Pseudokódy řadících algoritmů

Popis problému

VSTUP

Na vstupu máme soubor dat, která umíme porovnávat. Můžou to být například čísla nebo řetězce.

VÝSTUP

Na výstupu bude soubor dat ze vstupu uspořádaný od nejmenšího po největší.
Některé algoritmy data při řazení jen vyměňují a nepotřebují k řazení další paměť. Říkáme, že řadí na místě.
Na místě řadí Insert sort, Select sort i Quick sort.
Algoritmus Merge Sort (řazení sléváním) kopíruje data a potřebuje další místo v paměti.

Algoritmus Insert sort (řazení vkládáním)

VSTUP

data: seznam prvků

VÝSTUP

data: seřazený seznam (řadíme na místě)

PSEUDOKÓD

```
1. n <- length(data)
2. For i from 2 to n
3.   j <- i
4.   While j > 1 and data[j] < data[j-1]
5.     Vymen(data[j], data[j-1])
6.     j <- j-1
5. #data je seřazen
```

Algoritmus Vymen

VSTUP

prvky: a, b

VÝSTUP

prvky a, b, které si vyměnily hodnoty

PSEUDOKÓD

```
1. tmp <- a
2. a <- b
3. b <- tmp
4. #a, b si vyměnily hodnoty
```

Algoritmus Select sort (řazení výběrem)

VSTUP

data: seznam prvků

VÝSTUP

data: seřazený seznam (řadíme na místě)

PSEUDOKÓD

```
1. n <- length(data)
2. For i from 1 to n
3.   index <- NajdiIndexNejmensíhoPrvku(data, i, n)
4.   Vymen(data[i], data[index])
5. #data je seřazen
```

Algoritmus NajdiIndexNejmensíhoPrvku

VSTUP

data: seznam prvků
i: počáteční index
f: konečný index

VÝSTUP

index: index nejmenšího prvku

PSEUDOKÓD

```
1. min <- data[i]
2. index <- i
3. For j from i+1 to f
4.   If data[j] < min
5.     index <- j
6.     max <- min
7. #index je výstup algoritmu
```

Algoritmus Merge (slévání)

VSTUP

data1, data2: seřazené seznamy prvků

VÝSTUP

data: seřazený seznam (neřadíme na místě)

PSEUDOKÓD

```
 1. n1 <- length(data1)
 2. n2 <- length(data2)
 3. j1 <- 1
 4. j2 <- 1
 5. j <- 1
 6. While j1 <= n1 and j2 <= n2
 7.   If data1[j1] < data2[j2]
 8.     data[j] <- data1[j1]
 9.     j1 <- j1 + 1
10.   Else
11.     data[j] <- data2[j2]
12.     j2 <- j2 + 1
13.   j <- j + 1
14. While j1 <= n1
15.   data[j] <- data1[j1]
16.   j1 <- j1 + 1
17.   j <- j + 1
14. While j2 <= n2
15.   data[j] <- data2[j2]
16.   j2 <- j2 + 1
17.   j <- j + 1
18. #data je slitý ze seznamů data1, data2
```

Algoritmus Binárního vyhledávání

VSTUP

data: seřazený seznam prvků
prvek: položka, kterou hledáme v data

VÝSTUP

index: index prvku v data, případně -1, pokud prvek v data není

PSEUDOKÓD

```
 1. i <- 1
 2. f <- length(data)
 3. While i + 1 < f # Mezi indexy i, f je index i+1.
                    # Proto se s nerovná i ani f.
 4.   s = (i + f)//2 # // je celočíselné dělení
 5.   If data[s] = prvek
 6.     index <- s
 7.     Exit # ukončení programu, index je výstup programu
 8.   ElseIf data[s] < prvek
 9.     i <- s
10.   Else
11.     f <- s
12. If data[i] = prvek
13.   index <- i
14.   Exit
15. If data[f] = prvek
16.   index <- f
17.   Exit
18. index <- -1 # Nenašli jsme prvek v data
```