Algorithmen
bei Frau Prof. Dr. H. Ripphausen-Lipa

Berlin, den 30.11.2019
Matrikelnr.: 890251
Tobias Wagner

Einsendeaufgabe Nr. 2.1: Master-Theorem

Aufgabe 2.1. (Master-Theorem)

```java
public class Maximum {

    public static void main(String[] args) {
        int[] arrayElemente = new int[8]; // Definieren eines Arrays der Größe 8, um 8 Zahlen einzupflegen
        arrayElemente[0] = 1; // Eintragen der verschiedenen Zahlen
        arrayElemente[1] = 5;
        arrayElemente[2] = 10;
        arrayElemente[3] = 3;
        arrayElemente[4] = 7;
        arrayElemente[5] = 9;
        arrayElemente[6] = 8;
        arrayElemente[7] = 2;

        // Festlegung der Zählvariablen i und j und der
        // Variablen für das Maximum im Array
        int i;
        int iMax = 0;
        int j = 7;

        // Solange i kleiner j , prüfe ob die Zahl an Stelle i
        // größer ist, als das bisherige iMax, falls ja, speichere
        // die aktuell größte Zahl in iMax ab, erhöhe i um eins und fahre fort
        for (i = 0; i < j; i++) {
            if (iMax <= arrayElemente[i]) {
                iMax = arrayElemente[i];
            }
        }
        // Gebe am Ende über die Konsole als Antwort aus, welche Zahl die Größte im Array ist
        System.out.println("Das Maximum ist " + iMax);
    }
}
```

Aufwand einfache Ausführung	Anzahl Durchführungen	Insgesamt	
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(1)		
O(1)	O(n)	Ausführung for i: O(n)	
O(1)	O(n)		
O(1)	O(n)		
O(1)	O(1)		

Das oben gezeigte Verfahren, ist das iterative Verfahren. Dabei wird jedes Element im Array passiert und abgeglichen, ob es sich dabei um das größte Element / die größte Zahl in diesem Fall, im Array, handelt.

Der Zeitaufwand hierbei ist: O(n)

Algorithmen
bei Frau Prof. Dr. H. Ripphausen-Lipa

Berlin, den 30.11.2019
Matrikelnr.: 890251
Tobias Wagner

Einsendeaufgabe Nr. 2.1: Master-Theorem

Aufgabe 2.1. (Master-Theorem) (Pseudocode)

```
Search(A, l, m, r)
// Search sucht in Array A von Index l (links) bis r (rechts) das Maximum
// Teilarray A1 von A geht von Index l bis m
// Teilarray A2 von A geht von Index m+1 bis r
// Precondition: A1 ist unsortiert und A2 ist unsortiert
// H sei Hilfsarray
// l aktueller Index A1, m+1 aktueller Index A2
while (weder A1 noch A2 ist vollständig von links nach
rechts durchlaufen)
vergleiche aktuellen wert in A1 mit dem aktuellen Wert in A2, kopiere größeren aktuellen Wert von A1 oder A2 nach H
erhöhe akt. Index von beiden Teilarrays
Gebe den größten Wert aus beiden Teilarrays aus
```

Das oben gezeigte Verfahren, ist das divide and conquer Verfahren.
Es wird zunächst das array gesplittet (divide) und dann das Problem in den teilen des Arrays gelöst (conquer).

Der Zeitaufwand hierbei ist:

$$T(n) = \begin{cases} O(1) \text{ für } n = 1 \\ 2*T(n/2) + O(n) \text{ für } n > 1 \end{cases}$$

Die Lösung der Formel "2*T(n/2) + O(n) für n > 1" entspricht in unserem Fall der Lösung der Formel
Die Lösung der Formel "n^LOG(A;B)" (in Worten: n hoch Logarithmus A zur Basis B)

Daraus folgt: Wir haben es mit dem mittleren Fall des Master-Theorems zu tun.

Die Laufzeit beträgt:

$$T(n) = \Theta(n^{\log_b a} \log_2 n)$$

In unserem Beispiel, das Maximum aus einem unsortierten array zu finden, ist das Divide and Conquer Verfahren langsamer als das iterative Verfahren.

Algorithmen
bei Frau Prof. Dr. H. Ripphausen-Lipa

Berlin, den 02.12.2019
Matrikelnr.: 890251
Tobias Wagner

Einsendeaufgabe Nr. 2.2: Mergesort

Aufgabe 2.2. (Mergesort)

Schlussfolgerung:

Da beim Mergesort Algorithmus jedes Mal das vollständige Array geteilt und dann sortiert wird,
spielt es für die Länge der Laufzeit keine Rolle, ob es unsortiert oder sortiert ist.
Je größer das Array ist, desto länger ist die Laufzeit.