ADSO
6. Algoritmy řazení

(prezentace k učebnici)

Ivan Ryant

Agenda

- Základní pojmy
- Úloha abstraktních datových typů
- Posloupnosti a operace s nimi
- Vyhledávací datové struktury
- Vyhledávací posloupnost
- **Algoritmy řazení**
- Stromy
- Rozptýlené tabulky
- Prohledávání do hloubky a do šířky
- Práce s grafy
- Techniky návrhu efektivních algoritmů

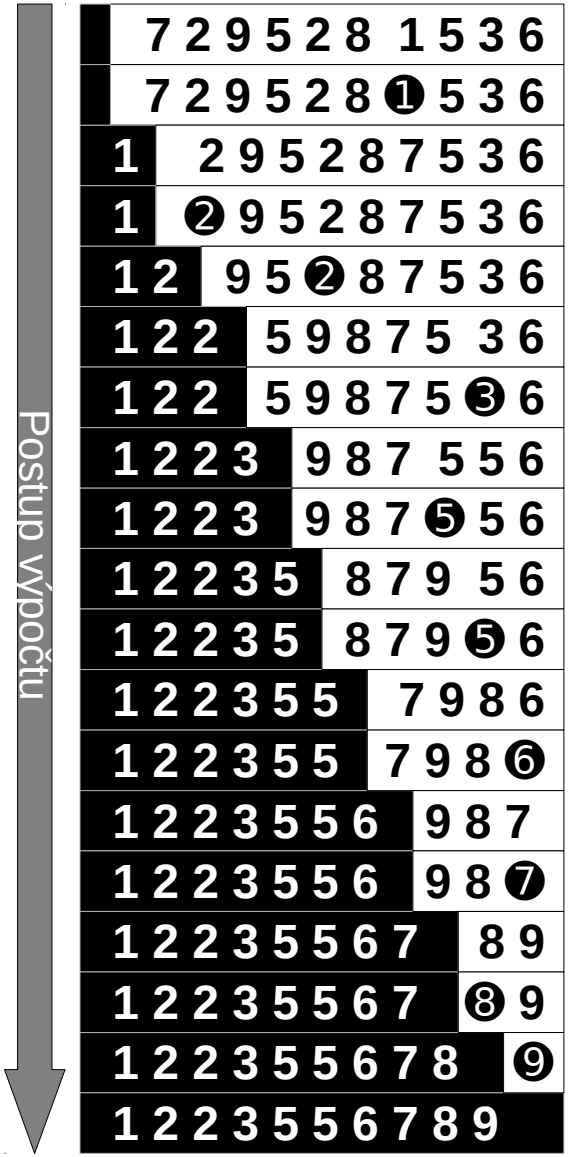
Agenda

- Základní pojmy
- Úloha abstraktních datových typů
- Posloupnosti a operace s nimi
- Vyhledávací datové struktury
- Vyhledávací posloupnost
- **Algoritmy řazení**
  - Řazení přímým výběrem
  - Řazení vkládáním
  - Bublinkové řazení
  - Řazení sléváním
  - Řazení rozdělováním
  - Counting sort
  - Řazení haldou
- Stromy
- Rozptýlené tabulky
- Prohledávání do hloubky a do šířky
- Práce s grafy
- Techniky návrhu efektivních algoritmů

Algoritmy řazení

- Řazení přímým výběrem
- Řazení vkládáním
- Bublinkové řazení
- Řazení sléváním
- Řazení rozdělováním
  - Výpočet mediánu
- Counting sort
- Řazení haldou

Řazení přímým výběrem



Vlevo je prázdná uspoř. posloupnost.
V usp. posl. vybrán prvek s min. klíčem.
Záměna prvků, roztažení uspoř. posl.
Vybereme další prvek s min. klíčem.
Vybraný zůstává na místě, vybereme další.
Prvky s klíčem 2 zachovávají pův. pořadí.
Vybereme další prvek s min. klíčem.
Pořadí prvků s klíčem 5 se změnilo!
Vybereme další prvek s min. klíčem.
Záměna, smrštění n.p., roztažení u.p.
Vybereme další prvek s min. klíčem.
Záměna, smrštění n.p., roztažení u.p.
Vybereme další prvek s min. klíčem.
Záměna, smrštění n.p., roztažení u.p.
Vybereme další prvek s min. klíčem.
Záměna, smrštění n.p., roztažení u.p.
Vybereme další prvek s min. klíčem.
Vybraný zůstává na místě, vybereme posl.
Poslední prvek zůstává na místě, hotovo.

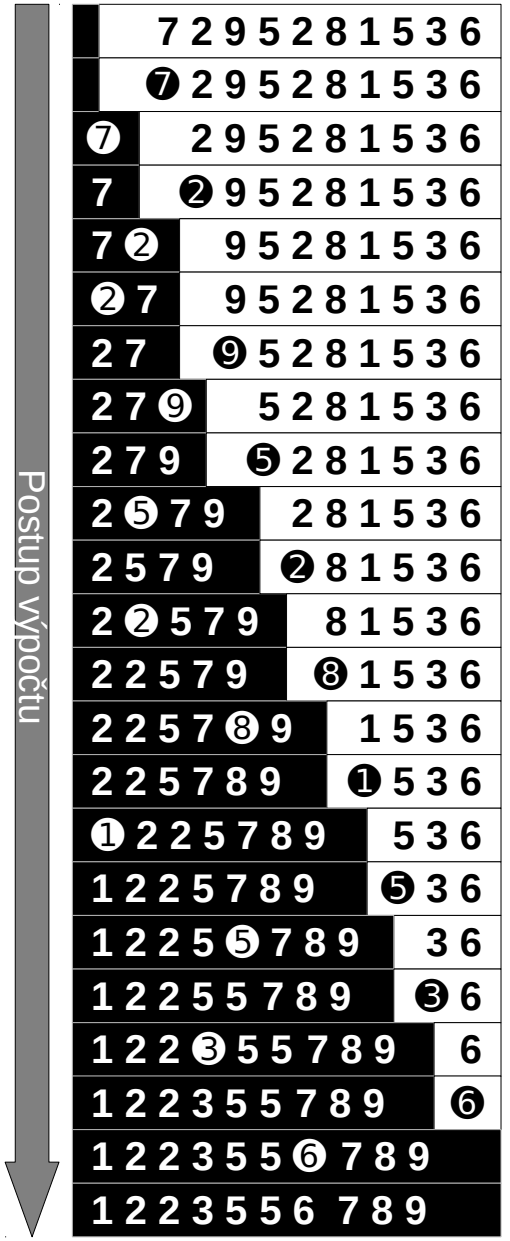
Příklad – scénář řazení posloupnosti přímým výběrem

Řazení přímým výběrem

Časová složitost:	**Kvadratická i lepší**
Paměťová složitost:	**lineární**
Na místě?	**ano, může být**
Stabilita?	**může být**

- Poznámky:
  - Časová a paměťová složitost vzhledem k počtu řazených prvků
  - Časová efektivita se může zlepšit až na lineárně logaritmickou použitím lepšího algoritmu pro výběr min. prvku (viz heapsort)
  - Nestabilní řazení může být časově nebo paměťově efektivnější

Řazení vkládáním



Vlevo je prázdná uspoř. posloupnost.
Vybereme první prvek z neusp. posl.
Usp. p. roztáhneme, neusp. smrštíme.
Vybereme první prvek z neusp. posl.
Usp. p. roztáhneme, neusp. smrštíme.
Zařadíme nový prvek do usp. p.
Vybereme první prvek z neusp. posl.
Usp. p. roztáhneme, neusp. smrštíme.
Vybereme první prvek z neusp. posl.
Zařadíme, roztáhneme, smrštíme.
Vybereme první prvek z neusp. posl.
Zachov. pův. poř. prvků se stejným kl.
Vybereme první prvek z neusp. posl.
Zařadíme, roztáhneme, smrštíme.
Vybereme první prvek z neusp. posl.
Zařadíme, roztáhneme, smrštíme.
Vybereme první prvek z neusp. posl.
Zachov. pův. poř. prvků se stejným kl.
Vybereme první prvek z neusp. posl.
Zařadíme, roztáhneme, smrštíme.
Vybereme první prvek z neusp. posl.
Zařadíme, roztáhneme, smrštíme.
Hotovo.

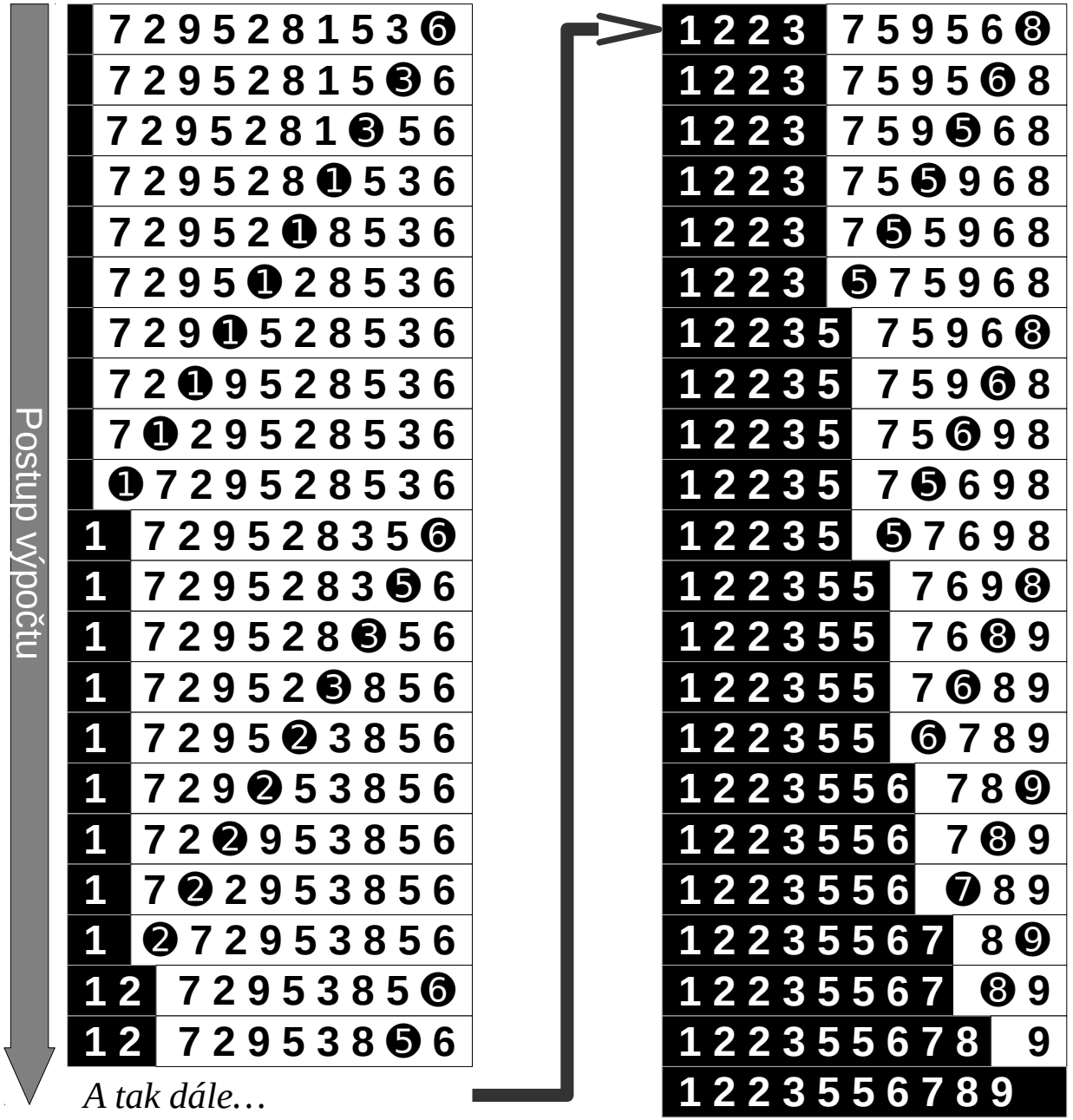
Příklad – scénář řazení posloupnosti vkládáním

Řazení vkládáním

Časová složitost:	**kvadratická**
Paměťová složitost:	**lineární**
Na místě?	**ano, může být**
Stabilita?	**může být**

- Poznámky:
  - Časová a paměťová složitost vzhledem k počtu řazených prvků
  - Nestabilní řazení může být časově nebo paměťově efektivnější

Bublinkové řazení



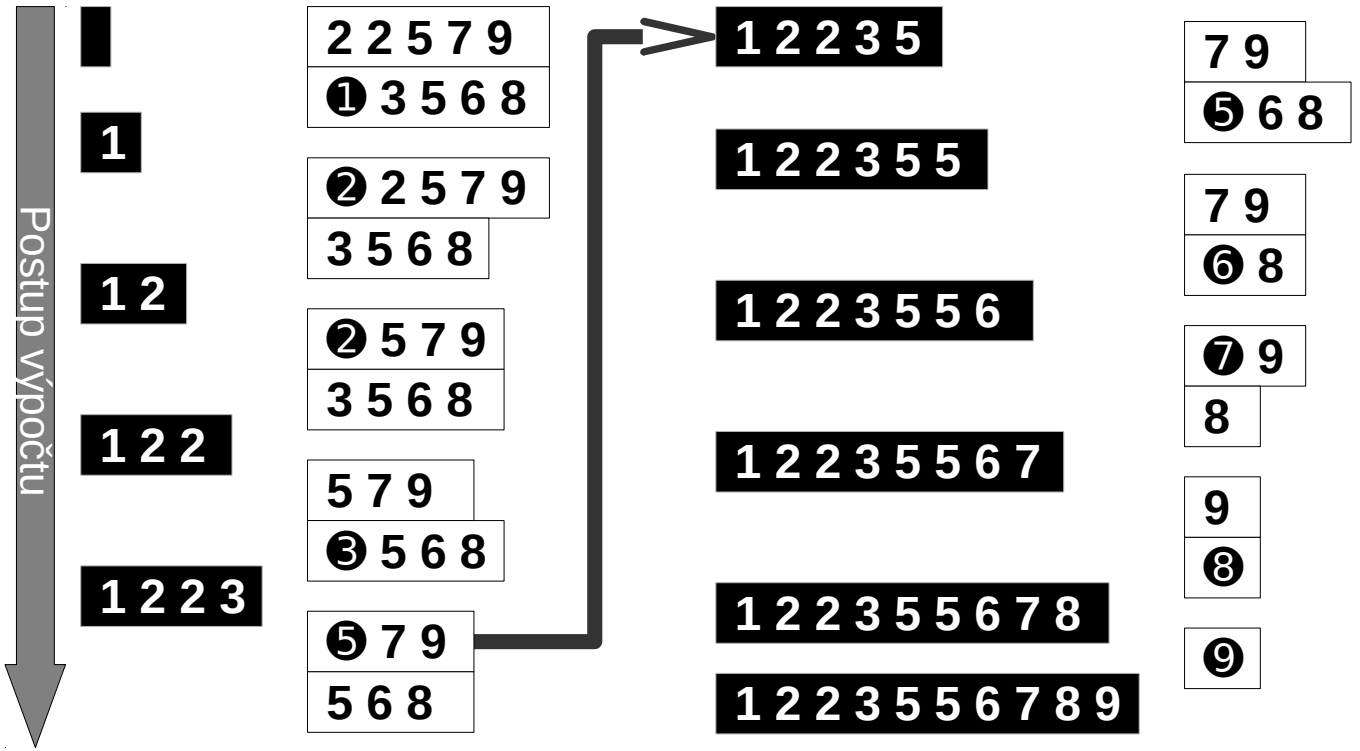
Příklad – scénář bublinkového řazení posloupnosti

Bublinkové řazení

Časová složitost:	**kvadratická**
Paměťová složitost:	**lineární**
Na místě?	**ano, může být**
Stabilita?	**může být**

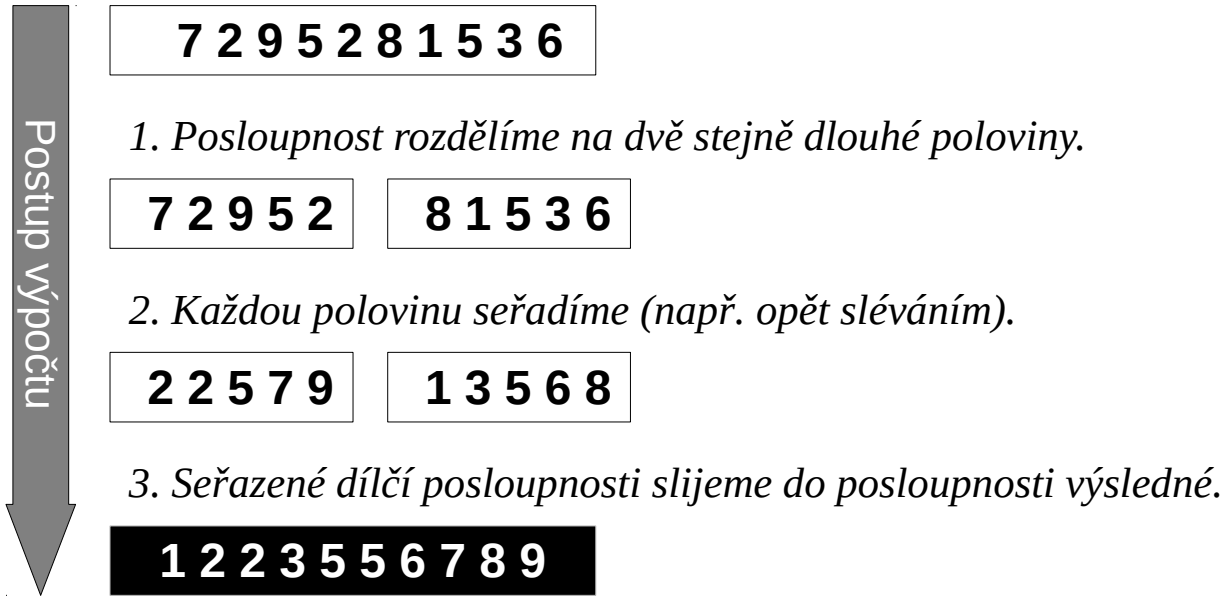
- Poznámky:
  - Časová a paměťová složitost vzhledem k počtu řazených prvků
  - Nestabilní řazení může být časově nebo paměťově efektivnější

Řazení sléváním



Příklad – scénář slití dvou seřazených posloupností

Řazení sléváním



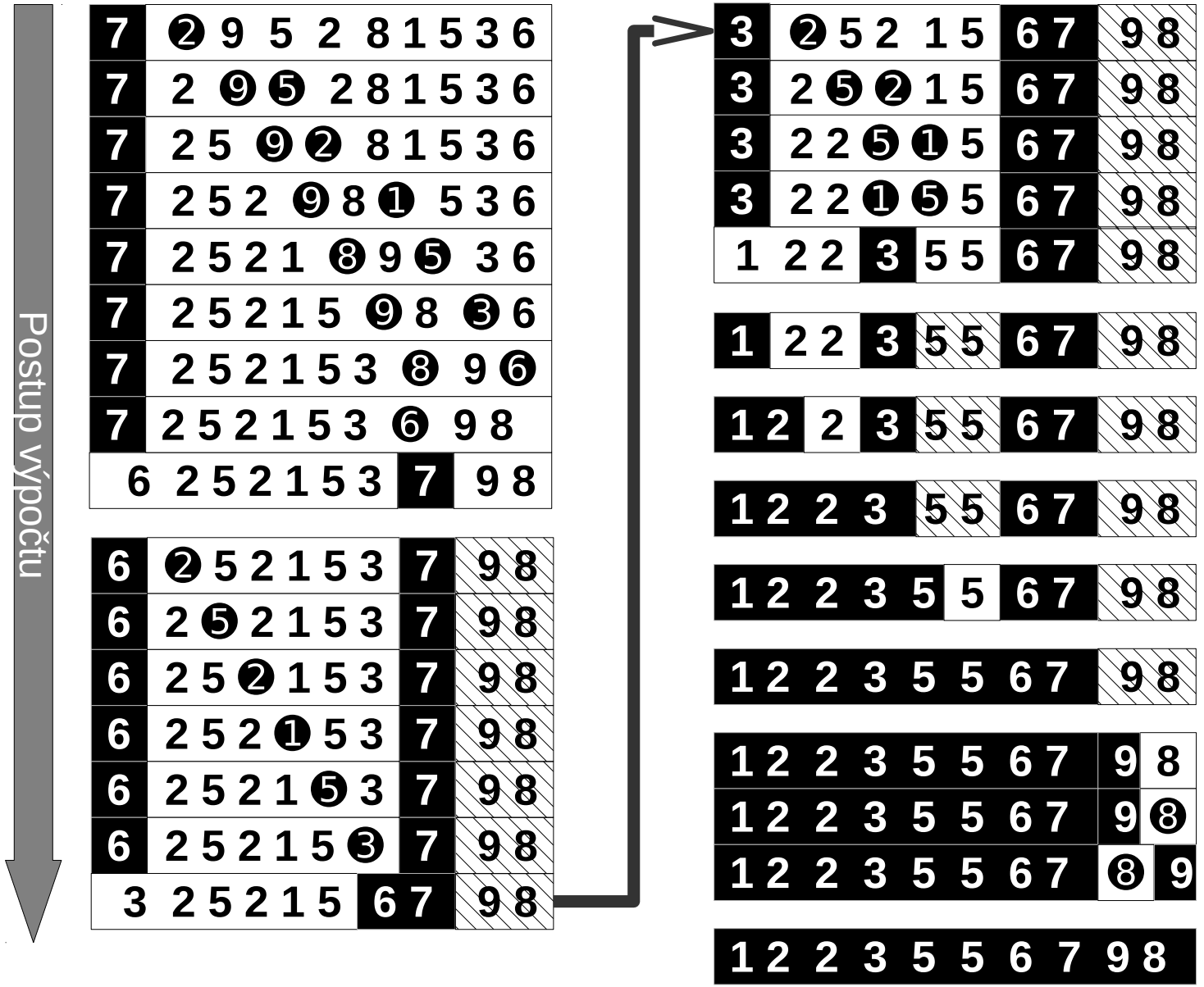
Příklad – scénář seřazení posloupnosti sléváním

Řazení sléváním

Časová složitost:	**lineárně logaritmická**
Paměťová složitost:	**lineární nebo horší**
Na místě?	**typicky ne, výjimečně ano**
Stabilita?	**ano, snadno**

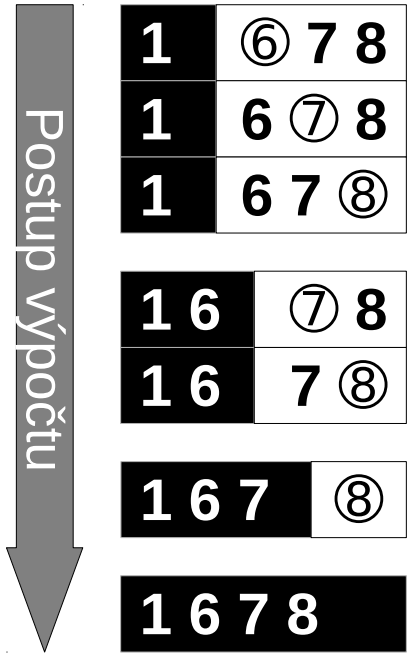
- Poznámky:
  - Časová a paměťová složitost vzhledem k počtu řazených prvků
  - Nestabilní řazení může být časově nebo paměťově efektivnější

Řazení rozdělováním



Příklad – postupné rozdělování posloupnosti

Řazení rozdělováním



V příkladu vidíme již seřazenou posloupnost 4 prvků, kterou chceme ještě zpracovat quicksortem. Musíme ji třikrát rekurzivně rozdělit a při jednotlivých děleních musíme projít (a porovnat s pivotem) postupně 3, 2 a 1 prvek, tj. v průměru 2 prvky při jednom dělení. Posloupnost N prvků bychom v případě, že jsou předem seřazené, museli dělit celkem N – 1 krát, hloubka rekurze by při tom dosáhla N – 1 a při jednom dělení bychom v průměru museli projít ½ (N – 1) prvků. Na celé degenerované řazení tedy potřebujeme celkem (N – 1) · ½ (N – 1) = ½ (N – 1)² kroků, tj. po zanedbání nedůležitých konstant bude časová složitost degenerovaného quicksortu *kvadratická*.

Příklad degenerovaného řazení čtyřprvkové posloupnosti quicksortem

Řazení rozdělováním – odhad časové složitosti

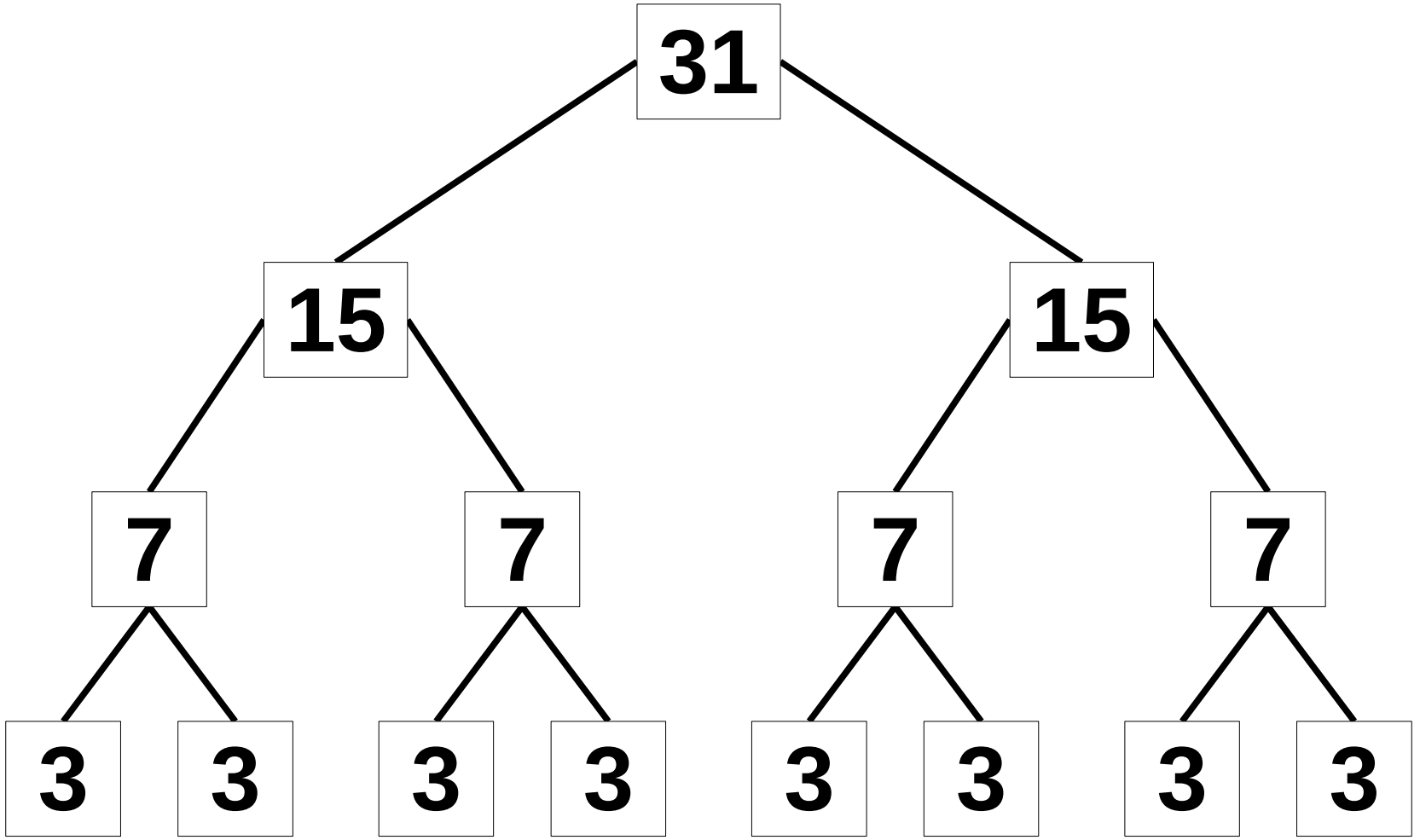


Schéma optimálního dělení posloupnosti quicksortem

Řazení rozdělováním

Časová složitost:	**až kvadratická**
Paměťová složitost:	**lineární**
Na místě?	**ano, lze**
Stabilita?	**obvykle ne**

- Poznámky:
  - Časová a paměťová složitost vzhledem k počtu řazených prvků

Řazení rozdělováním

- **Výpočet mediánu**
  - **medián** je prostřední prvek v seřazené posloupnosti
  - *quicksort* **nemusí seřadit** celou posloupnost (medián se nachází jen v jedné ze dvou dílčích posloupností pod nebo nad pivotem)
  - Rekurzi lze převést na časově efektivnější **cyklus** (bez použití zvláštního zásobníku)

Counting sort

Posloupnost:

7	2	9	5	2	8	1	5	3	6

Meze: min = 1, max = 9 (to se přepočítá na indexy 0 až 8).

Pole přihrádek:

1	2	3	4	5	6	7	8	9

klíče

0	1	2	3	4	5	6	7	8

indexy se počítají jako klíče – min

1	2	1	0	2	1	1	1	1

počty prvků pro jednotlivé klíče

0	2	3	3	5	6	7	8	9

indexy konců přihrádek

Pole výsledků:

0	1	2	3	4	5	6	7	8	9

indexy

1	2	2	3	5	5	6	7	8	9

výsledek řazení: seřazené pole

Příklad – scénář řazení posloupnosti countingsortem

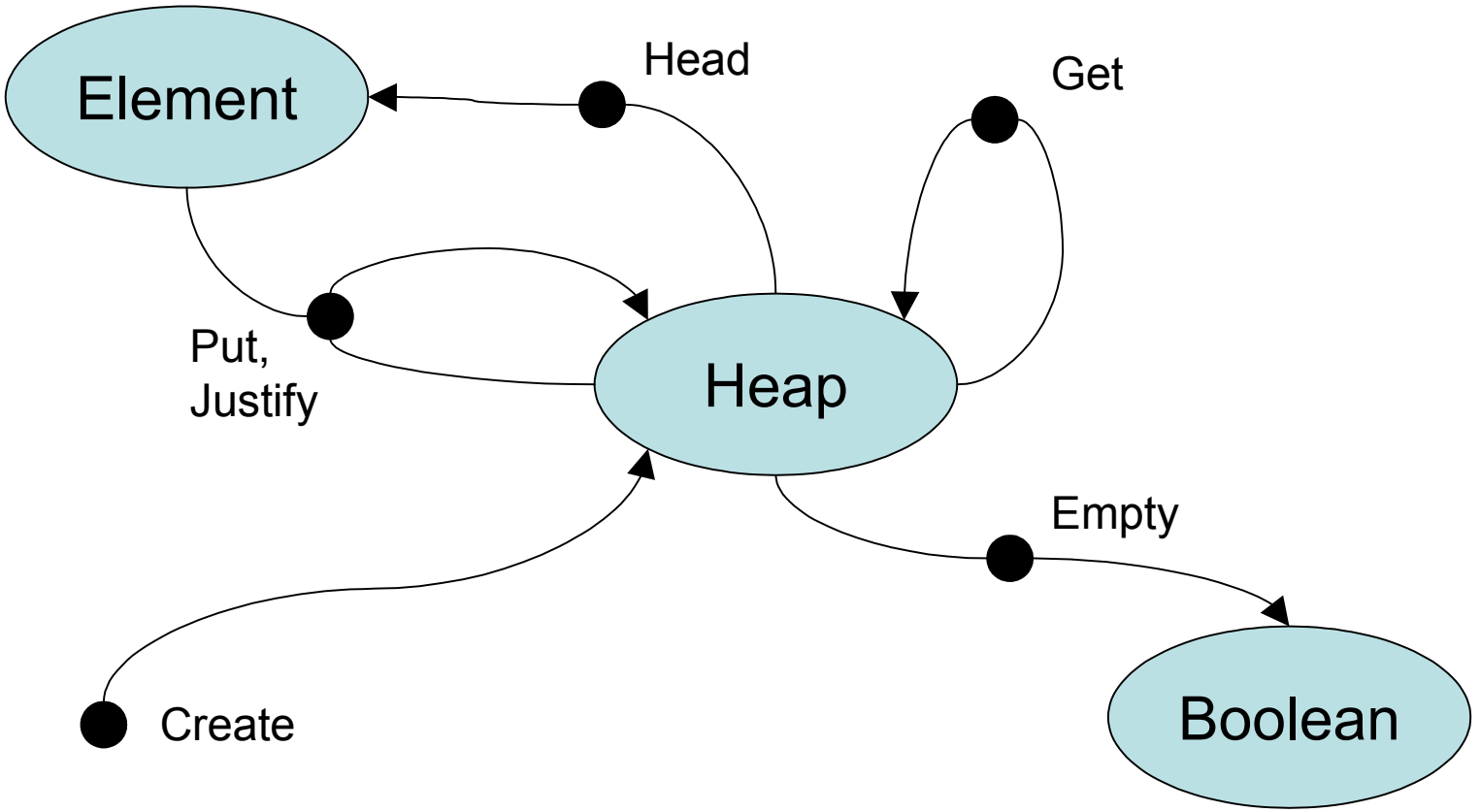
Counting sort

Časová složitost:	**lineární**
Paměťová složitost:	**lineární, ale POZOR!**
Na místě?	**ne**
Stabilita?	**typicky ano**

- Poznámky:
  - Časová a paměťová složitost závisí na počtu prvků + rozsahu hodnot klíčů

Řazení haldou

- Shoduje se algoritmem přímého výběru, ale
  - Neseřazená posloupnost je umístěna do haldy a vybírá se z ní efektivně (s logaritmickou složitostí)



Signatura datového typu *halda*

Řazení haldou

Časová složitost:	**lineárně logaritmická**
Paměťová složitost:	**lineární**
Na místě?	**ano, může být**
Stabilita?	**ne**

- Poznámky:
  - Časová a paměťová složitost závisí na počtu řazených prvků

Algoritmy řazení

- Algoritmy založené na
  - porovnávání prvků
    - Jednoduché algoritmy
      - se snadno programují
      - časově nejsou efektivní (kvadratická složitost)
    - Efektivní algoritmy (mergesort, heapsort)
      - jsou složitější na naprogramování
      - časově jsou efektivní (lineárně-logaritmická složitost)
    - Časová složitost algoritmu založeného na porovnávání nemůže být lepší než lineárně logaritmická
  - přímém adresování (indexování)
    - Časová i paměťová složitost může být lineární vzhledem k počtu prvků a rozsahu klíčů
      - Vícestupňové přihrádkové třídění je efektivnější prostorově, ale náročnější časově

Konec

Tato prezentace patří k učebnici *Algoritmy a datové struktury objektově* od Ivana Ryanta. Obsahuje texty a obrázky z této učebnice.

RYANT, Ivan. *Algoritmy a datové struktury objektově.* Praha: Ivan Ryant, 2017. ISBN 978-80-270-1660-0