Team: < 04 >, < Schäfer & Ahmad >

Aufgabenaufteilung:

1. <Aufgaben, für die Teammitglied 1 verantwortlich ist>, <Dateien, die komplett/zum Teil von Teammitglied 1 implementiert/bearbeitet wurden>
2. <Aufgaben, für die Teammitglied 2 verantwortlich ist>, <Dateien, die komplett/zum Teil von Teammitglied 2 implementiert/bearbeitet wurden>

Quellenangaben: <

http://de.wikipedia.org/wiki/Insertionsort

http://de.wikipedia.org/wiki/Selectionsort

>

Begründung für Codeübernahme: < Es wurde kein Code übernommen. >

Bearbeitungszeitraum: <

Für den Entwurf: 5 Stunden 30.10.2014

>

Aktueller Stand: < Entwurf fertig >

Änderungen im Entwurf: < KEINE >

sortNum:

Implementieren sie einen Zufallszahlengenerator. Eine

Zufallszahl ist Element Aller positiver Natürlicher Zahlen.

Die Funktion soll folgende Parameter erwarten:

- sortNum muss nach einer vorgegeben Anzahl Zufallszahlen erzeugen können.
    - Random: Erzeugte Zufallszahlen sind nicht sortiert.
    - WorstCase: Erzeugte Zufallszahlen sind absteigend vorsortiert.
    - BestCase: Erzeugte Zufallszahlen sind aufsteigend sortiert.
- Die erzeugten Zufallszahlen müssen in einer Datei zahlen.dat abgespeichert werden.

Implementieren sie den Sortieralgorithmus insertionSort.

Es soll eine einzige Funktion nach außen geliefert werden insertionS.

Diese Funktion erwartet drei Parameter:

Array -> Unser ADT Array mit unsortierten Zahlen

Von -> Kleinster Index vom Array

Bis -> Größter Index vom Array

Psydocode:

A = Unsortierte Liste

INSERTIONSORT(A)

1. for i ← 2 to Länge(A) do
2. einzusortierender_wert ← A[i]
3. j ← i
4. while j > 1 and A[j-1] > einzusortierender_wert do
5. A[j] ← A[j - 1]
6. j ← j – 1
7. A[j] ← einzusortierender_wert

Implementieren sie den Sortieralgorithmus selectionSort.

Es soll eine einzige Funktion nach außen geliefert werden selectionS.

Diese Funktion erwartet drei Parameter:

Array -> Unser ADT Array mit unsortierten Zahlen

Von -> Kleinster Index vom Array

Bis -> Größter Index vom Array

Pseudocode:

```
prozedur SelectionSort( A : Liste sortierbarer
Elemente )
  n = Länge( A )
  links = 0
  wiederhole
    min = links
    für jedes i von links + 1 bis n wiederhole
      falls A[ i ] < A[ min ] dann
          min = i
      ende falls
    ende für
    Vertausche A[ min ] und A[ links ]
    links = links + 1
  solange links < n
prozedur ende
```

Für beide Sortieralgorithmen gilt:

Zeitmessung:

Als erstes lassen sie den Algorithmus laufen und er protokolliert nur wie lange er zum sortieren gebraucht hat.

Aufwandberechnung:

Anschließend lassen sie ihn noch einmal laufen und er protokolliert den Aufwand für die Vergleiche und die Verschiebung.

Speicherung:

Der sortierte Array soll in die sortiert.dat Datei gespeichert werden.

Ausführung

Die Algorithmen sind 100-mal auszuführen, wenn möglich mit jeweils unterschiedlichen Zahlen:
- 80-mal Zufallszahlen
- 10-mal "best case"
- 10-mal "worst case"

Aus den Zeitmessungen sind anzugeben:

1. Anzahl eingelesener Elemente
2. Name des Algorithmus
3. Benötigte Zeit(Mittelwert), sowie maximal und minimal benötigteZeit.
4. Anzahl Vergleiche(Mittelwert), sowie maximale Anzahl und minimale Anzahl an Vergleichen.
5. Anzahl Verschiebungen(Mittelwert), sowie maximale Anzahl und minimale Anzahl an Verschiebungen.

Die Daten sind in einer Datei messung.log zu speichern. Zusätzlich, zur statistischen Aufbereitung, können auch *.csv dateien erzeugt werden.

Tipps:

- Das abspeichern von Zufallszahlen können realisiert werden durch ein Zusatz Modul wie z.B. myIO

- Speichert die generierten Zufallszahlen in einer Erlang Datenstruktur, somit kann man das Modul weiter geben und verwenden, beim raus holen aus der zahlen.dat, kann man den Inhalt wieder in die Erlang Datenstruktur konvertieren.

- Ausführung: Dieses empfehlen wir als eigenes Modus zu implementieren.