ohraničená rozumnosť

- ohraničenými výpočtovými prostriedkami (veľkosť pamäti, čas, dokedy treba rozhodnúť o ďalšom kroku),
- ohraničenými nákladmi na úsilie, ktoré možno vynaložiť na získanie údajov z prostredia (ohraničenie doby, ktorú získavanie môže najviac trvať, ohraničenie finančných nákladov získavania apod.),
- neúplnosťou a prípadnou protirečivosťou poznatkov v jeho báze,
- neurčitosťou niektorých poznatkov,
- nepresnosťou niektorých údajov.

Hľadanie riešenia problému

- Ak si rozumný agent prostredníctvom vnemu určí cieľ, môže problém vyriešiť vyhľadaním postupností akcií, vedúcich do cieľa.

```
function JEDNODUCHÝ-KONATEĽ-RIEŠIACI-PROBLÉM(vnem) returns akcia
    static: akcie, postupnosť akcií, na začiatku prázdna
            stav, nejaký opis súčasného stavu sveta
            cieľ, cieľ, na začiatku prázdny
            problém, vyjadrenie problému
    stav ← OBNOV-STAV(stav, vnem)
    if akcie je prázdna then
        cieľ      ← VYJADRI-CIEĽ(stav)
        problém ← VYJADRI-PROBLÉM(stav, cieľ)
        akcie     ← HĽADAJ(problém)
    akcia ← VYBER-PRVÚ(akcie, stav)
    akcie ← ZVYŠOK-AKCIÍ(akcie, stav)
return akcia
```


Hľadanie riešenia

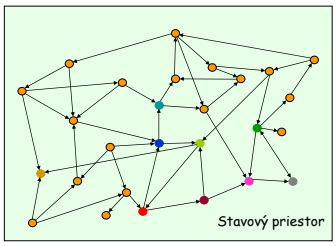
Hľadanie riešenia je prístup k riešeniu problémov, pri ktorom nevychádzame z algoritmu riešenia problému.
Buď ho nepoznáme (možno preto, že ani neexistuje), alebo ho poznáme, ale pre svoju neefektívnosť je prakticky nepoužiteľný. Namiesto toho vychádzame z algoritmu, ako riešenie hľadať.

Definícia typu problému

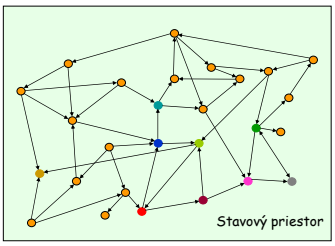
- Na vyjadrenie problému treba poznať niekoľko základných informácií:
  - Začiatočný stav
  - Množinu operátorov
  - Množinu všetkých stavov
  - Cieľový test
  - Cenu cesty

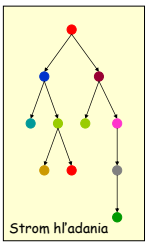
```
datatype PROBLÉM
        components:   STAVY, ZAČIATOČNÝ-STAV, OPERÁTORY,
                      CIEĽOVÝ-TEST, CENA-CESTY
```


Stavový priestor



Strom hľadania





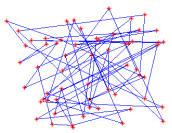
všimnime si, že pri hľadaní sa niektoré stavy môžu navštíviť viac ráz

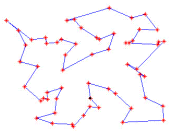
Reálne problémy – problém nájdenia cesty

- **problém naplánovania najvýhodnejšej cestovnej trasy z mesta *A* do mesta *B***
  - **stavy:** mestá, ktoré sa uvažujú pri hľadaní;
  - **začiatočný stav:** mesto *A*;
  - **operátory:** možné presuny z jedného mesta do druhého (existuje cesta na mape);
  - **cieľový test:** "Sme v meste *B*?";
  - **cena cesty:** aplikácia operátora, t.j. presun z jedného mesta do druhého, má cenu rovnajúcu sa vzdialenosti medzi týmito mestami .

7

Reálne problémy – problém obchodného cestujúceho





http://en.wikipedia.org/wiki/Traveling_salesman_problem

8

Reálne problémy – autonómne roboty

- Autonómny robot pri svojej činnosti rieši množstvo problémov:
  - Rozhodovanie, ktorú z možných akcií je treba vykonať
  - Predchádzanie kolíziám
  - Plánovanie trajektórií
  - Interpretácia veľkého množstva numerických dát, poskytovaných senzormi do kompaktnej zmysluplnej symbolickej reprezentácie
  - Diagnostikovanie, prečo niečo nedopadlo podľa očakávaní
  - Atď ...
- Na riešenie týchto problémov je nevyhnutné používať rôzne metódy prehľadávania, pričom v jednom časovom okamihu sa môže vykonávať viacero prehľadávaní súčasne

9

Hračkové problémy

10

Hračkové problémy – 8 hlavolam

1	8	3
6	2	7
4		5

Začiatočný stav

1	2	3
8		4
7	6	5

Cieľový stav

11

Hračkové problémy – 8 hlavolam

- **stavy:** všetky možné konfigurácie políčok na tabuli. Opis stavu obsahuje údaje o umiestnení každého z 8 políčok na tabuli.
- **začiatočný stav:** pre každý zvláštny prípad problému musí byť zadaný začiatočný stav, t.j. východisková konfigurácia na tabuli
- **operátory:** výmena prázdneho miesta s políčkom vpravo, vľavo, hore, dolu
- **cieľový test:** súčasný stav opisuje konfiguráciu, ktorá sa zhoduje s danou cieľovou konfiguráciou
- **cena cesty:** každý krok má cenu 1, takže cena cesty je jednoducho jej dĺžka

12

Hračkové problémy – 15 hlavolam

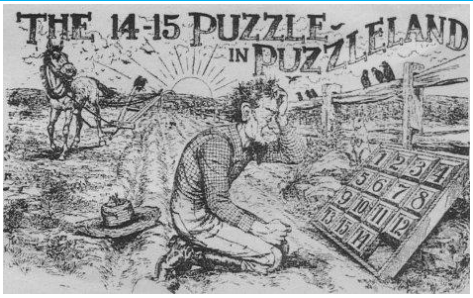
- Sam Loyd, ktorý sám seba označil za najväčšieho experta na puzzle v Amerike, v roku 1878 ponúkol prvému človeku, ktorý vyrieši tento hlavolam, odmenu 1000 dolárov.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

→ ?

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

Hračkové problémy – 15 hlavolam



NIKOMU SA TO VŠAK NEPODARILO ☺

Hračkové problémy – 15 hlavolam

- Aby bolo možné hlavolam vyriešiť, je nutné, aby bola hodnota **N mod 2** pre oba stavy rovnaká, pričom:

  N mod 2 = $n_2 + n_3 + \ldots + n_{15}$ + číslo riadku prázdneho políčka

  n_i - počet všetkých tých políčok j, pre ktoré platí i<j a zároveň sú umiestnené pred políčkom i

1	2	3	4
5	10	7	8
9	6	11	12
13	14	15	

$n_2 = 0$ $n_3 = 0$ $n_4 = 0$
$n_5 = 0$ $n_6 = 0$ $n_7 = 1$
$n_8 = 1$ $n_9 = 1$ $n_{10} = 4$
$n_{11} = 0$ $n_{12} = 0$ $n_{13} = 0$
$n_{14} = 0$ $n_{15} = 0$

→ N = 7 + 4

Hračkové problémy – 15 hlavolam

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

→ ?

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

N = 4 N = 5

4 mod 2= 0 0 ≠ 1 5 mod 2= 1

- Druhý stav teda nie je z prvého dosiahnuteľný a Sam Loyd sa o svoje peniaze nemusel báť.

Hračkové problémy – (n^2-1) hlavolam

- Aká je veľkosť stavového priestoru pre (n^2-1) hlavolam ?

	Počet stavov	Čas (10^8 stavov/sekunda)
9 hlavolam	9! = 362,880	0.036 sekundy
15 hlavolam	16! ~ 2.09 x 10^{13}	~ 55 hodín
24 hlavolam	25! ~ 10^{25}	> 109 rokov

- Ale iba **POLOVICA** týchto stavov je dosiahnuteľných z ľubovoľného stavu.

Hračkové problémy – šach, dáma, go

- Už v minulosti boli hry, ktorých úspešné vyriešenie vyžadovalo preskúmanie rôznych alternatív výzvou pre ľudskú inteligenciu.
  - Šach – pôvod Perzia, India, pred 4000 rokmi
  - Dáma – prvé zmienky na starých egyptských maľbách spred 3600 rokov
  - Go – pôvod Čína, pred 3000 rokmi

Hračkové problémy – šach

- Veľkosť stavového priestoru 10^{120}
  - 10^{120} > počet všetkých atómov vo vesmíre
- 200 miliónov pozícií za sekundu = 10^{100} rokov na vyhodnotenie všetkých možných hier
  - Vesmír existuje iba 10^{10} rokov
- 1957 – Newell a Simon: „Do **desiatich rokov** sa počítač stane šachovým veľmajstrom“
- predpoveď im celkom nevyšla. podcenili potrebný čas, ale podcenili aj umelú inteligenciu (dnes je už počítačový program s umelou inteligenciou nielen šachový veľmajster, ale dokonca poráža majstra sveta).

Hračkové problémy – šach

Kasparov		Deep Blue
165 cm	Výška	195 cm
80 kg	Hmotnosť	1,1 tony
34 rokov	Vek	4 roky
50 miliárd neurónov	Počítače	32 RISC procesorov + 256 VLSI šach. “enginov“
2 pozícií/s	Rýchlosť	200,000,000 pozícií/s
Obrovské	Znalosti	Primitívne
Electrické/chemické	Napájanie	Electrické
Enormné	Ego	Žiadne

Deep Blue vyhráva po 3 výhrach, 1 prehre a 2 remízach

šach

- 10. 2. 1996 Philadelphia: Deep Blue porazil Kasparova v normálnej partii = vôbec prvý raz zvíťazil počítač nad úradujúcim majstrom sveta (celkovo ale zápas na 6 partií Kasparov vyhral 3 a 2 remizoval, 4:2)
- **11.5.1997** – Gary Kasparov prehráva s počítačom Deep Blue zápas na 6 partií: 3½-2½, prehral druhú a poslednú rozhodujúcu partiu. V poslednej spravil jasnú chybu, v druhej sa mu zdal počítač príliš tvorivý, IBM poprela ľudskú intervenciu.
- 2.8.2003 – Gary Kasparov remizuje s programom Deep Junior
  - cena je približne 100 dolárov
  - 3 milióny pozícií/s
  - knižnica otvorení
  - zaujímavejšie ťahy sa hlbšie prehľadávajú
  - modelovanie protihráča

šach

- 2002: Kramnik – Deep Fritz 4:4
- 2003: Kasparov – Deep Fritz 2:2
- 2006: Kramnik – Deep Fritz 2:4
  - program Fritz dnes (~~2011~~, 2012) stojí ~~99.90 €~~ 49.90 €
  - minimálne požiadavky: Pentium 300 MHz, 64 MB RAM, Windows Vista or XP (SP 2), DVD ROM drive, Windows Media Player 9.

Rybka

- vyhral viacero turnajov šachových programov, vrátane majstrovstiev sveta 2007, 2008, 2009, 2010
- vyhral partie s veľmajstrami, ktorí dostali výhodu pešiaka
- vyhral zápas s veľmajstrom, ktorý mal všetky možné výhody okrem pešiaka: dvojnásobný čas na rozmýšľanie, naopak Rybka databázu otvorení obmedzenú len na 3 ťahy, obmedzenie na heš tabuľku 1/2 GB, bez databázy koncových hier. 4a1/2:1a1/2

Rybka

- verzie
  - vývoj začal 2003
  - 1 beta 2005 ELO=2885
  - 2.2 ELO=3110
  - 3 2008 ELO= cca +100 (v 2010 najvyššie vyhodnotený šachový program s ELO=3227)
  - 4 2010
  - 4+ cluster
  - 5 vraj sa pripravuje na 2012

Rybka

- α/β hľadanie
- používa reprezentáciu stavu pomocou bitových dosák
  - bitová doska (bitboard) dátová štruktúra, zvláštny prípad bitovej množiny
  - jedna bitová doska má 64 bitov (toľko má šachovnica políčok)
- stav partie sa reprezentuje:
  - dvanástimi bitovými doskami:
    - biele/čierne pešiaky, strelci, jazdci, veže, dáma, kráľ
  - niekoľkými stavovými bitmi:
    - b/č dostal šach,
    - b/č ešte môže robiť rošádu.
- tím:
  - Vaclav „Vasik“ Rajlich, jr. medzinárodný majster v šachu
  - Iweta Radziewicz Rajlich , medzinárodná majsterka v šachu
  - Larry Kaufman, majster sveta v šachu 2008 (vyhodnocovacia funkcia)
  - Jeroen Noomen, Dagh Nielsen (otvorenia)

25

Vasik Rajlich

- 1971 Cleveland
- MIT
- medzinárodný majster v šachu
- autor šachového programu Rybka http://www.rybkachess.com/
- &&&&&&&&&&&&&&&&&&
- VÚMS Praha, U Michigan, CMU
- Wayne State U
- profesor, softvérové inžinierstvo

Rybka

- podľa stavu 2010: najvyššie hodnotený program, Elo = 3227
- vyhral viacero turnajov šachových programov, vrátane majstrovstiev sveta World Computer Chess Championships (WCCC): 2007, 2008, 2009, 2010
- June 29, 2011 at 1:03 pm:
  - Rybka has been disqualified and banned for the plagiarizing of two other chess engines, Crafty and Fruit.
  - International Computer Games Association (ICGA):
  - anulovala výsledky Rybky spätne za roky 2006-2010
  - odobrala tituly majstra sveta 2007-2010
  - V Rajlich má doživotný zákaz súťažiť na akomkoľvek turnaji organizovanom ICGA
  - musí vrátiť sošky a peniaze

27

Rybka - spor

- obvinenie:
  - VR nedodržal pravidlo č. 2 pre turnaje ICGA. *Vyžaduje, aby autor šachového programu v prihláške na turnaj uviedol všetky zdroje, ktoré použil pri tvorbe svojho programu.*
  - *"2. Each program must be the original work of the entering developers. Programming teams whose code is derived from or including game-playing code written by others must name all other authors, or the source of such code, in the details of their submission form. Programs which are discovered to be close derivatives of others (e.g., by playing nearly all moves the same), may be declared invalid by the Tournament Director after seeking expert advice. For this purpose a listing of all game-related code running on the system must be available on demand to the Tournament Director."*
- argumentácia:
  - nemusí ísť o presné, doslovné skopírovanie
  - treba priznať aj to, že nejaká časť je odvodenina (derivative) od niečo iného bez ohľadu na zhodu/podobnosť na úrovni zdrojového programu
  - rozsah: nedá sa určiť presne (6? 12? 20? 200? riadkov), rozumné posúdenie funkcionality
- ak by uviedol, že zdrojom pre časť Rybky je program Fruit (autor Letouzey):
  - Letouzey by sa musel vyjadriť ako spoluautor Rybky, ktorý z jeho programov (Fruit alebo Rybka) má súťažiť

28

Rybka - spor

- obrana:
  - VR tvrdil, že je úplne nevinný,
  - obvinil vyšetrovanie, že nebolo nestranné
  - odmietol spolupracovať pri vyšetrovaní aj brániť sa.
  - verejne prehlásil iba:
  - *Rybka neobsahuje časť programu, ktorá rieši ako hrať (game-playing code), ktorú by napísal niekto iný okrem štandardných výnimiek, ktoré sa nedajú chápať ako programovanie hry.*
- Søren Riis:
  - implementácia podobných koncepcií a algoritmov vyhodnocovania v šachovom programe povedie spravidla ku podobnostiam v zdrojových programoch aj ak sa zdrojové časti nekopírovali. Rybka implementuje koncepcie a algoritmy, získané z programu Fruit.

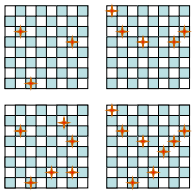
29

šach

- v súčasnosti (2012) sa za najlepší šachový program považuje Houdini (autor Robert Houdart)
- Rybka sa predáva...

30

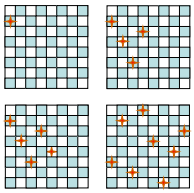
Hračkové problémy – 8 dám (1.formulácia)



- **Stavy:** všetky možné konfigurácie ľubovoľného možného (0-8) počtu dám na šachovnici
- **Počiatočný stav:** žiadna dáma na šachovnici
- **Operátory:** položenie dámy na ľubovoľné políčko šachovnice
- **Cena cesty:** 0
- **Cieľový test:** 8 dám na šachovnici umiestnených tak, že sa navzájom neohrozujú

→ 64x63x...x57 ~ $3x10^{14}$ stavov

Hračkové problémy – 8 dám (2. formulácia)



- **Stavy:** všetky možné konfigurácie ľubovoľného možného (0-8) počtu dám na šachovnici také, že ani jedna z dám nie je ohrozená
- **Počiatočný stav:** žiadna dáma na šachovnici
- **Operátory:** položenie dámy na ľubovoľné políčko v najľavejšom prázdnom stĺpci také, že ju na ňom neohrozuje žiadna iná dáma
- **Cena cesty:** 0
- **Cieľový test:** 8 dám na šachovnici umiestnených tak, že sa navzájom neohrozujú

→ 2057 stavov

Hračkové problémy – kryptografické problémy

SEND	9567
+MORE	+1085
MONEY	10652

- **stavy:** všetky možné zápisy hlavolamu (zápis výrazu a hodnoty), v ktorých sa vyskytujú písmená a/alebo číslice
- **začiatočný stav:** hlavolam (zápis výrazu a hodnoty), v ktorom sa vyskytujú iba písmená
- **operátory:** náhrada všetkých výskytov nejakého písmena za číslicu, ktorá sa ešte medzi číslicami v hlavolame nevyskytuje
- **cieľový test:** hlavolam obsahuje iba číslice a je aritmeticky správny
- **cena cesty:** 0, pretože všetky riešenia sú rovnako dobré

Charakteristiky problémov

- riešením problému je stav alebo cesta
- problém rozložiteľný na samostatne riešiteľné podproblémy

- problémy s ignorovateľnými krokmi riešenia
- problémy s odčiniteľnými krokmi riešenia
- problémy s neodčiniteľnými krokmi riešenia

Hľadanie riešenia

Hľadanie riešenia je prístup k riešeniu problémov, pri ktorom nevychádzame z algoritmu riešenia problému.

Buď ho nepoznáme (možno preto, že ani neexistuje), alebo ho poznáme, ale pre svoju neefektívnosť je prakticky nepoužiteľný. Namiesto toho vychádzame z algoritmu, ako riešenie hľadať.

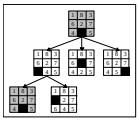
Hľadanie riešenia - algoritmus

```
function VŠEOBECNÉ-HĽADANIE(problém, stratégia)
    returns riešenie alebo neúspech

    inicializuj strom hľadania použitím začiatočného stavu z problém
    loop do
        if nie sú nerozvité uzly then return neúspech
        vyber list za uzol na rozvitie podľa stratégia
        if uzol predstavuje cieľový stav then return zodpovedajúce riešenie
        else rozvi uzol a pripíš vygenerované uzly do stromu hľadania
end
```

Stavový priestor a graf (strom) hľadania

- Reprezentácia uzla:

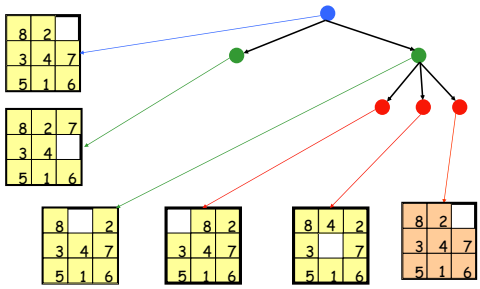


- zodpovedajúci stav zo stavového priestoru,
- uzol v strome hľadania, z ktorého sa daný uzol vygeneroval,
- operátor, ktorý sa aplikoval pri generovaní uzla (rodičovský uzol),
- počet uzlov na ceste z koreňa do daného uzla (hĺbka uzla),
- cena cesty zo začiatočného uzla do daného uzla.

datatype UZOL
components: STAV, RODIČOVSKÝ-UZOL, OPERÁTOR, HĹBKA, CENA-CESTY

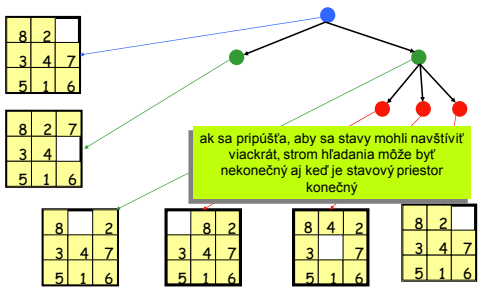
37

Uzly v strome hľadania ≠ stavy



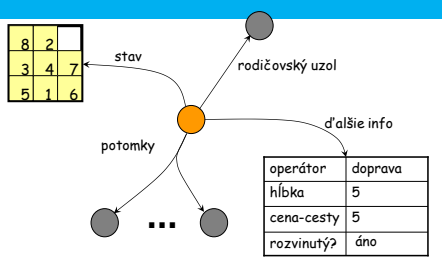
38

Uzly v strome hľadania ≠ stavy



39

dátová štruktúra pre uzol



hĺbka uzla N
= dĺžka cesty z koreňa do N
(hĺbka koreňa = 0)

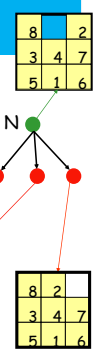
40

rozvinutie uzla

rozvinutie uzla N v strome hľadania pozostáva z:

1) vyhodnotenia funkcie nasledovníka na STAV(N)
2) vygenerovania potomka/nasledovníka uzla N pre každý stav, ktorý vráti funkcia nasledovníka

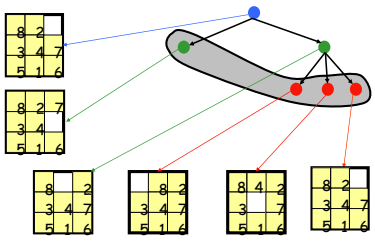
generovanie uzla ≠ rozvinutie uzla



41

Okraj stromu hľadania

- okraj je množina všetkých uzlov (v strome hľadania), ktoré ešte nie sú rozvinuté



Je okraj totožný s množinou listov?

42

Front – štruktúra na zápis množiny uzlov

- Nad frontom definujeme tieto operácie:
  - VYTVOR-FRONT(*prvky*) vytvorí front s danými prvkami;
  - PRÁZDNY(*front*) vráti true práve vtedy, ak front neobsahuje žiadne prvky;
  - VYBER(*front*) odstráni prvok z frontu a vráti ho;
  - ZARAĎ-DO-FRONTU(*prvky*, *front*) vráti front po zaradení prvkov do pôvodného frontu. Rôzne druhy tejto funkcie určujú rôzne algoritmy hľadania.

Všeobecný algoritmus hľadania

```
function VŠEOBECNÉ-HĽADANIE(problém, ZARAĎ-DO-FRONTU)
    returns riešenie alebo neúspech
    static: front,  front obsahujúci vygenerované a nerozvité uzly,
                    na začiatku prázdny
            uzol, uzol stromu hľadania

    front ← VYTVOR-FRONT(VYTVOR-UZOL(ZAČIATOČNÝ-STAV[problém]))
    loop do
       if front je prázdny  then return neúspech
       uzol ← VYBER(front)
       if CIEĽOVÝ-TEST[problém] aplikovaný na STAV(uzol) je úspešný
        then return VYBER-RIEŠENIE(uzol)
    front ← ZARAĎ-DO-FRONTU(ROZVI(uzol, OPERÁTORY[problém]), front)
    end
```

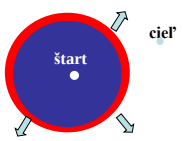
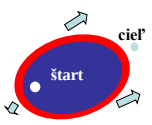

Stratégie hľadania

- **Neinformované (slepé)**
  - Nemajú k dispozícii nejakú doplňujúcu informáciu o probléme
  - Poradie generovania stavov závisí iba od informácií získaných hľadaním a nie je ovplyvnené ani nepreskúmanou časťou grafu ani vlastnosťami cieľového stavu.
- **Informované (heuristické)**
  - Majú k dispozícii nejakú doplňujúcu informáciu o probléme
  - Heuristická informácia sa často využíva na to, aby sa zvýšila efektívnosť hľadania (t.j. znížila časová a/alebo pamäťová zložitosť) aj za cenu, že nebude dodržané ďalšie kritérium, a síce prípustnosť a/alebo úplnosť

heuristika

- ηὕρηκα [heuréka] = našiel (objavil) som to – Archimedes
- ηὑρίσκω = nájsť, objaviť
- spôsob riešenia problému, pre ktorý nemáme algoritmus alebo presný postup ⇒ heuristické riešenie problémov
- Polya: Ako to vyriešiť. 1954:
  - ak nerozumiete riešenému problému, skúste si ho nakresliť
  - ak neviete nájsť riešenie, predstavte si, že ho máte a pozrite sa, či z neho neviete odvodiť postup (pracovať odzadu)
  - ak je problém abstraktný, skúste najprv riešiť konkrétny príklad
  - skúste najprv riešiť všeobecnejší problém (paradox vynálezcu: ambicióznejší plán môže mať lepšie vyhliadky na jeho vyriešenie)
- heuristika v informatike: postup, ktorý zvyčajne vedie k dobrému riešeniu, avšak nezaručuje, že sa nájde najlepšie riešenie, ani že sa nájde v krátkom čase, ani že sa vôbec nájde.

Stratégie hľadania

Neinformované hľadanie	Informované hľadanie
	

Stratégie hľadania

- **Úplnosť**
  Zaručuje hľadanie s danou stratégiou, že sa nájde riešenie, ak existuje?
- **Časová zložitosť**
  Ako dlho trvá, kým sa nájde riešenie?
- **Pamäťová zložitosť**
  Koľko pamäti treba na vykonanie hľadania?
- **Prípustnosť**
  Nájde sa pomocou danej stratégie najlepšie riešenie, ak existuje aspoň jedno riešenie?