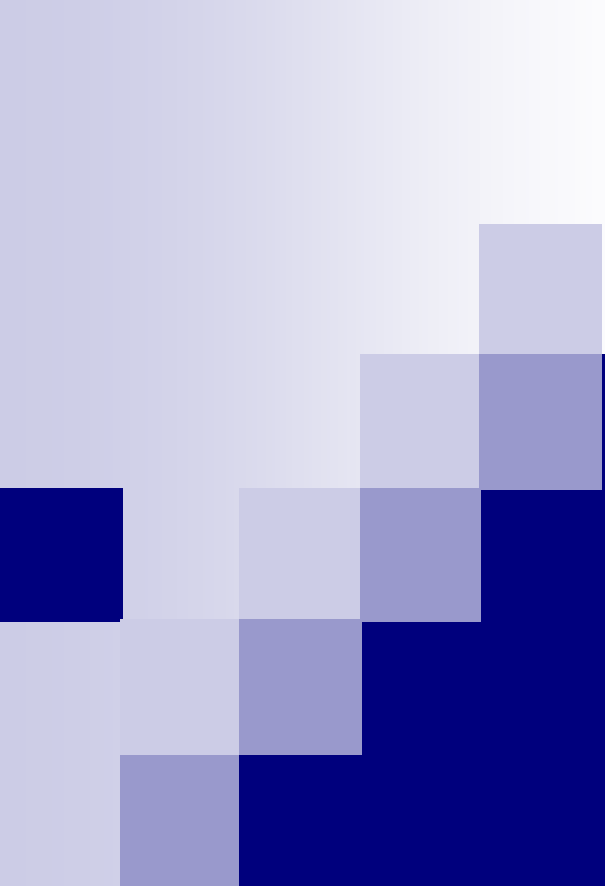




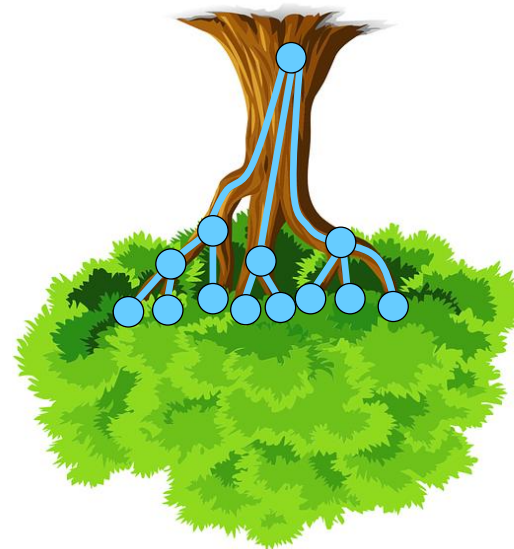
Algoritmizace

Daniel Průša, Robert Pěnička

2025

Dnešní téma

- Stromy
- Prohledávání s návratem (backtracking)
- Druhá domácí úloha



slido



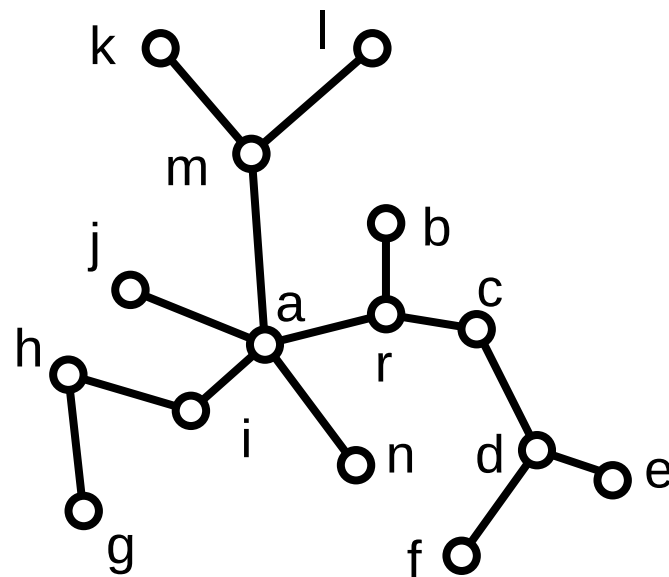
**Join at slido.com
#3115574**

① Start presenting to display the joining instructions on this slide.

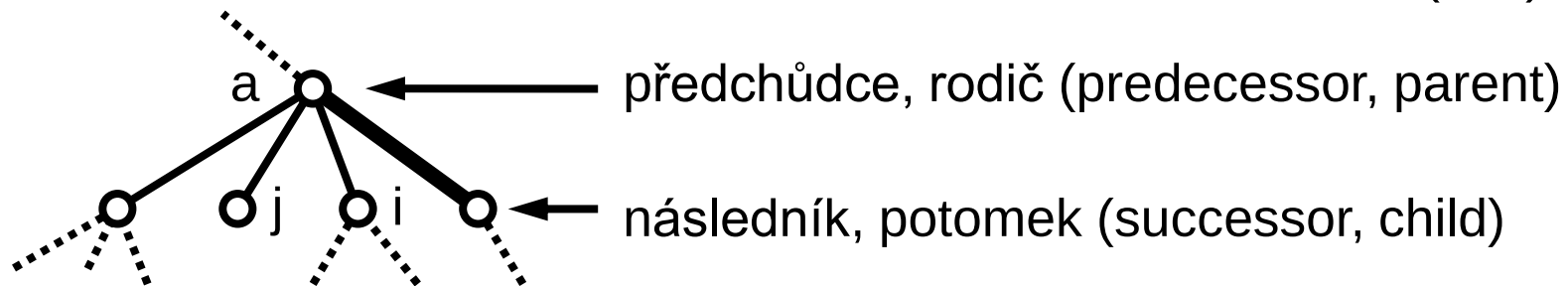
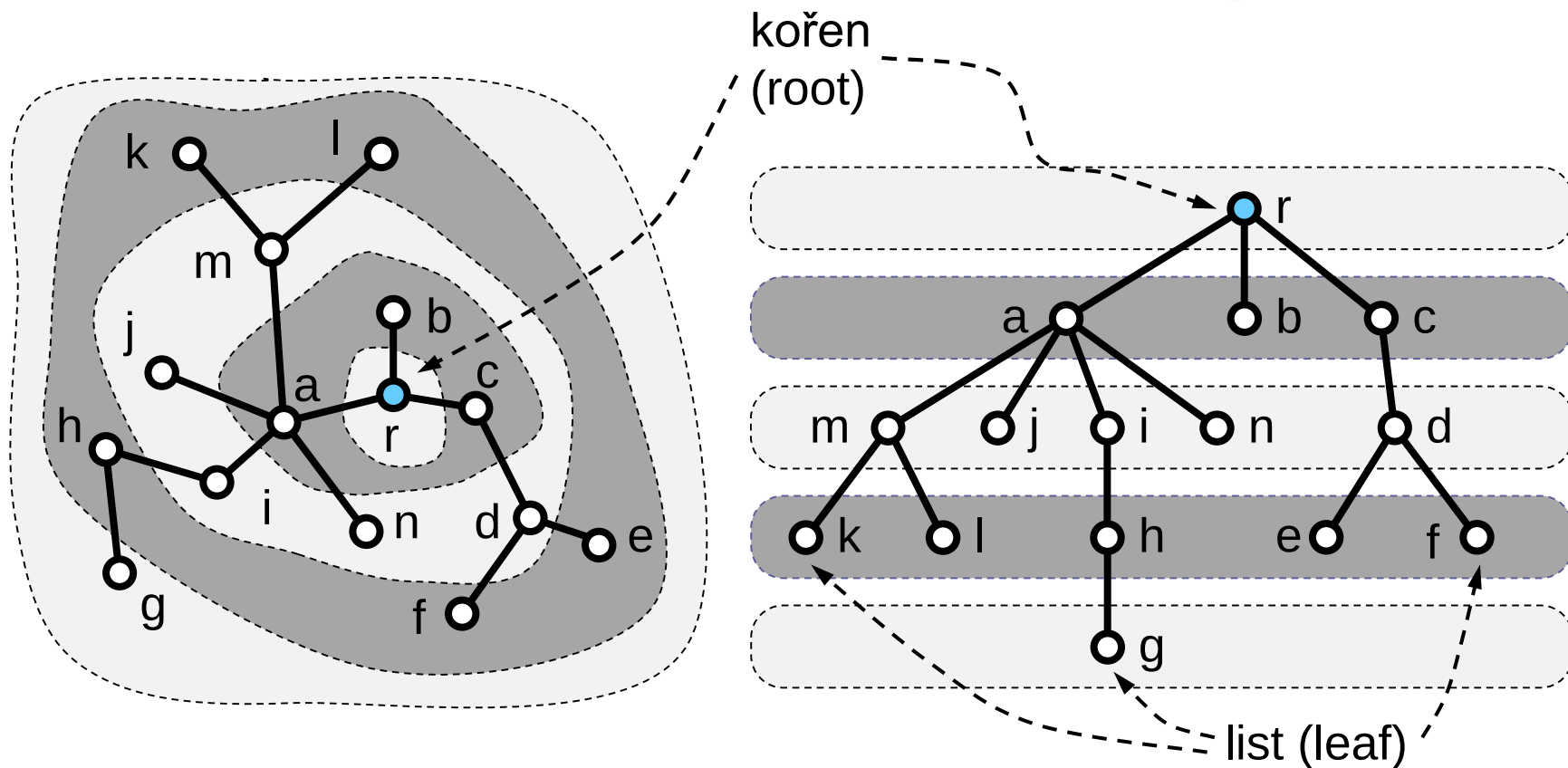
Strom jako graf

■ $G = (V, E)$

- Souvislý graf bez cyklů.
- Každé dva uzly spojuje právě jedna cesta.
- $|E| = |V| - 1$.



Kořenový strom



Co lze stromem reprezentovat?

Aritmetický výraz

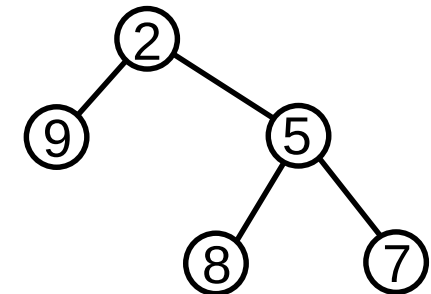
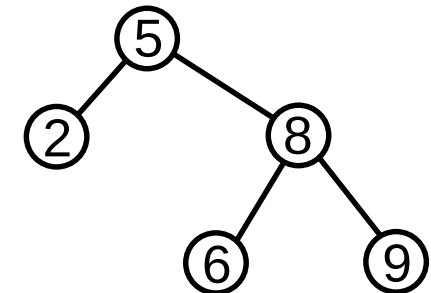
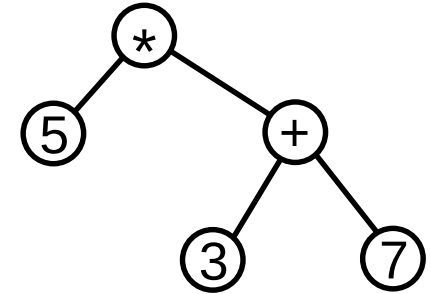
Datová struktura (BVS, halda)

Rozhodovací strom (botanický klíč)

Hierarchie entit (zaměstnanci, rodokmen)

Rekurzivní volání

Stavový prostor



slido



Co z následujícího znáte?

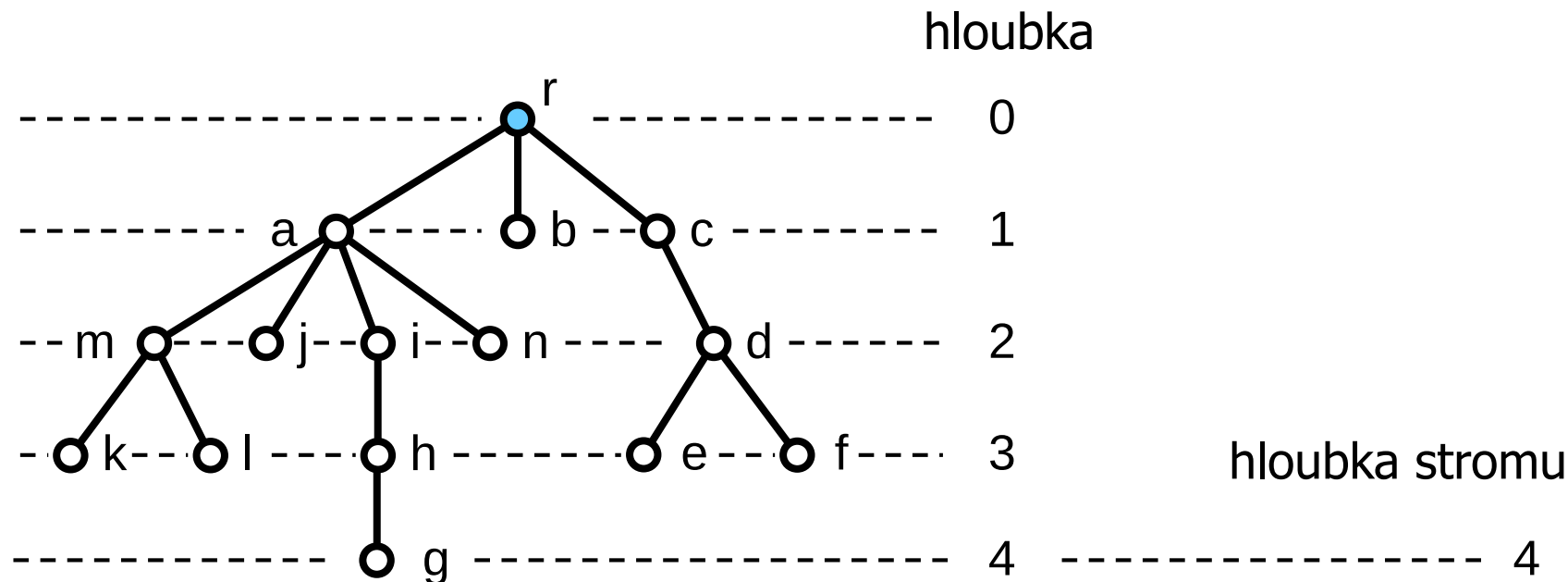
① Start presenting to display the poll results on this slide.



Audience Q&A Session

① Start presenting to display the audience questions on this slide.

Hloubka kořenového stromu

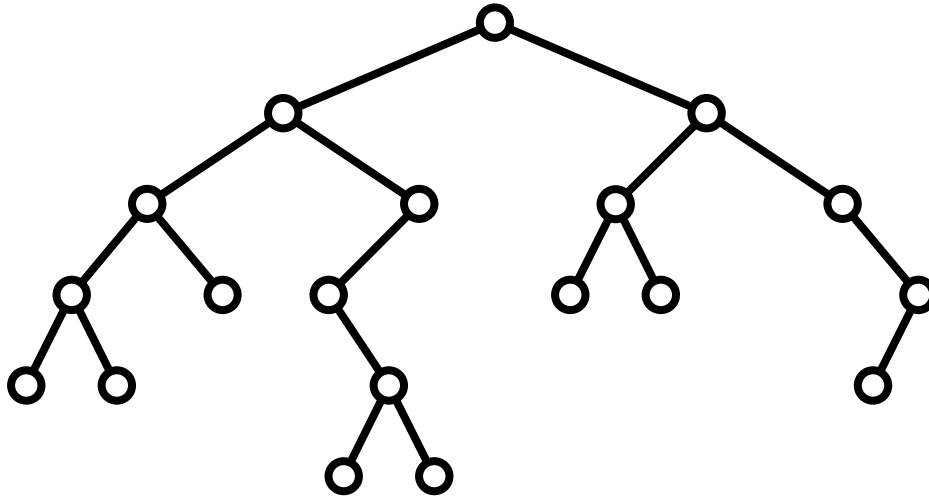


Hloubka uzlu u je hranová vzdálenost u od kořene.

Hloubka stromu T je maximum z hloubek uzlů v T .

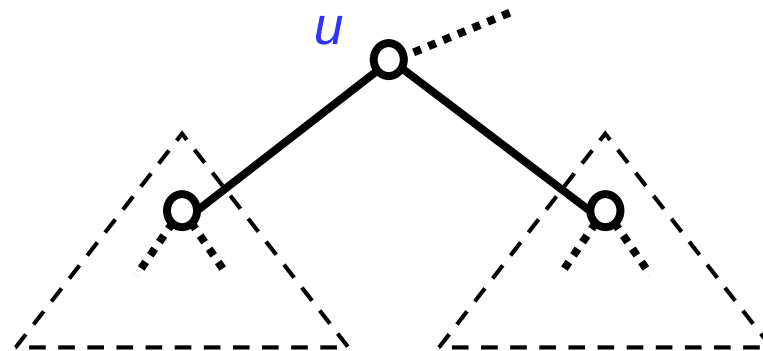
Hloubku prázdného stromu definujeme jako -1 .

Binární strom



Počet potomků každého uzlu je 0, 1 nebo 2.

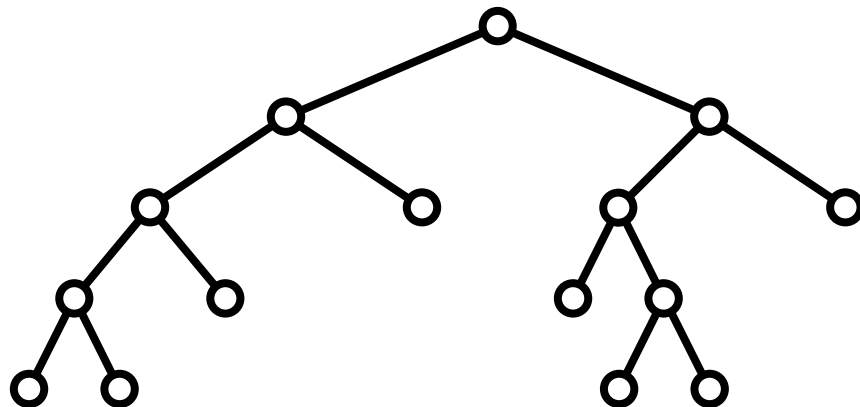
Levý a pravý podstrom
(left and right subtree):



Podstrom uzlu *u* levý pravý

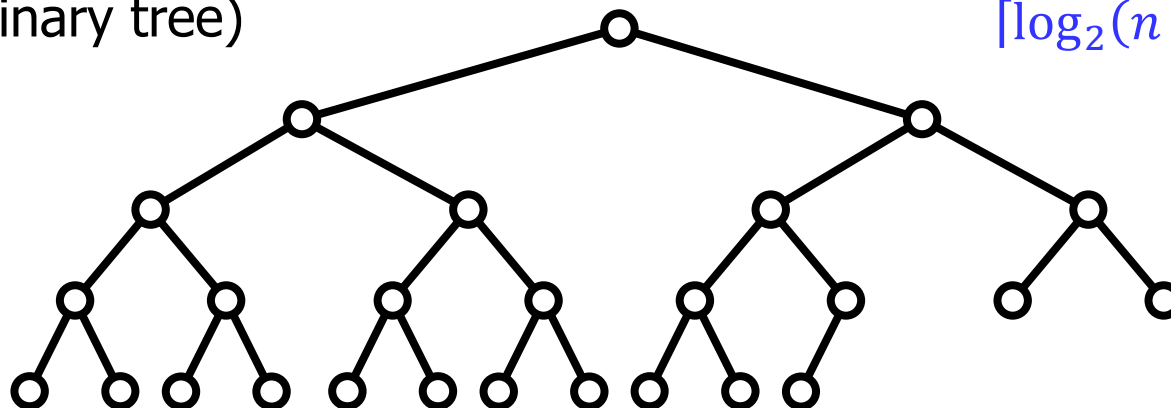
Binární strom

Pravidelný binární strom
(regular binary tree)



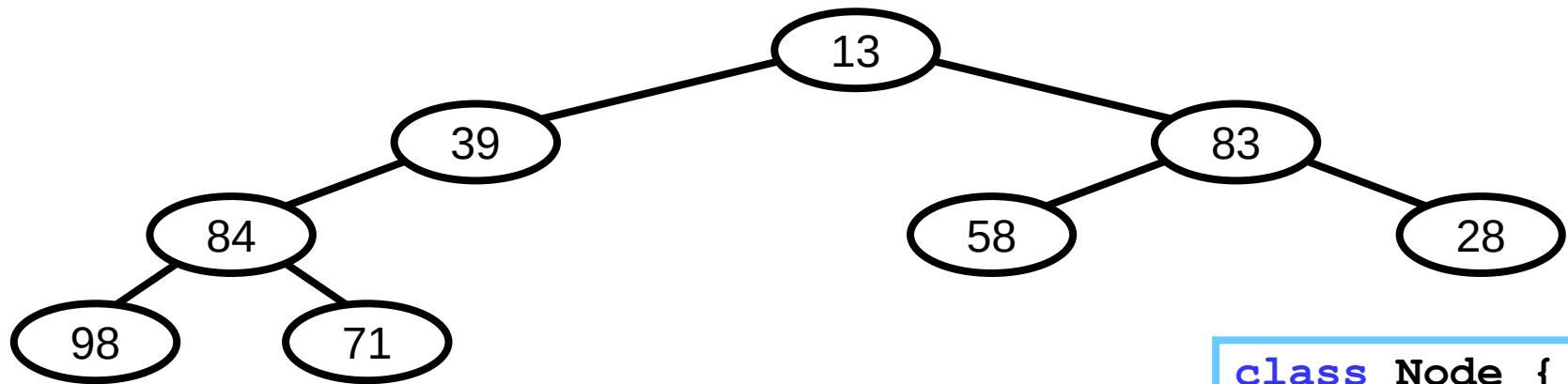
Počet potomků každého uzlu
je jen 0 nebo 2.

Vyvážený binární strom
(balanced binary tree)

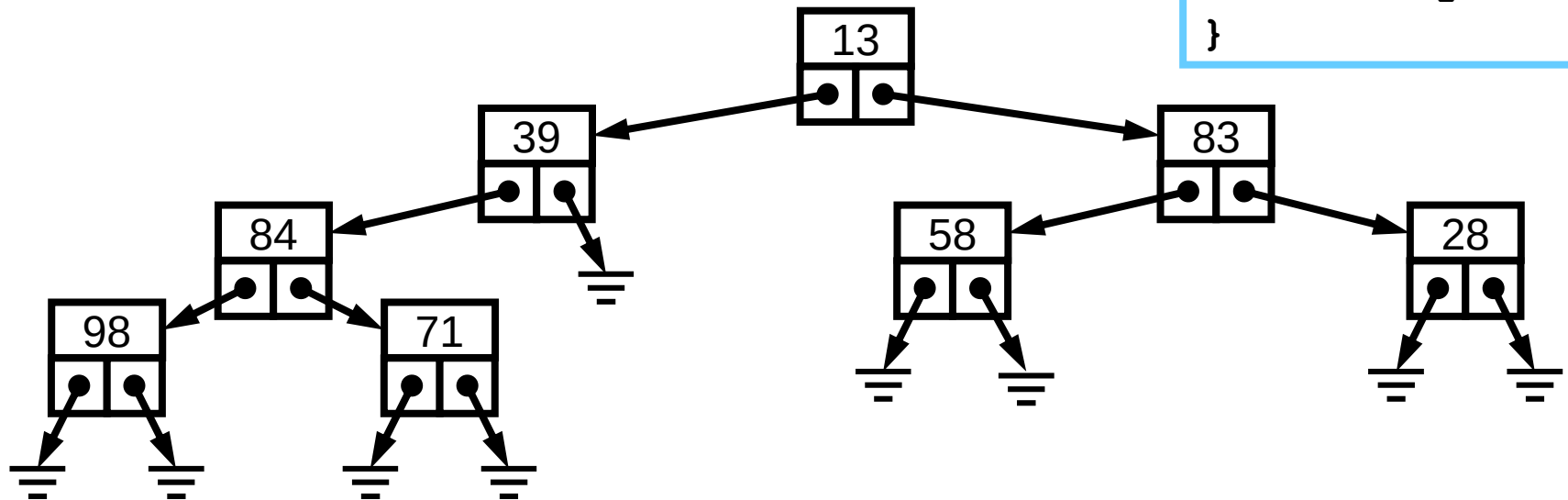


hloubka pro n uzlů je
 $\lceil \log_2(n + 1) \rceil - 1$

Reprezentace binárního stromu



```
class Node {  
    Node left;  
    Node right;  
    int key;  
}
```

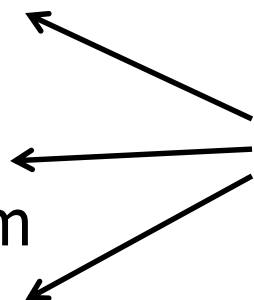


Průchod stromem do hloubky

Začni v kořeni stromu a opakuj:

Pro aktuální uzel u

- projdi rekurzivně levý podstrom
- projdi rekurzivně pravý podstrom



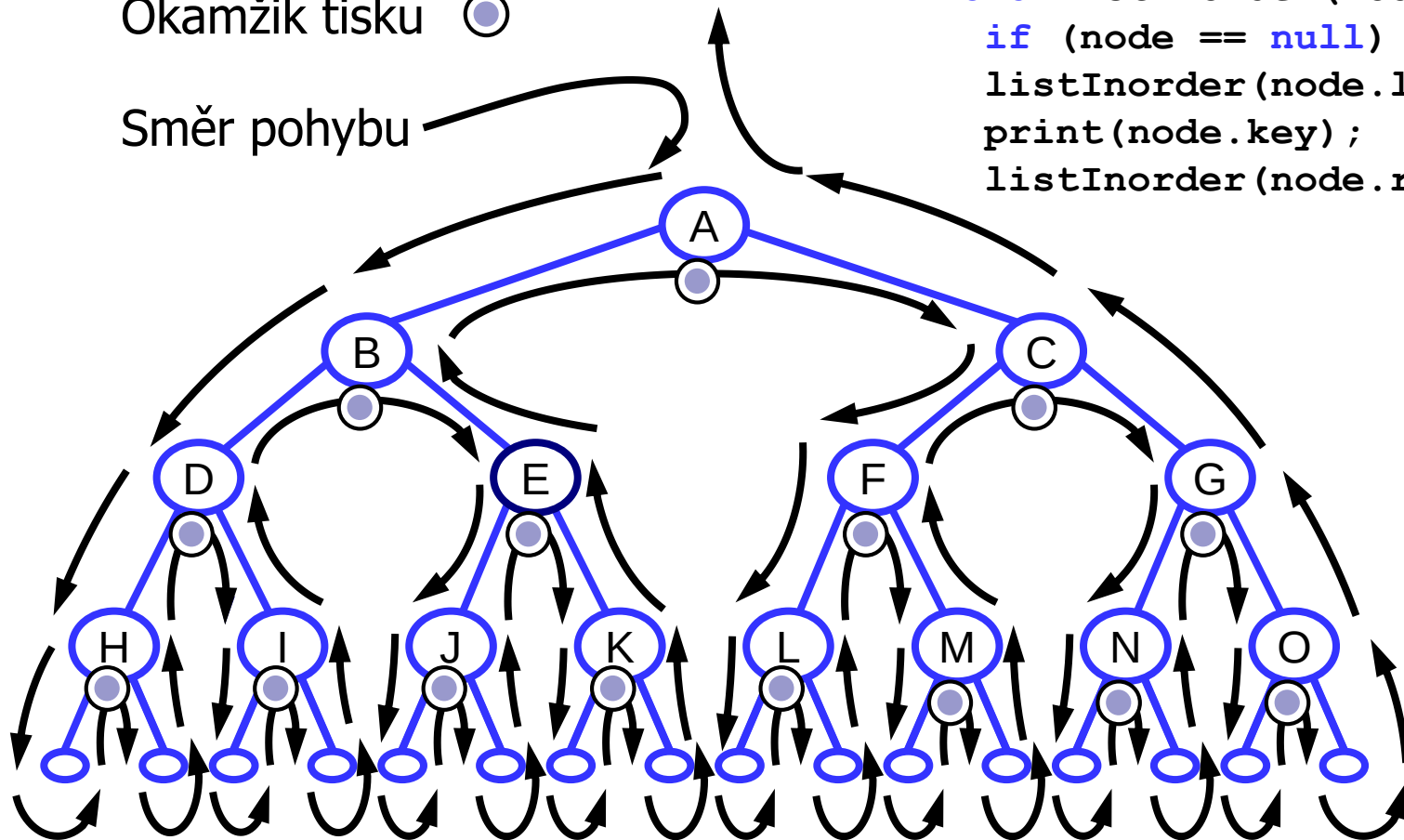
zpracuj uzel u
(je více možností,
kdy to provést)

Průchod v pořadí Inorder

Okamžik tisku ○

Směr pohybu

```
void listInorder(Node node) :  
    if (node == null) return;  
    listInorder(node.left);  
    print(node.key);  
    listInorder(node.right);
```



Výstup

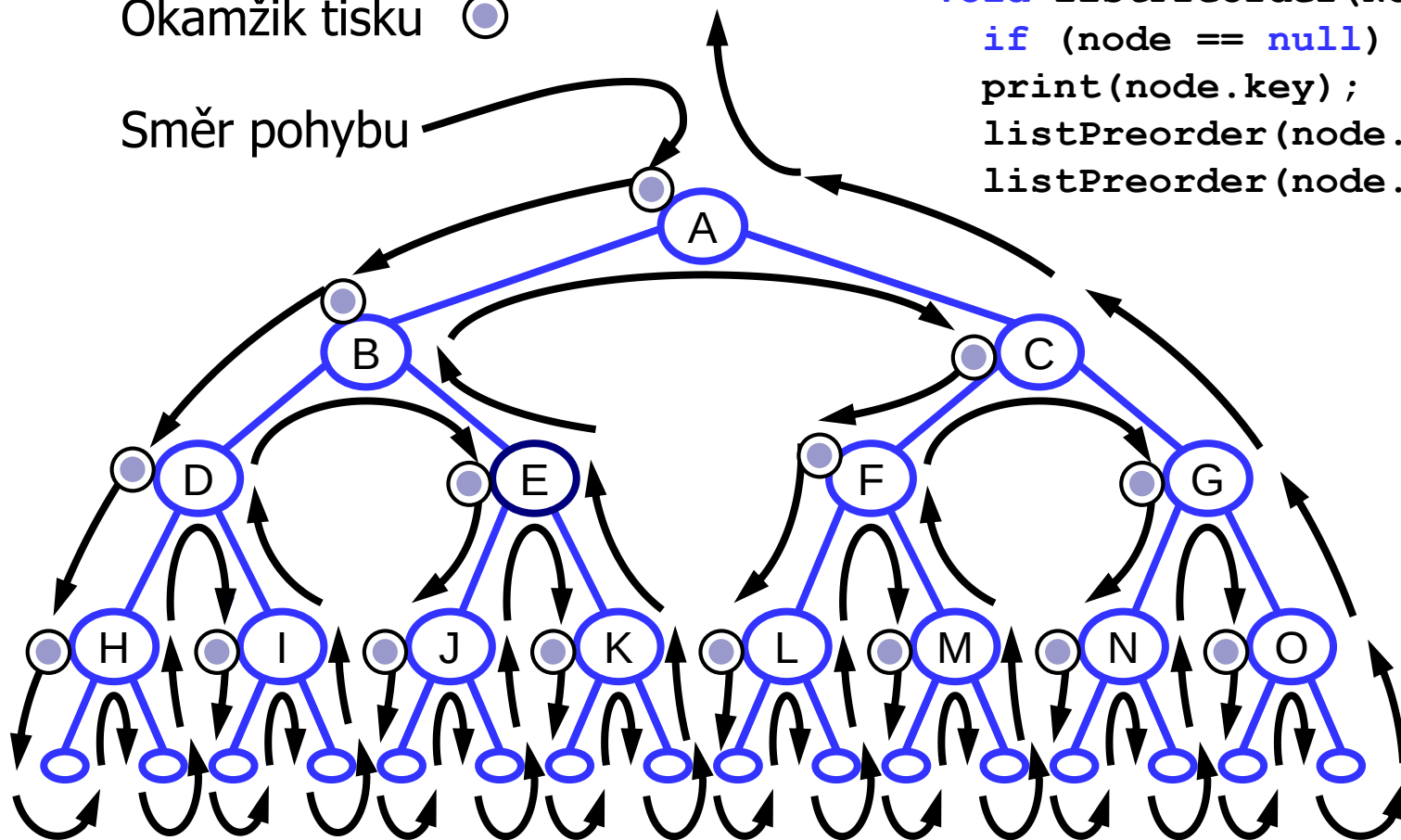
H D I B J E K A L F M C N G O

Průchod v pořadí Preorder

Okamžik tisku ○

Směr pohybu

```
void listPreorder(Node node) :  
    if (node == null) return;  
    print(node.key);  
    listPreorder(node.left);  
    listPreorder(node.right);
```



Výstup

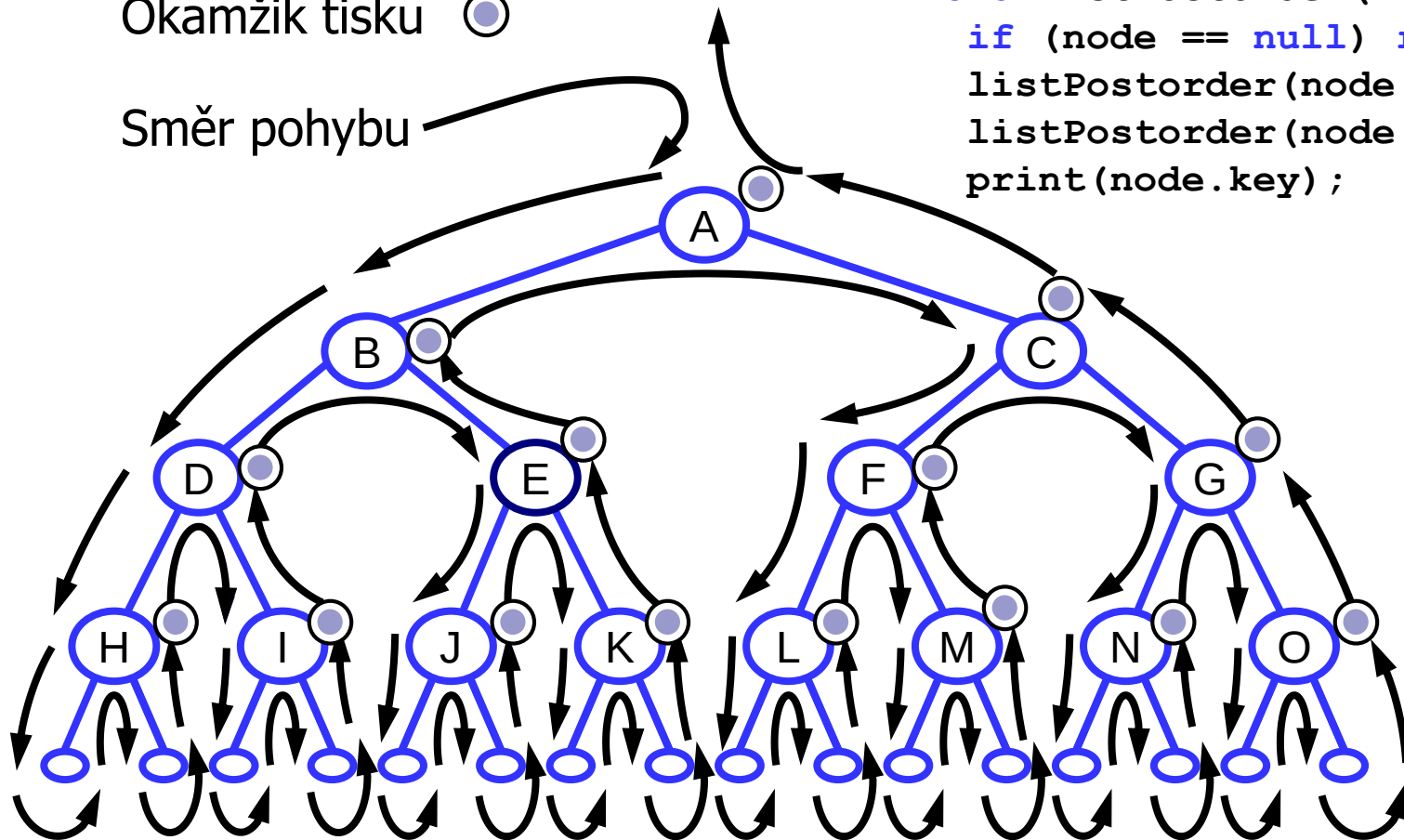
A B D H I E J K C F L M G N O

Průchod v pořadí Postorder

Okamžik tisku ○

Směr pohybu

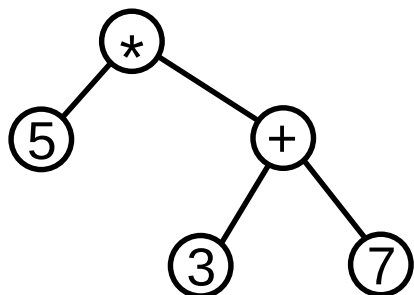
```
void listPostorder(Node node) :  
    if (node == null) return;  
    listPostorder(node.left);  
    listPostorder(node.right);  
    print(node.key);
```



Výstup

H I D J K E B L M F N O G C A

Použití průchodů



strom reprezentuje výraz

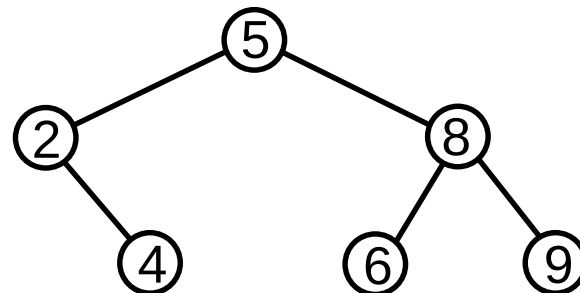
$5 * (3 + 7)$

- Inorder: $5 * 3 + 7$
 - reprezentovaný výraz nelze určit
- Preorder: $* 5 + 3 7$
 - Lisp
- Postorder: $5 3 7 + *$
 - reverzní polská notace, zásobníkový kalkulátor

Zásobník implementuje rekurzi

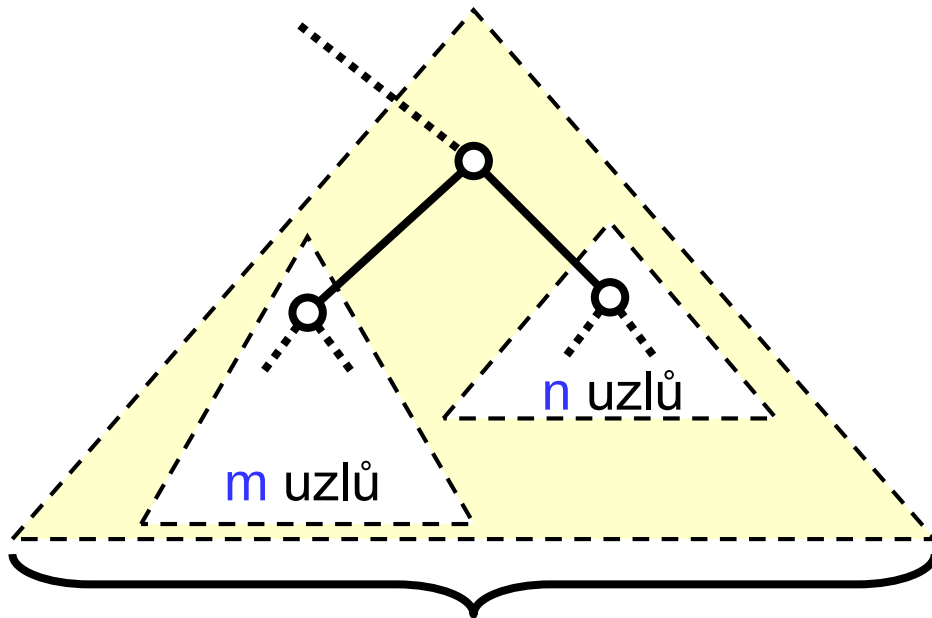
```
void inorderIterative(Node root) {  
    Stack<Node> stack = new Stack();  
    Node curr = root;  
    while (!stack.empty() || curr != null) {  
        if (curr != null) {  
            stack.push(curr);  
            curr = curr.left;  
        } else {  
            curr = stack.pop();  
            System.out.print(curr.key + " ");  
            curr = curr.right;  
        }  
    }  
}
```

Uzel uložíme na zásobník v okamžiku jeho objevení a odebereme jej po zpracování jeho levého podstromu.



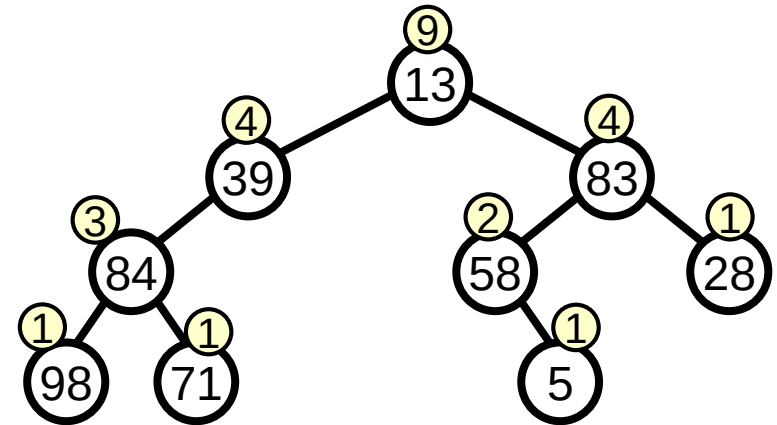
Počítání uzlů

Strom nebo podstrom



celkem ... $m+n+1$ uzlů

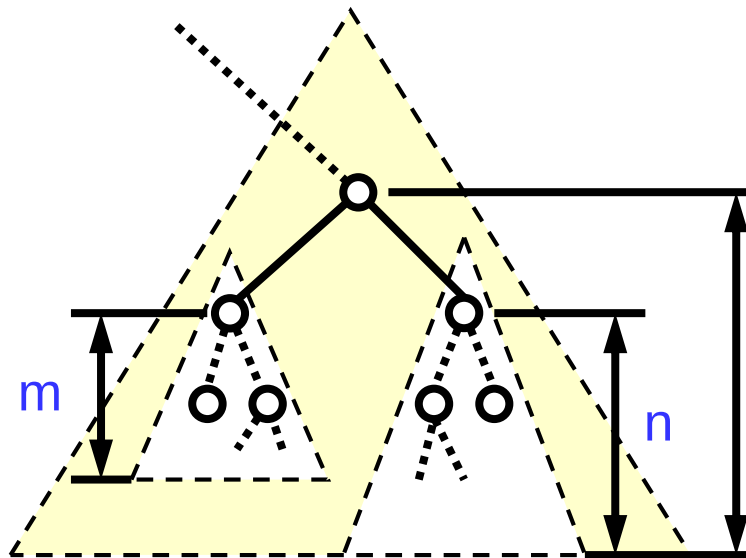
Příklad



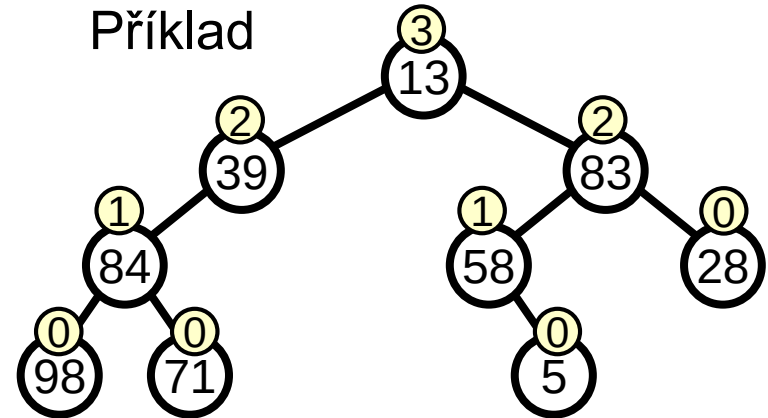
```
int count(Node node) {  
    if (node == null) return 0;  
    return (count(node.left) + count(node.right) + 1);  
}
```

Počítání hloubky

Strom nebo podstrom



Příklad



$\max(m,n)+1$

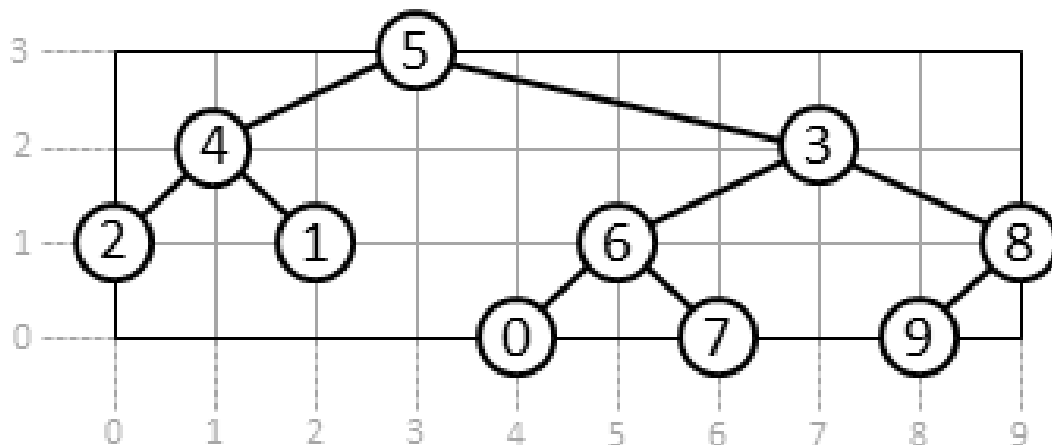
```
int depth(Node node) {  
    if (node == null) return -1;  
    return max(depth(node.left), depth(node.right)) + 1;  
}
```

Kreslení binárního stromu

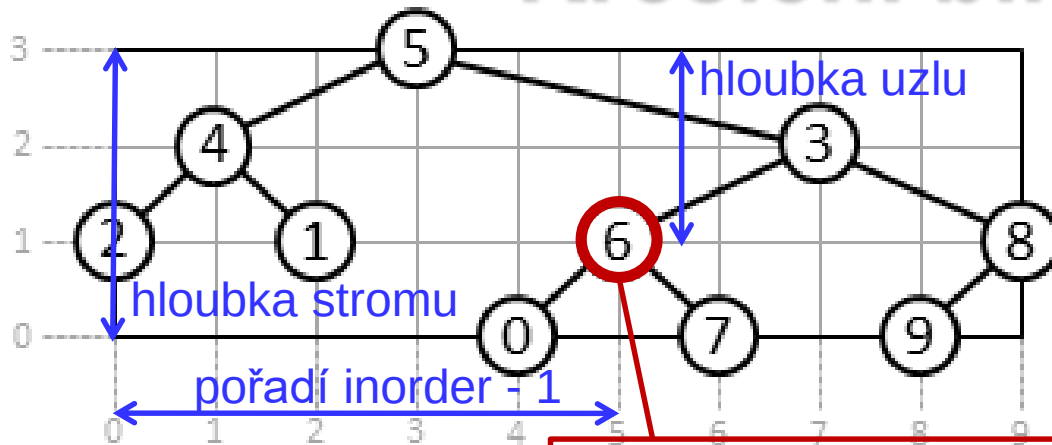
Vstup: binární strom T s n uzly.

Jak určíme souřadnice uzlů (středů kružnic), aby

- každý uzel měl celočíselnou y -ovou souřadnici o 1 větší než jeho potomci,
- všechny uzly měly navzájem různé celočíselné x -ové souřadnice od 0 do $n - 1$,
- každý levý (pravý) potomek měl x -ovou souřadnici menší (větší) než jeho rodič?



Kreslení binárního stromu



```
static int treeDepth = 0;
```

```
static int counter = 0;
```

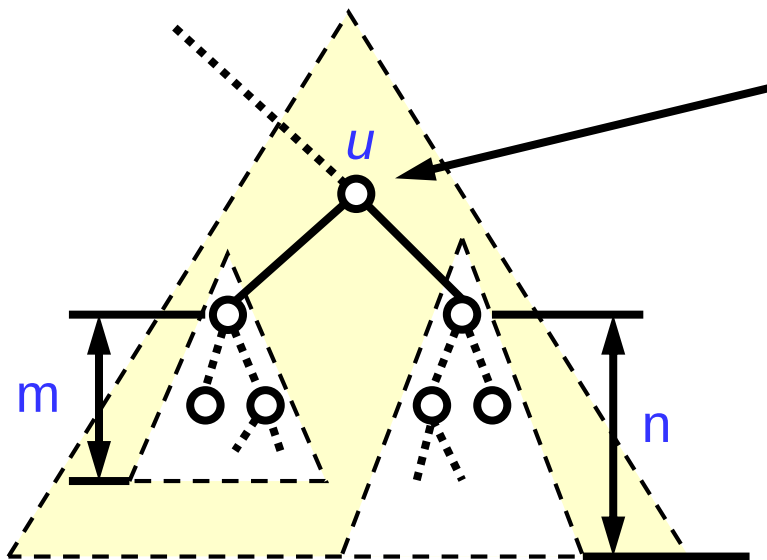
```
class Node {  
    Node left, right;  
    int key;  
    int depth, inorder; }
```

```
void process(Node node, int depth) {  
    if (node == null) return;  
    node.depth = depth;  
    if (depth > treeDepth) treeDepth = depth;  
    process(node.left, depth + 1);  
    node.inorder = ++counter;  
    process(node.right, depth + 1); }
```

Call: `process(root, 0);`

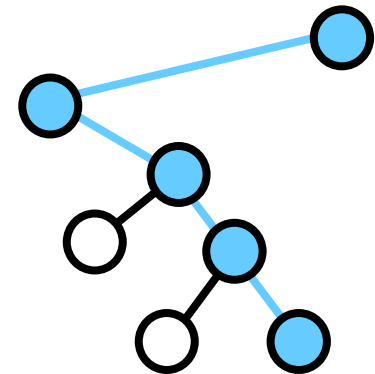
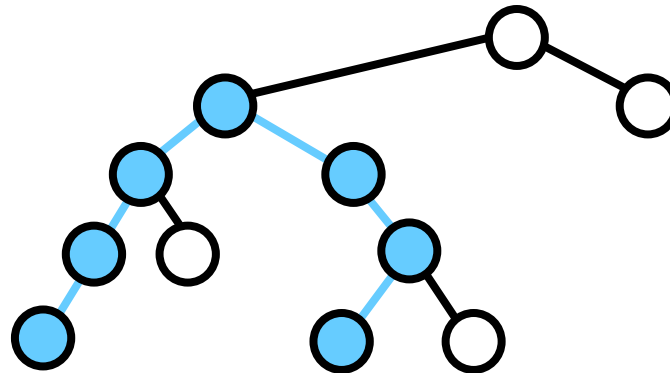
Počítání nejdelší cesty

Strom nebo podstrom



$m+n+2$... délka nejdelší cesty s nejvyšším bodem v uzlu u

Příklady





**Kolik různých cest
nenulové délky existuje v
binárním stromě s 'n'
uzly?**



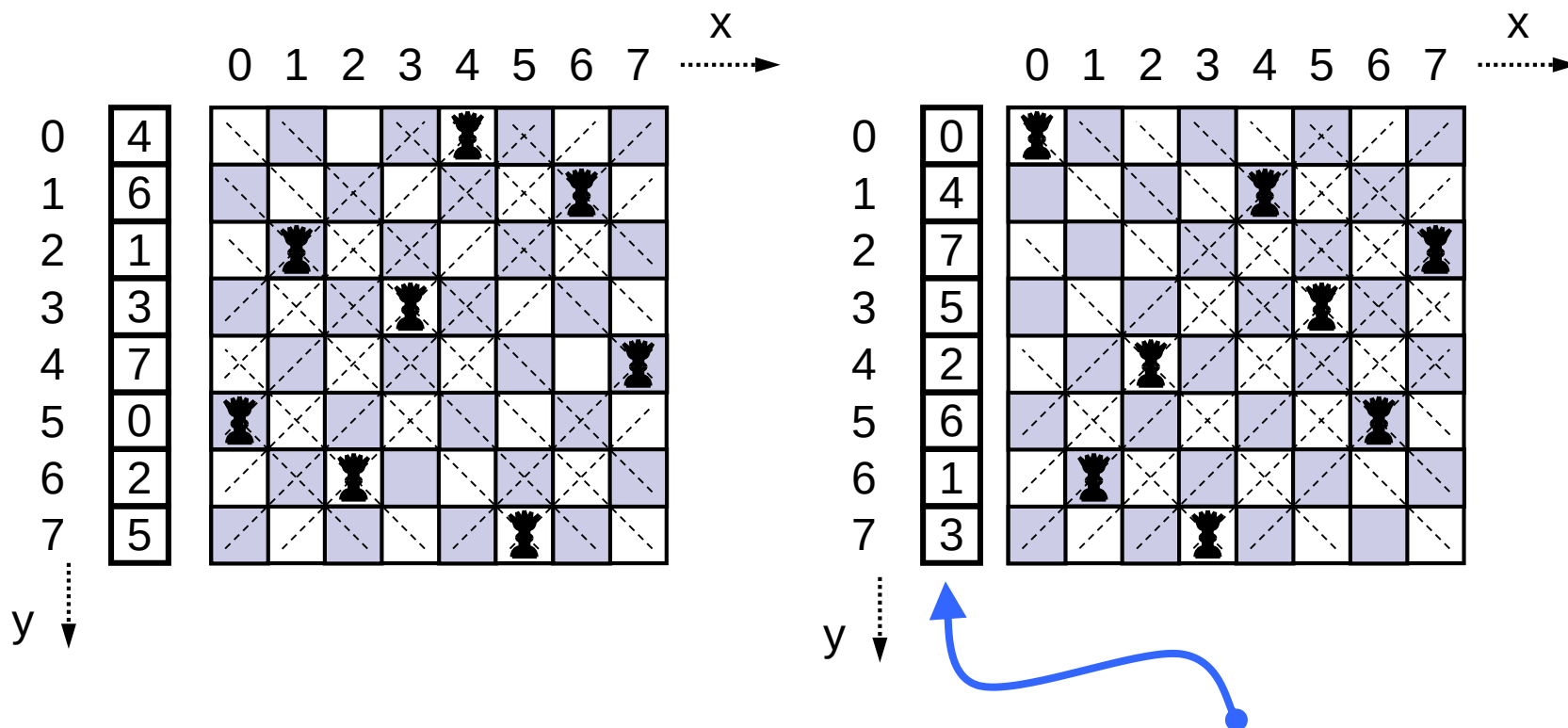
Audience Q&A Session

① Start presenting to display the audience questions on this slide.

Prohledávání s návratem

Problém osmi dam na šachovnici (obecně N dam na šachovnici NxN)

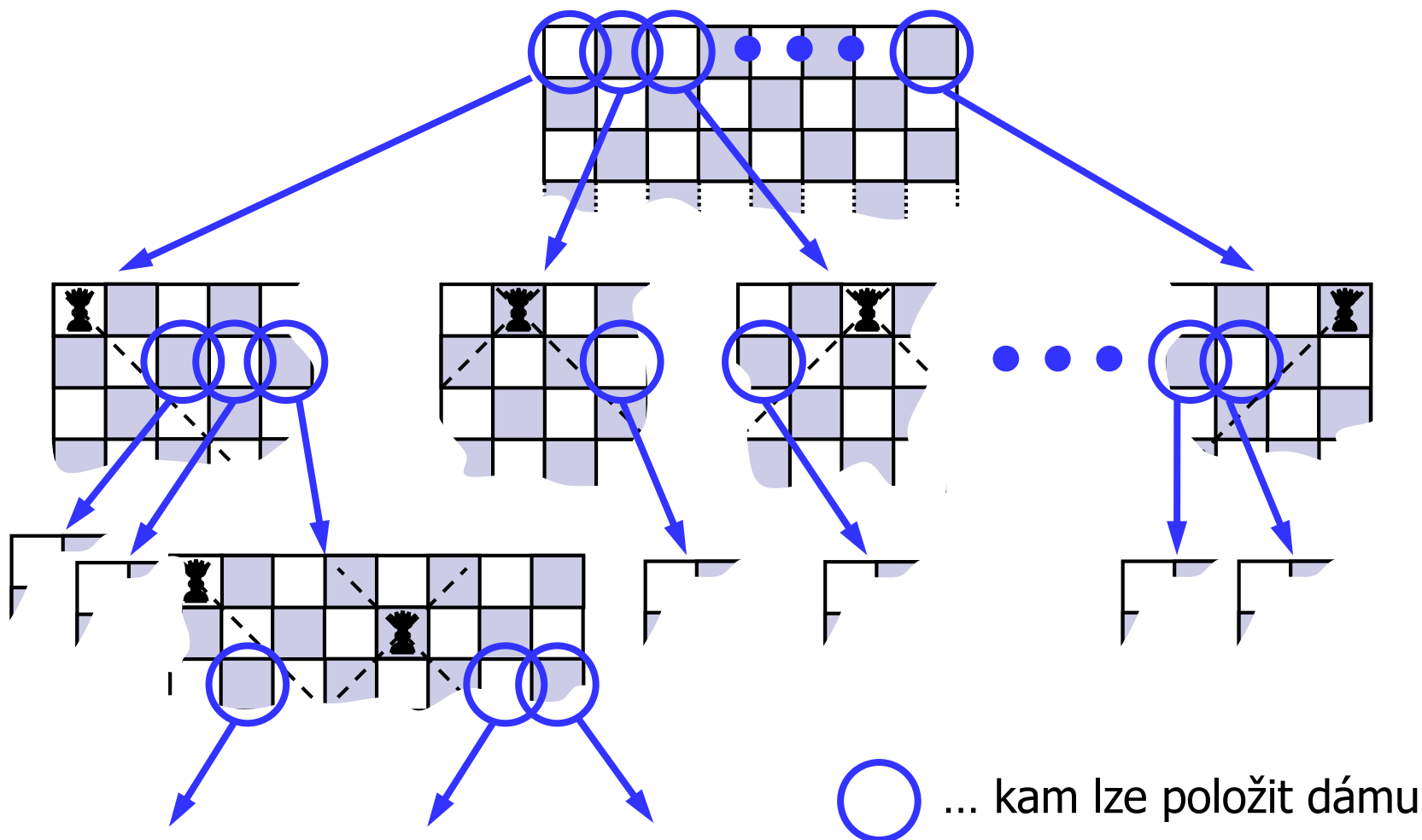
Některá řešení:



Jediná datová struktura `xPosArray[ ]` (viz kód)

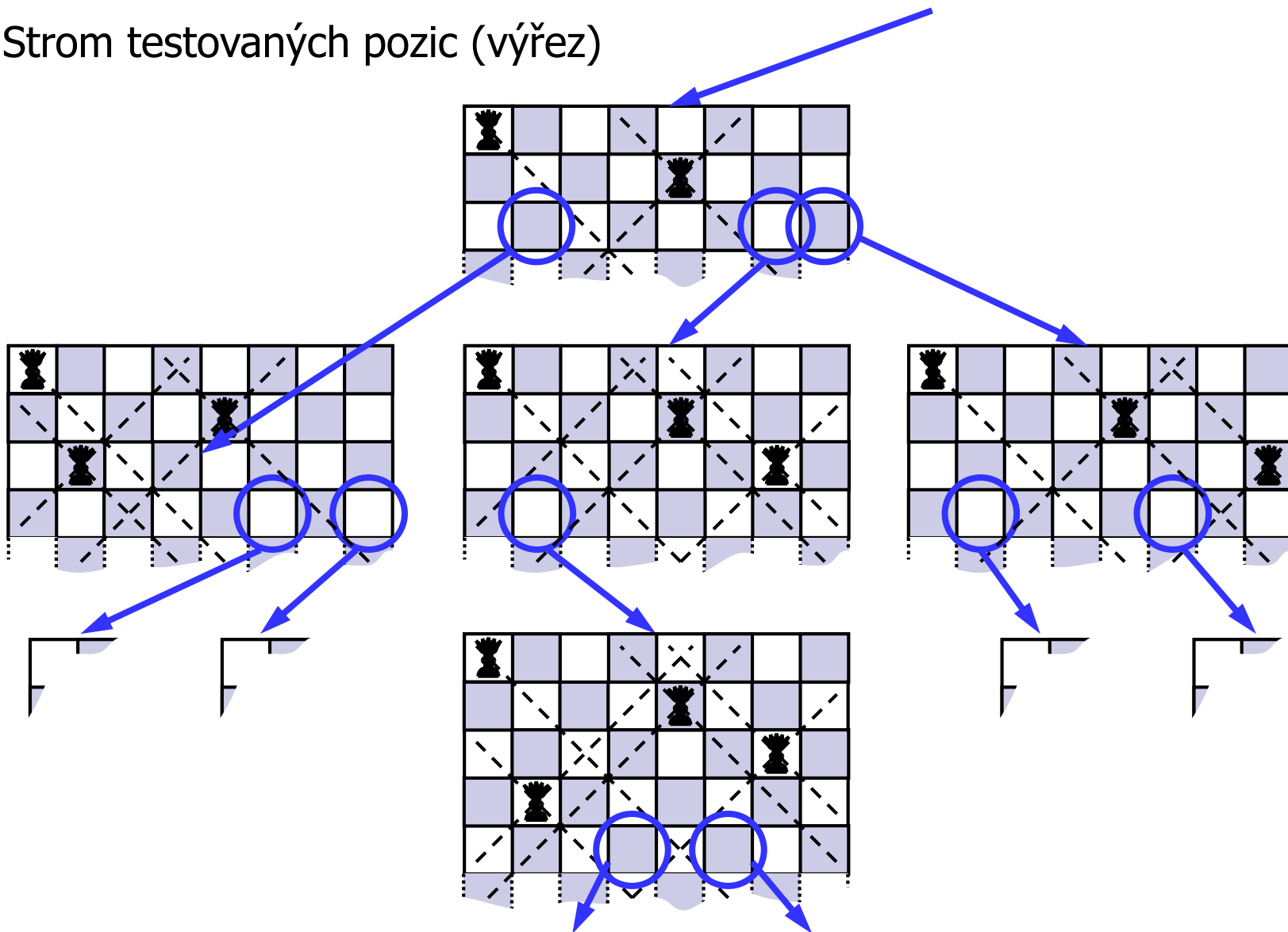
Prohledávání s návratem

Strom testovaných pozic (kořen a několik potomků)



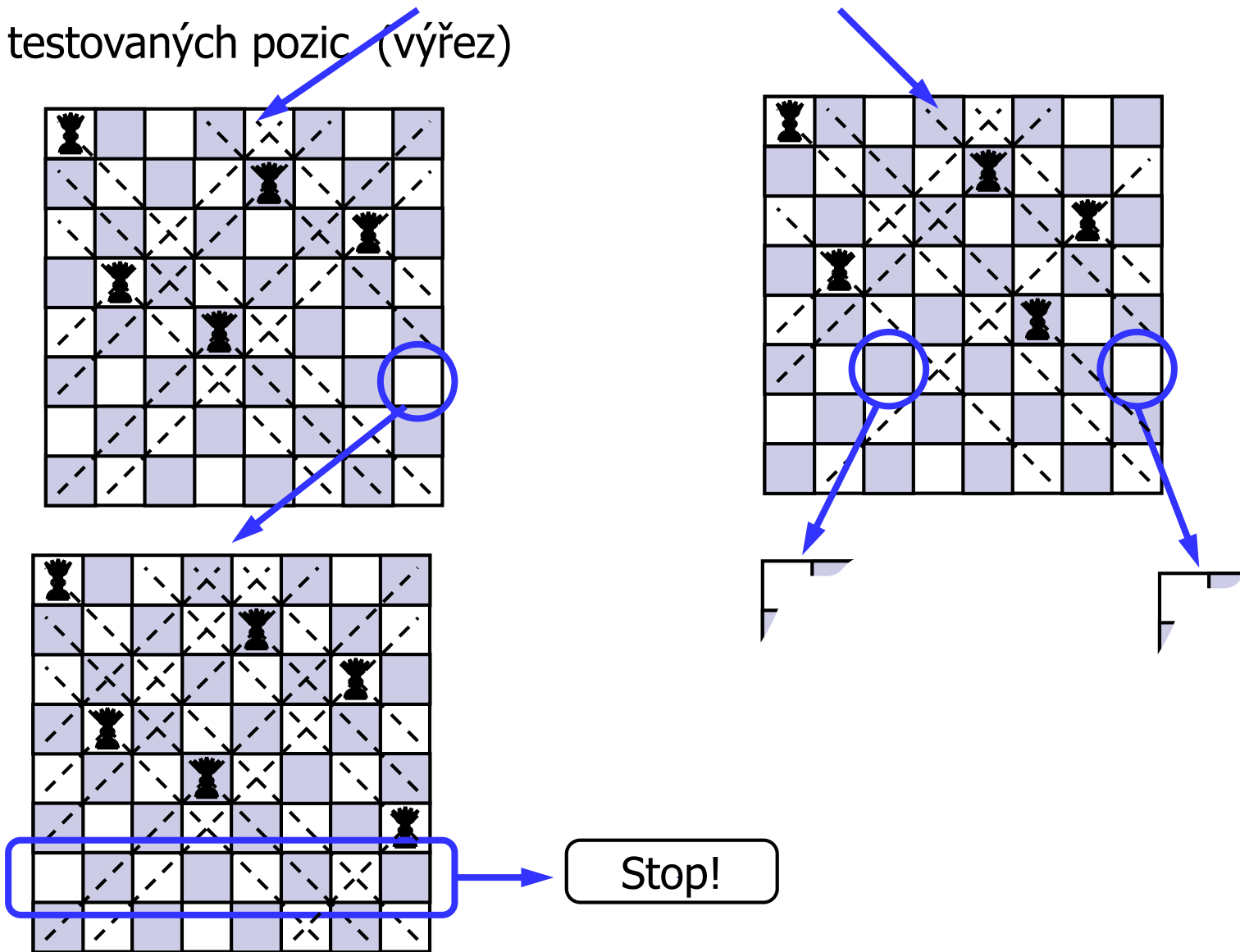
Prohledávání s návratem

Strom testovaných pozic (výřez)



Prohledávání s návratem

Strom testovaných pozic (výřez)



Prohledávání s návratem

```
boolean posOK(int x, int y) {
    int i;
    for (i = 0; i < y; i++)
        if ((xPosArr[i] == x) || // stejný sloupec
            (abs(y-i) == abs(xPosArr[i]-x))) // nebo diagonála
            return false;
    return true;
}

void tryPutRow(int y) {
    int x;
    if (y == N) print_xPosArr(); // řešení
    else
        for (x = 0; x < N; x++) // testuj sloupce
            if (posOK(x, y)) { // když je volno,
                xPosArr[y] = x; // dej tam dámu
                tryPutRow(y + 1); // a volej rekurzi
            }
}
```

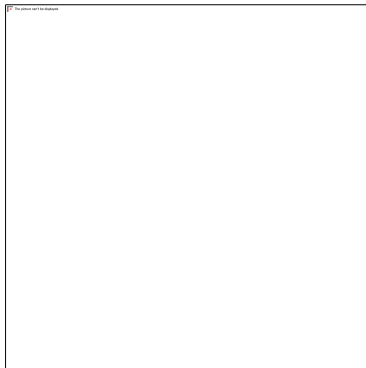
Call: tryPutRow(0);

Prohledávání s návratem

N poč. dam	Počet řešení	Počet testovaných pozic dámy		Zrychlení
		Hrubá síla (N^N)	Backtrack	
4	?	256	240	1.07
5	10	3 125	1 100	2.84
6	4	46 656	5 364	8.70
7	40	823 543	25 088	32.83
8	92	16 777 216	125 760	133.41
9	352	387 420 489	651 402	594.75
10	724	10 000 000 000	3 481 500	2 872.33
11	2 680	285 311 670 611	19 873 766	14 356.20
12	14 200	8 916 100 448 256	121 246 416	73 537.00



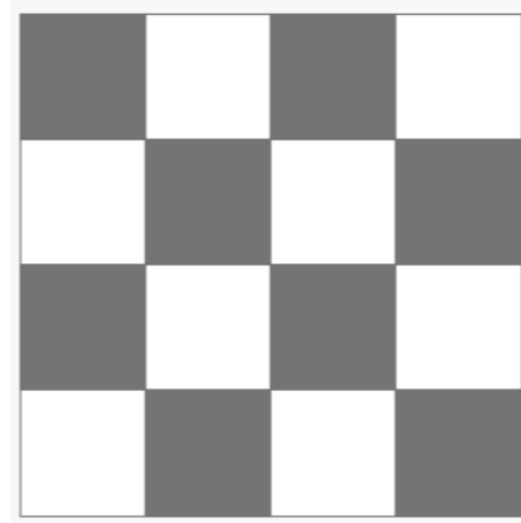
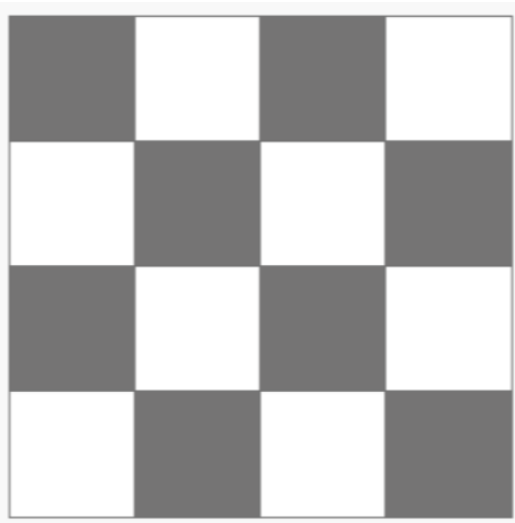
Please download and
install the Slido app on
all computers you use



**Kolika způsoby lze umístit
4 dámy na šachovnici 4x4
tak, aby se žádné dvě
navzájem neohrožovaly?**

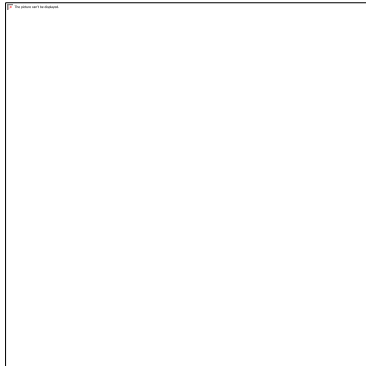
① Start presenting to display the poll results on this slide.

Šachovnice 4x4





Please download and
install the Slido app on
all computers you use

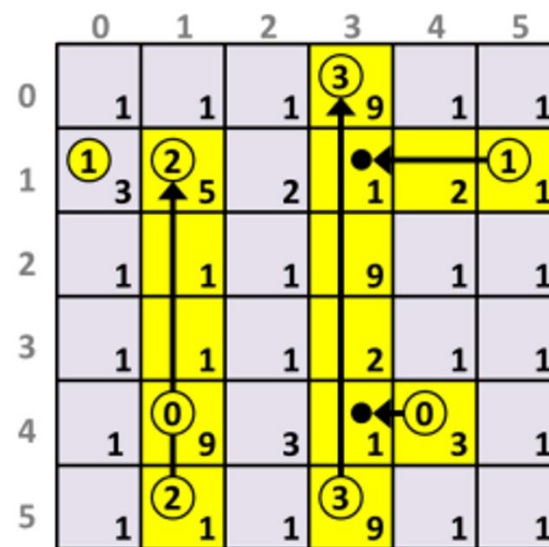
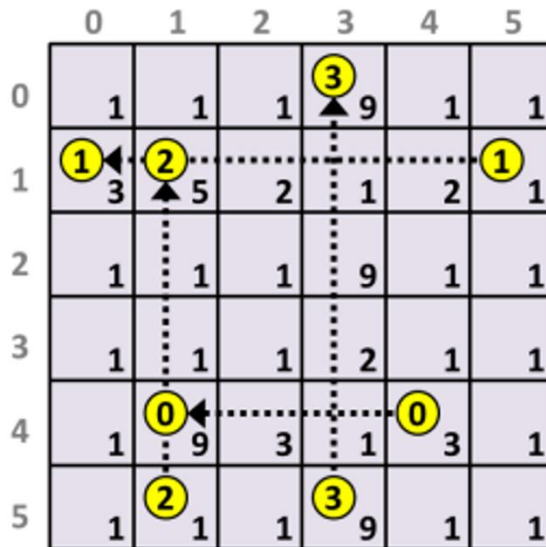


Audience Q&A

① Start presenting to display the audience questions on this slide.

Úloha o nanorobotech

- Nanoroboti shromažďují vzorky v poli sektorů.
- Vzorky v sektorech mají různé hodnoty.
- Každý nanorobot má určený startovní a koncový sektor.
- Činnost nanorobota kontaminuje procházené sektory, jsou poté neprůchozí pro ostatní nanoroboty.
- V jakém pořadí nanoroboty sekvenčně aktivovat, aby hodnota sesbíraných vzorků byla maximální?



Zvolené pořadí:

3, 2, 0, 1

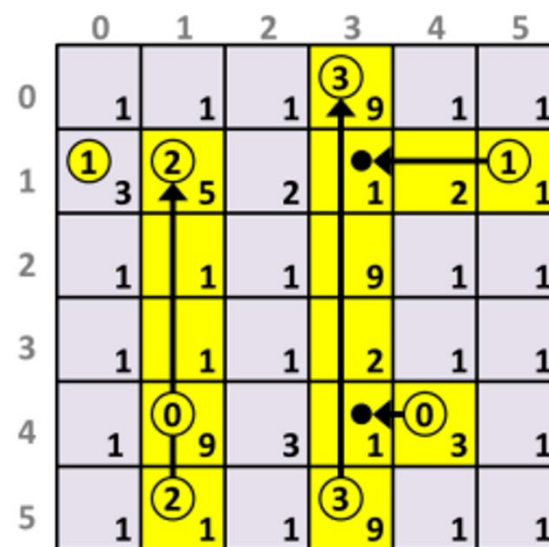
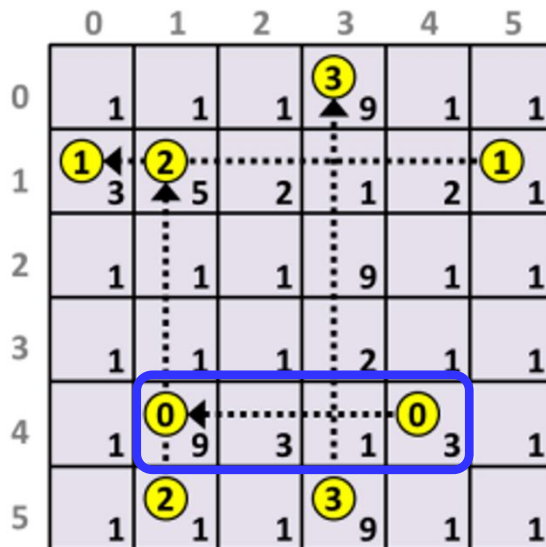
Hodnota vzorků:

$31+17+3+3=54$

Úloha o nanorobotech

- Ořezávání stavového prostoru
 - Pamatujeme si dosud nejlepší nalezené řešení.
 - Pokud aktuálně prohledávané možnosti nemohou toto řešení vylepšit, provedeme návrat (k tomuto účelu stanovíme vhodný horní (dolní) odhad ceny řešení dosažitelného z aktuálního stavu).

Příklad: Je-li nanorobot 0 aktivován jako první, nebude hodnota nasbíraných vzorků větší než $16 + (1 + 9 + 14) = 40$.



Zvolené pořadí:

3, 2, 0, 1

Hodnota vzorků:

$31 + 17 + 3 + 3 = 54$

Úloha o nanorobotech

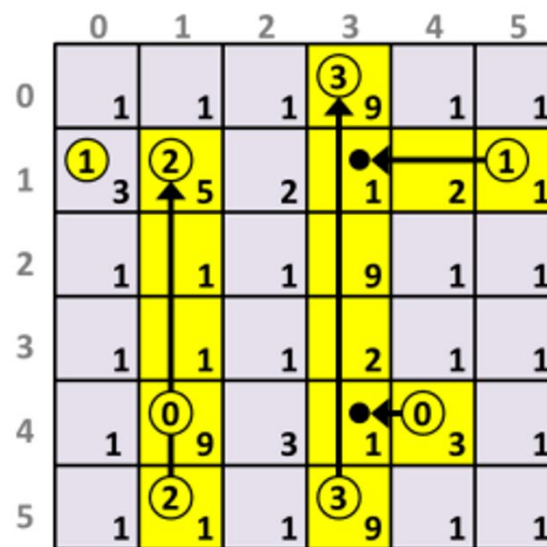
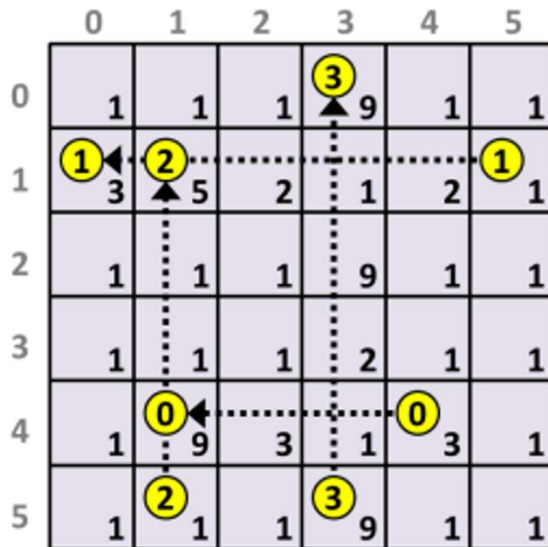
■ Heuristiky

- „Dobré“ řešení chceme najít co nejdříve, aby ořezávání bylo co nejefektivnější.

Příklad heuristiky: Zjistíme hodnotu vzorků, které jednotlivé nanoroboty sesbírají, pokud navštíví všechny sektory na jejich naplánované cestě.

Podle těchto hodnot nanoroboty uspořádáme sestupně. Stavový prostor procházíme podle získaného pořadí.

3 (31), 2 (17), 0 (16), 1 (14)



Zvolené pořadí:

3, 2, 0, 1

Hodnota vzorků:

31+17+3+3=54

slido



Audience Q&A Session

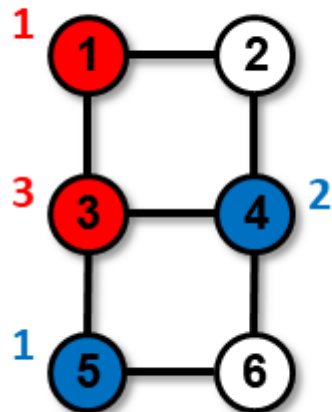
① Start presenting to display the audience questions on this slide.

Druhá domácí úloha

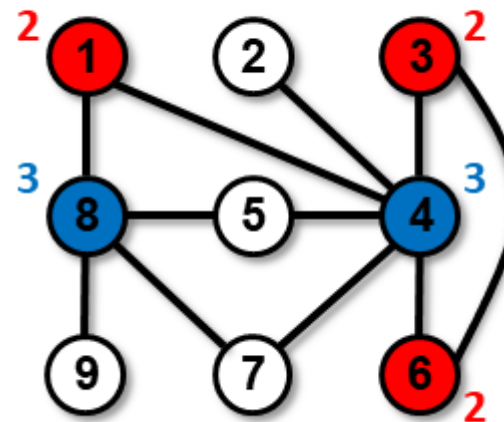
V distribuované architektuře optimálně umístit několik agentů dvou typů

- **T1** (červený) – jeho skóre je počet sousedů obsazených agentem
- **T2** (modrý) – jeho skóre je počet neobsazených sousedů

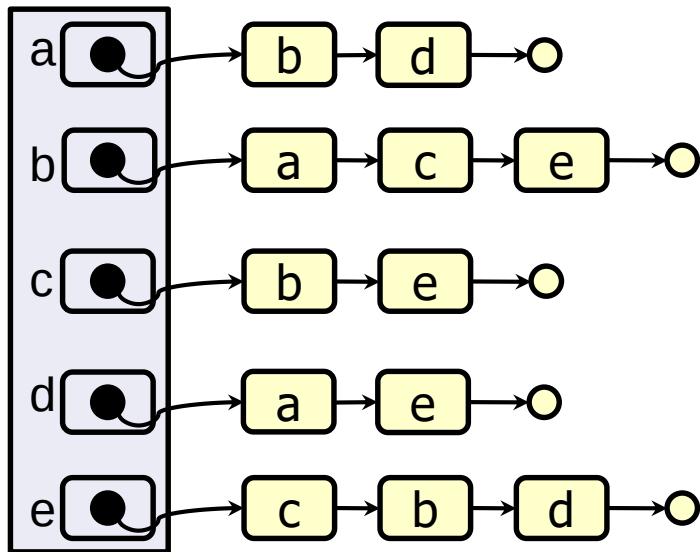
a)



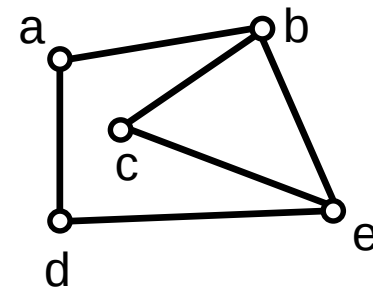
b)



Reprezentace grafu v paměti



$$G = (V, E)$$





Audience Q&A Session

① Start presenting to display the audience questions on this slide.