Řazení, třídění dat

Řadící algoritmus

Řadící algoritmus

- Zajišťuje seřazení daného souboru dat podle specifikovaného pořadí
  - Nejčastěji dle velikosti nebo abecedně
  - Důležitá, často užívaná úloha – hromadné zpracování dat – vyhledávání jednodušší a hlavně rychlejší v seřazených datech
  - Načtená data musí být před řazením uložena !
- **Třídící algoritmus (jednodušší)**
  - Rozdělení vstupní množiny dat do několika skupin dle zadaného kritéria bez potřeby určení jejich vzájemného pořadí

Definice problému

- **Na vstupu** je posloupnost $S = (S_1, S_2, \ldots, S_n)$
- **cílem** je najít takovou posloupnost

$$S' = (S'_1, S'_2, \ldots, S'_n)$$

pro kterou platí dvě základní kritéria:

1. Tato posloupnost je seřazená:

$$S'_1 \leq S'_2 \leq \cdots \leq S'_n$$

2. Posloupnost S' je permutací původní posloupnosti S (obsahuje tedy stejná data, jen v jiném pořadí).

Nejdůležitější kritérium algoritmů

Časová složitost – funkce, která udává, kolik elementárních operací (kroků algoritmu) je třeba provést, aby se provedla úloha s určitým množstvím vstupních dat. (průměrně)

Malé množství dat – pomalejší algoritmus

Velké množství dat – rychlejší algoritmus

Skutečnou dobu výpočtu ovlivňuje i rychlost procesoru, použitý programovací jazyk, kvalita překladače apod., neovlivňuje časovou složitost

Další kritérium

- **Paměťová složitost** - funkce, která udává, jaký počet paměťových míst (jednoduchých proměnných) je potřeba k uskutečnění výpočtu pro velikost vstupních dat N

časová a paměťová složitost -> hlavní hledisko praktické použitelnosti algoritmu - ***efektivita výpočtu***

Rychlost algoritmu

- Velmi dobrá: NlogN
- Rozumně zvládnutelná : N, N^2, N^3..
- Nerozumně – k ničemu : exponenciální fce

Základní skupiny řadících algoritmů

- **Řazení výběrem**
  - V souboru se vždy najde nejmenší ze zbývajících položek a uloží na konec postupně budovaného seřazeného souboru.
- **Řazení vkládáním**
  - Ze souboru neseřazených dat se postupně bere položka po položce a vkládá se na správné místo v seřazeném souboru (zpočátku prázdném).
- **Řazení záměnou**
  - V souboru se vždy nalezne (nějakou metodou závislou na konkrétním algoritmu) nějaká dvojice prvků, která je ve špatném pořadí, a tyto prvky se navzájem zamění.
- **Řazení slučováním**
  - Vstupní soubor se rozdělí na části, které se (typicky rekurzivně) seřadí; výsledné seřazené části se poté sloučí takovým způsobem, aby i výsledek byl seřazený.

Nejběžnější algoritmy

- **SELECT SORT (řazení výběrem)**
- **BUBBLE SORT (bublinkové řazení)**
- **SHAKE SORT (řazení přetřásáním)**
- **MERGESORT (rozděl a panuj - slučováním)**
- **QUICKSORT (rychlé rekurzivní řazení)**
- **HEAPSORT (řazení haldou)**
- **SHELLSORT (řazení se snižujícím se přírůstkem)**

SELECT SORT
řazení přímým výběrem

- *Časová náročnost : $O(N^2)$*
- *Pro menší množství dat*
- *Velmi jednoduchá metoda*

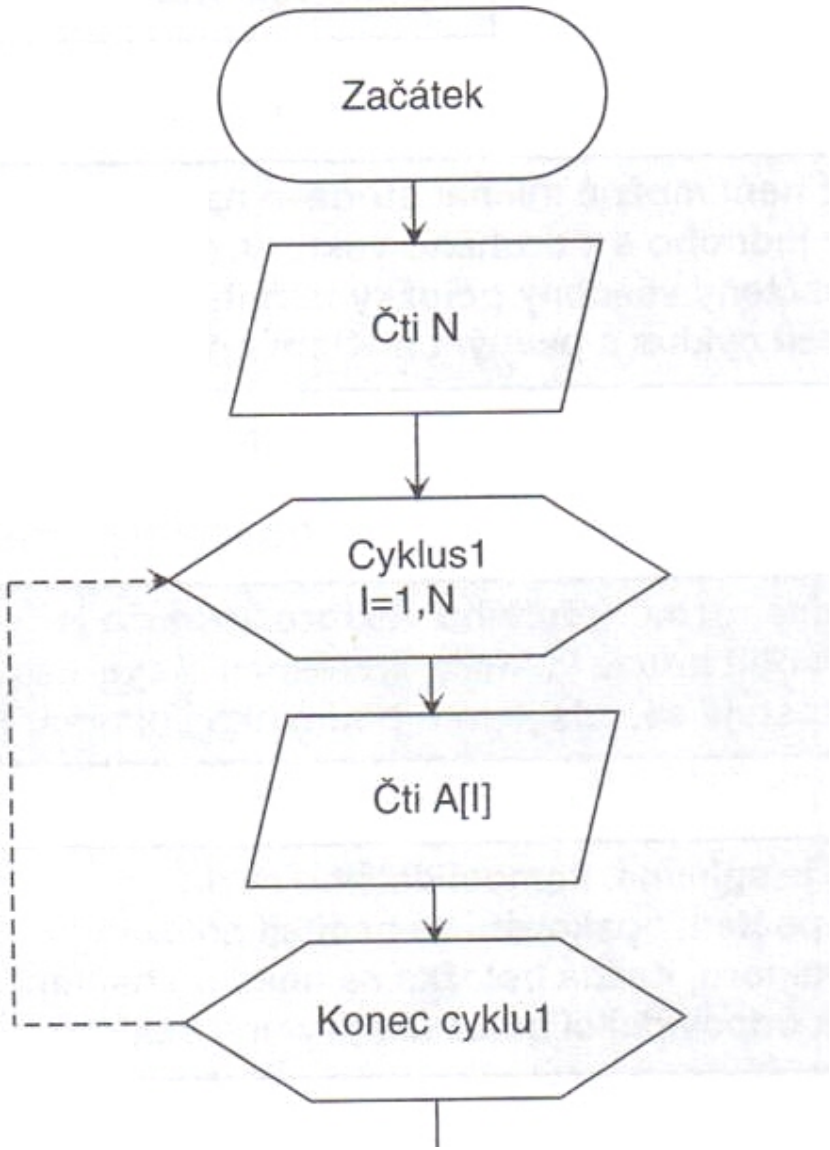
1. Načteme všech n čísel do pole A – A[N]
2. Najdeme minimum v posloupnosti (vzestupné řazení)
3. Zaměníme minimum s prvkem na prvním místě A[1]
4. Zbytek uspořádáme opakováním těchto kroků pro (n-1) prvků

- **Vnitřní cyklus** – hledání minima (FOR I:=1 to N-1)
- **Vnější cyklus** – nastavení dolní meze řídícího cyklu

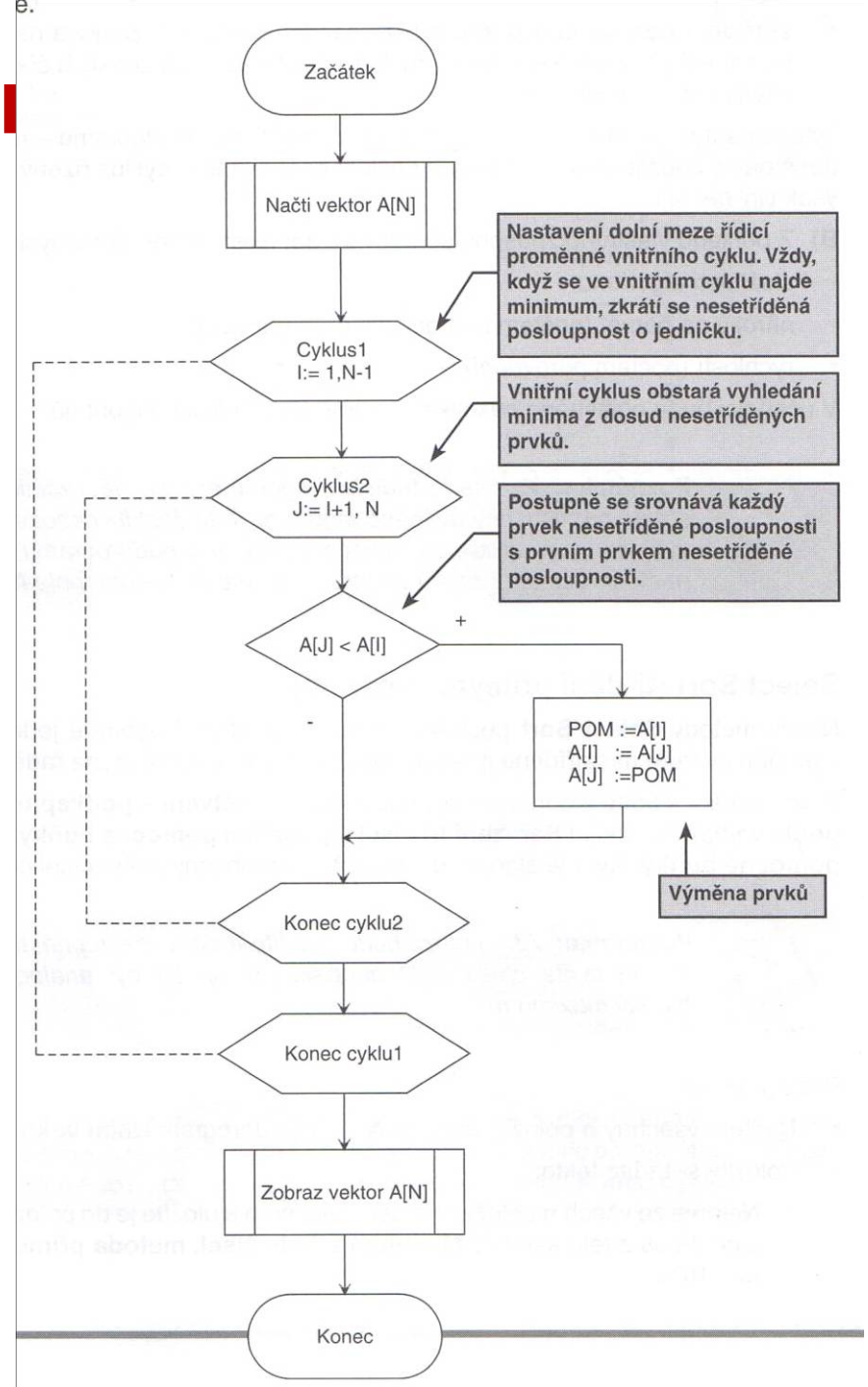
  (FOR J:=I+1 to N)

  (je-li nalezeno minimum, číslo se vyřadí a hledá se ze zbylých čísel)

Načtení řady čísel - vektoru



Třídění – Select Sort



BUBBLE SORT
bublinkové řazení

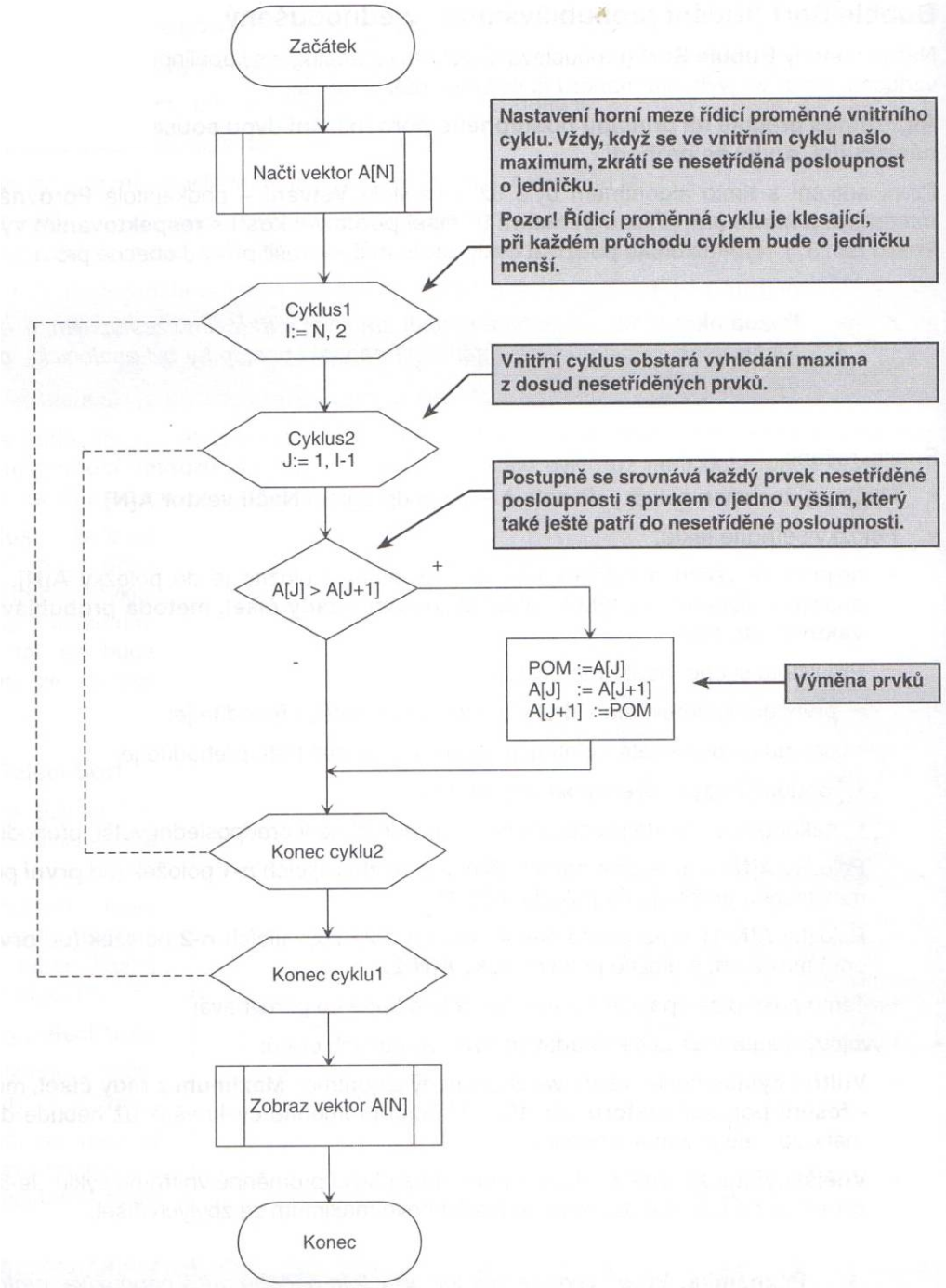
- *Velká časová náročnost : $O(N^2)$*
- *Pro menší množství dat, výuková jednoduchá metoda*

- Princip postupného porovnávání dvou sousedních prvků; je-li prvek větší než následující, prvky se prohodí (vzestupné řazení)
- + test, zda je řada prvků už setříděna – proměnná ZM

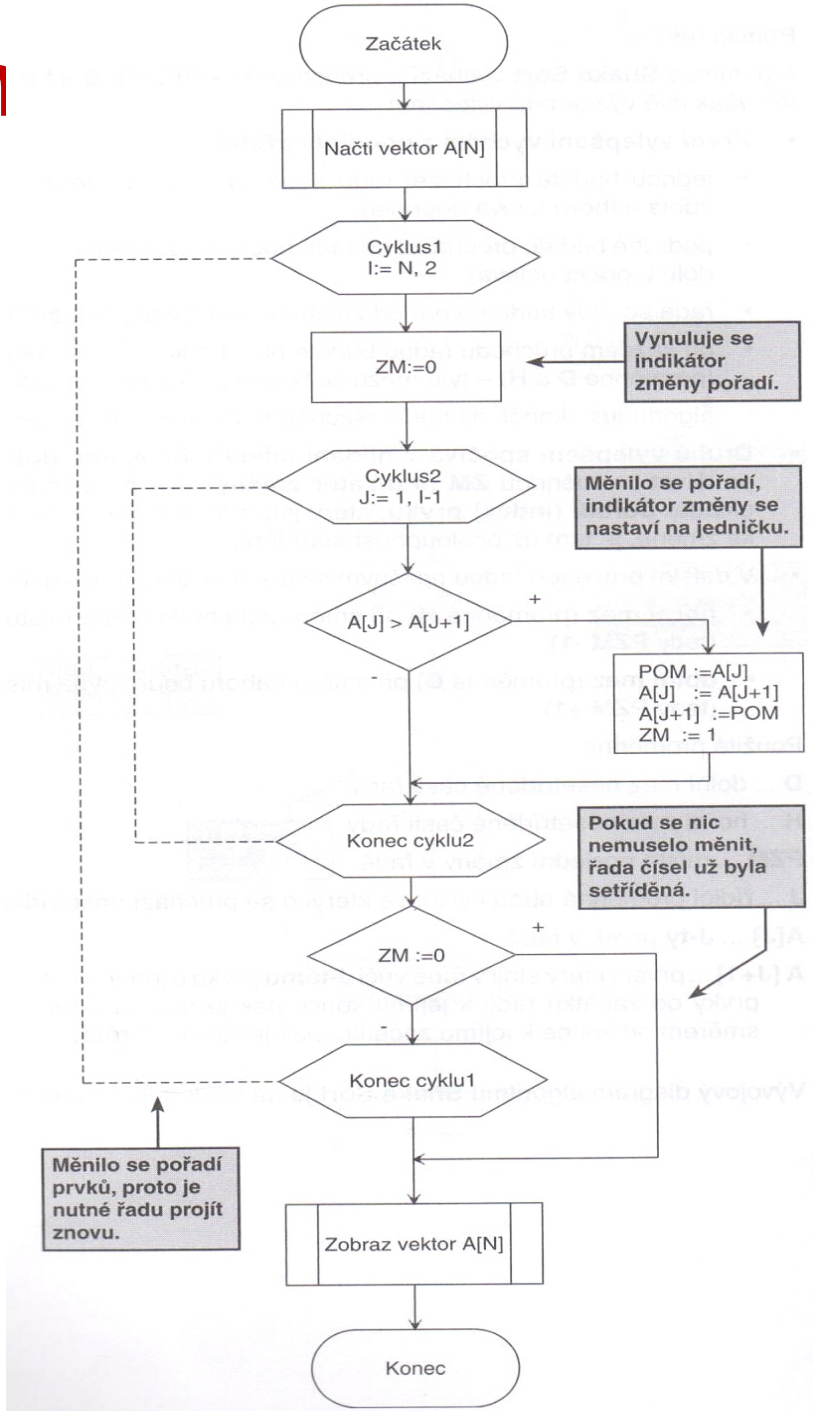
1. Načteme všech n čísel do pole A – A[N]
2. Najdeme maximum v posloupnosti - do A[N]
3. Vyhledáváme podobně od 1 do N-1 prvku a opakujeme tyto kroky dokud bude co prohledávát

- **Vnitřní cyklus** – hledání maxima pobubláváním
- **Vnější cyklus** – nastavení dolní meze řídícího cyklu

Tří ble Sort

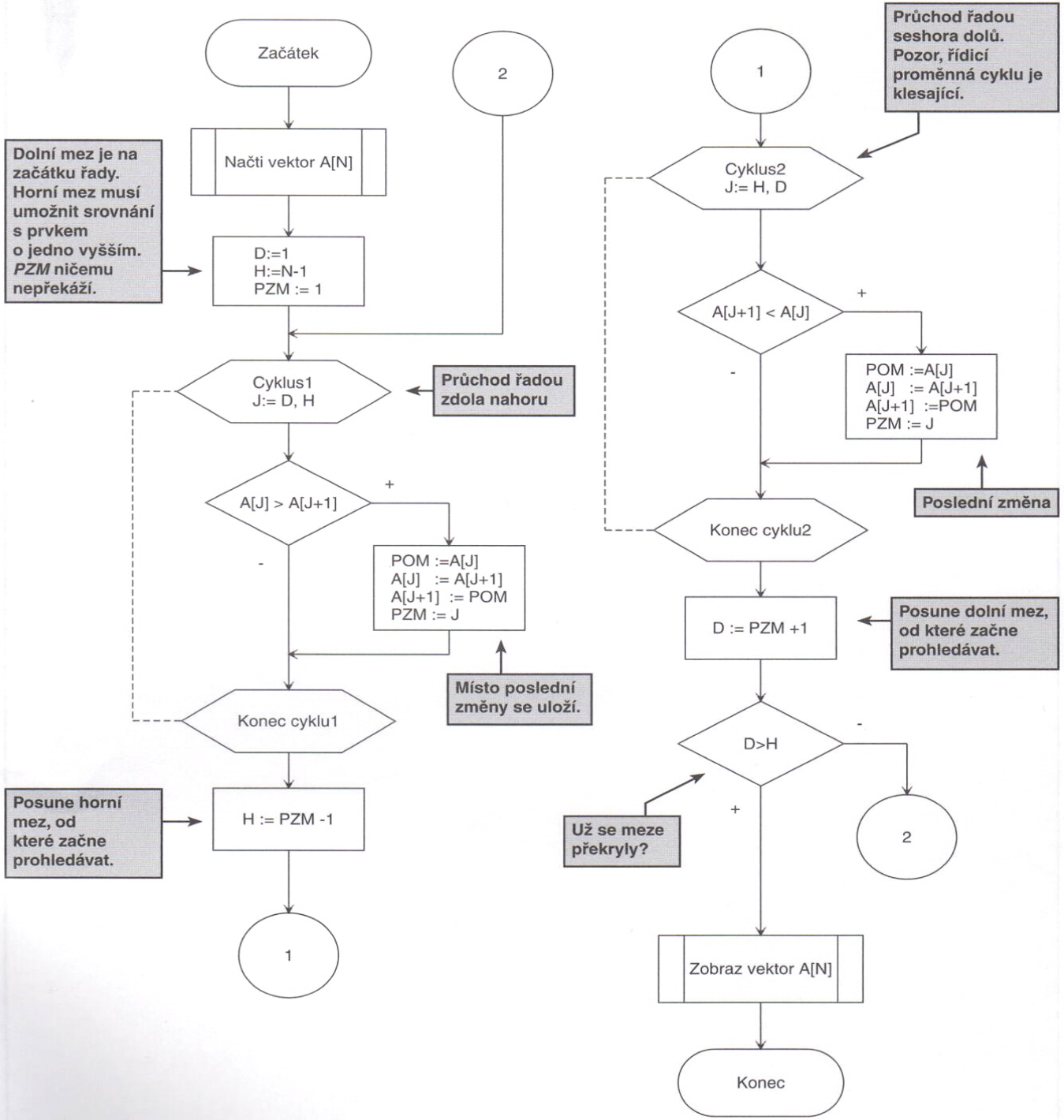

Začátek
Načti vektor A[N]
Nastavení horní meze řídicí proměnné vnitřního cyklu. Vždy, když se ve vnitřním cyklu našlo maximum, zkrátí se nesetříděná posloupnost o jedničku.
Pozor! Řídicí proměnná cyklu je klesající, při každém průchodu cyklem bude o jedničku menší.
Cyklus1
I:= N, 2
Vnitřní cyklus obstará vyhledání maxima z dosud nesetříděných prvků.
Cyklus2
J:= 1, I-1
Postupně se srovnává každý prvek nesetříděné posloupnosti s prvkem o jedno vyšším, který také ještě patří do nesetříděné posloupnosti.
A[J] > A[J+1]
+
-
POM :=A[J]
A[J] := A[J+1]
A[J+1] :=POM
Výměna prvků
Konec cyklu2
Konec cyklu1
Zobraz vektor A[N]
Konec

Třídění - Buble Sort



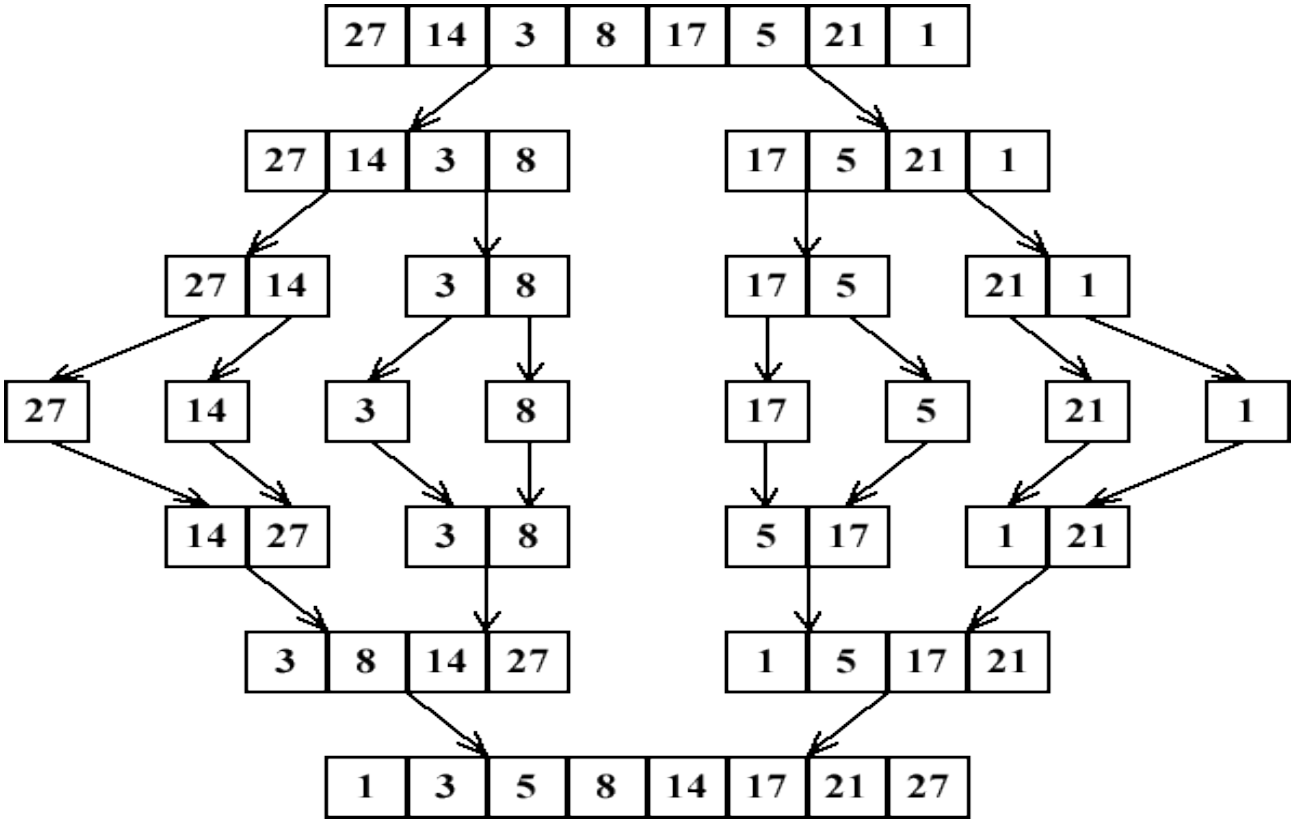
SHAKE SORT
řazení přetřásáním

- Vychází z bublinkového řazení, 2 vylepšení :
  1. Procházením zleva doprava necháme probublat největší prvek
  - Zpátky zprava doleva necháme probublat nejmenší prvek
  - Nejdéle zůstane nesetříděná posloupnost uprostřed
  - Při každém průchodu řadou hlídáme D dolní mez a H horní mez nesetříděného zbytku uprostřed řady
  - Algoritmus končí, až se meze D a H překryjí (D>H)
  2. Hlídání místa v řadě (index prvku), kde došlo k poslední změně místa PZM
  - Zpětným průchodem můžeme začít až od PZM
  - Směr dolů: horní mez H = PZM – 1
  - Směr nahoru: dolní mez D = PZM + 1


Začátek
Načti vektor A[N]
Dolní mez je na začátku řady. Horní mez musí umožnit srovnání s prvkem o jedno vyšším. PZM ničemu nepřekáží.
D:=1
H:=N-1
PZM := 1
Cyklus1
J:= D, H
Průchod řadou zdola nahoru
A[J] > A[J+1]
+
-
POM :=A[J]
A[J] := A[J+1]
A[J+1] := POM
PZM := J
Místo poslední změny se uloží.
Konec cyklu1
Posune horní mez, od které začne prohledávat.
H := PZM -1
1
2
1
Cyklus2
J:= H, D
Průchod řadou seshora dolů. Pozor, řídicí proměnná cyklu je klesající.
A[J+1] < A[J]
+
-
POM :=A[J]
A[J] := A[J+1]
A[J+1] :=POM
PZM := J
Poslední změna
Konec cyklu2
D := PZM +1
Posune dolní mez, od které začne prohledávat.
D>H
-
+
Už se meze překryly?
2
Zobraz vektor A[N]
Konec

MERGESORT
(rozděl a panuj -řazení slučováním)

- *Časová složitost (NlogN) – rychlý*
- *Potřebuje navíc pole o velikosti N*
- Načteme všech n čísel do pole A – A[N]
- Rozdělí neseřazenou posloupnost na dvě podobně velké podmnožiny
- Seřadí obě podmnožiny
- Porovnává prvky z obou podmnožin a spojuje je do nového pole
- 1945 John von Neumann



Algoritmus třídění Merge sort sestává z několika základních částí:

1.Rozdělení neseřazené množiny dat na dvě přibližně stejně velké podmnožiny
2.Seřazení obou podmnožin
3.Spojení seřazených podmnožin do jedné seřazené množiny
Tento postup je nejčastěji realizován rekurzivně.

QUICK SORT

- *Časová složitost (NlogN) – Jeden z nejrychlejších algoritmů řazení založený na porovnávání prvků, je jednoduchý*
- Zvolí se nějaká hodnota – **pivot**

(problém: zvolíme-li nějaké číslo blízké průměru – velmi rychlá metoda, jsou na to různé metody)

- Rozdělíme posloupnost na dvě přibližně stejně veliké části podle pivota

  1.Část – čísla menší než pivot

  2.Část – čísla větší než pivot
- Obě části samostatně seřadíme a spojíme
- Obě části se rekurzivně řadí stejným postupem