Algoritmizace a programování

Algoritmy řazení dat

Problematika řazení dat v poli

Řazení přímým výběrem

Řazení přímým vkládáním

Řazení přímou výměnou

Verze pro akademický rok 2011/2012

Úloha řazení (třídění)

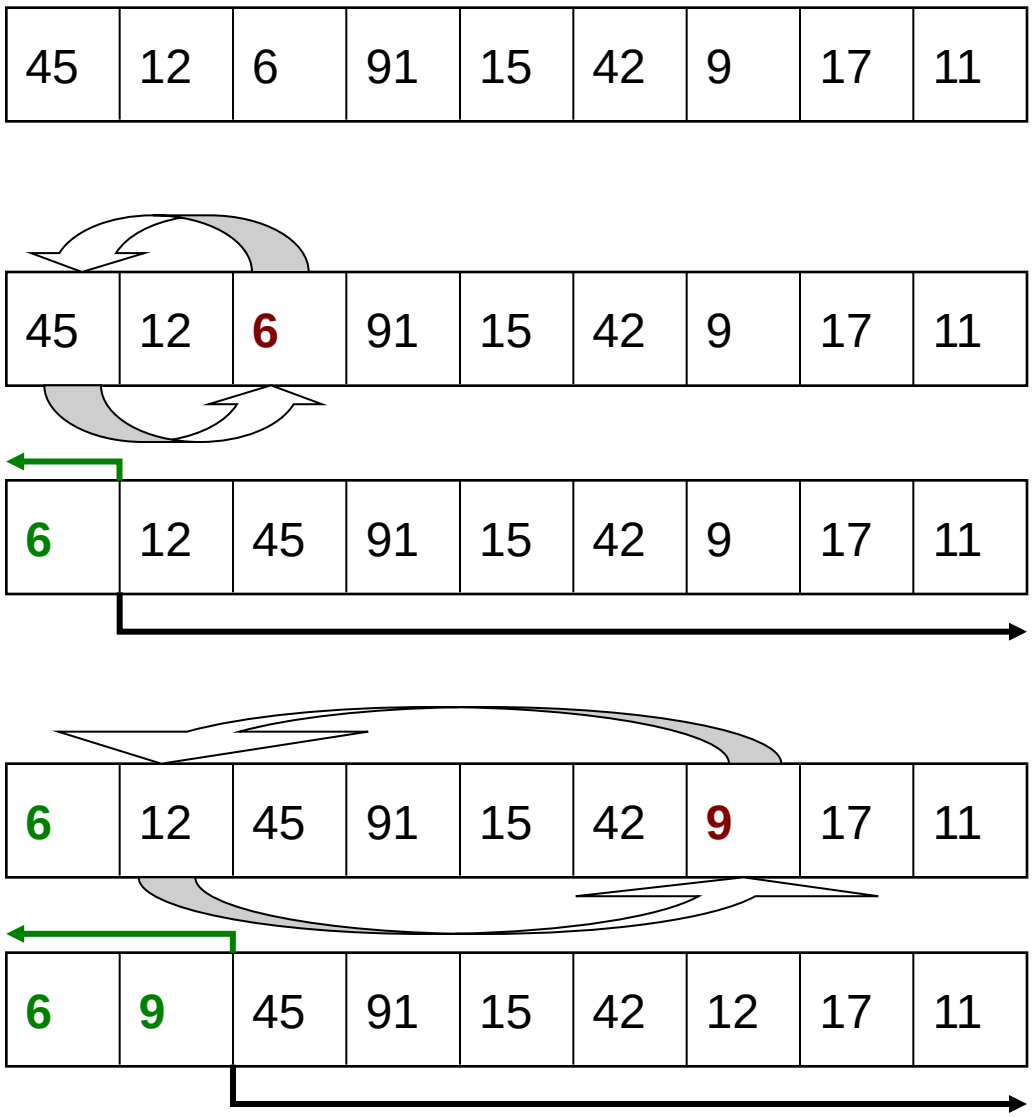
- Pole hodnot libovolného typu
- Pro hodnoty daného typu je definována relace „uspořádání“ s operací $\leq$ (resp. >)
  - Pole čísel
  - Pole komplexních čísel
  - Pole textových řetězců
  - Pole bodů
  - Pole obsahující informace o studentech fakulty

- Naším úkolem je uspořádat hodnoty v tomto poli (od indexu 0 až do indexu *n*) tak, aby pro každé dvě sousední hodnoty a_i, a_{i+1} platilo $a_i \leq a_{i+1}$
  - Vzestupné uspořádání hodnot

- Různé metody řazení – přímé metody, nepřímé metody
- Uvedeme si tři různé algoritmy – všechny dále použité algoritmy jsou přímé

- Pro absolvování zkoušky z předmětu ALP je mimo jiné nutná znalost všech tří dále uvedených algoritmů a schopnost jejich aplikace pro uspořádání daných hodnot (například uspořádání bodů dle vzrůstající vzdálenosti od počátku)

Řazení přímým výběrem



- Pole číselných hodnot
- V prvním kroku hledáme prvek na první pozici
- Na první pozici patří minimální prvek z celého rozsahu od indexu 0 do n-1
  - Vyhledáme minimální prvek a zajistíme jeho přesun na první pozici (výměnou dvou prvků pole)
- Po prvním kroku
- Ve druhém kroku hledáme prvek na druhou pozici – minimální prvek od druhého do n-tého prvku v poli

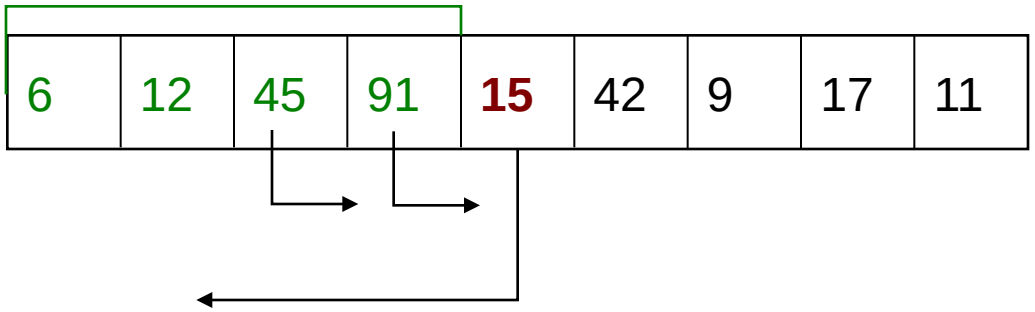
Řazení přímým výběrem

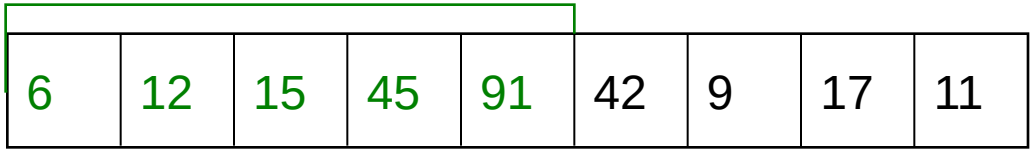
```
...

for (int i = 0; i < n-1; i++){
  imin = i;
  for (int j = i+1; j < n; j++) {
    if (a[j] < a[imin]) imin = j;
  }
  if (i != imin) {
    pom = a[i];
    a[i] = a[imin];
    a[imin] = pom;
  }
}
```

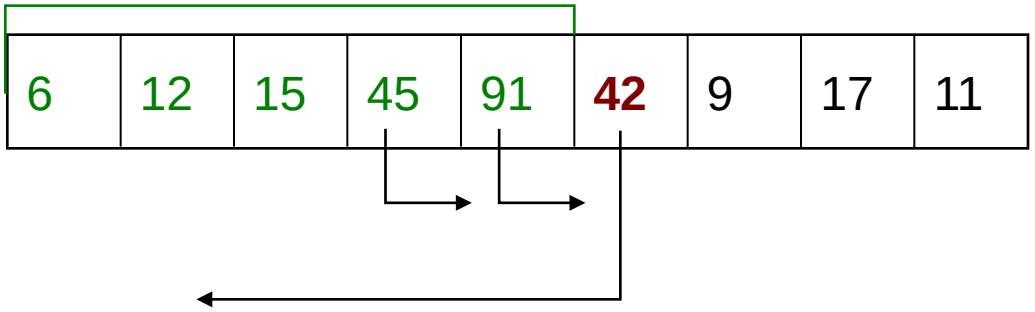
- Pole a hodnot primitivního datového typu
- V poli je n hodnot
- Indexy od `0` do `n-1`
- Pomocná proměnná `pom` je stejného datového typu jako položky pole

- **Přímý výběr** – postupně vybíráme minimální prvek v daném rozsahu pole a přesouváme jej na správné místo

Řazení přímým vkládáním







- Každý prvek se postupně vkládáme do zleva vzestupně setříděné posloupnosti
- i-tý krok
- Prvek a_i vkládáme do setříděné posloupnosti a_0 .. a_{i-1}
  - každý větší prvek odsuneme vpravo
  - pokud nalezneme prvek, který je menší nebo roven vkládané hodnotě – vložíme hodnotu **za** něj
- Tento postup aplikujeme pro všechny prvky posloupnosti

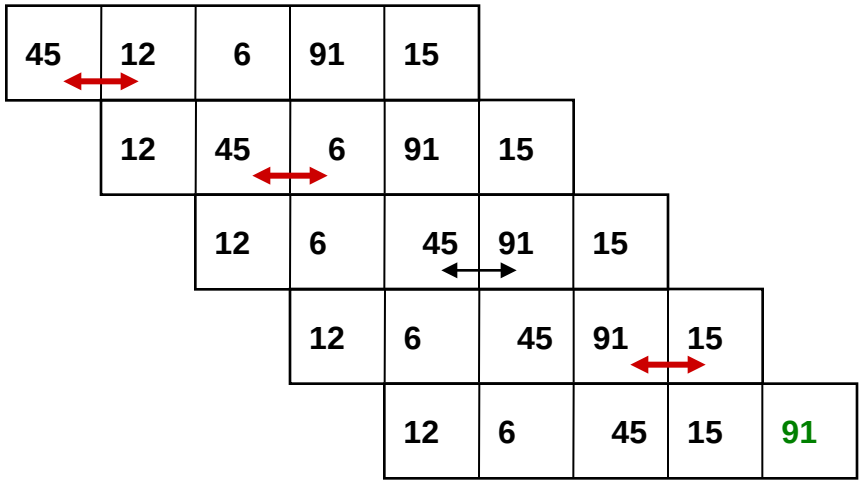
Řazení přímým vkládáním

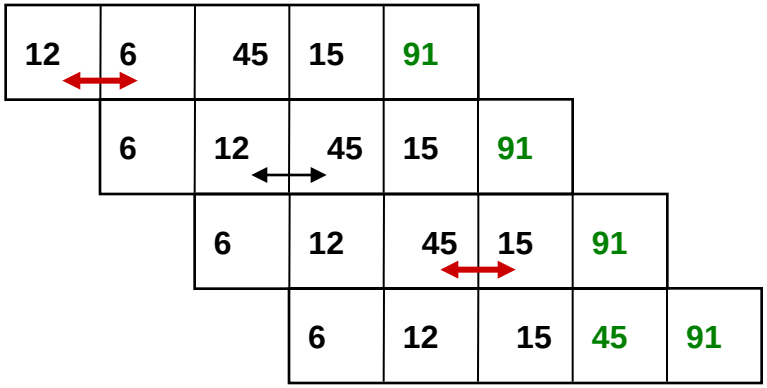
```
...

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

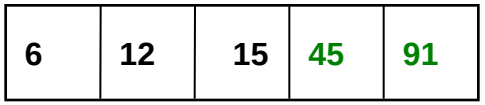
- Pole a hodnot primitivního datového typu
- V poli je n hodnot
- Indexy od `0` do `n-1`
- Pomocná proměnná `pom` je stejného datového typu jako položky pole

- **Přímé vkládání** – prvky postupně vkládáme do vzestupně setříděné podposloupnosti vytvářené průběžně od indexu 0 v příslušném poli

Řazení přímou výměnou







...

- Projděme celé pole a porovnejme vždy dva sousední prvky – pokud neodpovídají uspořádání potom je navzájem vyměníme
- Výsledkem operace bude, že minimálně na poslední místo se dostane největší hodnota celého pole
- Udělejme totéž podruhé, nyní stačí projít pole do předposledního prvku
- Pokud uvedeným způsobem projdeme pole (n-1)x (s postupně snižující se horní hranicí indexu pole), potom výsledkem celé operace bude setřídění hodnot v poli

Řazení přímou výměnou

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

- Pole a hodnot primitivního datového typu
- V poli je n hodnot
- Indexy od `0` do `n-1`
- Pomocná proměnná `pom` je stejného datového typu jako položky pole

- **Přímá výměna** – postupně testujeme a vzájemně vyměňujeme sousední prvky v poli

Řazení přímou výměnou

- Uvedený algoritmus třídění přímou výměnou není nejvhodnější
- Obecně se jedná o nejméně efektivní algoritmus třídění hodnot v poli

- Využijeme přirozených vlastností tohoto algoritmu a provedeme drobné úpravy, které zlepší jeho efektivitu

- Při každém průchodu pole zaznamenáme
  - Zda nastala nějaká výměna
  - Kde k příslušné výměně došlo
- Při dalším průchodu
  - Další průchod provedeme pouze v případě, že při minulém (průchodu) došlo k výměně; v opačném případě je pole setříděné
  - Při dalším průchodu bude horní hranice indexu odvozena od místa poslední výměny při průchodu předešlém (od poslední výměny do konce je pole setříděné)

Řazení přímou výměnou

```
...
vymena = n-1;
while (vymena > 0) {
  kam = vymena;
  vymena = 0;
  for (int i = 0; i < kam; i++){
    if (a[i] > a[i+1]){
      pom = a[i];
      a[i] = a[i+1];
      a[i+1] = pom;
      vymena = i;
    }
  }
}
```

- Pole a hodnot primitivního datového typu
- V poli je n hodnot
- Indexy od `0` do `n-1`
- Pomocná proměnná `pom` je stejného datového typu jako položky pole

- **Přímá výměna** – postupně testujeme a vzájemně vyměňujeme sousední prvky v poli

- Takto upravený algoritmus využívá přirozených vlastností použité metody