Programování

Řadící algoritmy

Myšlenka třídících algoritmů

- Práce s velkým objemem dat
- Data jsou na vstupu neuspořádaná
- Netříděna data nelze snadno popsat

- Seřazená data lze jednodušeji organizovat a analyzovat
- Výsledný algoritmus je optimalizovaný

Dělení řadících algoritmů

- Porovnávací – Neporovnávací
- Stabilní – Nestabilní
- In situ – Ex situ
  - Paměťová náročnost
- Podle složitosti výpočtů

Bublinkové řazení

- Univerzální řadící algoritmus
- Velmi neefektivní řešení
- Využívá principu prohazování sousedních hodnot

- Stabilní, in situ
- Kvadratická složitost
- Konstantní paměť navíc

Bublinkové řazení

```
public void BubbleSortAlgorithm(int[] pole)
{
    int n = pole.Length;
    for (int i = 0; i <= pole.Length; i++)
    {
        for (int j = 0; j < pole.Length - i - 1; j++)
        {
            if (pole[j] > pole[j + 1])
            {
                /// prohození prvků
                int tmp = pole[j];
                pole[j] = pole[j + 1];
                pole[j + 1] = tmp;
            }
        }
    }
}
```

Řazení vkládáním

- Insertion sort
- Jednoduchá implementace
- Efektivní na malých a částečně seřazených množinách

- Kvadratická složitost
- In situ, stabilní
- Konstantní paměť navíc

Řazení vkládáním

```
public void InsertionSortAlgorithm(int[] pole)
{
    for (int i = 1; i < pole.Length; i++)
    {
        int pivot = pole[i];
        int j = i - 1;

        /// přesunutí setřízených prvků, které jsou větší než pivot
        while (j >= 0 && pole[j] > pivot)
        {
            pole[j + 1] = pole[j];
            j--;
        }
        pole[j + 1] = pivot;
    }
}
```

Řazení slučováním

- Merge sort
- Využívá principu rozděl a panuj

- Stabilní, ex situ
- Složitost **n log n**
- Navíc potřeba **log n** paměti

Řazení slučováním

```
public void MergeSortAlgorithm(int[] pole)
{
    if (pole.Length > 1)

        int stred = pole.Length / 2;
        int[] leve = new int[stred];
        int[] prava = new int[pole.Length - stred];
        // naplenění levého a pravého pole
        for (int i = 0; i < mid; i++)
        {
            leve[i] = pole[i];
        }
        for (int i = stred; i < pole.Length; i++)
        {
            prave[i - strd] = pole[i];
        }
        // Rekurzivní volání řazení nad levým a pravým polem
        MergeSortAlgorithm(left);
        MergeSortAlgorithm(right);
```

```
    // Sloučení seřazených polí
    int i = 0, j = 0, k = 0;
    while (i < leve.Length && j < prave.Length)
    {
        if (leve[i] < prave[j])
        {
            pole[k] = leve[i];
            i++;
        }
        else
        {
            pole[k] = prave[j];
            j++;
        }
        k++;
    }
         while (i < leve.Length)
         {
             pole[k] = leve[i];
             i++;
             k++;
         }
         while (j < prave.Length)
         {
             pole[k] = leve[j];
             j++;
             k++;
         }
     }
}
```

Rychlé řazení

- Quick sort
- Jeden z nejrychlejších algoritmů používaný pro řazení
- Rovněž využívá metody rozděl a panuj

- Nestabilní, ex situ
- Složitost **n log n**, v nejhorším možném případě kvadratická
- Navíc potřeba **log n** paměti

- Zásadní je volba **pivota** posloupnosti

Rychlé řazení

```
void QuickSort(int[] pole, int leva, int prava)
{
    int pivot = pole[(leva + prava) / 2];
    int levyIndex, pravyIndex, tmp;
    levyIndex = leva;
    pravyIndex = prava;
    do
    {
        while (pole[levyIndex] < pivot && levyIndex < prava)
            levyIndex++;
        while (pole[pravyIndex] > pivot && pravyIndex > leva)
            pravyIndex--;

        if (levyIndex <= pravyIndex)
        {
            tmp = pole[levyIndex];
            pole[levyIndex++] = pole[pravyIndex];
            pole[pravyIndex--] = tmp;
        }
    } while (levyIndex < pravyIndex);

    if (pravyIndex > leva)
        QuickSort(pole, leva, pravyIndex);
    if (levyIndex < prava)
        QuickSort(pole, levyIndex, prava);
}
```

Další řadící algoritmy

- Řazení haldou (heap sort)
- Kyblíkové řazení (bucket sort)
- Výběrové řazení (Selection sort)
- Shellovo řazení (Shell sort)
- Číslicové řazení (Radix sort)