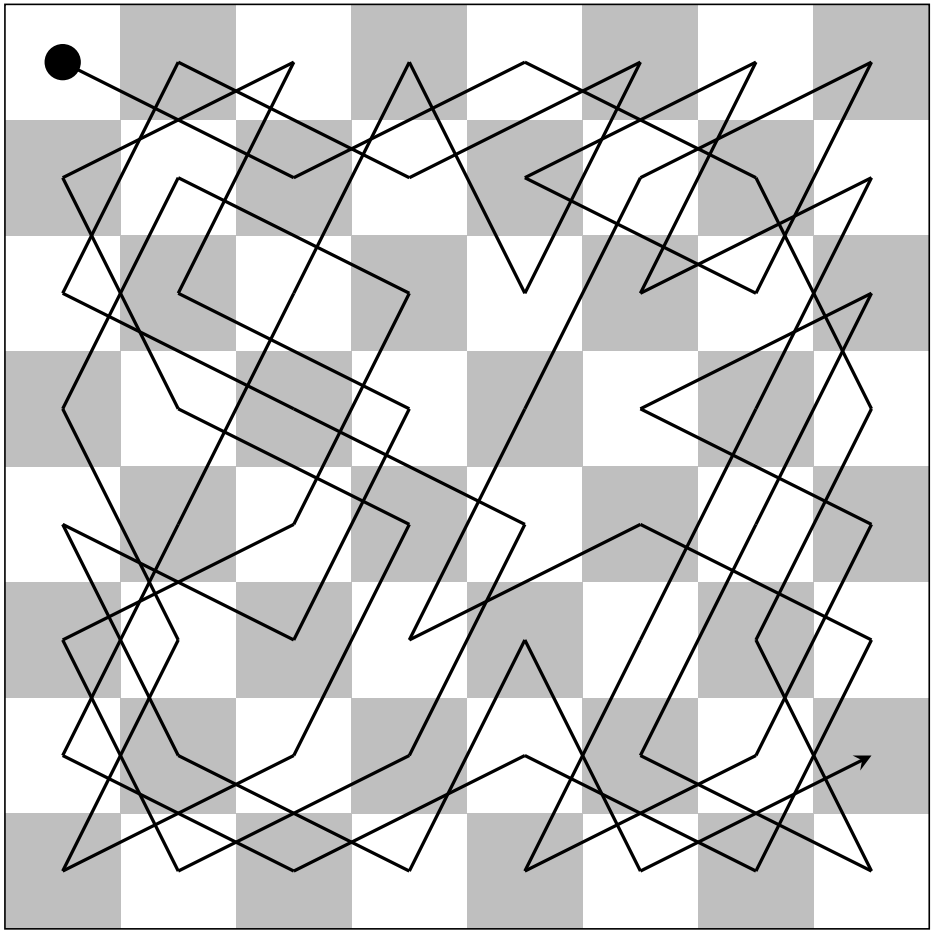
Jezdcova procházka

- šachový a matematický problém popsaný pomocí šachové figury jezdce a šachovnice,
- jezdec se pohybuje v souladu s šachovými pravidly po prázdné šachovnici,
- úkolem je, aby každé pole navštívil právě jednou,
- úloha již z 9. století,
- pro šachovnici 8×8 existuje 26 534 728 821 064 řešení, kdy kůň ohrožuje své výchozí postavení,
- řešení uzavřená vs otevřená.

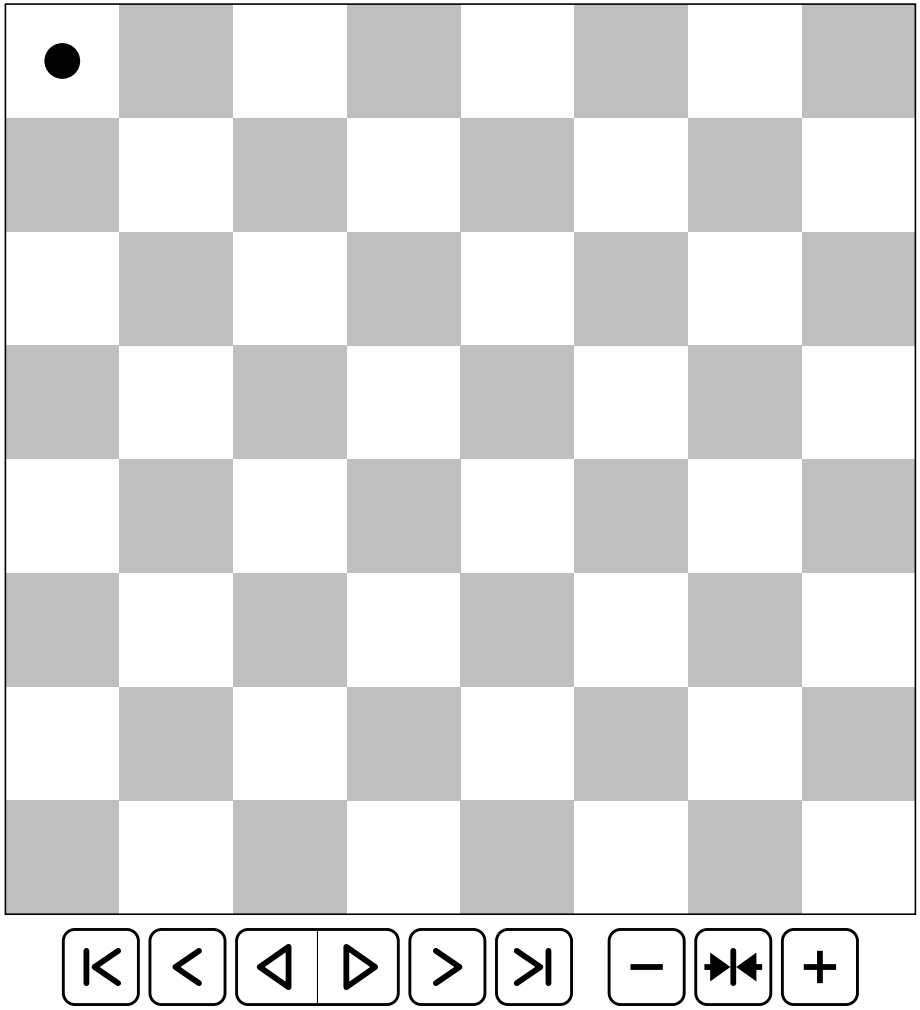
Možné tahy šachovým koněm



Výsledné řešení



Animace výsledného řešení



Zdrojové kódy řešení, soubor KnightsTour.h

```
1  #pragma once
2
3  #include <iomanip>
4  #include <iostream>
5
6  using namespace std;
7
8  /**
9   * Demonstracni ukazka backtrackingu.
10  * Cilem ukazky je implementace sachove ulohy zname pod nazvem Jezdcova
        prochazka, anglicky Kniht's tour.
11  * Informace o uloze lze najit napriklad na http://cs.wikipedia.org/
      wiki/Jezdcova_proch%C3%A1zka
12  *
13  * @author     Jiri Dvorsky <jiri.dvorsky@vsb.cz>
14  *
15  * Prostor jmen pro projekt KnightsTour.
16  */
17 namespace KnightsTour
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
18 {
19     /**
20      * Velikost sachovnice
21      */
22     const int BoardSize = 8;
23
24     /**
25      * Hodnota volneho, dosud nenavstiveneho, pole sachovnice
26      */
27     const int FreeCell = 0;
28
29     /**
30      * Pocet vsech potencialne moznych tahu konem z policka sachovnice
31      */
32     const int NumberOfKnightMoves = 8;
33
34     /**
35      * Zmeny hodnoty radku pro jednotlive mozne tahy konem
36      */
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
37    const int RowMoveDelta[NumberOfKnightMoves] = {-2, -1, +1, +2, +2,
          +1, -1, -2};
38
39    /**
40     * Zmeny hodnoty sloupce pro jednotlive mozne tahy konem
41     */
42    const int ColumnMoveDelta[NumberOfKnightMoves] = {+1, +2, +2, +1,
          -1, -2, -2, -1};
43
44    /**
45     * Struktura pro ulozeni pozice kone na sachovnici
46     */
47    struct Position
48    {
49       /**
50        * radek dane pozice
51        */
52       int Row;
53       /**
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
54          * Sloupec dane pozice
55          */
56         int Column;
57     };
58
59
60     /**
61      * Inicializace sachovnice.
62      *
63      * Funkce vyplni vsechna policka sachovnice nulami, coz znamena, ze
            na dane pole kun dosud neskocil.
64      *
65      * @param Board Inicializovana sachovnice
66      */
67     void InitBoard(int Board[BoardSize][BoardSize]);
68
69
70     /**
71      * Vystup sachovnice na standardni vystup (obrazovku).
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
72      *
73      * @param Board Vypisovana sachovnice
74      */
75     void PrintBoard(const int Board[BoardSize][BoardSize]);
76
77
78     /**
79      * Posun konem po sachovnici.
80      *
81      * Funkce provede jeden z NumberOfKnightMoves moznych tahu.
82      * Funkce nezkouma, jestli provedeny tah je platny (napriklad, zda
            se kun neocitnul mimo sachovnici).
83      *
84      * @param CurrentPos Soucasna poloha kone na sachovnici
85      * @param MoveIndex cislo tahu, ktery z NumberOfKnightMoves moznych
            tahu chceme provest.
86      * @return Funkce vraci novou polohu kone na sachovnici.
87      */
88     Position Move(const Position& CurrentPos, const int MoveIndex);
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
89
90
91      /**
92       * Test, zda je tah platny.
93       *
94       * Tah je platny pokud se kun neocitnul mimo sachovnici a pokud
             neskocil na policko, kde jiz jednou byl.
95       *
96       * @param Pos Soucasna poloha kone na sachovnici
97       * @param Board sachovnice
98       * @return Funkce vraci true pokud je tah platny, jinak vraci false.
99       */
100     bool IsValidMove(const Position& Pos, const int Board[BoardSize][
             BoardSize]);
101
102
103     /**
104      * Rekurzivni hledani reseni ulohy, implementace pomoci
             backtrackingu.
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
105     *
106     * Funkce postupne provadi jednotlive mozne tahy konem ze soucasne
            polohy kone na sachovnici (celkem az NumberOfKnightMoves tahu).
107     * Pokud ja takto provedeny tah platny, tah se zaznamena do
            sachovnice a rekurzivne se pokracuje dal.
108     * Pokud funkce vycerpa vsechny mozne tahy ze soucasne pozice, vraci
             funkce false jako signal, ze reseni
109     * z teto polohy kone na sachovnici neni mozne v dalsich tazich
            najit. Prohledavani se tedy vraci o jeden tah zpet (vraceni
            zpet - backtracking).
110     *
111     * Funkce hleda prvni mozne reseni, nikoliv vsechna mozna reseni.
112     *
113     * @param Board sachovnice, v sachovnici jsou zaznamenany vsechny
            tahy pocinaje prvnim (cislo 1) a konce tahem s cislem
            MoveNumber-1.
114     * @param CurrentPos Soucasna poloha kone na sachovnici, poloha v
            tahu s cislem MoveNumber-1.
115     * @param MoveNumber cislo tahu, ktery prave resime
```

Zdrojové kódy řešení, soubor KnightsTour.h (pokrač.)

```
116     * @return Funkce vraci true pokud bylo v naslednych tazich po tahu
            s cislem MoveNumber nalezeno reseni, jinak vraci false.
117     */
118    bool SolveKnightsTour(int Board[BoardSize][BoardSize], const
           Position& CurrentPos, const int MoveNumber);
119 }
```

Zdrojové kódy řešení, soubor KnightsTour.cpp

```
1  #include "KnightsTour.h"
2
3  namespace KnightsTour
4  {
5     void InitBoard(int Board[BoardSize][BoardSize])
6     {
7        for(int i = 0; i < BoardSize; i++)
8        {
9           for(int j = 0; j < BoardSize; j++)
10          {
11             Board[i][j] = FreeCell;
12          }
13       }
14    }
15
16    void PrintBoard(const int Board[BoardSize][BoardSize])
17    {
18       for(int i = 0; i < BoardSize; i++)
19       {
```

Zdrojové kódy řešení, soubor KnightsTour.cpp (pokrač.)

```
20          for(int j = 0; j < BoardSize; j++)
21          {
22             cout << setw(4) << Board[i][j];
23          }
24          cout << endl;
25       }
26       cout << endl;
27    }
28
29    Position Move(const Position& CurrentPos, const int MoveIndex)
30    {
31       Position NewPos;
32       NewPos.Row = CurrentPos.Row + RowMoveDelta[MoveIndex];
33       NewPos.Column = CurrentPos.Column + ColumnMoveDelta[MoveIndex];
34       return NewPos;
35    }
36
37    bool IsValidMove(const Position& Pos, const int Board[BoardSize][
          BoardSize])
```

Zdrojové kódy řešení, soubor KnightsTour.cpp (pokrač.)

```
38    {
39       if (0 <= Pos.Row && Pos.Row < BoardSize && 0 <= Pos.Column && Pos
             .Column < BoardSize)
40       {
41          return Board[Pos.Row][Pos.Column] == FreeCell;
42       }
43       return false;
44    }
45
46    bool SolveKnightsTour(int Board[BoardSize][BoardSize], const
          Position& CurrentPos, const int MoveNumber)
47    {
48       if (MoveNumber == BoardSize*BoardSize + 1)
49       {
50          return true;
51       }
52       for(int i = 0; i < NumberOfKnightMoves; i++)
53       {
54          Position p = Move(CurrentPos, i);
```

Zdrojové kódy řešení, soubor KnightsTour.cpp (pokrač.)

```
55          if (IsValidMove(p, Board))
56          {
57             Board[p.Row][p.Column] = MoveNumber;
58             if (SolveKnightsTour(Board, p, MoveNumber+1))
59             {
60                return true;
61             }
62             Board[p.Row][p.Column] = FreeCell;
63          }
64       }
65       return false;
66    }
67 }
```

Zdrojové kódy řešení, soubor main.cpp

```
1  #include "KnightsTour.h"
2
3  using namespace KnightsTour;
4
5  void main()
6  {
7     int Board[BoardSize][BoardSize];
8     InitBoard(Board);
9     Position InitPos;
10    // vychozi pozice je zvolena nahodne
11    InitPos.Row = 0;
12    InitPos.Column = 0;
13    Board[InitPos.Row][InitPos.Column] = 1;
14    bool HasSolution = SolveKnightsTour(Board, InitPos, 2);
15    if (HasSolution)
16    {
17       PrintBoard(Board);
18    }
19    else
```

Zdrojové kódy řešení, soubor main.cpp (pokrač.)

```
20      {
21          cout << "Reseni nenalezeno!" << endl;
22      }
23  }
```

Děkuji za pozornost