Řadící algoritmy I.

Metodický koncept k efektivní podpoře klíčových odborných kompetencí s využitím cizího jazyka ATCZ62 - CLIL jako výuková strategie na vysoké škole

Interreg
EVROPSKÁ UNIE
Rakousko-Česká republika
Evropský fond pro regionální rozvoj

Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj

VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Řadící algoritmy

- Seřazení daného souboru dat do specifického pořadí (abecedně, podle čísel)
- Dvojice klíč-hodnota – řadí se podle klíče, na hodnotu není brán zřetel
- Rozdělení
  - Stabilní vs. nestabilní – zachovává pořadí položek se stejným klíčem
  - Podle typu řazení
    - Výběrem
    - Vkládáním
    - Záměnou
    - Slučováním

Interreg
Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Řadící algoritmy

- Nejznámější algoritmy
  - Buble sort — Bublinkové řazení
  - Heap sort — Řazení haldou
  - Insertion sort — Řazení vkládáním
  - Merge sort — Řazení slučováním
  - Quicksort — Rychlé řazení
  - Selection sort — Řazení výběrem
- Další algoritmy založené na jiném principu
  - Bucket sort — Přihrádkové řazení
  - Radix sort — Třídění podle základu
  - Counting sort — Řazení počítáním četnosti

Interreg Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Bubble sort

- Implementačně jednoduchý algoritmus
- Opakovaně prochází seznam a porovnává dva sousední prvky
- Univerzální, pracuje lokálně (nepotřebuje dodatečnou paměť)
- Prvky s nejvyšší hodnotou probublávají na konec seznamu

Interreg
Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Bubble sort

- Algoritmus:

```
opakuj
        bylo seřazeno := ano;
        pro i od 1 do (počet prvků - 1) opakuj:
                pokud seznam[i] > seznam[i + 1]
                        zaměň(seznam[i], seznam[i + 1])
                                bylo seřazeno := ne;
dokud není bylo seřazeno == ano;
```

Heap sort

- Jeden z nejlepších obecných algoritmů založených na porovnávání prvků
- Není stabilní
- Využívá datovou strukturu halda a její vlastnosti

Interreg
Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Heap sort

- Postup:
  - Vytvoření hlady z pole
  - Nejmenší prvek je kořenem – umístíme na první místo v poli a kořen odebereme
  - downheap() – obnovení haldy podle pravidel
  - Opakujeme odebírání kořene a obnovení haldy, dokud halda není prázdná

Interreg
Rakousko-Česká republika
Evropský fond pro regionální rozvoj
Europäische Union
Evropská unie
Europäischer Fonds für regionale Entwicklung
Evropský fond pro regionální rozvoj
VŠTE
UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Insertion sort

- Jednoduchá implementace
- Efektivní na malých množinách
- Efektivní na částečně seřazených množinách
- Stabilní
- Dokáže řadit data tak, jak přicházejí na vstup
- Postup:
  1. Posloupnost rozdělíme na seřazenou a neseřazenou tak, že seřazená obsahuje první prvek posloupnosti
  2. Z neseřazené části vybereme první prvek a zařadíme jej na správné místo v seřazené posloupnosti
  3. Prvky v seřazené posloupnosti posuneme o jednu pozici doprava
  4. Seřazenou část zvětšíme o jeden prvek. Naopak neseřazenou část o jeden prvek zleva zmenšíme
  5. Kroky 2–5 aplikujeme až do úplného seřazení neseřazené části

Insertion sort

- Algoritmus:

```
insertionSort(A)
        pro i od 1 do počtu prvků opakuj:
                hodnota = A[i];
                j = i-1;
                pokud j >= 0 a zároveň A[j] > hodnota opakuj:
                        A[j+1] = A[j];
                        A[j] = hodnota;
                        j--;
```