Algoritmizace a programování

Vyhledávání, vkládání, odstraňování

Vyhledání hodnoty v nesetříděném poli

Vyhledání hodnoty v setříděném poli

Odstranění hodnoty z pole

Vkládání hodnoty do pole

Verze pro akademický rok 2012/2013

Základní algoritmy zpracování sady dat

- Sada dat
  - záznamy o studentech fakulty (jméno, příjmení, datum narození, datum přijetí, průběh studia)
  - kniha jízd – datum, SPZ vozidla, počet ujetých kilometrů, čerpání phm atd.
  - objednávky firmy – objednatel, položky objednávky
- Základní operace, které potřebujeme provádět s obdobnou sadou dat
  - přidat záznam (studenta)
  - odstranit záznam (například v případě chybného založení)
  - modifikovat aktuálně zaznamenané údaje
  - vyhledat konkrétní záznam (na základě stanoveného kriteria)
  - seřadit záznamy dle požadovaného kriteria (dle jména, dle studijních výsledků)
- Uchování záznamů aktuálně zpracovávaných programem
  - soubor
  - pole
  - další struktury jako je seznam
- Ukažme si jednotlivé operace **vyhledávání, přidávání** a **odstraňování** za předpokladu, že jednotlivé záznamy jsou uloženy v poli – pro jednoduchost budeme prozatím pracovat pouze s polem hodnot primitivního datového typu

Úloha vyhledávání hodnoty v poli

- Vyhledání
  - Vyhledání konkrétní hodnoty `h` v poli `a`, které obsahuje `n` hodnot, předpokládáme, že obecně `n ≤ a.length`
  - Výsledkem má být index od `0` do `n-1` jako pozice výskytu hledané hodnoty, popřípadě hodnota `-1`, pokud se daný prvek v poli nevyskytuje
  - Při řešení konkrétní úlohy je nutné uvažovat, zda se hledaná hodnota může v poli vyskytovat vícekrát či nikoli; pokud ano, zda se má vyhledat první výskyt, nebo první výskyt a následně další
  - Variantně, pokud požadujeme pouze zjistit, zda se daná hodnota v poli vyskytuje či nikoli, by mohla být výsledkem hodnota `true` nebo `false`

- **Nesetříděné pole**
  - Aplikujeme algoritmus **sekvenční vyhledávání** – tj. postupné procházení a testování všech hodnot v poli od nejnižšího indexu

- **Setříděné pole – vzestupné uspořádání hodnot**
  - **Sekvenční vyhledávání** – ukončení hledání, kdy nalezneme hodnotu ≥ hledané
  - **Binární vyhledávání** (*metoda půlení intervalu*) – efektivnější využití toho, že hodnoty jsou vzestupně uspořádané

Vyhledávání v nesetříděném poli

```
// vyhledani prvniho vyskytu
p = -1;
for (int i = 0; (p == -1) && (i < n); i++){
  if (a[i] == h)
    p = i;
}
// (p = -1) => hodnota nenalezena

// vyhledani dalsiho vyskytu
d = p;
for (int i = p+1; (d == p) && (i < n); i++){
  if (a[i] == h)
    d = i;
}
// (d = p) => dalsi hodnota nenalezena
```

- Pole a s n hodnotami
- Hodnoty od indexu 0 do indexu n-1
- Hledaná hodnota je v proměnné h
- Výsledná pozice prvního výskytu hledané hodnoty je v proměnné p
- Pozice dalšího výskytu hledané hodnoty je v proměnné d za předpokladu, že pozice předchozího výskytu je v proměnné p

Vyhledávání v setříděném poli

```
// vyhledani vyskytu
pos = -1;
d = 0;
h = n-1;
do {
  p = (d + h) / 2;
  if (a[p] == cislo) pos = p;
  else if (a[p] > cislo) h = p - 1;
  else d = p + 1;
} while ((pos == -1) && (d <= h));
```

- Pole a s n hodnotami
- Hodnoty od indexu 0 do indexu `n-1`
- Hledaná hodnota je v proměnné `cislo`
- Výsledná pozice výskytu hledané hodnoty je v proměnné `pos`

- Pokud `pos` $\neq$ `-1`, potom `pos` obsahuje pozici hledaného prvku
- Pokud `pos = -1`, potom d obsahuje pozici, na kterou by patřila hledaná hodnota v setříděné posloupnosti

Úloha odstranění hodnoty z pole

- Předpoklady
  - Mějme pole `a`, které obsahuje `n` hodnot, délka pole nemusí být nutně shodná s velikostí alokované paměti, tedy obecně $n \leq a.length$
  - V rámci tohoto pole byla nalezena pozice (index) prvku `pos`, který se má odstranit
  - Příslušný index by měl být v rozsahu od `0` do `n-1`

- Požadavky na řešení
  - Naším úkolem je odstranit prvek z pole tak, aby
    - prvek se v poli nevyskytoval
    - pole bylo o jednu položku kratší

- Řešení
  - Pokud **nezáleží na pořadí hodnot v poli** – kopírování posledního prvku na danou pozici `pos`, aktualizace délky
  - **Pořadí hodnot v poli je důležité** – od identifikované pozice `pos` až do konce pole provedeme zkopírování každého prvku na pozici předcházející, aktualizace délky pole

Odstranění hodnoty z pole

```
// odstraneni prvku z usporadaneho pole

// pos

if ((pos >= 0) && (pos < n)) {
  for (int i = pos; i < n-1; i++){
    a[i] = a[i+1];
  }
  n--;
}
```

- Pole a s n hodnotami
- Hodnoty od indexu 0 do indexu `n-1`
- Hodnota, kterou požadujeme odstranit má index `pos`
- Po provedení operace obsahuje proměnná a v rozsahu od `0` do `n` modifikované pole

Úloha vložení hodnoty do pole

- Předpoklady
  - Mějme pole `a`, které obsahuje `n` hodnot, délka pole nemusí být nutně shodná s velikostí alokované paměti, tedy obecně `n ≤ a.length`
  - V rámci tohoto pole byla nalezena pozice (index) `pos`, na kterou je nutné vložit novou hodnotu `h`
  - Příslušný index `pos` je v rozsahu od `0` do `n`

- Požadavky na řešení
  - Naším úkolem je přidat novou hodnotu do pole tak, aby
    - hodnota byla vložena na požadovanou pozici
    - všechny stávající hodnoty zůstaly v poli zachovány
    - pole bylo o jednu položku delší

- Řešení
  - Při řešení zadané úlohy je nutné vzít v úvahu kapacitu aktuálně alokované paměti pro dané pole – operaci provedeme ve dvou krocích
    - Zvětšíme aktuální délku pole
    - Od identifikované pozice až do konce pole provedeme zkopírování každého prvku na pozici následující

Vložení hodnoty do pole

```
// vlozeni noveho prvku do pole
// pos
 final int DELTA = 10;

if ((pos >= 0) && (pos <= n)) {
  if (n == a.length){
    b = new int[a.length + DELTA];
    for (int i = 0; i < n; i++) b[i] = a[i];
    a = b; b = null;
  }
  n++;
  for (int i = n-1; i > pos; i--){
    a[i] = a[i-1];
  }
  a[pos] = h;
}
```

- Pole a s n hodnotami
- Hodnoty od indexu 0 do indexu `n-1`
- Novou hodnotu h vkládáme na pozici `pos`
- Po provedení operace obsahuje proměnná a v rozsahu indexů od `0` do `n-1` modifikované pole

Kopírování pole

- Metoda třídy systém pro kopírování části pole do jiného pole

```
System.arraycopy(a, srcPos, b, destPos, length);
```

- Výsledek stejný jako

```
for (int i = 0; i < length; i++)
     b[destPos + i] = a[srcPos + i]
```

- Efektivita systémové funkce vyšší

```
b = new int[a.length + DELTA];
System.arraycopy(a, 0, b, 0, pos);
b[pos] = h;
System.arraycopy(a, pos, b, pos+1, n-pos);
a = b; b = null;
n++;
```