Jednoduché řadicí algoritmy (řazení primitivních datových typů, problematika řazení objektů)

Jednoduché řadicí algoritmy

Každý algoritmus pracuje na trochu jiném principu a rychlost jejich vykonání se různí.

Bubble sort (bublinkové řazení)

V bubble sortu se postupně systematicky porovnávají dvojice sousedních prvků, které se vzájemně prohodí v momentě, když menší číslo následuje po větším. Z hlediska naprogramování je bubble sort nejjednodušším algoritmem, pro praktické účely je však neefektivní. Částečně seřazené pole zpracuje rychleji než neseřazené. Používá se hlavně pro výukové účely.

```
public static void bubbleSort(int[] array){
    for (int i = 0; i < array.length - 1; i++) {
        for (int j = 0; j < array.length - i - 1; j++) {
            if(array[j] < array[j+1]){
                int tmp = array[j];
                array[j] = array[j+1];
                array[j+1] = tmp;
            }
        }
    }
}
```

Select sort (třídění přímým výběrem)

Principem tohoto řazení je výběr mezního prvku (maxima nebo minima) z tříděné posloupnosti a jeho záměna s prvním (nebo posledním) prvkem. V dalším kroku se třídí pole o n-1 prvcích a je opakován tentýž průběh. Takto je postupováno až do úplného seřazení posloupnosti. Získaná posloupnost bude seřazená vzestupně v případě, že budeme vybírat z nesetříděné posloupnosti vždy nejmenší prvek a zařadíme ho na začátek posloupnosti, a sestupně v případě, že vybereme největší prvek. Setříděná část pole se při tomto řazení postupně zvětšuje, zatímco nesetříděná se postupně zmenšuje.

Nevýhodou tohoto algoritmu je, že nachází-li se na vstupu již částečně setříděná posloupnost, algoritmus takový fakt ignoruje a proběhne vždy maximální počet kroků.

Popis algoritmu v krocích:

1. v posloupnosti je nalezen nejmenší prvek a je vyměněn s prvkem na první pozici, čímž dojde k rozdělení posloupnosti na dvě části; setříděná část zatím obsahuje pouze jeden prvek, nesetříděná n-1
2. v nesetříděné části se opět najde nejmenší prvek a je vyměněn s prvkem na první pozici nesetříděné části, čímž dojde však zároveň k zařazení prvku do části setříděné
3. obsahuje-li nesetříděná část více než jeden prvek, pokračuje se, od bodu dva, jinak je třídění ukončeno

```
public static void SelectionSort(int[] array) {
    for (int i = 0; i < array.Length - 1; i++) {
```

```
        int maxIndex = i;
        for (int j = i + 1; j < array.Length; j++) {
            if (array[j] > array[maxIndex]) maxIndex = j;
        }
        int tmp = array[i];
        array[i] = array[maxIndex];
        array[maxIndex] = tmp;
    }
}
```

Insert sort (přímé vkládání)

Pole se dělí na setříděný a nesetříděný úsek. Na začátku je setříděný úsek tvořen pouze prvním prvkem pole. První prvek nesetříděného úseku se vždy zařadí podle velikosti do setříděného úseku, čímž se úsek nesetříděný zleva zkrátí o jeden prvek.

```
public static void insertSort(int[] array) {
    for (int i = 0; i < array.length - 1; i++) {
        int j = i + 1;
        int tmp = array[j];
        while (j > 0 && tmp > array[j-1]) {
            array[j] = array[j-1];
            j--;
        }
        array[j] = tmp;
    }
}
```

Řazení primitivních datových typů

V programovacím jazyce je obvykle již implementováno, jak posuzovat hodnoty primitivních datových typů. Jazyk tak při řazení ví, že 5 je menší než 14, 0 větší než -22 či že false (0) má nižší hodnotu než true (1). Pro řazení pole s primitiv. datovými typy slouží metoda Arrays.sort(pole).

Problematika řazení objektů

Hlavním problémem u řazení objektů je ten, že program sám o sobě neví, podle jaké vlastnosti takového objektu by měl kolekci objektů seřadit. Existují pak dvě možnosti, jak objekty řadit.

1. **přirozené řazení** – je řazeno pomocí stejných metod jako neprimitivní datové typy, tedy sort() a binarySearch(); porovnání objektů musí být definováno ve třídě těchto objektů
2. **absolutní řazení** – je tu možnost si zvolit, podle čeho se má kolekce objektů řadit, metodě sort() předáme druhý parametr, který vlastnost, podle které řadit, určuje

Přirozené řazení

Třída implementuje rozhraní java.lang.Comparable s metodou compareTo(Trida t). Tato metoda je pak vnitřně při snaze řadit objekty volána a každé dva objekty jsou postupně porovnávány, až jsou seřazeny. Funkce vrací vždy -1, 0 nebo 1 podle toho, zda má být porovnávaný objekt chápán jako *nižší*, *rovný* nebo *vyšší*.

Absolutní řazení

Třída implementuje rozhraní java.lang.Comparator, které má dvě metody. První z nich je metoda `equals(Object o)` – dědí ji automaticky každá třída ze třídy Object, není tedy nutno ji definovat, druhou metodou je pak `int compare(Trida t1, Trida t2)`.