Metody návrhu algoritmů

IB111 Programování a algoritmizace

2010

Návrhu algoritmů

- vybrané metody:
  - hladové algoritmy
  - dynamické programování
  - rekurze
  - hrubá síla
- tato přednáška:
  - především ilustrativní
  - ukázky na hrách, log. úlohách, ...

Rozměňování peněz

- vstup: částka X
- výstup: kolik nejméně mincí (bankovek) potřebujeme k vyplacení částky
- varianty:
  1. klasické hodnoty peněz: 1, 2, 5, 10, 20, 50, 100, ...
  2. libovolné (zadané) hodnoty

Rozměňování peněz I

- hladový algoritmus: „použij vždy nejvyšší minci, která je menší než cílová částka“
- funguje pro klasické hodnoty
- nefunguje pro obecný případ – najděte příklad

Rozměňování peněz II

- dynamické programování
- „vyplňování tabulky“, budujeme řešení „od spodu“
- pro hodnoty mincí 1, 3, 4

částka	1	2	3	4	5	6	7	8	9	10
počet mincí										

Rozměňování peněz II

- dynamické programování
- „vyplňování tabulky“, budujeme řešení „od spodu“
- pro hodnoty mincí 1, 3, 4

částka	1	2	3	4	5	6	7	8	9	10
počet mincí	1	2	1	1	2	2	2	2	3	3

Hra Nim

- máme n sirek (např. 20)
- povolený tah: odebrat 1, 2 nebo 3 sirky
- hráči se střídají
- kdo nemůže táhnout (nejsou žádné sirky), prohrává

Hra Nim II

- varianty:
  - tahy: 1, 3 nebo 4 sirky
  - tahy: 1, 2 nebo 3 sirky; ale zakázáno odebrat tolik, kolik sebral oponent v posledním tahu
- najděte vítěznou strategii pomocí dynamického programování

Nejdelší rostoucí úsek, podposloupnost

Nejdelší rostoucí úsek

vstup: posloupnost čísel
výstup: nejdelší rostoucí úsek (po sobě jdoucí čísla)

Nejdelší rostoucí podposloupnost

vstup: posloupnost čísel
výstup: nejdelší rostoucí podposloupnost (čísla nemusí jít po sobě)

Nejdelší rostoucí úsek, podposloupnost

příklad: 4, 1, 8, 6, 5, 11, 2, 4, 9, 12, 5, 13, 7, 8, 12, 10, 11

Nejdelší rostoucí úsek, podposloupnost

příklad: 4, 1, 8, 6, 5, 11, 2, 4, 9, 12, 5, 13, 7, 8, 12, 10, 11

nejdelší rostoucí úsek: 2, 4, 9, 12

nejdelší rostoucí podposloupnost: 1, 2, 4, 5, 7, 8, 10, 11

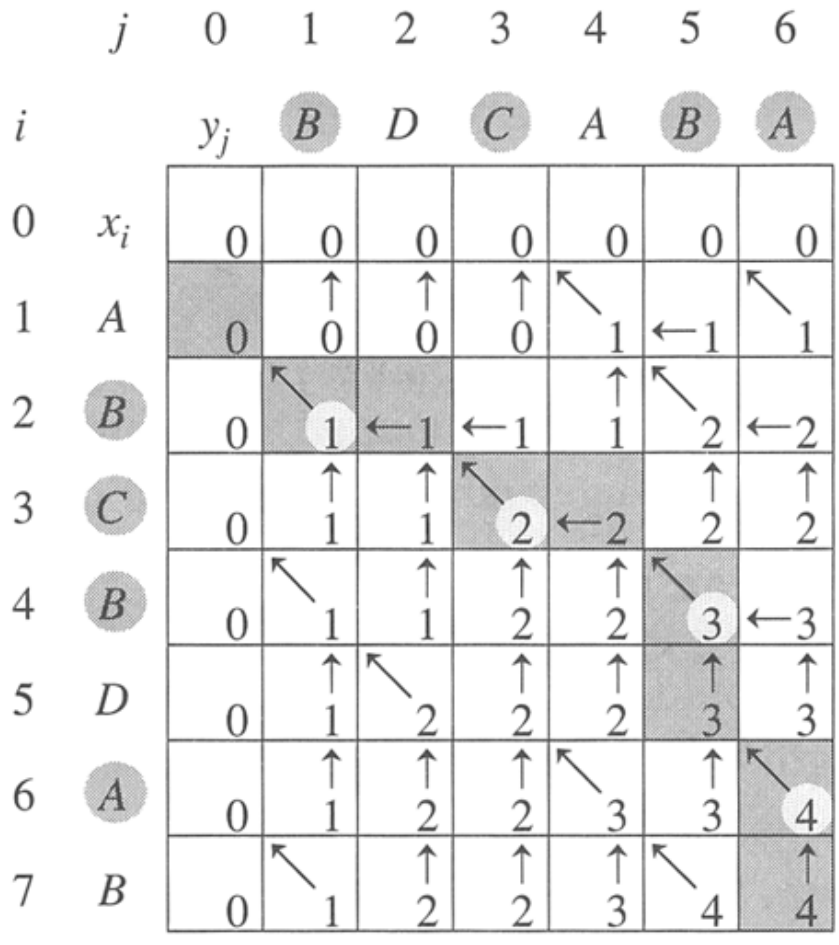
Algoritmy

- nejdelší rostoucí úsek:
  - jeden průchod pole, lineární složitost $O(n)$
  - pamatuji si aktuální délku rostoucí podposloupnosti + dosavadní maximum
- nejdelší rostoucí podposloupnost:
  - „dynamické programování“
  - pro každou pozici si pamatuji délku nejdelší podposloupnosti končící v tom místě
  - přímočaře: kvadratická složitost $O(n^2)$
  - efektivní implementace $O(n \log n)$

Nejdelší společný podřetězec, podposloupnost

- vstup: dva řetězce s_1, s_2
- výstup: nejdelší společný úsek/podposloupnost (rozdíl podobně jako u předchozího)
- příklad:
  - BDCABA
  - ABCBDAB

Nejdelší společná podposloupnost



i		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
0	x_i		0	0	0	0	0	0	0
1	A		0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B		0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C		0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B		0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D		0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A		0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B		0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Nejdelší společná podposloupnost

„dynamické programování“

Algorithm 7.1 LCS
Input: Two strings A and B of lengths n and m, respectively, over an alphabet Σ.
Output: The length of the longest common subsequence of A and B.

1. **for** $i \leftarrow 0$ **to** n
2. $\quad L[i,0] \leftarrow 0$
3. **end for**
4. **for** $j \leftarrow 0$ **to** m
5. $\quad L[0,j] \leftarrow 0$
6. **end for**
7. **for** $i \leftarrow 1$ **to** n
8. $\quad$ **for** $j \leftarrow 1$ **to** m
9. $\quad\quad$ **if** $a_i = b_j$ **then** $L[i,j] \leftarrow L[i-1,j-1]+1$
10. $\quad\quad$ **else** $L[i,j] \leftarrow \max\{L[i,j-1], L[i-1,j]\}$
11. $\quad\quad$ **end if**
12. $\quad$ **end for**
13. **end for**
14. **return** $L[n,m]$

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Hrubá síla

- „Vyzkoušej všechny možnosti.“
- kdy použít:
  - problém relativně malý, možností málo – i když by to šlo chytřeji, nemá cenu se s tím babrat (např. Sudoku)
  - nic lepšího prostě neznáme (NP-úplné problémy)

Heuristiky

- heuristika = přístup, který většinou pomůže, ale nemusí fungovat vždy
- dobrá heuristika může řádově zlepšit hrubou sílu

Prohledávání stavového prostoru

připomenutí: stavový prostor = graf

použití např.:

- Good Old-Fashioned Artificial Intelligence (GOFAI)
- verifikace protokolů

Logická úloha: Misionáři a kanibalové

- 3 misionáři, 3 kanibalové
- řeka, 1 loďka (max 2 lidé)
- víc kanibalů jak misionářů na jednom místě $\Rightarrow$ problém

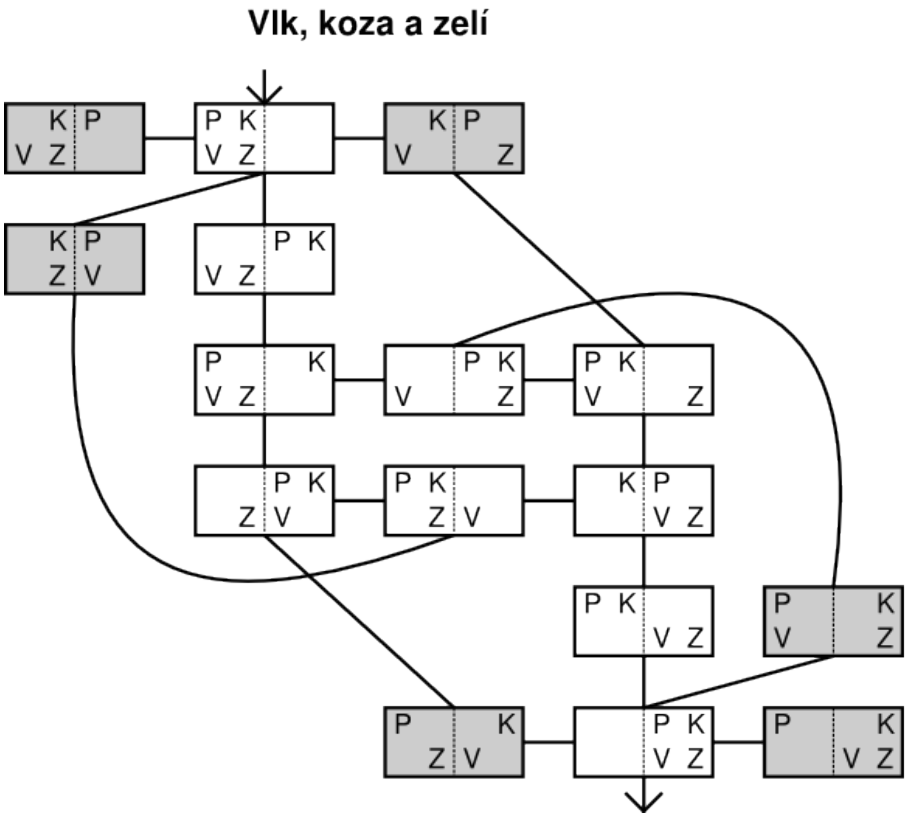
Vlk, koza, zelí

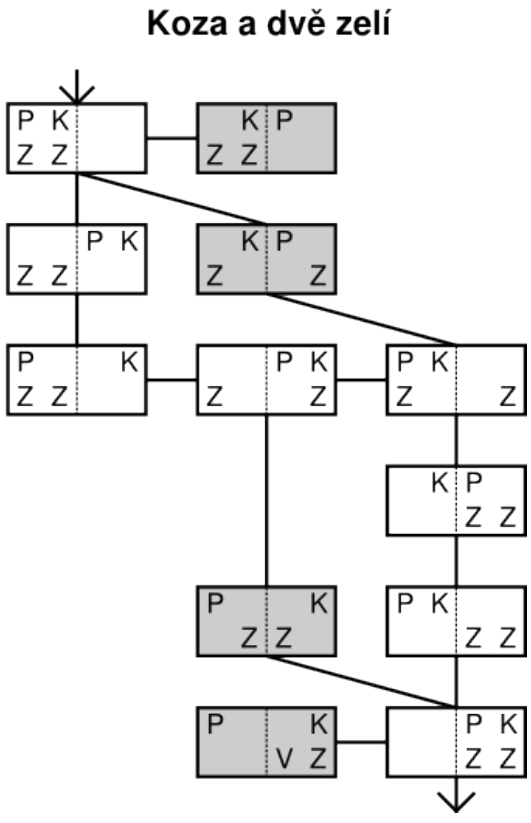
- klasická úloha typu převážení přes řeku
- zakreslit stavový prostor
- co úloha „koza a 2 zelí“?

Vlk, koza, zelí

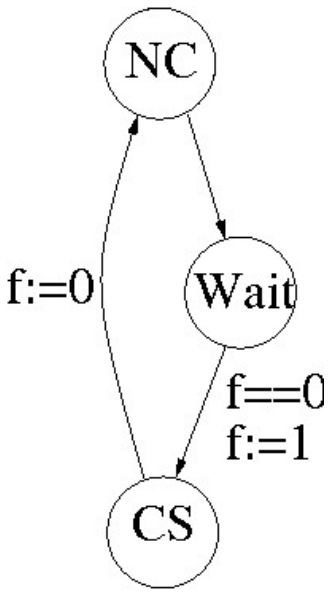
- klasická úloha typu převážení přes řeku
- zakreslit stavový prostor
- co úloha „koza a 2 zelí“?
- „izomorfní“ úlohy

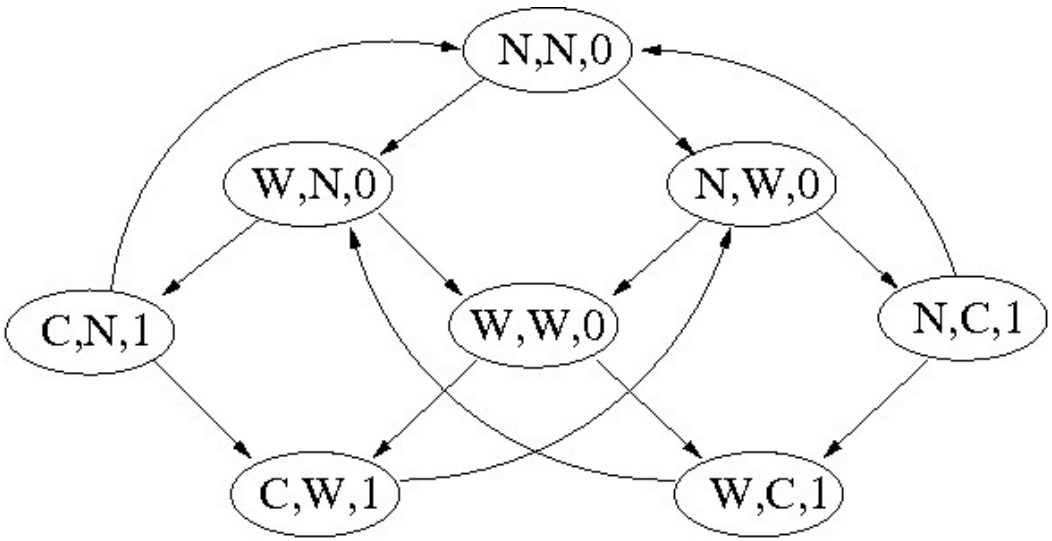
Vlk, koza, zelí





Verifikace protokolů

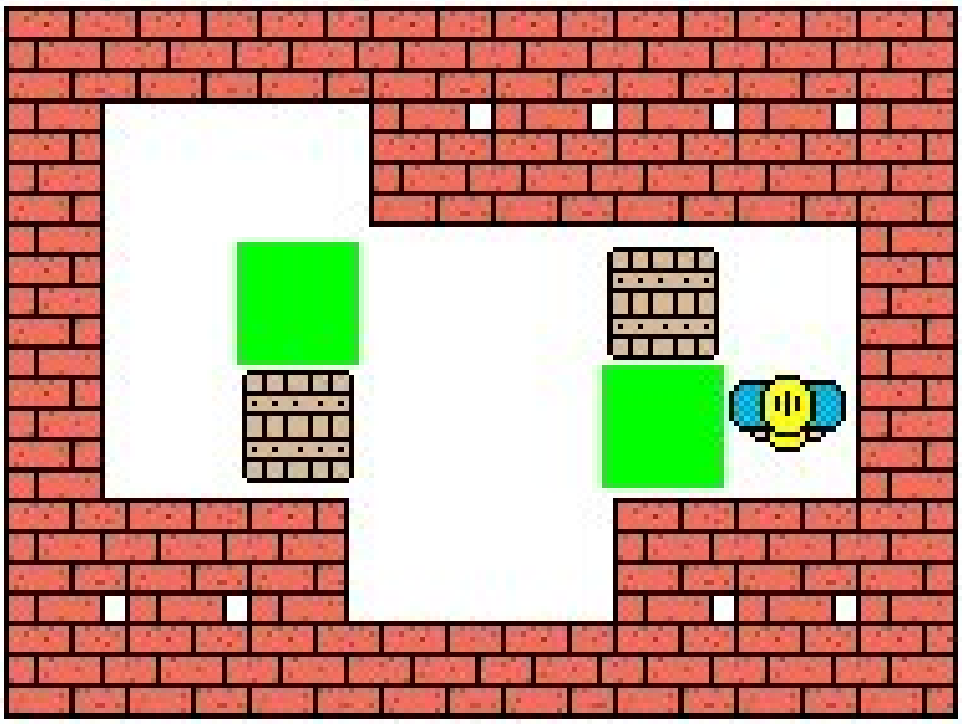




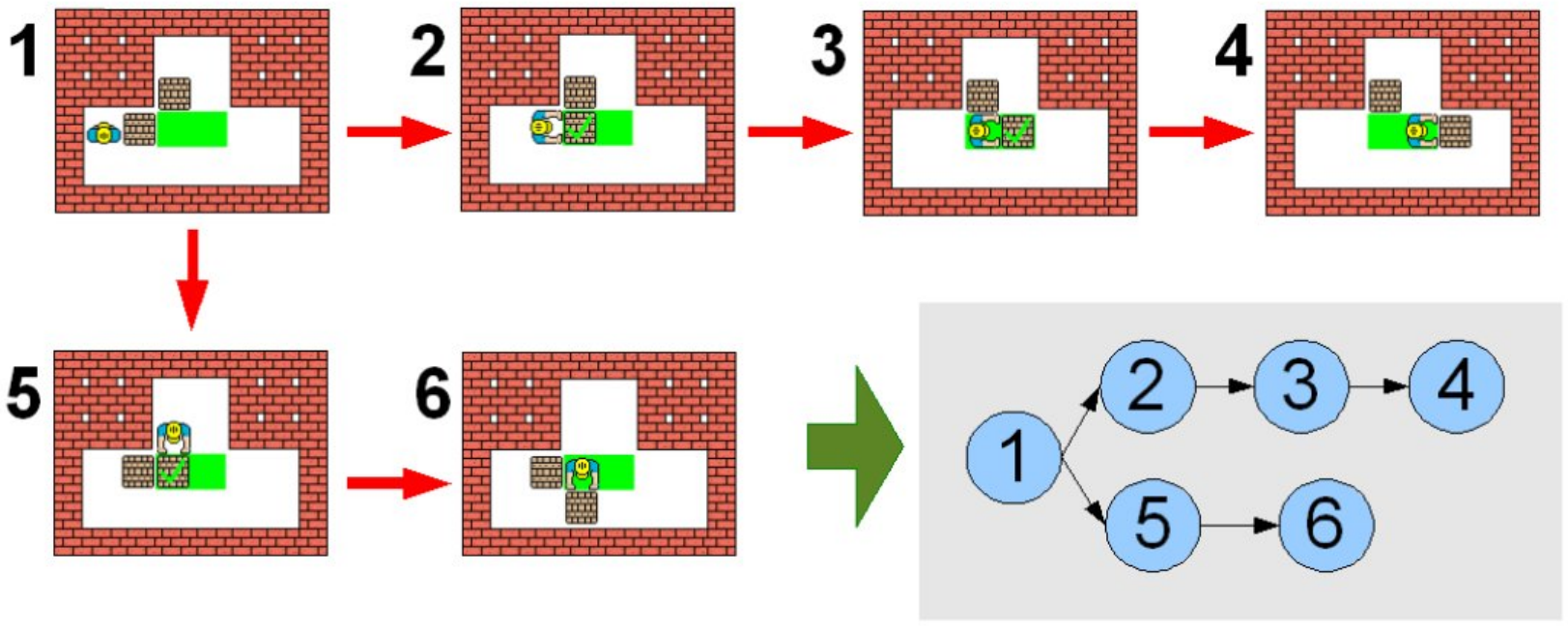
Počet stavů

- zmíněné úlohy mají jen pár stavů – lze nakreslit celý stavový prostor i ručně
- praktické problémy – mnoho stavů – nelze vygenerovat ani počítačem
- $\Rightarrow$ heuristiky

Sokoban



Sokoban



Základní algoritmus

- stavový prostor = graf
- prohledávání grafu pomocí klasických prohledávání: BFS, DFS
- trochu sofistikovanější prohledávání:
  - IDA – iterative deepening
  - obousměrné hledání

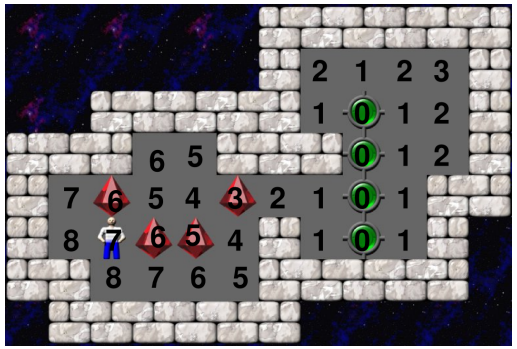
Praktické poznámky

- reprezentace – co je jeden stav (viz Sokoban)
- co potřebuji umět:
  - generování následníků stavu
  - ukládání stavů, test na přítomnost stavu (datový typ množina)

A* search

- úprava prohledávání do šířky (BFS)
- BFS postupuje rovnoměrně (povodeň)
- A* upřednostňuje „nadějnější“ stavy
- heuristické ohodnocení stavů

Heuristika – příklad Sokoban

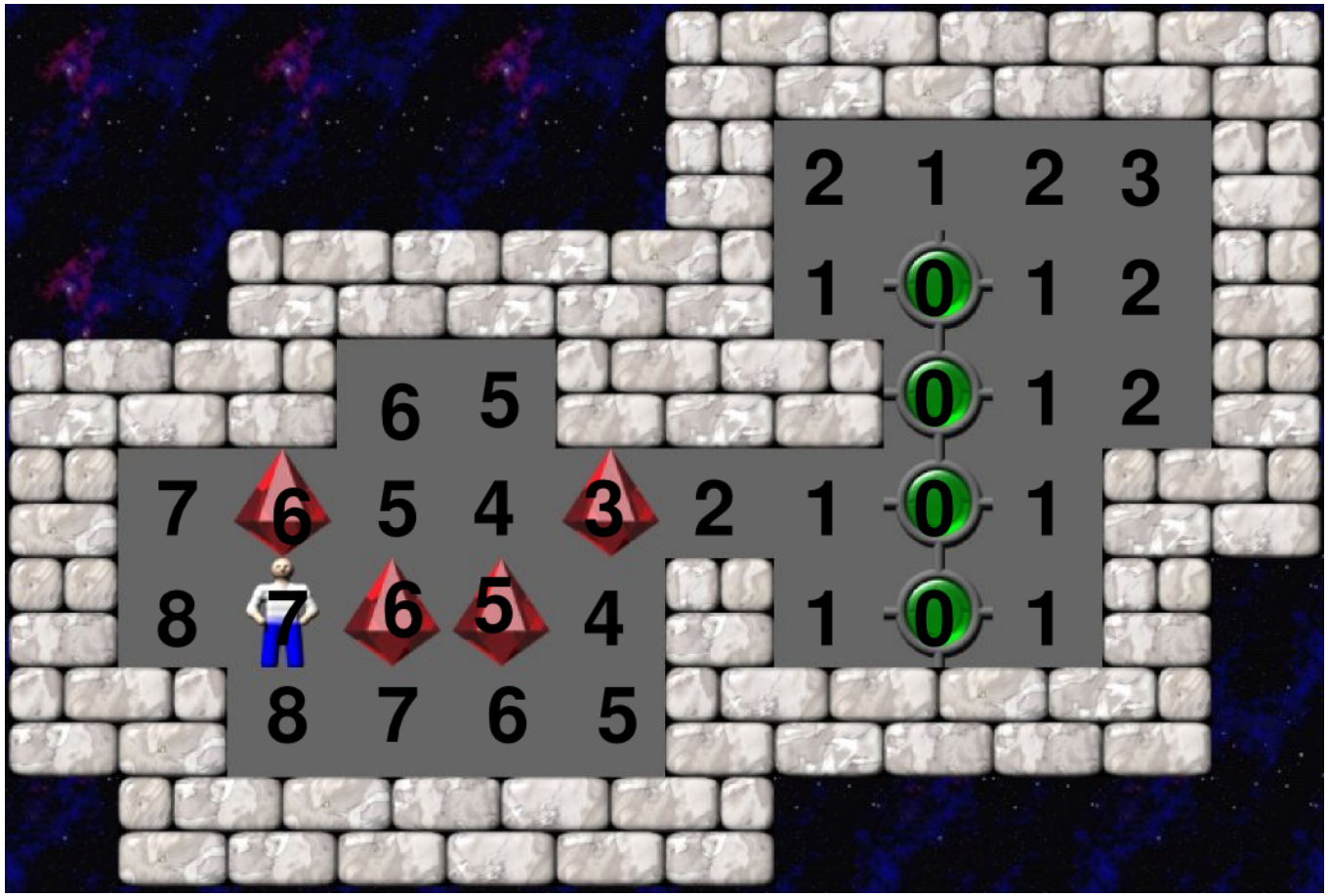


příklady:

- napočítat staticky vzdálonosti jednotlivých políček k cílovým polím, sečíst
- hledat minimální párování bedna – cílové pole
- brát v potaz polohu beden

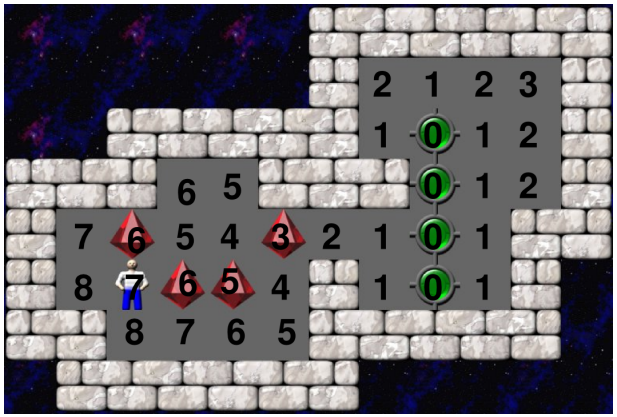
„trade-off“ přesnost heuristiky a časová náročnost jejího výpočtu

Heuristika – příklad Sokoban

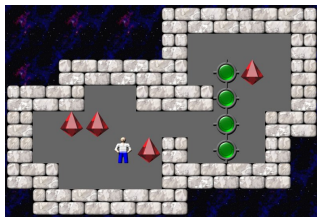


Heuristika – příklad Sokoban

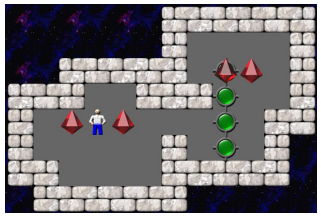
Vzdálenosti od cílových políček:



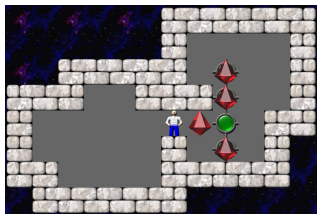
Ohodnocení: 16
= 1+4+5+6



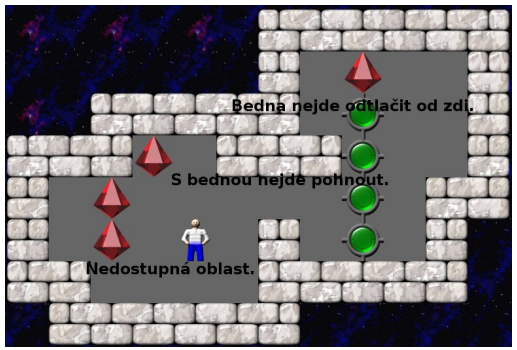
Ohodnocení: 11
= 0+1+4+6



Ohodnocení: 1
= 0+0+0+1

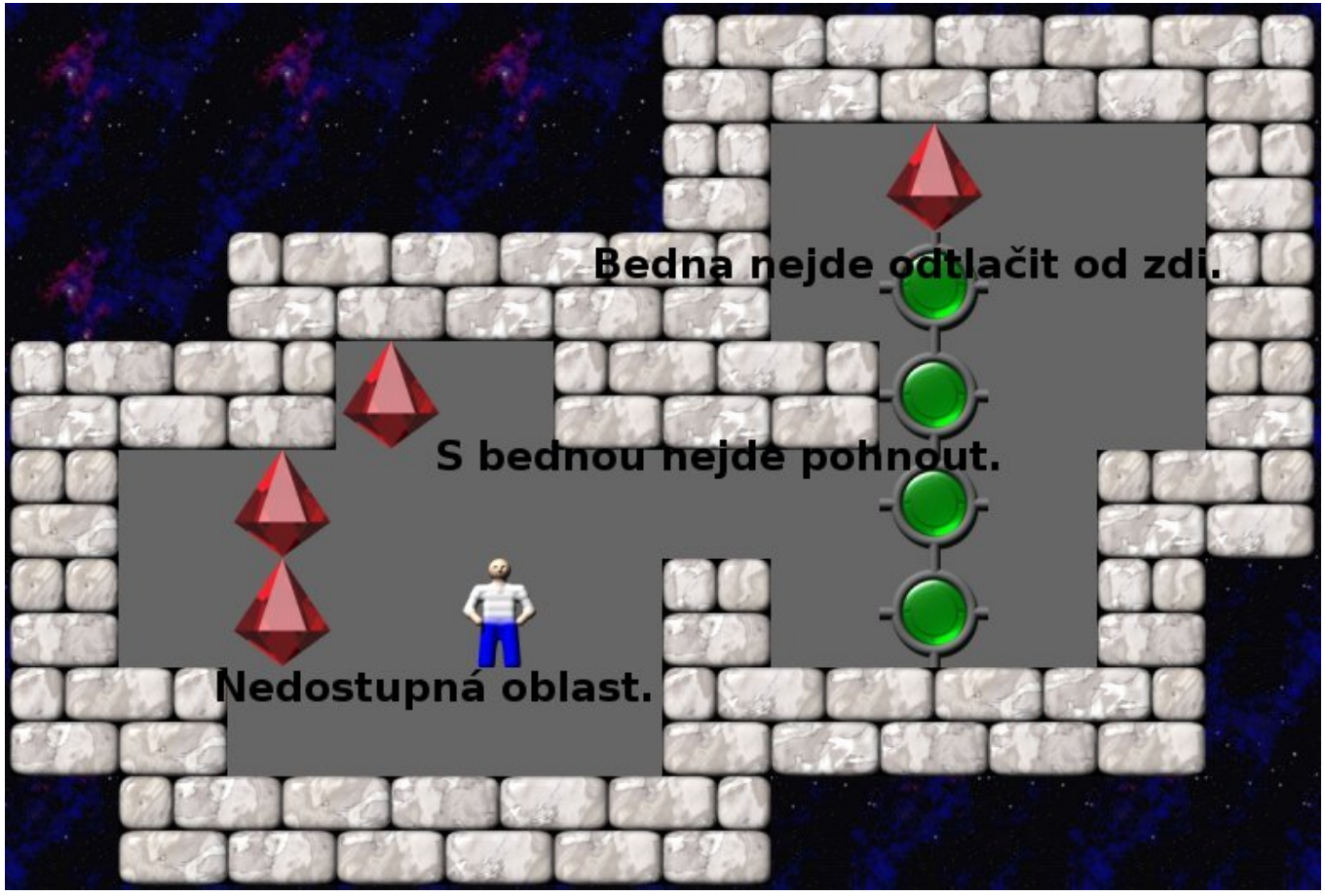


Ořezávání neperspektivních stavů



- dokážu rozhodnout, že ze stavu se nedá dostat do cíle $\Rightarrow$ netřeba z tohoto stavu dál prohledávat
- příklad Sokoban:
  - bedna v rohu
  - bedna u zdi, od které se nemůže dostat

Ořezávání neperspektivních stavů



Problém splnění podmínek

Constraint satisfaction problem (CSP)

- množina proměnných
- domény hodnot
- podmínky

Sudoku

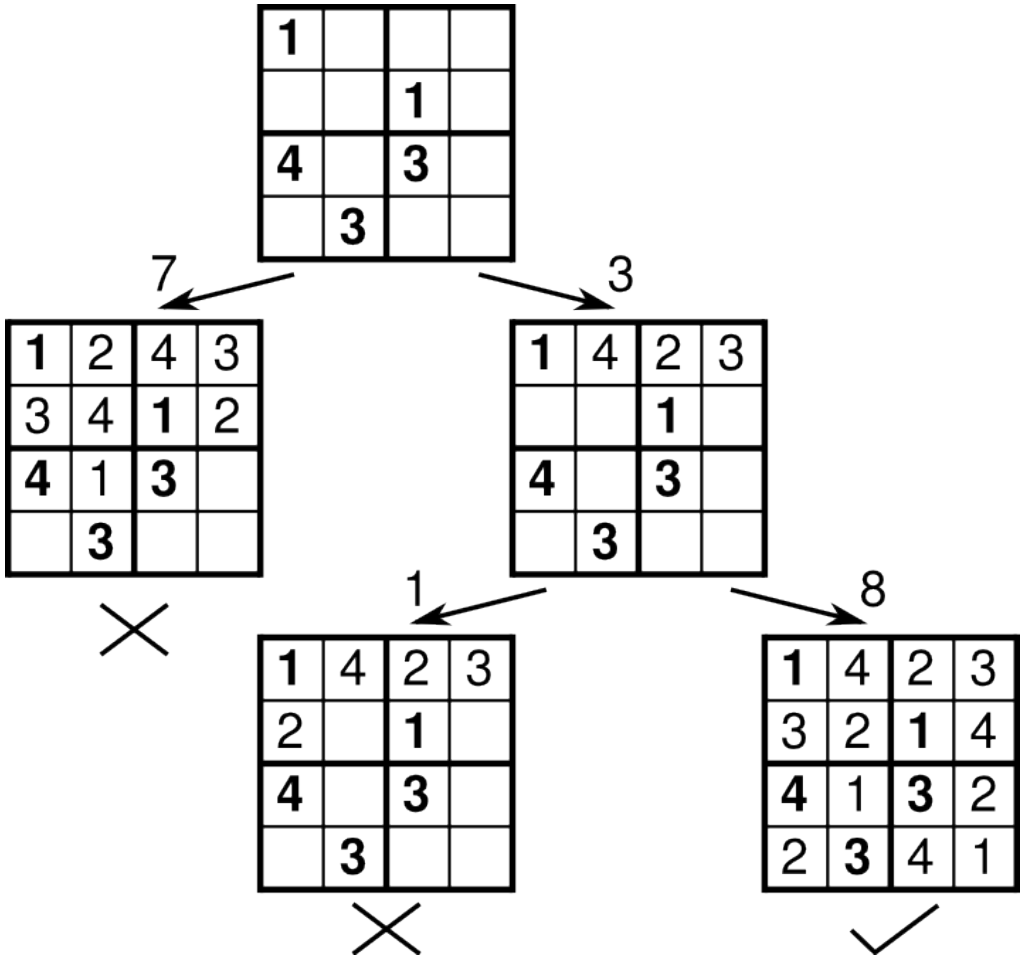
1	a	b	c
d	e	**1**	f
4	g	**3**	h
i	**3**	j	k

- proměnné: $a, b, c, d, \ldots$
- domény hodnot: $\{1, 2, 3, 4\}$
- podmínky (pro j): $j \neq b, j \neq h, j \neq i, j \neq k, j \neq 3, j \neq 1$

Backtracking

- systematické hledání řešení metodou pokus-omyl
- začít s prázným přiřazením hodnot proměnným
- postupně přiřazujeme proměnným hodnoty (systematicky)
- porušení podmínek – vrátit se zpět (backtrack) a zkusit jinou hodnotu
- podmínky kontrolujeme průběžně
- prohledávání do hloubky (DFS)

Backtracking a Sudoku



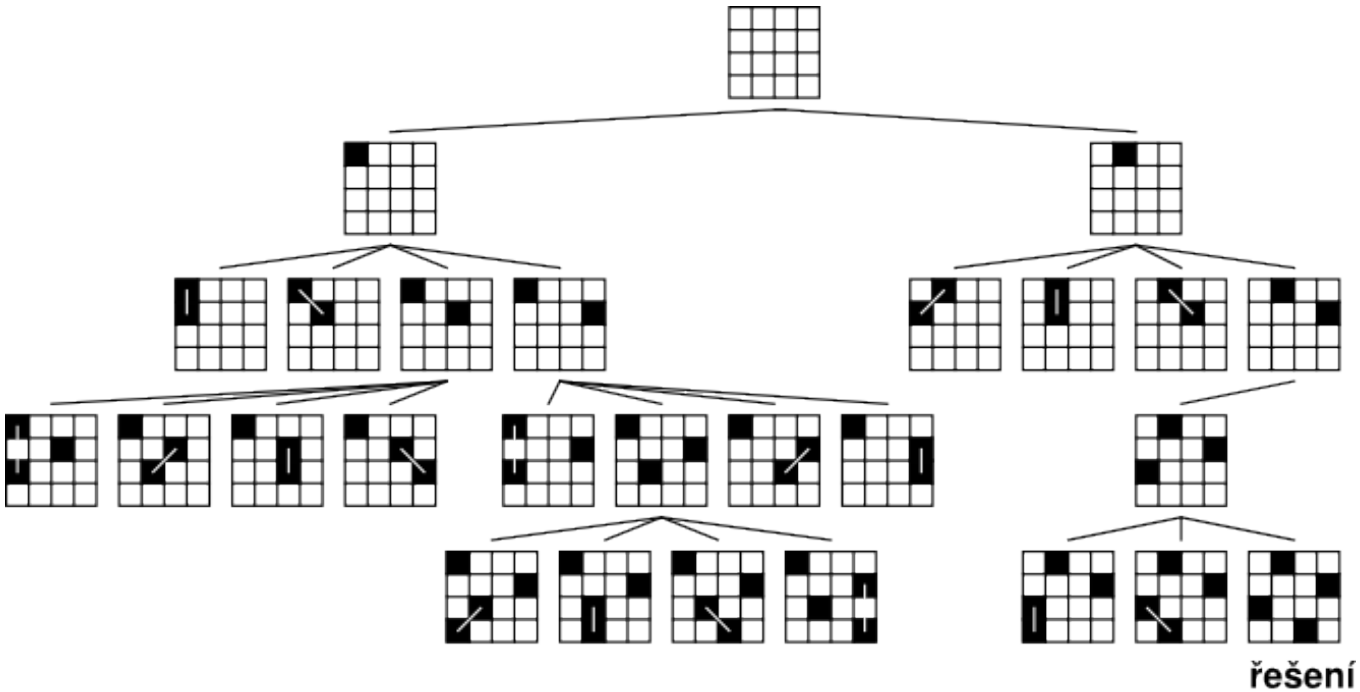
Heuristiky

- forward checking, back jumping
- pořadí výběru proměnných

Příklad: dámy na šachovnici

- rozmístit n dam na šachovnici $n \times n$
- nesmí se vzájemně ohrožovat
- klasické zadání $n = 8$
- zkuste vyřešit pro $n = 4$
- zakreslete strom prohledání při bactrackingu

Čtyři dámy



Propagace podmínek

- pokus o „inteligentní“ řešení
- množina kandidátů pro každou proměnnou
- odvozování nutných hodnot
- efektivnější, ale nezaručuje řešení
- praxe: kombinace propagace podmínek a backtrackingu

Propagace podmínek

1	2,4	2,4	2,3, 4
2,3	2,4	**1**	2,3, 4
4	1,2	**3**	1,2
②	**3**	2,4	1,2, 4

Další problémy CSP

- SAT – splnitelnost logických formulí
- obarvení grafu
- rozvrhování

Shrnutí

- hladový algoritmus
- dynamické programování – „vyplňování tabulky“
- hrubá síla
- prohledávání stavového prostoru, backtracking
- heuristiky – ohodnocení stavů, ořezávání neperspektivních, ...