



UMELÁ INTELIGENCIA

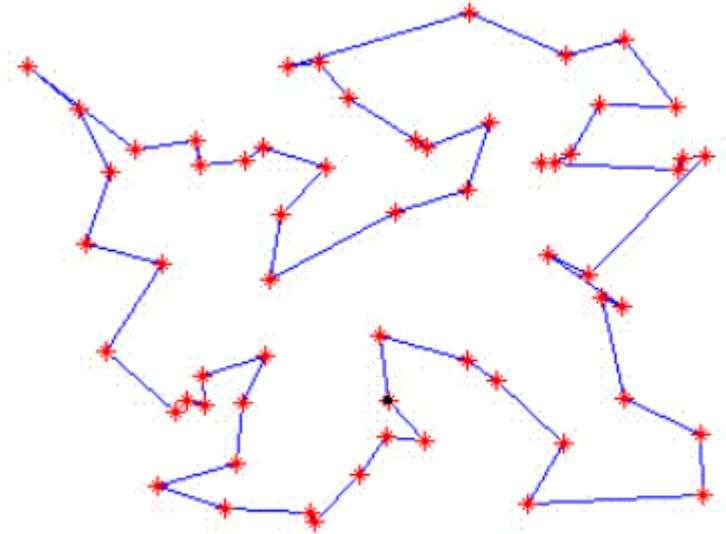
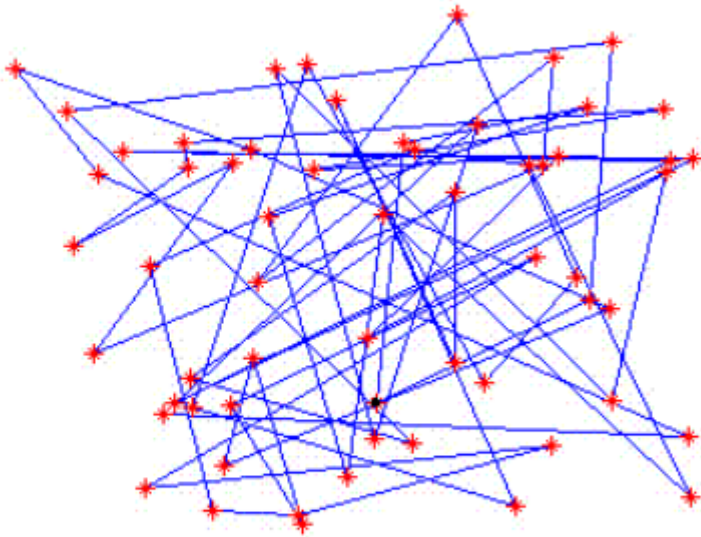
Reálne problémy – problém nájdania cesty

2

- Problém naplánovania najvýhodnejšej cestovnej trasy z mesta A do mesta B
 - ▣ **stavy:** mestá, ktoré sa uvažujú pri hľadaní
 - ▣ **začiatkový 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

Reálne problémy – problém obchodného cestujúceho

3



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

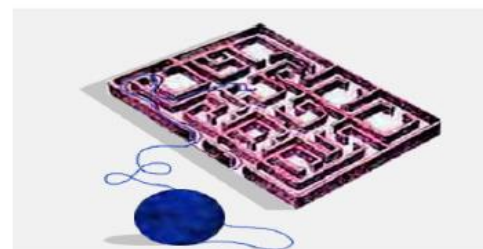
Reálne problémy – autonómne roboty

4

- 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
 - ▣ Atd'. ...
- Na riešenie týchto problémov je nevyhnutné používať rôzne metódy prehľadávania, pričom v jednom časovom okamihu sa môže vykonávať viacero prehľadávaní súčasne

Hračkové problémy

5



Hračkové problémy – 8 hlavolam

6

1	8	3
6	2	7
4		5

Začiatočný stav

1	2	3
8		4
7	6	5

Cieľový stav

Hračkové problémy – 8 hlavolam

7

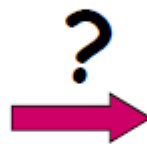
- **Stavy:** všetky možné konfigurácie políček na tabuli. Opis stavu obsahuje údaje o umiestnení každého z 8 políček na tabuli.
- **Začiatočný stav:** pre každý zvláštny prípad problému musí byť zadaný začiatkový stav, t.j. východisková konfigurácia na tabuli
- **Operátory:** výmena prázdneho miesta 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

Hračkové problémy – 15 hlavolam

8

- Sam Loyd, ktorý sám seba označil za najväčšieho experta na puzzle v Amerike, v roku 1896 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

9



NIKOMU SA TO VŠAK NEPODARILO 😊

Hračkové problémy – 15 hlavolam

10

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

$N = n_2 + n_3 + \dots + n_{15} + \text{číslo riadku prázdneho políčka}$

n_i - počet všetkých tých políčok j , pre ktoré platí $v_i < v_j$ a zároveň sú umiestnené pred políčkom i , t.j. $j < i$

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	

Hračkové problémy – 15 hlavolam

11

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	

napríklad:

$v_1=1, v_2=2, \dots,$
 $v_{10}=6, \dots$

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

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

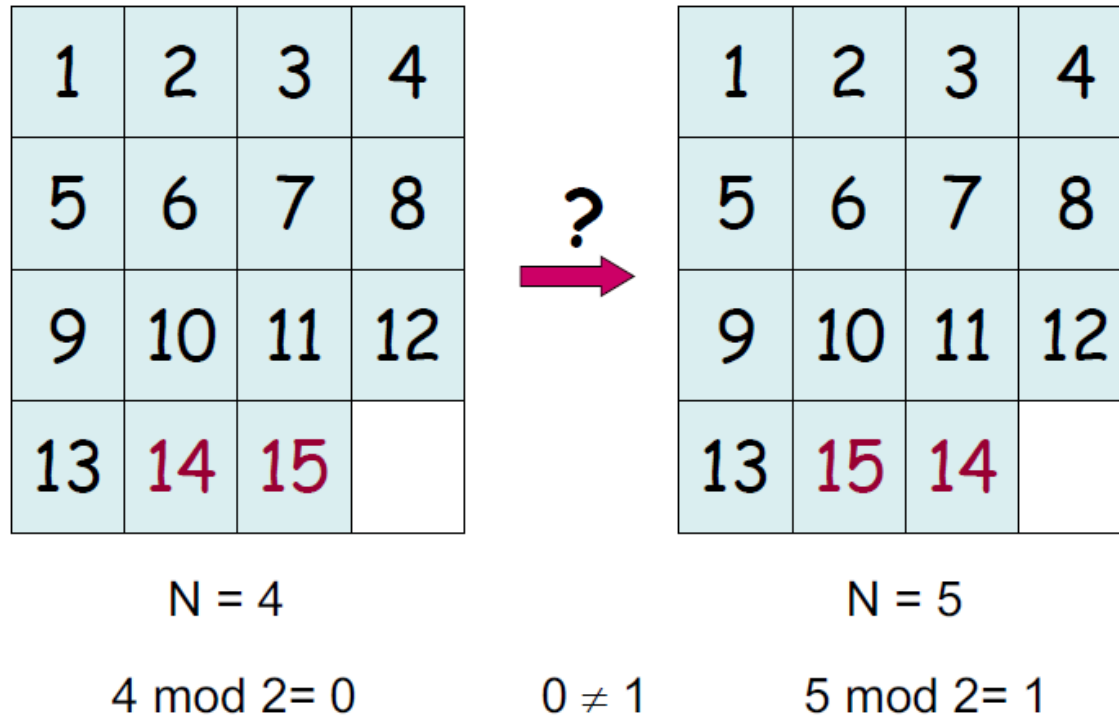
$$\rightarrow N = 7 + 4 = 11$$

pomôcka: $n_{10} = 4$ – prečo?

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

Hračkové problémy – 15 hlavolam

12



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

Hračkové problémy – 15 hlavolam

13

- 15 – hlavolam v skutočnosti vymyslel niekto iný. Pôvodný nápad mal vraj jeden poštár zo štátu N.Y. v roku 1874.
- Prvýkrát cenu tiež vypísal niekto iný. Zubár z Massachusetts v roku 1880. A za niečo iné. Za postup na vyriešenie hlavolamu. Cenou boli falošné zuby a 100\$.
- Loyd si začal pripisovať vymyslenie hlavolamu až neskôr (1891)
- Loyd naozaj vypísal cenu za vyriešenie tej zvláštnej inštancie hlavolamu (zámena 14 – 15). Jeho formulácia vraj bola:
 - *Cieľom je posúvať kamene, vždy po jednom, tak, aby sa dostali do dokonalej postupnosti, vrátane kameňov s číslami 14 a 15.*

Hračkové problémy – $(N^2 - 1)$ hlavolam

14

- 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	$16! \sim 2.09 \times 10^{13}$	~ 55 hodín
24 hlavolam	$25! \sim 10^{25}$	> 109 rokov

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

Hračkové problémy – šach, dáma, go

15

- 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 mal'bách spred 3600 rokov
 - ▣ go – pôvod Čína, pred 3000 rokmi

Hračkové problémy – šach

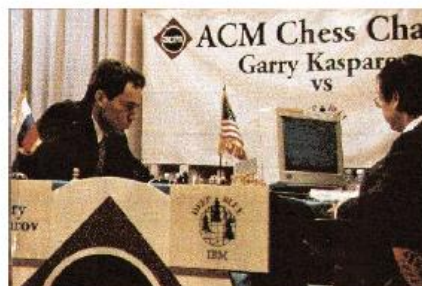
16

- 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

17

Kasparov



165 cm

80 kg

34 rokov

50 miliárd neurónov

2 pozícií/s

Obrovské

Electrické/chemické

Enormné

Výška

Hmotnosť

Vek

Počítače

Rýchlosť

Znalosti

Napájanie

Ego



Deep Blue

195 cm

1,1 tony

4 roky

32 RISC procesorov
+ 256 VLSI šach. "enginov"

200,000,000 pozícií/s

Primitívne

Electrické

Žiadne

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

Hračkové problémy – šach

18

- 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

Deep Fritz

19

- ❑ 2002: Kramnik – Deep Fritz 4:4
- ❑ 2003: Kasparov – Deep Fritz 2:2
- ❑ 2006: Kramnik – Deep Fritz 2:4
 - ▣ Fritz 17 stojí dnes (10.9.2020) €79.90
 - ▣ minimálne požiadavky: Dual Core, 2 GB RAM,
 - ▣ Windows 7 or 8.1, DirectX11, grafická karta 256 MB alebo viac.



Rybka

20

- Vyhrál viacero turnajov šachových programov, vrátane majstrovstiev sveta 2007, 2008, 2009, 2010
 - Vyhrál partie s veľmajstrami, ktorí dostali výhodu pešiaka
 - Vyhrál 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

21

□ 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 mala vzniknúť v r. 2012 ale dodnes nevyšla

Rybka

22

- α/β hľadanie (alpha–beta pruning)
- Používa reprezentáciu stavu pomocou bitových dosiek
 - ▣ bitová doska (bitboard) dátová štruktúra, zvláštny prípad bitovej množiny
 - ▣ jedna bitová doska má 64 bitov (tol'ko 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)

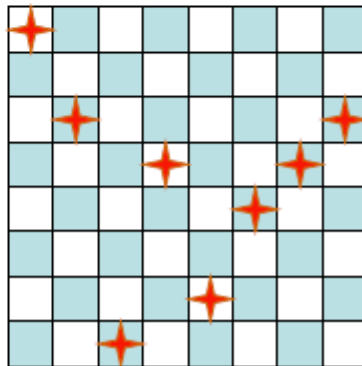
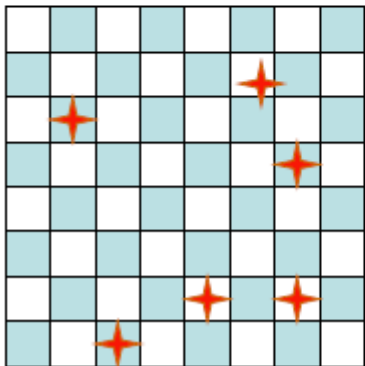
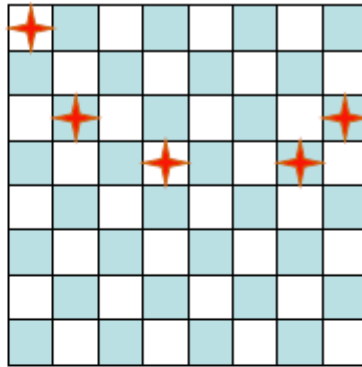
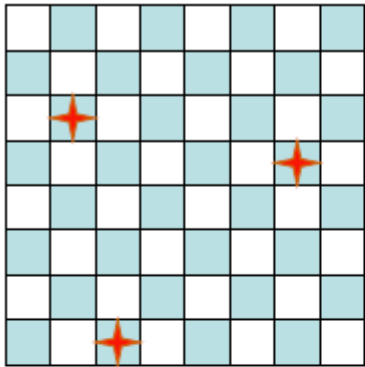
Šach

23

- V súčasnosti (09/2022) sa za najlepšie šachový program považuje Stockfish 14
 - ▣ ELO = 3535
 - ▣ completely free, open source and cross-platform
- 2. miesto si drží Dragon by Komodo 3.1
 - ▣ ELO = 3529
- 3. miesto má aktuálne Fat Fritz 2
 - ▣ ELO = 3514

Hračkové problémy – 8 dām (1. formulácia)

24

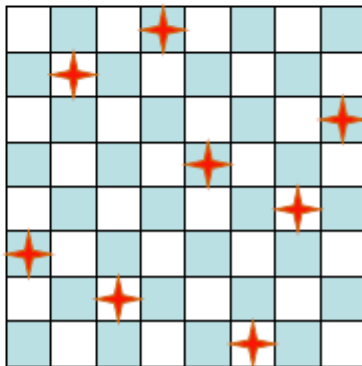
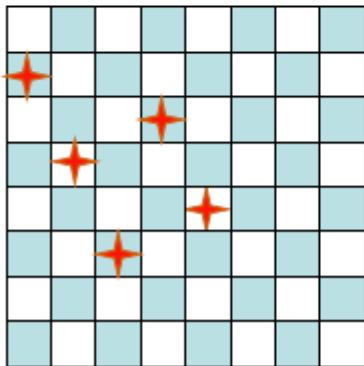
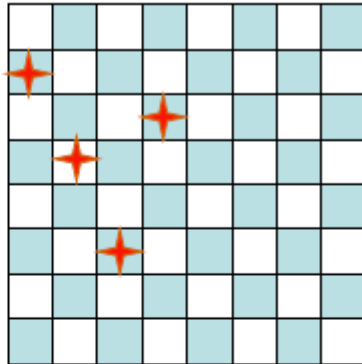
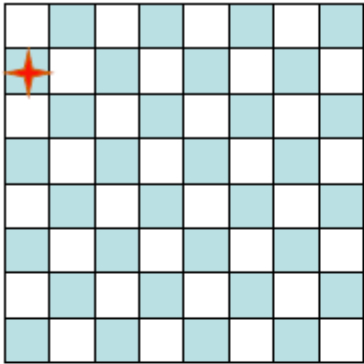


- **Stavy:** všetky možné konfigurácie ľubovoľného možného (0-8) počtu dām na šachovnici
- **Počiatočný stav:** žiadna dáma na šachovnici
- **Operátory:** polozenie dámy na ľubovoľné políčko šachovnice
- **Cena cesty:** 0
- **Cieľový test:** 8 dām na šachovnici umiestnených tak, že sa navzájom neohrozujú

$64 \times 63 \times \dots \times 57 \sim 3 \times 10^{14}$ stavov

Hračkové problémy – 8 dām (2. formulácia)

25



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

2057 stavov

Charakteristiky problémov

26

- 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

27

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

28

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*

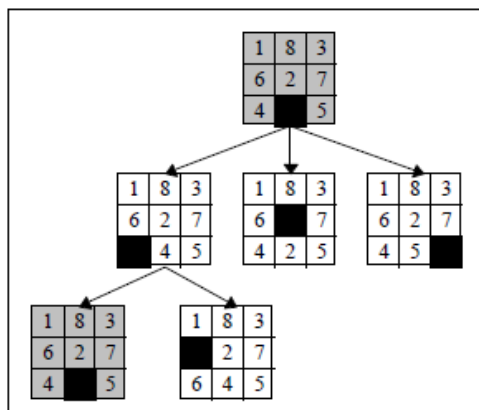
else rozvi uzol a pripíš vygenerované uzly do stromu hľadania

end

Stavový priestor a graf (strom) hľadania

29

■ Reprezentácia uzla:



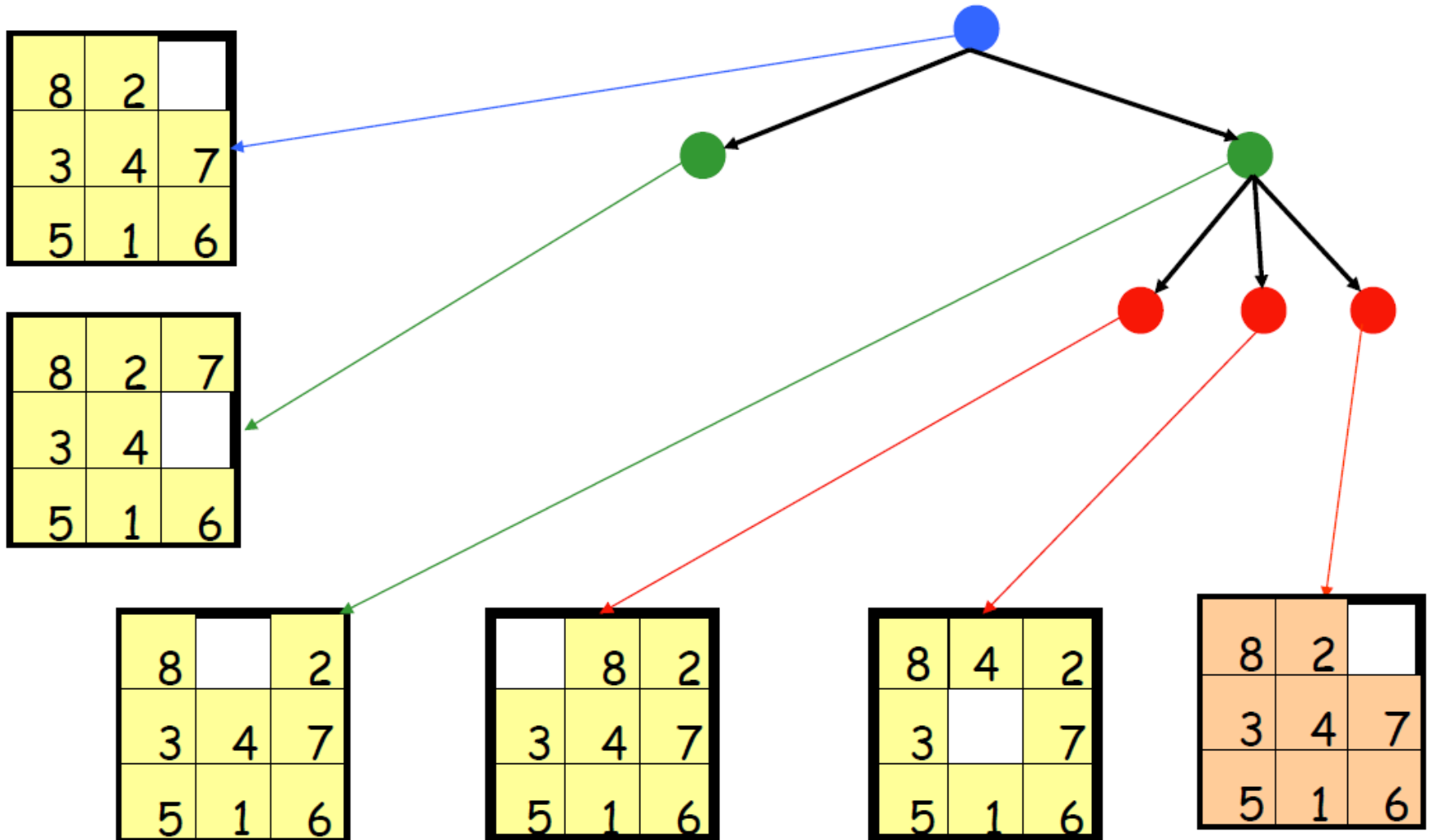
- zodpovedajúci stav zo stavového priestoru,
- uzol v strome hľadania, z ktorého sa daný uzol vygeneroval,
- operátor, ktorý sa aplikoval pri generovaní uzla (na rodičovský uzol),
- počet uzlov na ceste z koreňa do daného uzla (hĺbka uzla),
- cena cesty zo začiatočného uzla do daného uzla.

datatype UZOL

components: STAV, RODIČOVSKÝ-UZOL, OPERÁTOR,
HĽBKA, CENA-CESTY

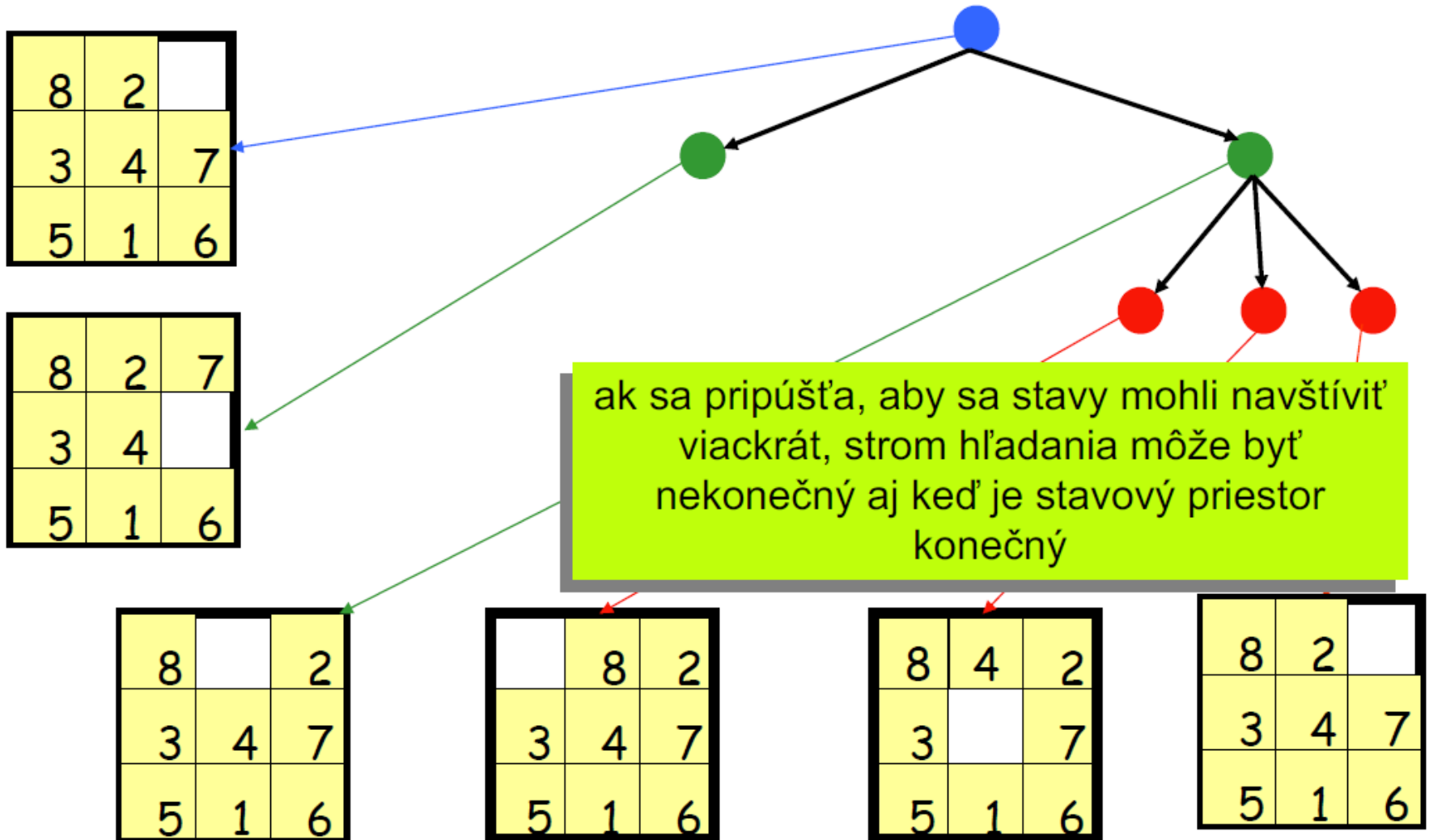
Uzly v strome hl'adania a stavy

30



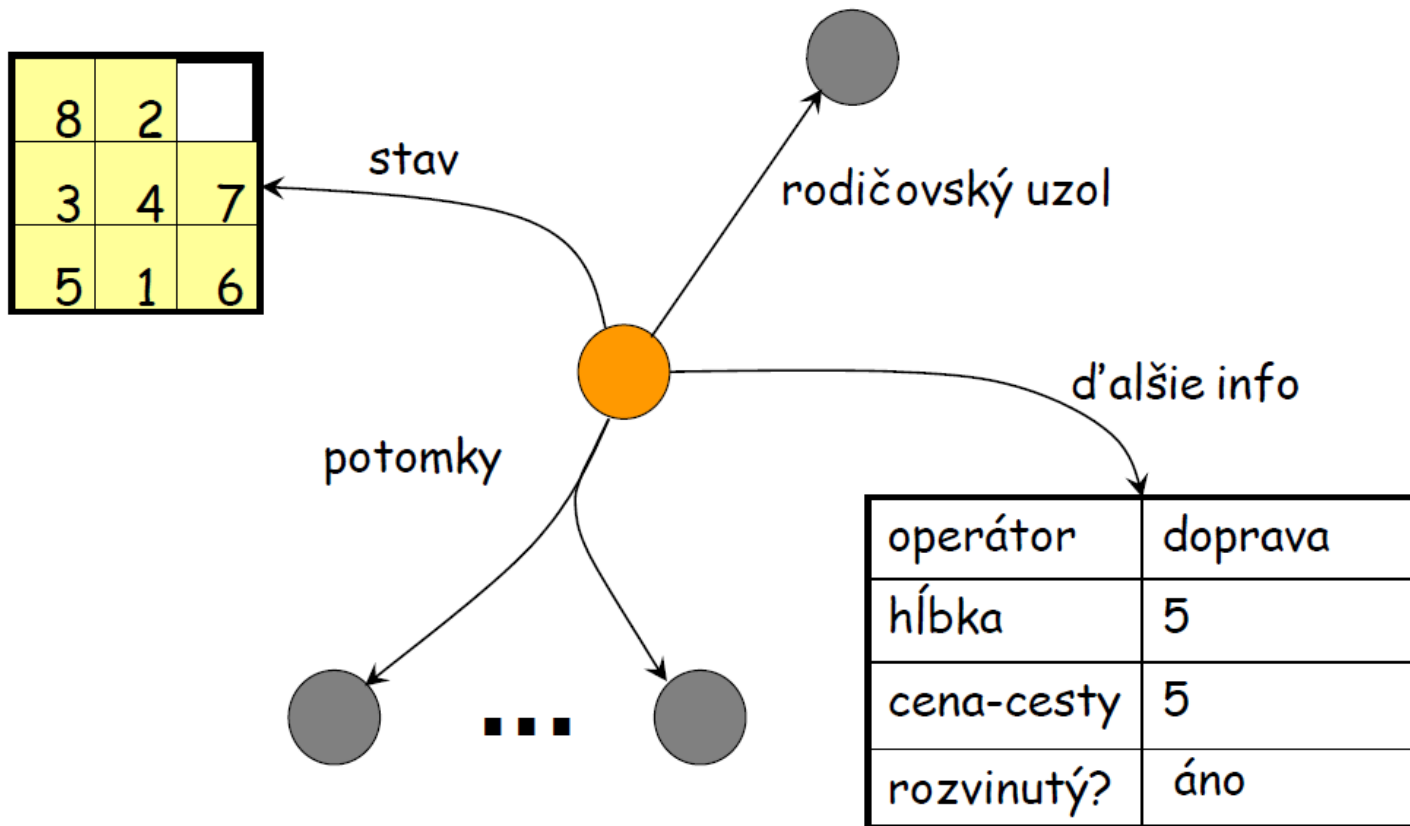
Uzly v strome hľadania a stavy

31



Dátová štruktúra pre uzol

32



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

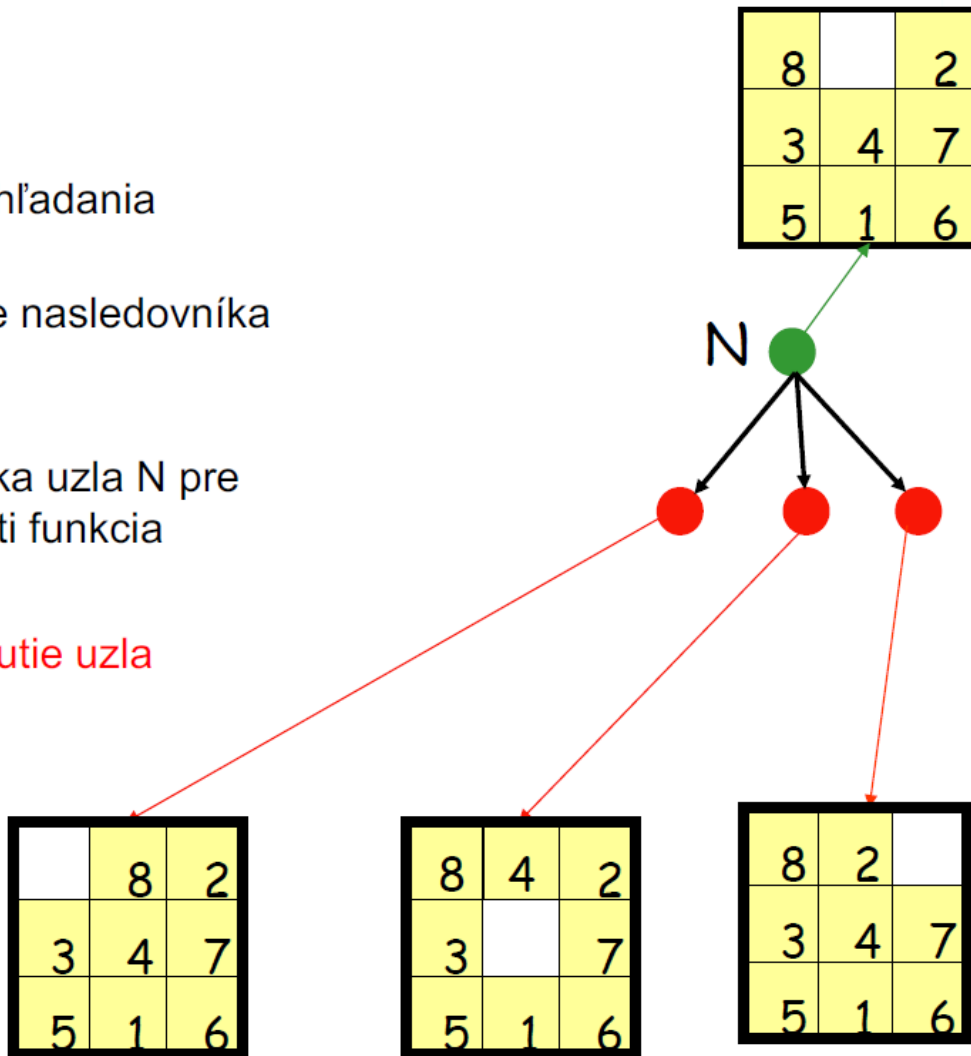
Rozvinutie uzla

33

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

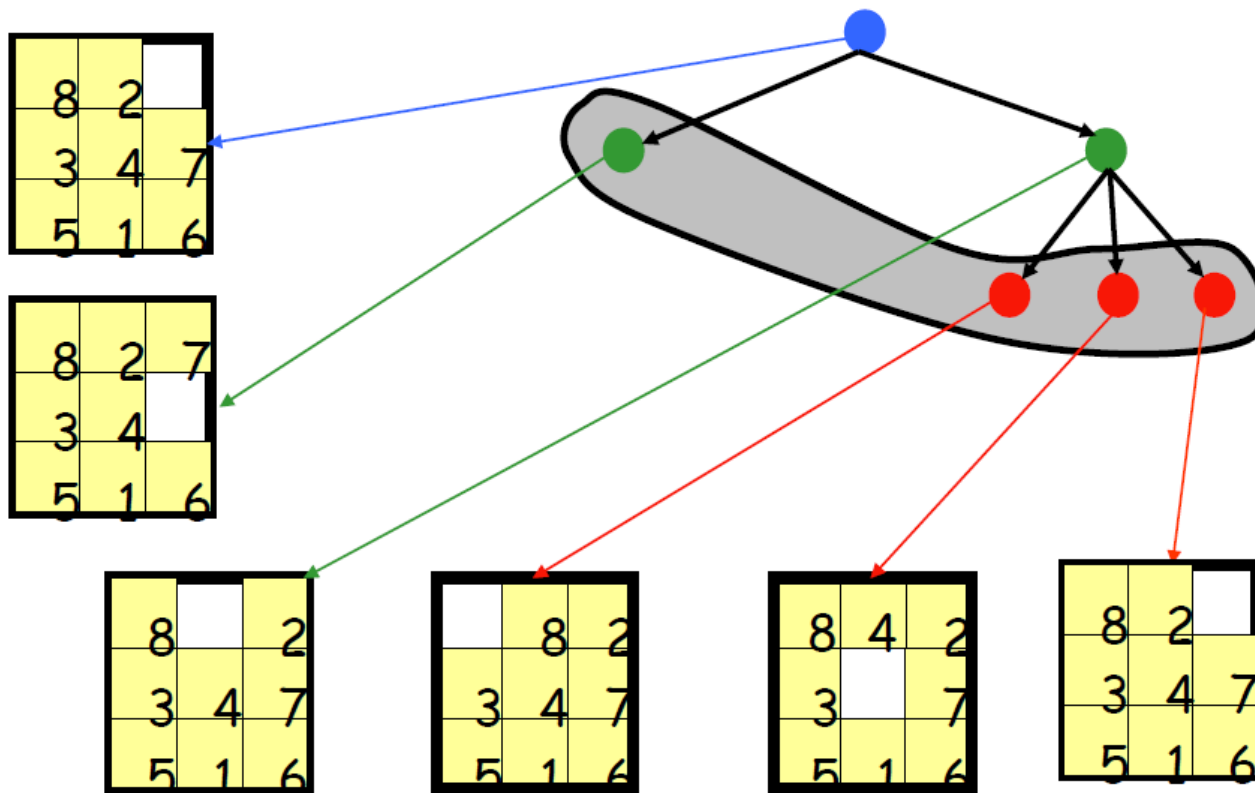
generovanie uzla \neq rozvinutie uzla



Okraj stromu hľadania

34

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



Je okraj
totožný s
množinou
listov?

Front – štruktúra na zápis množiny uzlov

35

- **Angl. Queue**
- **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 (prvok)
 - ▣ **ZARAĎ-DO-FRONTU(prvky, front)** vráti front po zaradení prvkov do pôvodného frontu. Rôzne druhy tejto funkcie určujú rôzne algoritmy hľadania

Všeobecný algoritmus hľadania

36

```
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, OPERÁTOR[problém]), front)
  end
```

Stratégie hľadania

37

□ **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

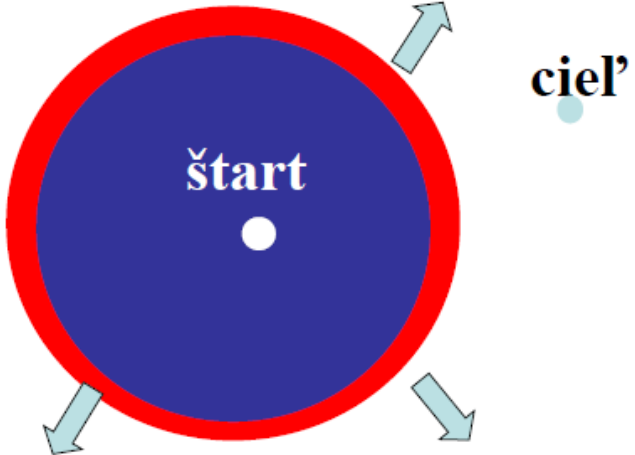
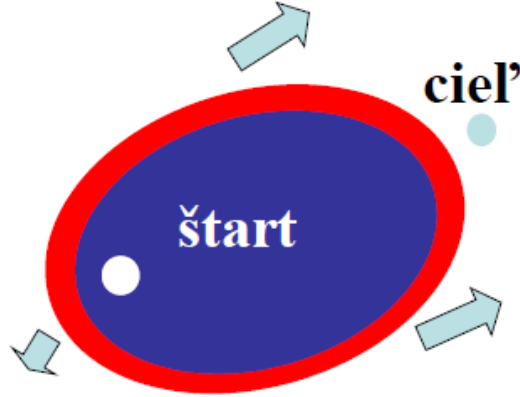
Heuristika

38

- ηύρηκα [heuréka] = našiel (objavil) som to – Archimedes
- ηύρίσκω = 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

39

Neinformované hľadanie	Informované hľadanie
 <p>The diagram illustrates uninformed search. A large blue circle represents the search space, with a red border. A white dot in the center is labeled 'štart'. Three light blue arrows point outwards from the red border, indicating a uniform expansion from the start point. A small blue dot to the right is labeled 'cieľ'.</p>	 <p>The diagram illustrates informed search. An oval blue shape represents the search space, with a red border. A white dot inside is labeled 'štart'. Three light blue arrows point outwards from the red border, indicating a non-uniform expansion based on some heuristic. A small blue dot to the right is labeled 'cieľ'.</p>

Stratégie hľadania

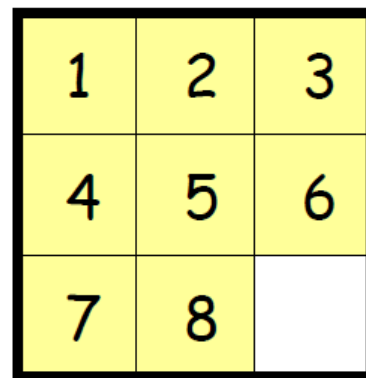
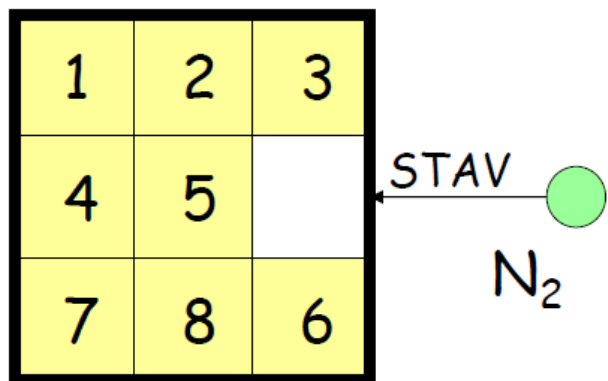
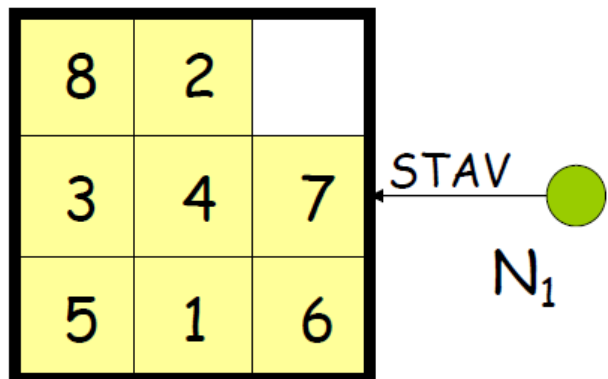
40

- ❑ **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 (Optimálnosť)** – nájde sa pomocou danej stratégie najlepšie riešenie, ak existuje aspoň jedno riešenie?

Príklad

41

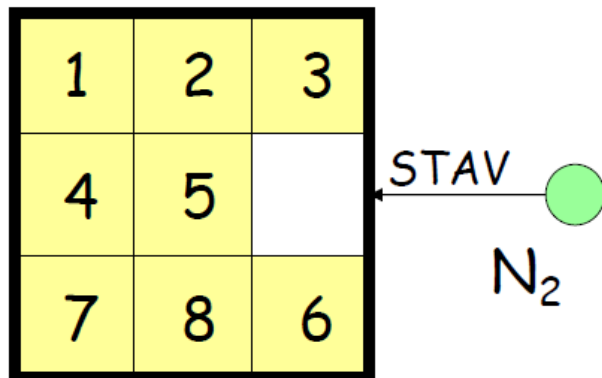
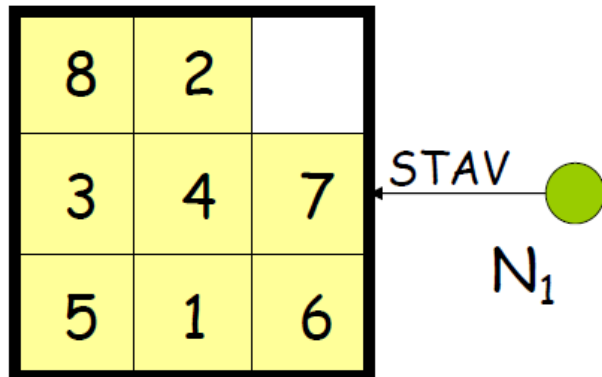
pre slepú stratégiu, N_1 a N_2 sú len dva uzly (s nejakou polohou v strome hľadania)



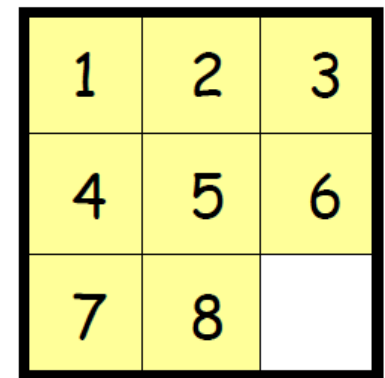
cieľový stav

Príklad

42



pre heuristickú stratégiu, počítajúcu počet kameňov, ktoré nie sú na svojom mieste, N_2 je sľubnejší uzol než N_1



cieľový stav

Poznámka

43

- problémy, ktoré uvažujeme, ako napr. (n^2-1) -hlavolam, sú NP-ťažké
 - neočakávajme, že budeme vedieť vyriešiť ľubovoľnú (t.j. každú) inštanciu takého problému v čase lepšom než exponenciálnom
 - môžeme sa usilovať vyriešiť každú inštanciu čo najefektívnejšie
- to je účelom stratégie hľadania

Slepé stratégie

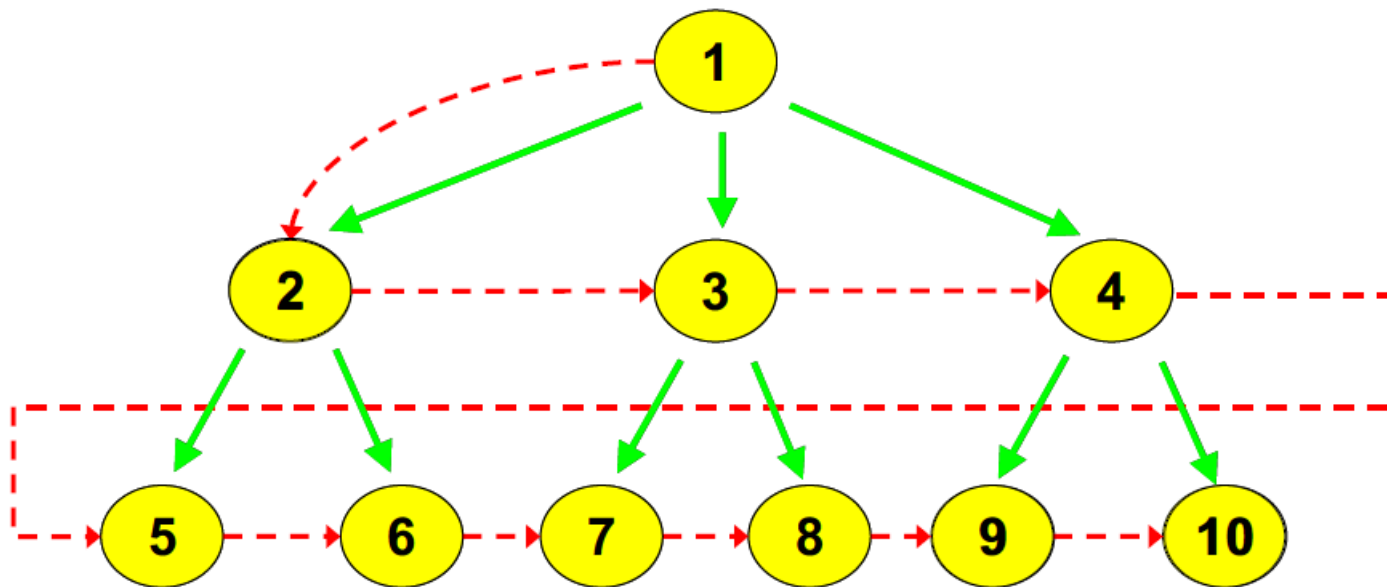
44

- do šírky (Breadth-First Search)
 - obojsmerne
 - do hĺbky (Depth-First Search)
 - obmedzené
 - iteratívne sa prehľbujúce
 - do hĺbky s návratom
 - rovnomerná cena (varianta do šírky)
- cena hrany = 1
- cena hrany
= $c(\text{operátor}) \geq \varepsilon > 0$

Hľadanie do šírky

45

- úplné, prípustné, exponenciálna zložitosť

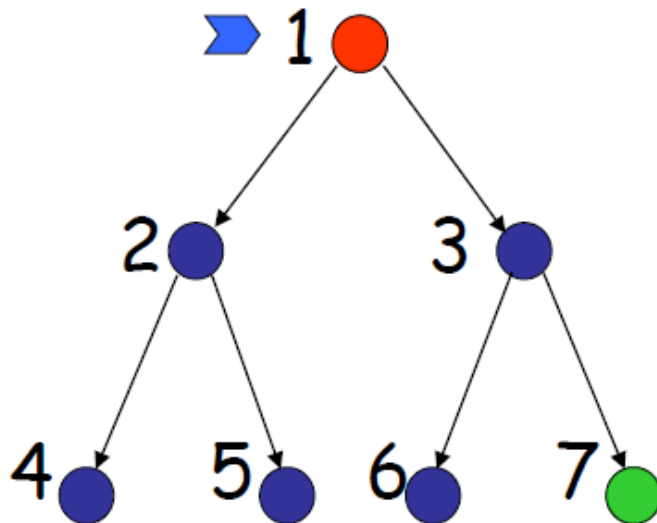


```
function HLADANIE-DO-ŠÍRKY(problém) returns riešenie alebo neúspech  
return VŠEOBECNÉ-HLADANIE(problém, ZARAĎ-NA-KONIEC)
```

Hľadanie do šírky

46

Nové uzly sa pridávajú na koniec OKRAJa

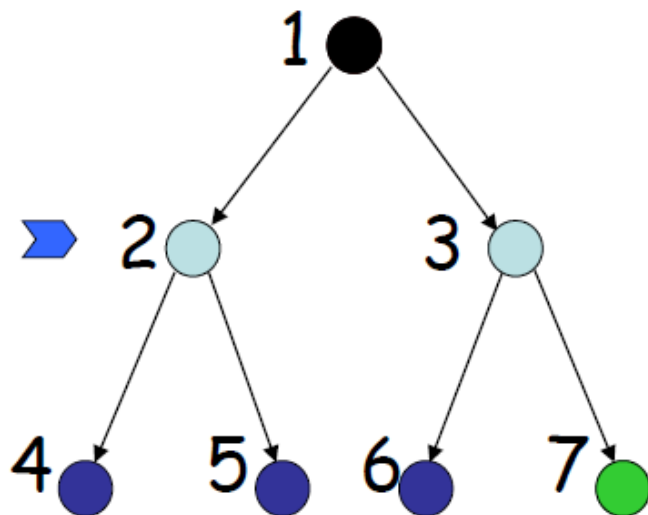


OKRAJ = (1)

Hľadanie do šírky

47

Nové uzly sa pridávajú na koniec OKRAJA

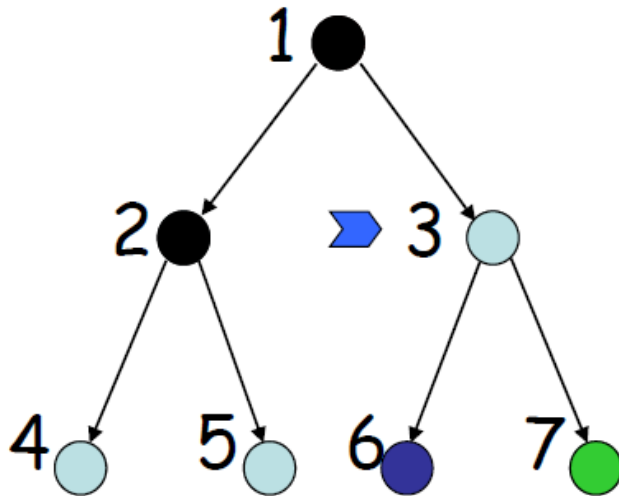


OKRAJ = (2, 3)

Hľadanie do šírky

48

Nové uzly sa pridávajú na koniec OKRAJA

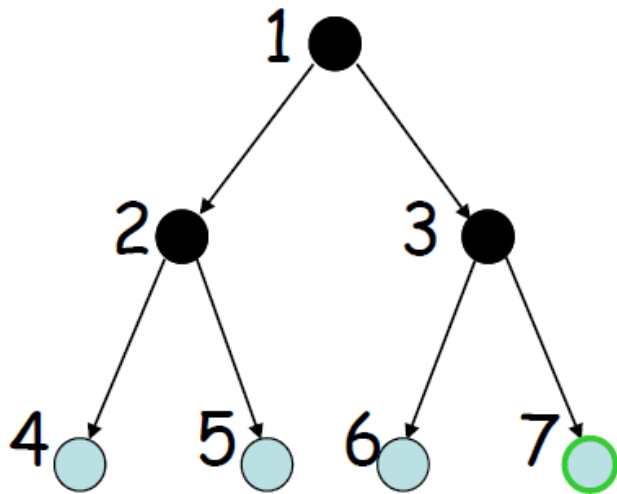


OKRAJ = (3, 4, 5)

Hľadanie do šírky

49

Nové uzly sa pridávajú na koniec OKRAJa



OKRAJ = (4, 5, 6, 7)

Dôležité parametre

50

- 1) Maximálny počet nasledovníkov ktoréhokoľvek stavu
→ faktor vetvenia **b** prehľadávaného stromu
- 2) Minimálna dĺžka (\neq cena) cesty medzi počiatočným a cieľovým stavom
→ hĺbka **d** najplytšieho cieľového uzla v strome

Vyhodnotenie

51

- **b**: Vetviaci faktor
- **d**: hĺbka najplytšieho cieľového uzla
- Hľadanie do šírky je:
 - úplné
 - optimálne, ak je krok 1
- Počet vygenerovaných uzlov
???

Vyhodnotenie

52

- **b**: Vetviaci faktor
- **d**: hĺbka najplytšieho cieľového uzla
- Hľadanie do šírky je:
 - úplné
 - optimálne ak je krok 1
- Počet vygenerovaných uzlov
 $1 + b + b^2 + \dots + b^d = ???$

Vyhodnotenie

53

- **b**: Vetviaci faktor
- **d**: hĺbka najplytšieho cieľového uzla
- Hľadanie do šírky je:
 - úplné
 - optimálne ak je krok 1
- Počet vygenerovaných uzlov
$$1 + b + b^2 + \dots + b^d = (b^{d+1} - 1) / (b - 1) = O(b^d)$$
- → Časová a priestorová zložitosť je $O(b^d)$

Časové a pamäťové nároky hľadania do šírky

54

Hĺbka	Počet uzlov	Čas	Pamäť
0	1	0.01 milisekundy	100 slabík
1	35	0.3 milisekundy	3.4 kiloslábík
2	1225	0.01 sekundy	119 kiloslábík
3	42 875	0.4 sekundy	4 megaslabiky
4	1.5×10^6	15 sekúnd	143 megaslabík
5	52×10^6	8.7 minúty	4.8 gigaslabík
6	1.8×10^9	5 hodín	171 gigaslabík
7	64×10^9	7 dní	5.8 teraslabík
8	2.2×10^{12}	261 dní	204 teraslabík
9	78×10^{12}	25 rokov	7 168 teraslabík
10	2.7×10^{15}	874 rokov	250 888 teraslabík
12	3.3×10^{18}	10^6 rokov	8.7×10^6 teraslabík
...
20	7.6×10^{30}	2.4×10^{18} rokov	6.9×10^{20} teraslabík

Predpoklady: faktor vetvenia 35, 100000 uzlov / sekunda, 100 slabík / uzol

Poznámka

55

Ak problém nemá riešenie, hľadanie do šírky sa môže vykonávať *donekonečna* (ak stavový priestor je nekonečný alebo stavy môžu byť znovu navštívené ľubovoľný počet ráz)

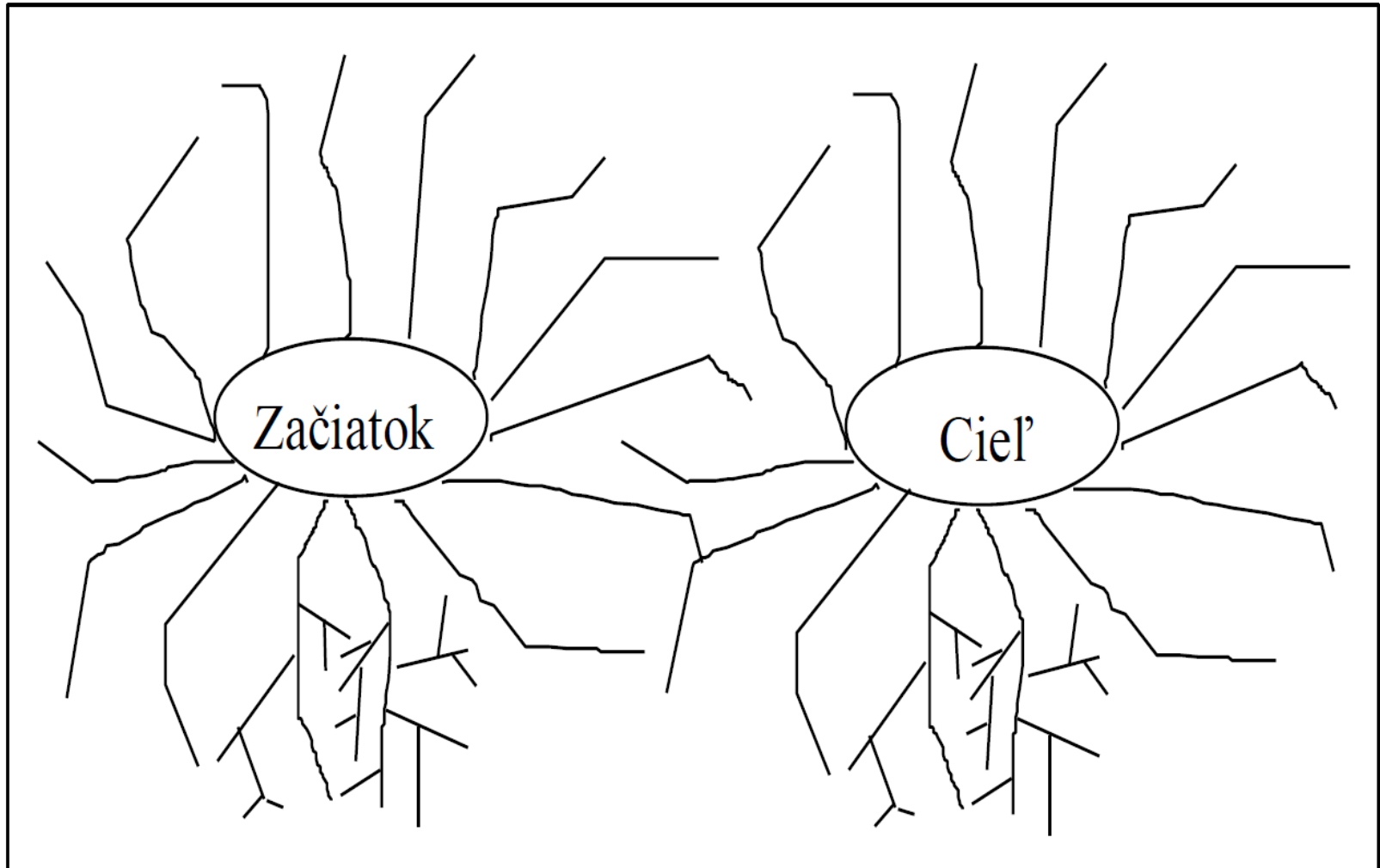
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	



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

Obojsmerné hľadanie

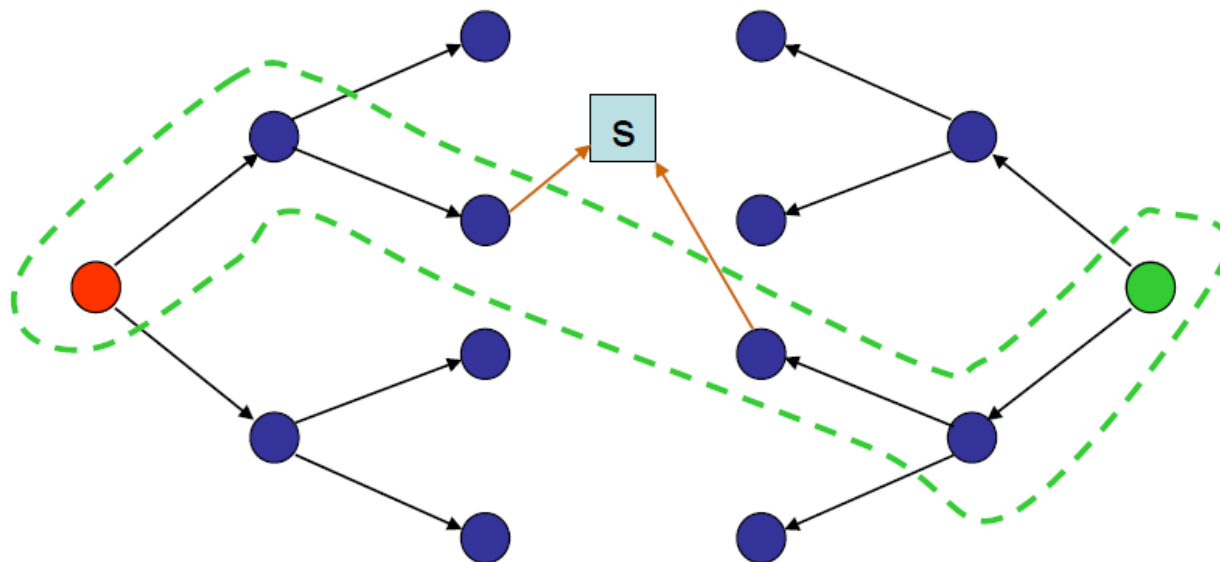
56



Obojsmerná stratégia

57

fronty dvoch okrajov: OKRAJ1 a OKRAJ2



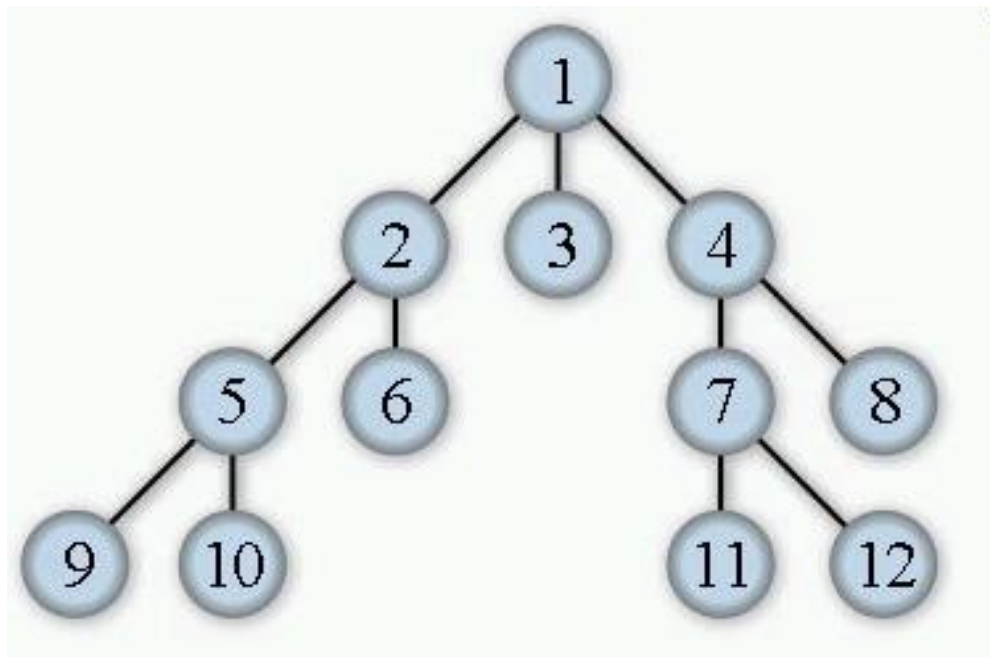
Časová a priestorová zložitosť je $O(b^{d/2}) \ll O(b^d)$
ak oba stromy majú rovnaký vetviaci faktor b

Otázka: Čo sa stane ak vetviaci faktor je rôzny
od každého smeru?

Hľadanie do hĺbky

58

□ Depth-first search

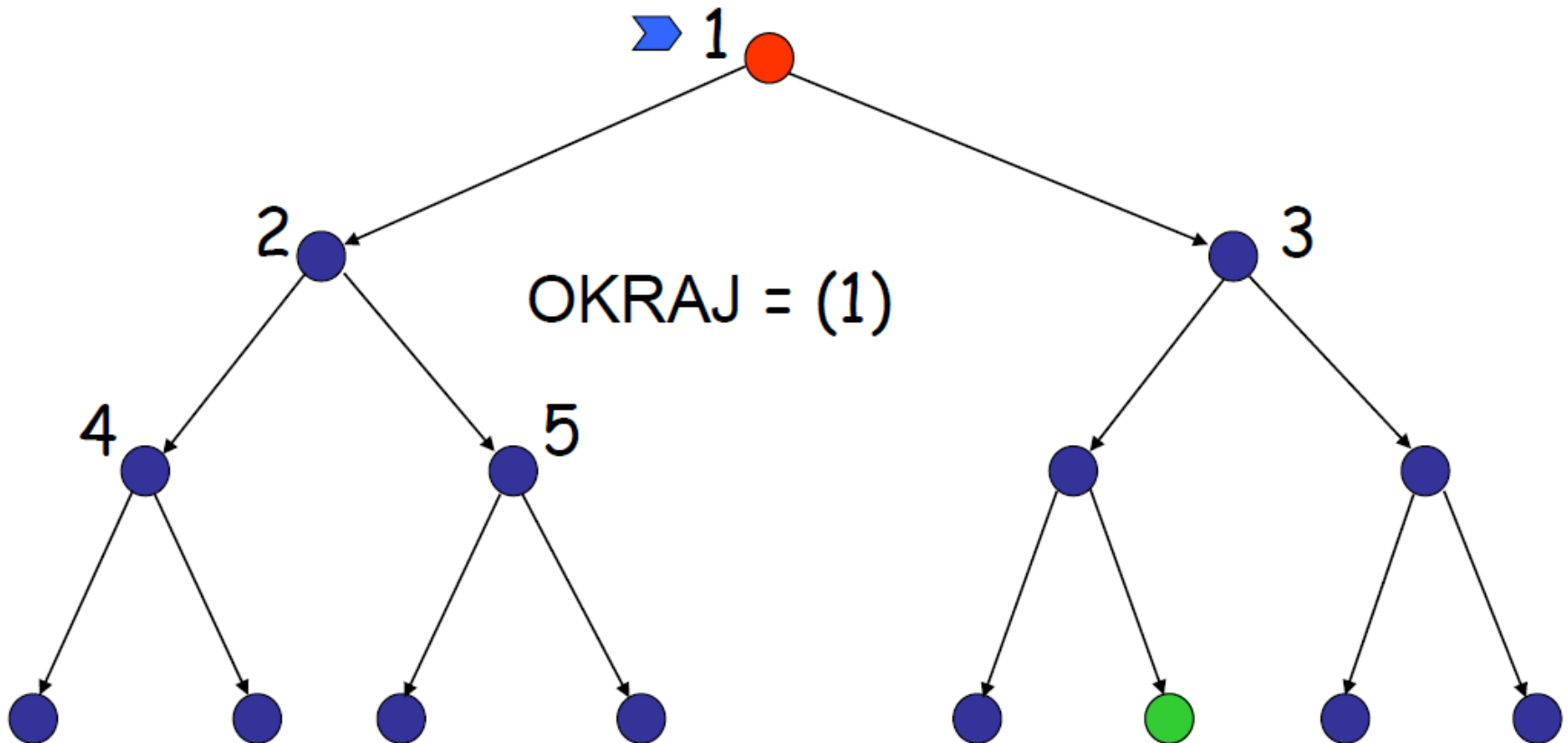


```
function HLADANIE-DO-HLBKY(problém) returns riešenie alebo neúspech  
return VŠEOBECNÉ-HLADANIE(problém, ZARAĎ-NA-ZAČIATOK)
```

Hľadanie do hĺbky

59

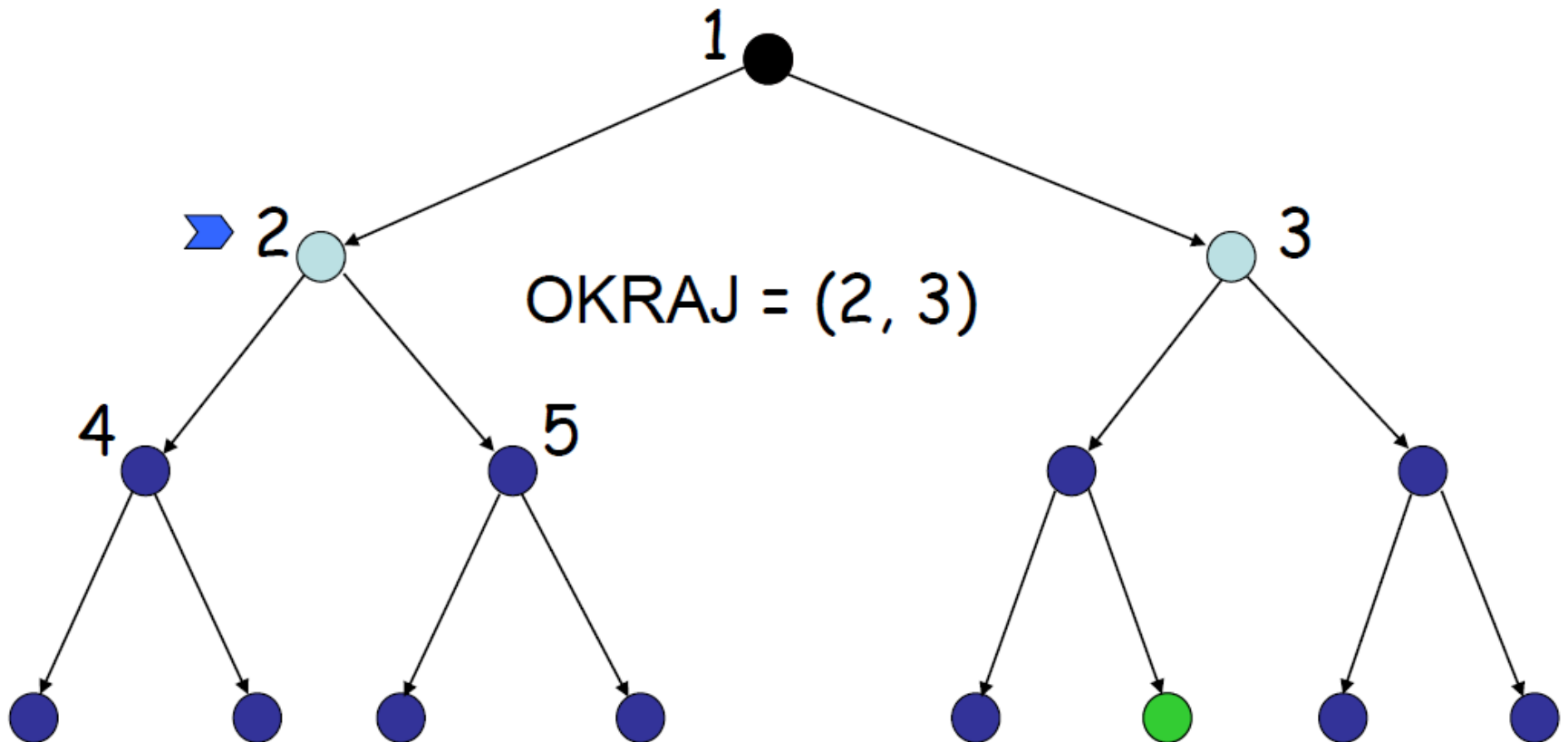
Nové uzly sa vkladajú na začiatok OKRAJA



Hľadanie do hĺbky

60

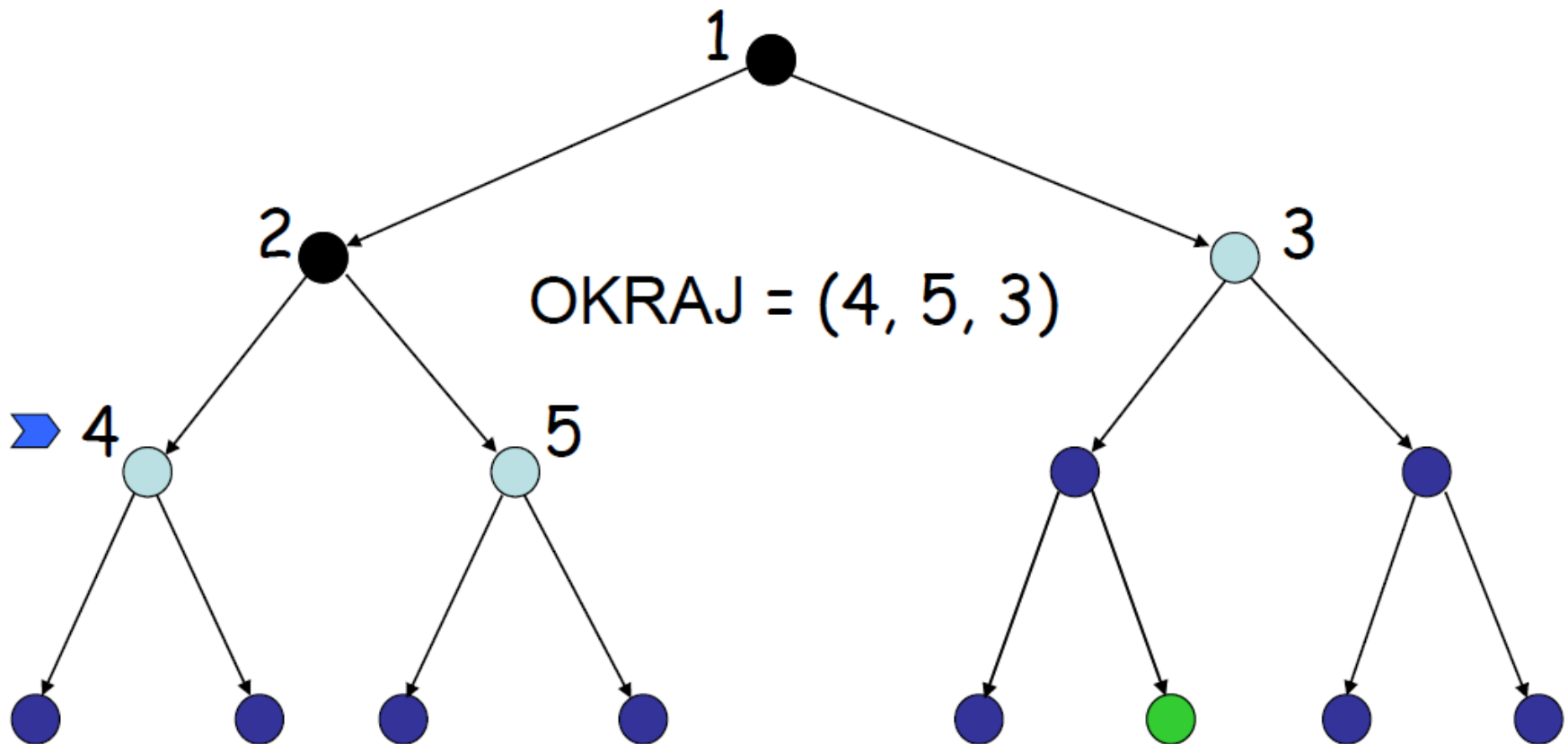
Nové uzly sa vkladajú na začiatok OKRAJA



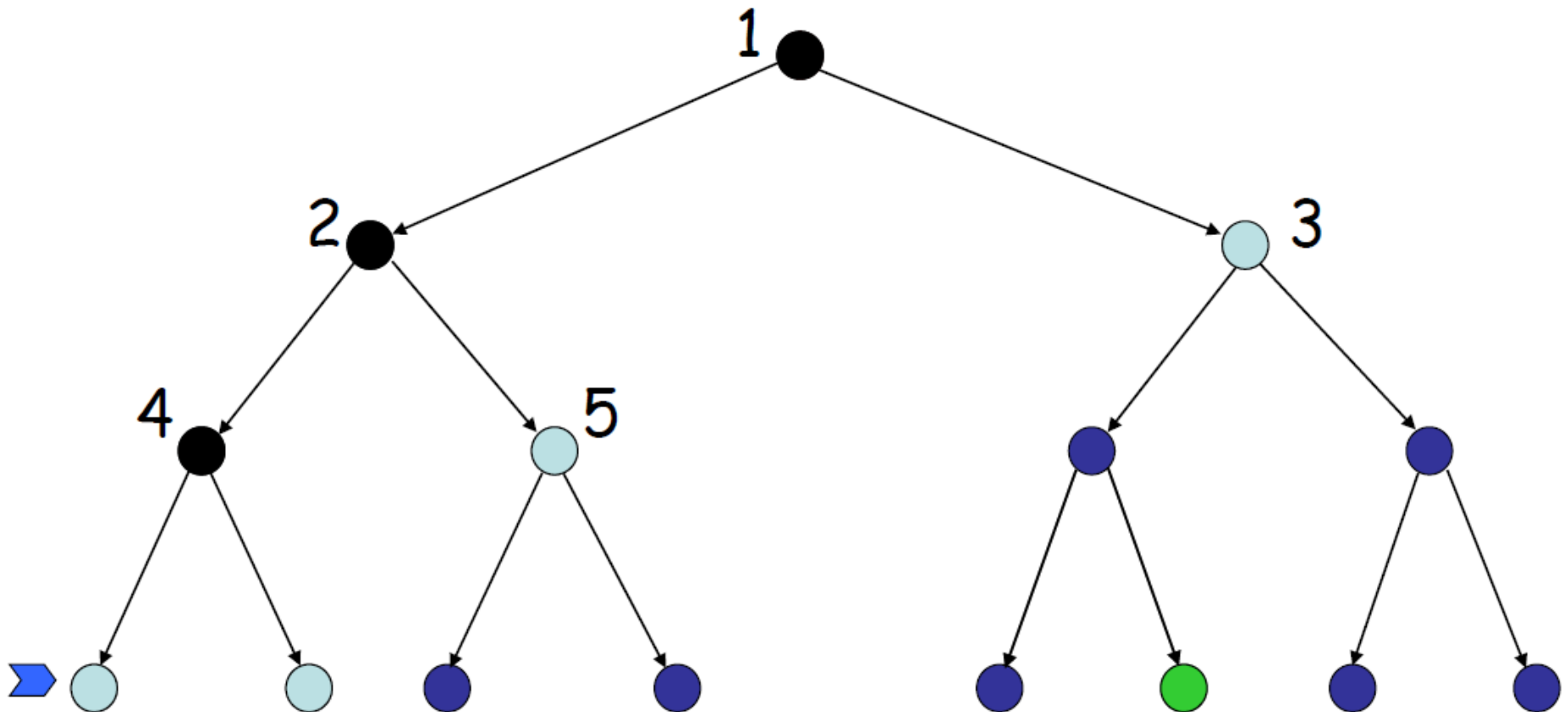
Hľadanie do hĺbky

61

Nové uzly sa vkladajú na začiatok OKRAJA



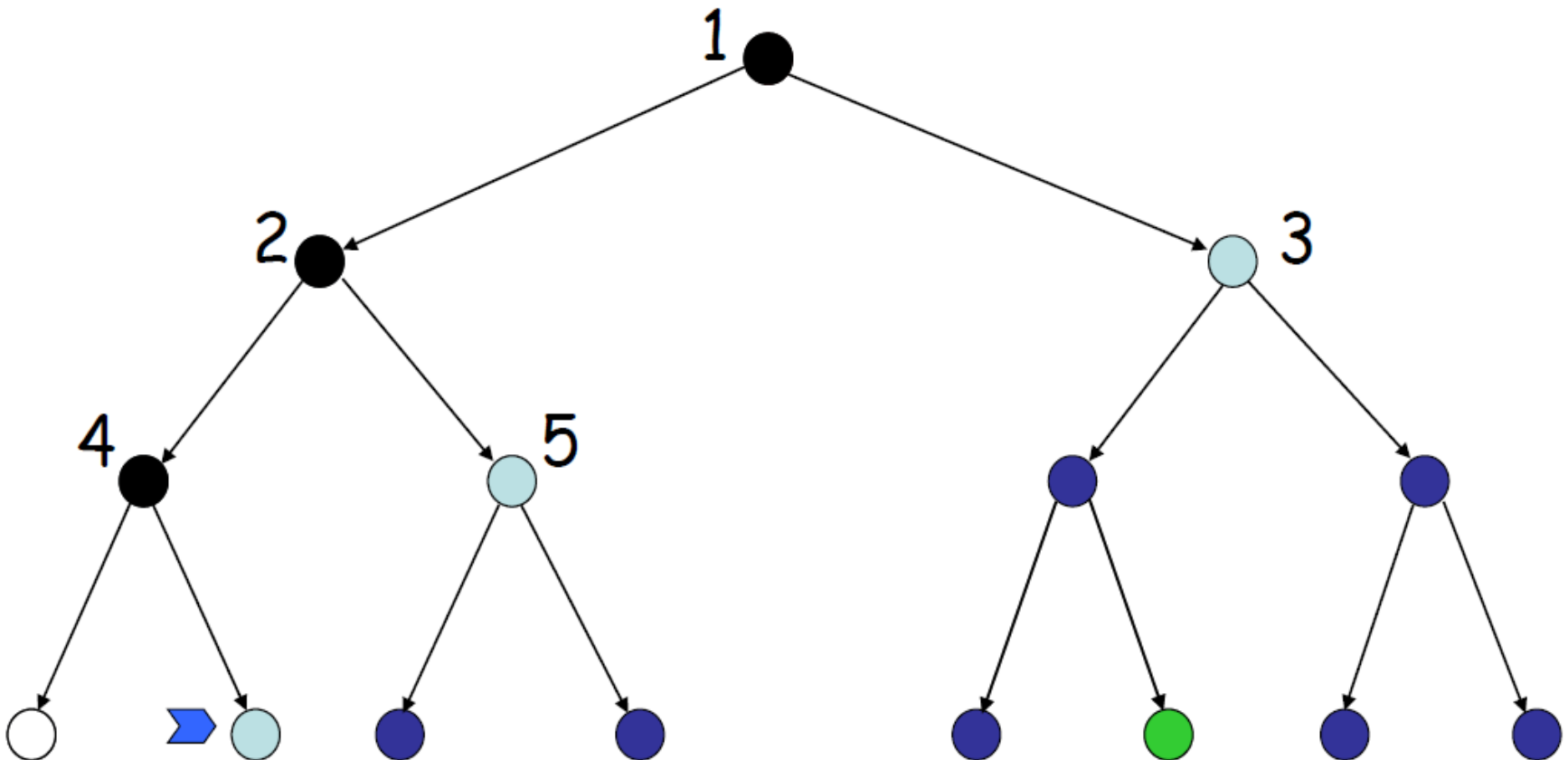
62



Hľadanie do hĺbky

63

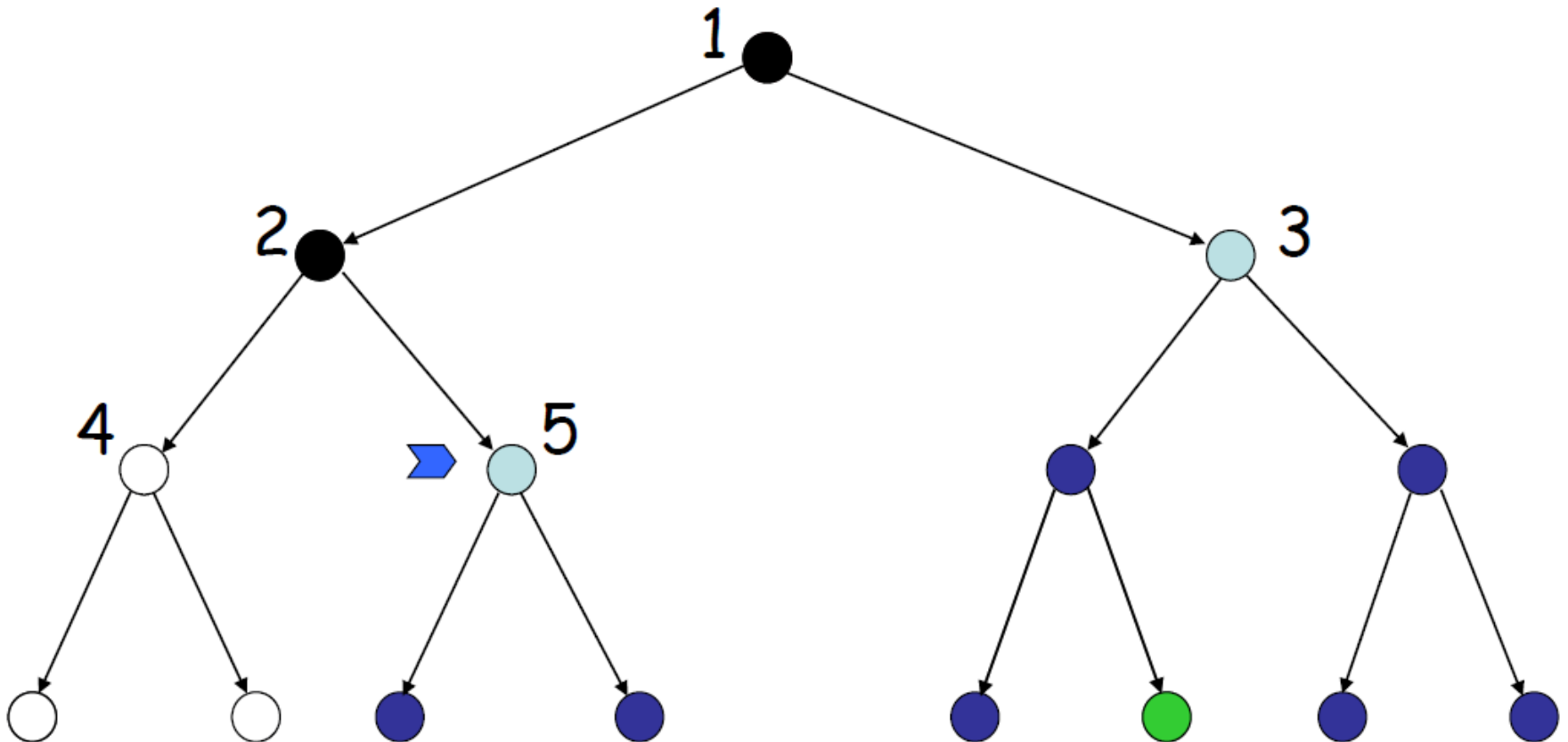
Nové uzly sa vkladajú na začiatok OKRAJA



Hľadanie do hĺbky

64

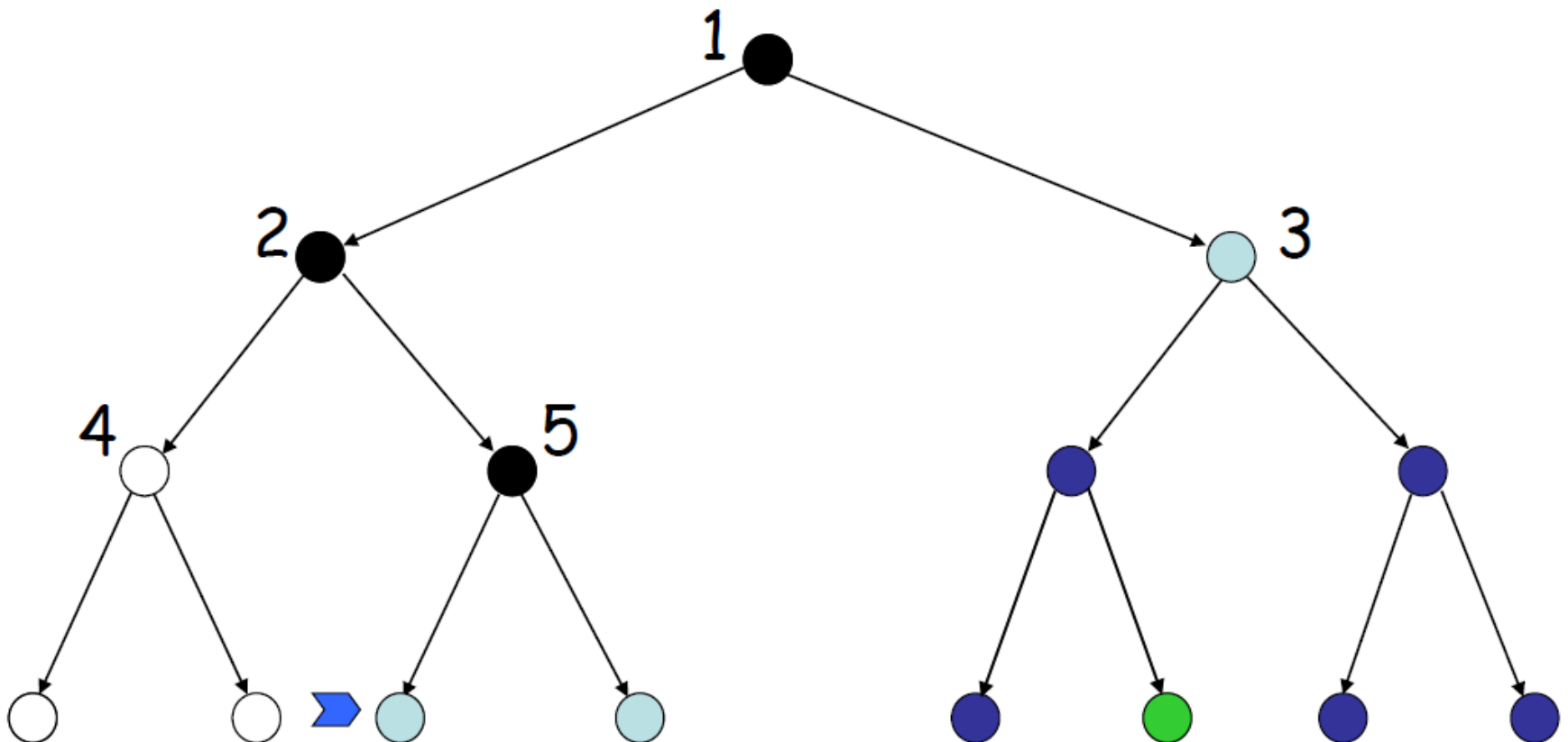
Nové uzly sa vkladajú na začiatok OKRAJA



Hľadanie do hĺbky

65

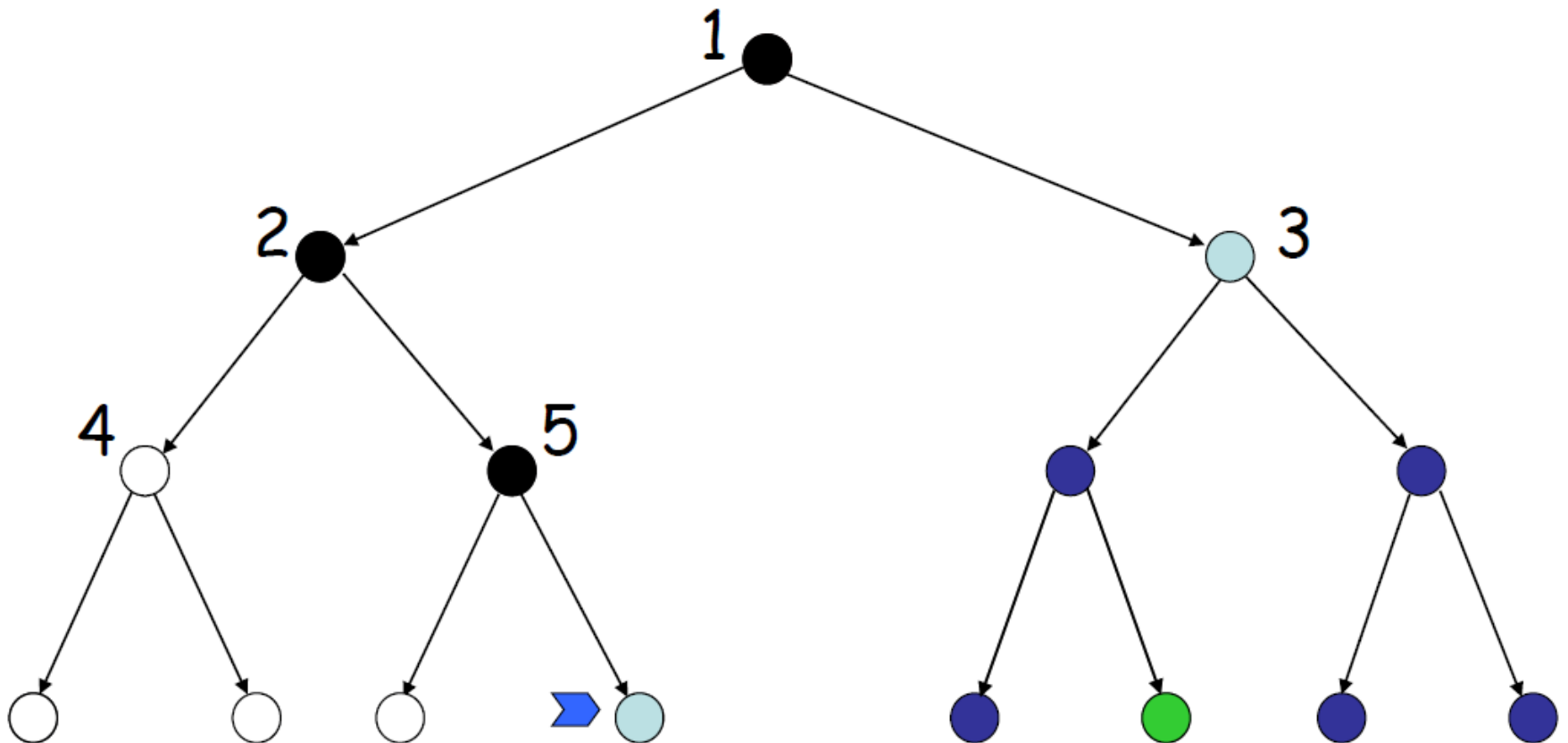
Nové uzly sa vkladajú na začiatok OKRAJA



Hľadanie do hĺbky

66

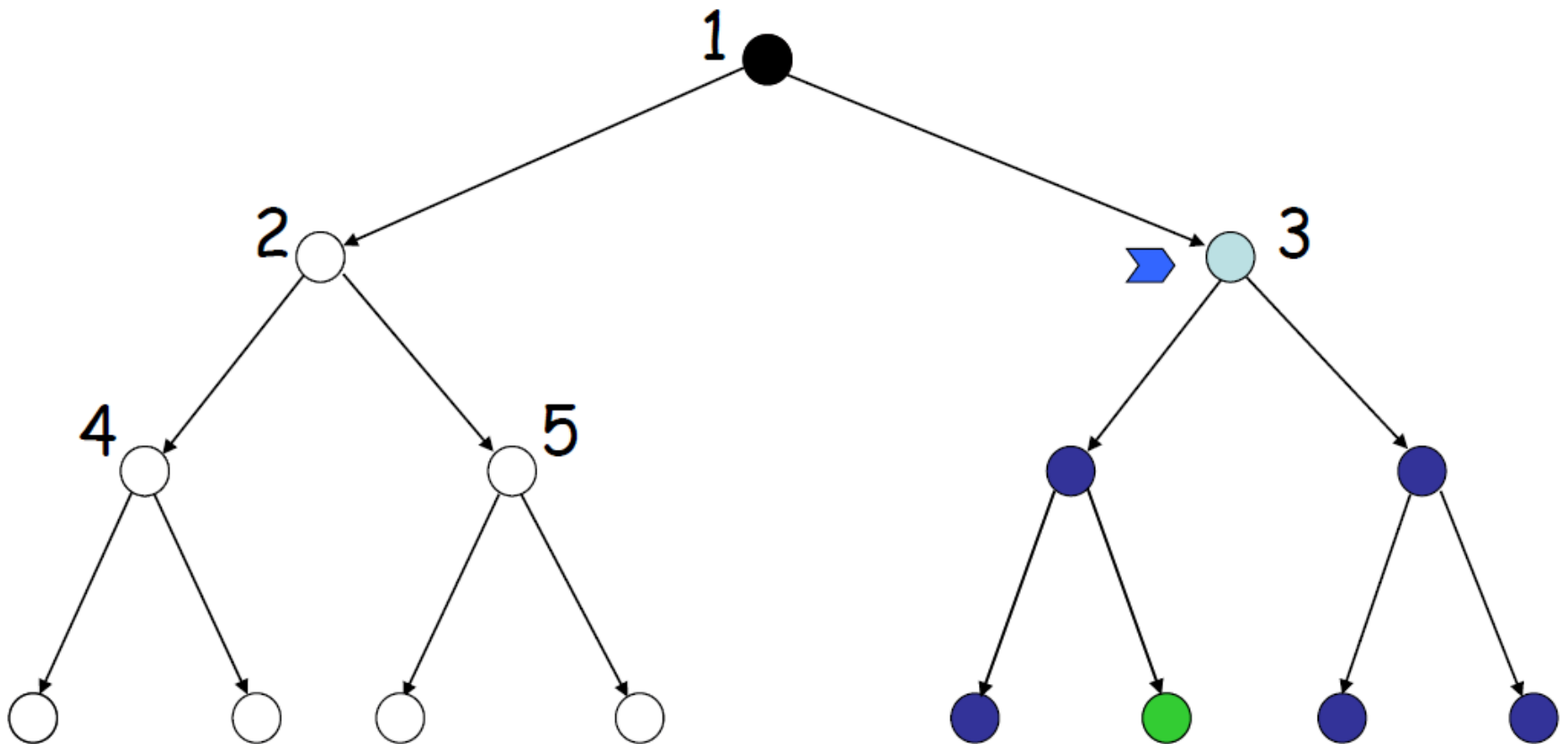
Nové uzly sa vkladajú na začiatok OKRAJA



Hľadanie do hĺbky

67

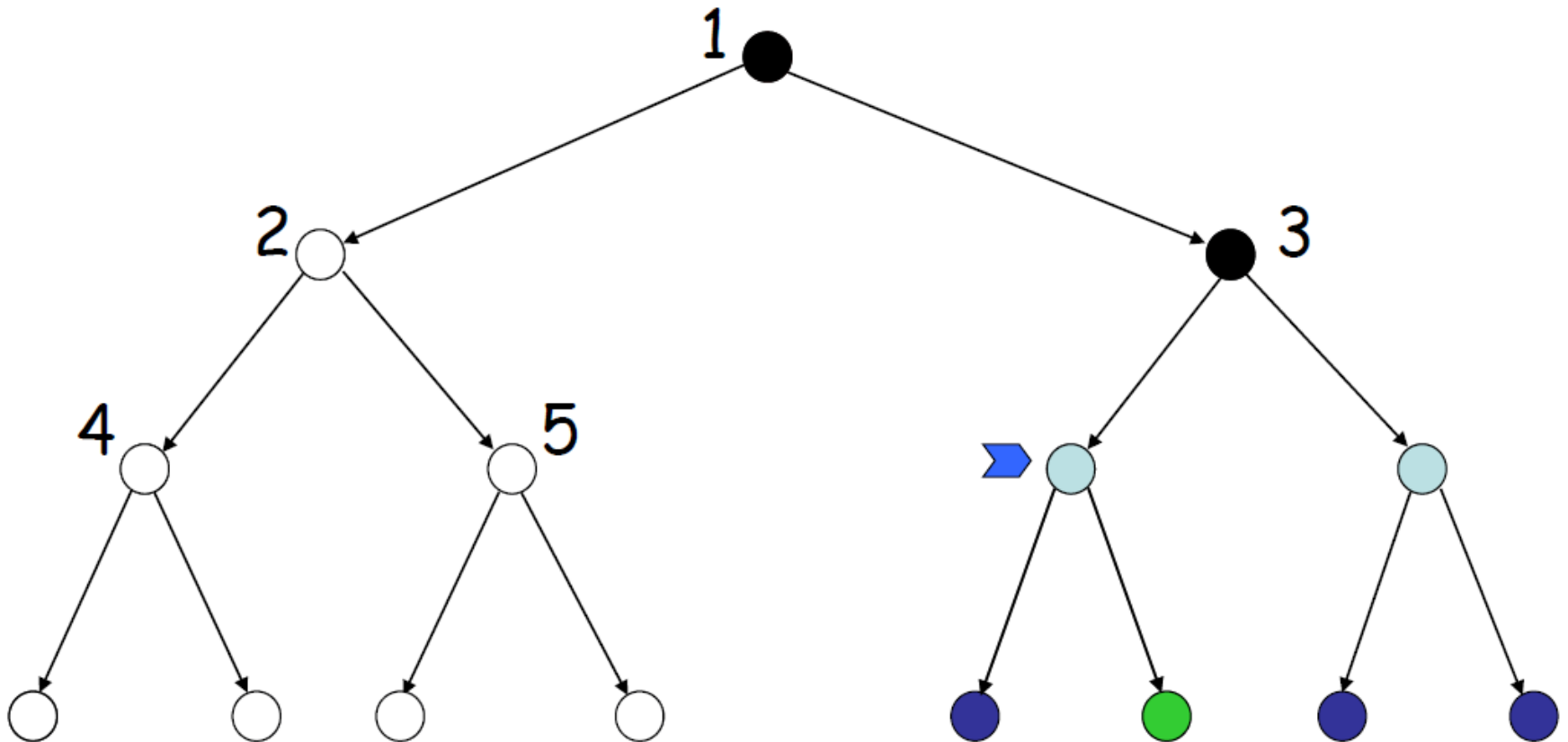
Nové uzly sa vkladajú na začiatok OKRAJa



Hľadanie do hĺbky

68

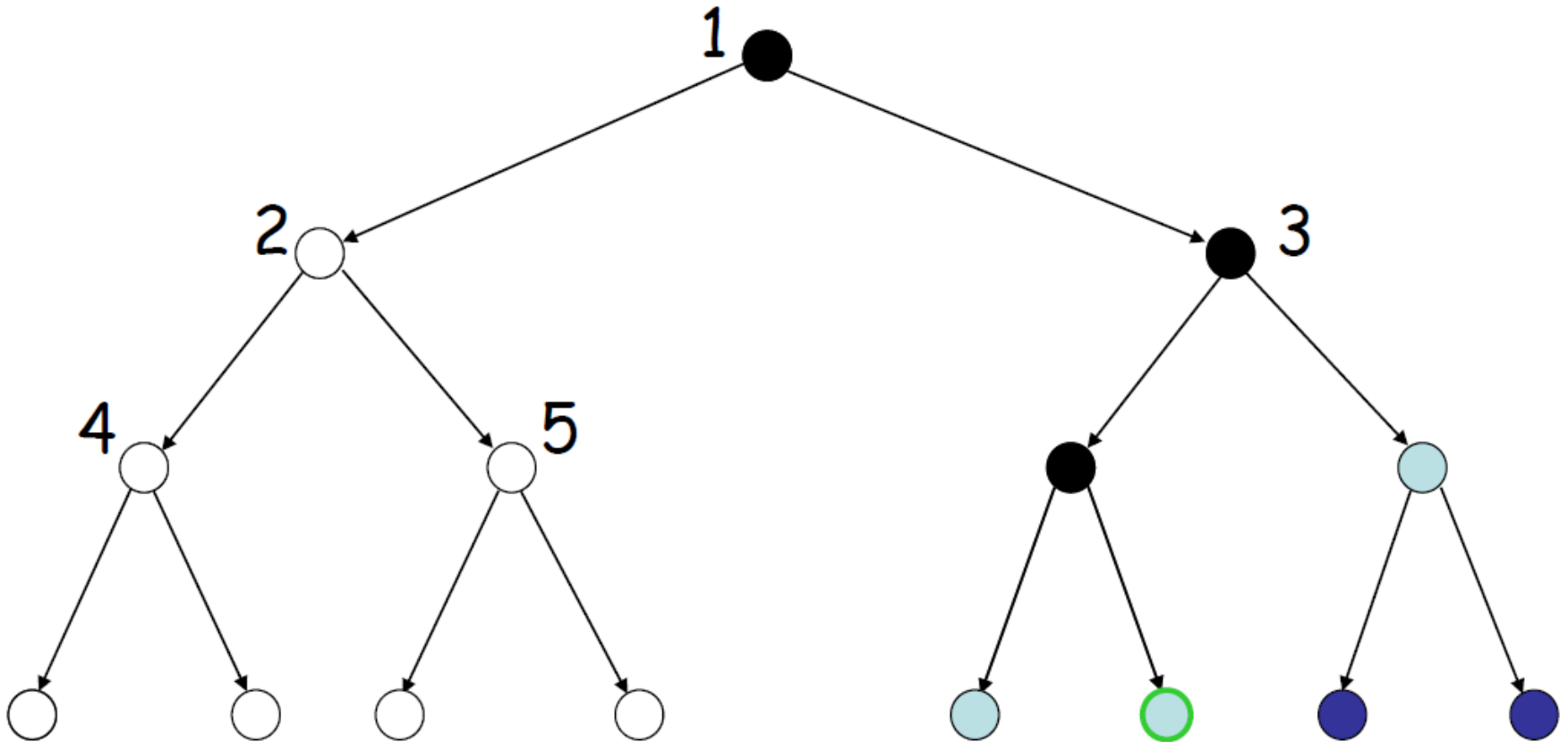
Nové uzly sa vkladajú na začiatok OKRAJA



Hľadanie do hĺbky

69

Nové uzly sa vkladajú na začiatok OKRAJa



Vyhodnotenie

70

- **b**: vetviaci faktor
- **d**: hĺbka najplytšieho cieľového uzla
- **m**: maximálna hĺbka listového uzla
- Hľadanie do hĺbky je:
 - úplné?
 - optimálne?

Vyhodnotenie

71

- **b**: vetviaci faktor
 - **d**: hĺbka najplytšieho cieľového uzla
 - **m**: maximálna hĺbka listového uzla
 - Hľadanie do hĺbky je:
 - úplné iba pre konečný strom hľadania
 - nie je optimálne
 - Počet vygenerovaných uzlov (najhorší prípad) :
 $1 + b + b^2 + \dots + b^m = O(b^m)$
 - Časová zložitosť: $O(b^m)$
 - Priestorová zložitosť: $O(bm)$ [alebo $O(m)$]
- [pripomienka: Vyhľadávanie do šírky vyžaduje $O(b^d)$ čas a pamäť]

Cyklicky sa prehlbujúce hľadanie

72

```
function CYKLICKY-SA-PREHLBUJÚCE-HĽADANIE(problém)  
returns riešenie alebo neúspech  
  
  for hlbka  $\leftarrow$  0 to  $\infty$  do  
    if OBMEDZENÉ-HĽADANIE(problém, hlbka) je úspešné  
      then return jeho riešenie  
  
  end  
  return neúspech
```


Obmedzené prehľadávanie do hĺbky

73

- Hľadanie do hĺbky s odseknutím v hĺbke k
 - ▣ hĺbka, za ktorou sa uzly nerozvíjajú
- Tri možné prípady
 - ▣ riešenie
 - ▣ zlyhanie – žiadne riešenie
 - ▣ odseknutie hĺbky

Cyklicky sa prehľbujúce hľadanie

74

Poskytuje to najlepšie z hľadania do šírky a do hĺbky

Hlavná idea:

IDS

Pre $k = 0, 1, 2, \dots$ do:

 Vykonaj hľadanie do hĺbky s odseknutím v hĺbke k

 (napr., generuj iba uzly s hĺbkou $\leq k$)

Vyhodnotenie

75

- Cyklicky sa prehľbujúce hľadanie je:
 - úplné
 - optimálne ak cena kroku = 1
- Časová zložitosť:
$$(d+1)(1) + db + (d-1)b^2 + \dots + (1)b^d = O(b^d)$$
- Priestorová zložitosť: $O(bd)$ alebo $O(d)$

Počet generovaných uzlov (hľadanie do šírky a cyklické prehlbovanie)

76

$$d = 5 \text{ a } b = 2$$

v hĺbke	do šírky	cykl. prehľb.
0	1	$1 \times 6 = 6$
1	2	$2 \times 5 = 10$
2	4	$4 \times 4 = 16$
3	8	$8 \times 3 = 24$
4	16	$16 \times 2 = 32$
5	32	$32 \times 1 = 32$
spolu	63	120

$$120/63 \sim 2$$

Počet generovaných uzlov (hľadanie do šírky a cyklické prehlbovanie)

77

$$d = 5 \text{ a } b = 10$$

v hĺbke	do šírky	cykl. prehľb.
0	1	6
1	10	50
2	100	400
3	1,000	3,000
4	10,000	20,000
5	100,000	100,000
spolu	111,111	123,456

$123,456 / 111,111 \sim 1.111$ t.j. len o 11% viac generovaných uzlov

Porovnanie stratégií

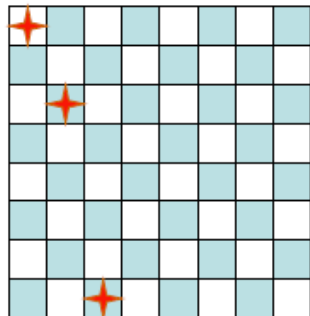
78

- Hľadanie do šírky
 - ▣ úplné a prípustné, ale má vysokú pamäťovú zložitosť
- Hľadanie do hĺbky
 - ▣ pamäťovo efektívne, ale nie je úplné ani prípustné
- Cyklické prehľbovanie
 - ▣ úplné, prípustné, s rovnakou pamäťovou zložitosťou ako prehľadávanie do hĺbky a má skoro rovnakú časovú zložitosť ako prehľadávanie do šírky

Znovunavšτίvené stavy

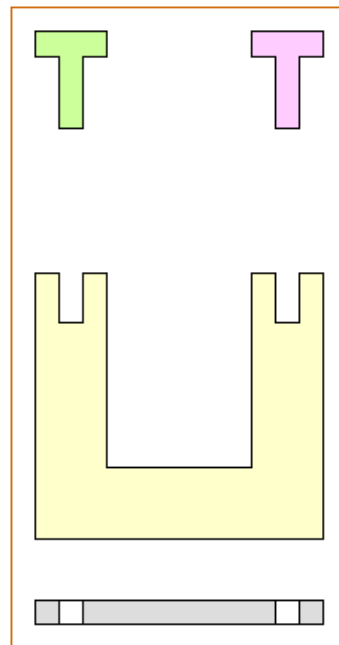
79

Žiadne



8 dām

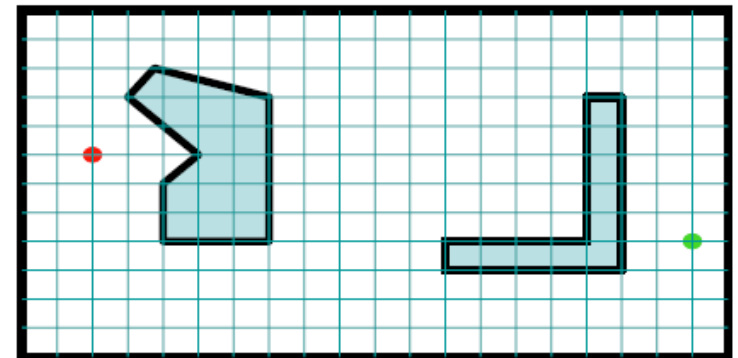
Málo



plánovanie
montáže

Veľa

1	2	3
4	5	
7	8	6



8-puzľa a navigácia robota

Znovunavštievané stavy

80

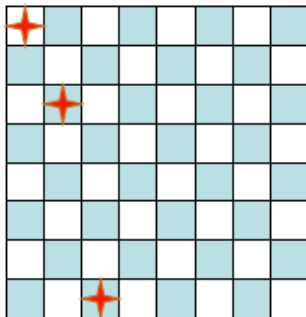
Žiadne

Málo

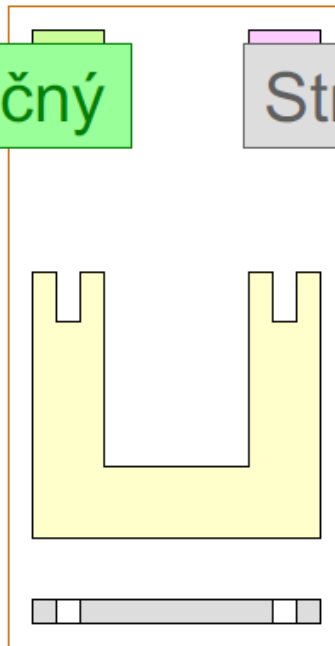
Veľa

Strom je konečný

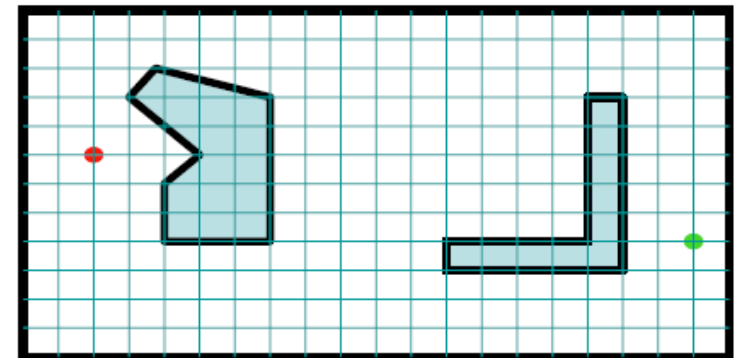
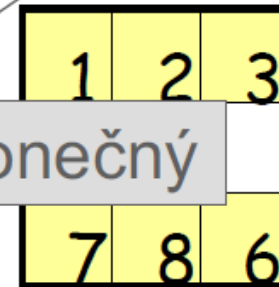
Strom je nekonečný



8 dám



plánovanie
montáže



8-puzľ a navigácia robota

Vyhýbanie sa znovunavštičeným stavom

81

- Vyžaduje porovnávanie opisov stavov
- Hľadanie do šírky:
 - ▣ Ulož všetky stavy združené s generovanými uzlami do NAVSTIVENE
 - ▣ Ak stav nového uzla je v NAVSTIVENE, tak zruš uzol

Vyhýbanie sa znovunavšτίveným stavom

82

□ Hľadanie do hĺbky:

▣ Riešenie 1:

- ukladaj všetky stavy asociované s uzlami v aktuálnej ceste do NAVSTIVENE
- ak stav nového uzlu je v NAVSTIVENE, tak zruš uzol
- tým sa iba vyhneme slučkám

▣ Riešenie 2:

- ukladaj všetky generované stavy do NAVSTIVENE
- ak stav nového uzlu je v NAVSTIVENE, tak zruš uzol
- rovnaká pamäťová zložitosť ako pri hľadaní do šírky!

Porovnanie neinformovaných stratégií hľadania

83

Kritérium	Do šírky	Rovnomernej ceny	Do hĺbky	Obmedzené do hĺbky	Cyklicky sa prehĺbujúce	Obojsmerné
Čas	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Pamäť	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Prípustná?	áno	áno	nie	nie	áno	áno
Úplná?	áno	áno	nie	áno, ak $l \geq d$	áno	áno

- b je faktor vetvenia,
- d je hĺbka riešenia,
- m je maximálna hĺbka stromu hľadania,
- l je hraničná hĺbka (pri obmedzenom hľadaní do hĺbky)



ĎAKUJEM ZA POZORNOSŤ