Aufgabe 2: Sortieren in Linearzeit (2+2+2 Punkte)

Wir betrachten ein Feld aus n unterschiedlichen Zahlen. In der Vorlesung wurde gezeigt, dass es kein vergleichsbasiertes Sortierverfahren geben kann, dass für alle n! möglichen Eingabefelder eine asymptotisch lineare Laufzeit besitzt.

Verallgemeinern Sie diese Aussage und zeigen Sie, dass es kein vergleichsbasiertes Sortierverfahren geben kann, welches für mehr als

(a) die Hälfte

(b) 1/n

(c) $1/(2^n)$

aller n! möglichen Eingabefelder eine asymptotisch lineare Laufzeit besitzt.

Lemma

Sei T der Entscheidungsbaum für den Algorithmus A und $C(n)$ bzw. $\overline{C}(n)$ die Worst Case bzw. Average Case (bei Gleichverteilung) Anzahl von Vergleichen bei n zu sortierenden Komponenten. Dann gilt:

(a) $C(n) = \max\{h(v) \mid v \text{ Blatt von } T\} = h(T)$, *und*

(b) $\overline{C}(n) = \frac{1}{n!} \sum_{v \text{ Blatt von } T} h(v) = \frac{1}{n!} \cdot H(T)$.

Lemma

Sei T ein binärer Baum mit b Blättern. Dann gilt:

(a) $h(T) \geq \log_2 b$ *und*

(b) $H(T) \geq b \cdot \log_2 b$.

(c) b = Anzahl Blätter im Entscheidungsbaum $= \frac{1}{2^n} \cdot n!$

$$C(n) = h(T) \geq \log_2 b = \log_2\left(\frac{1}{2^n} \cdot n!\right) =$$

$$= \log_2\left(\frac{1}{2^n}\right) + \log(n!)$$

$\log(a \cdot b) = \log(a) + \log(b)$

$$= \log_2(2^{-n}) + \log(n!)$$

$\log_2(2^x) = x$, $2^{\log_2(x)} = x$

$$= -n + \log_2(n!)$$

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^{n+(1/12n)}$$

$$\geq -n + \log_2\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)$$

$\log(a^b) = b \cdot \log(a)$

$$= -n + \log_2(\sqrt{2\pi}) + \underbrace{\log_2(\sqrt{n})}_{= \frac{1}{2} \cdot \log_2(n)} + n \cdot \left(\log_2(n) - \log_2(e)\right)$$

$\log\left(\frac{a}{b}\right) = \log(a) - \log(b)$

$$\in \Omega(n \cdot \log_2(n))$$

$C \cdot n$ ist gegenüber $n \cdot \log_2(n)$ asymptotisch vernachlässigbar

Aufgabe 3: Sortieren in Linearzeit (2+2+2+2 Punkte)

Beweisen oder widerlegen Sie: Es gibt einen vergleichsbasierten Sortieralgorithmus, der ein beliebiges Feld von n Elementen in $\mathcal{O}(n)$ Laufzeit sortiert, falls

a) ... ein Element existiert, das mehr als $n/2$ mal im Feld vorkommt.

b) ... die erste Hälfte bereits aufsteigend und die zweite absteigend sortiert ist.

c) ... im Feld nur eine konstante Anzahl an unterschiedlichen Elementen existiert.

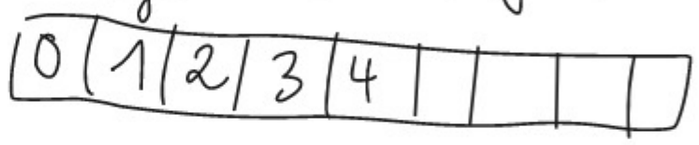
d) ... 99% aller Elemente den gleichen Wert besitzen.

Skizzieren Sie jeweils einen solchen Sortieralgorithmus oder beweisen Sie, warum es einen solchen nicht geben kann.

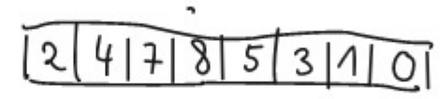
Tipp: Für die Nicht-Existenz eines Sortieralgorithmus bietet sich ein Widerspruchsbeweis an: Nehmen Sie an, es gäbe einen solchen Algorithmus und zeigen Sie, wie Sie damit *beliebige* Felder (also Felder ohne diese besonderen Voraussetzungen) in Linearzeit sortieren können.

b) Algorithmus

Lege neues Array an




```
void sort(int[] A){
  int m = A.length/2;
  int i=0;
  int j=A.length-1;
  int k=0
  int[] z = new int[A.length];
  while(i<m && j>=m){
    if(A[i]<=A[j]){
      z[k]=A[i];
      i++;
    }
    else{
      z[k]=A[j];
      j--;
    }
    k++;
  }
  while(i<m){
    z[k]=A[i];
    i++;
  }
  while(j>=m){
    z[k]=A[j];
    j--;
  }
```

b) 

Array hat Länge n wobei n gerade

#Permutationen = $\binom{n}{\frac{n}{2}} = \frac{n!}{(\frac{n}{2})! \cdot (\frac{n}{2})!}$

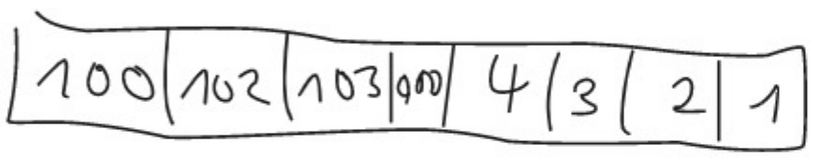
"Anzahl"

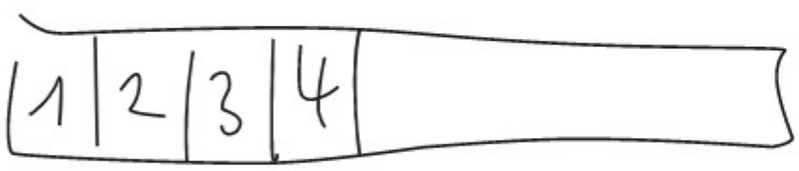
$C(n) = h(T) \geq \log_2(h) = \log_2$

$= \log_2(n!) - 2 \cdot \log_2((\frac{n}{2})!)$

$\geq$

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^{n+(1/12n)}$$





$\in \mathcal{O}(n)$

9)

1	5	1	8	1	2	3	2	1

Angenommen es gäbe so einen Algorithmus ≙ Methode `void sort1(int[] A)`

```
void sort2(int[] A){
  int[] z = new int[2·A.length];
  for(int i=0; i<A.length; ++i){
    z[i] = A[i];
    z[A.length+i] = Integer.MAX_VALUE;
  }
  sort1(z);
  for(int i=0; i<A.length; ++i){
    A[i] = z[i];
  }
}
```

$\in O(n)$

Widerspruch, da

Theorem

Jeder deterministische vergleichsbasierte Sortieralgorithmus benötigt sowohl im schlimmsten Fall als auch im Mittel (bei Gleichverteilung) für Eingaben der Länge n mindestens $n \log n - n \log_2 e \approx n \log_2 n - 1{,}44n$ Vergleiche.

Aufgabe 3: Sortieren in Linearzeit (2+2+2+2 Punkte)

Beweisen oder widerlegen Sie: Es gibt einen vergleichsbasierten Sortieralgorithmus, der ein beliebiges Feld von n Elementen in $\mathcal{O}(n)$ Laufzeit sortiert, falls

a) ... ein Element existiert, das mehr als $n/2$ mal im Feld vorkommt.

b) ... die erste Hälfte bereits aufsteigend und die zweite absteigend sortiert ist.

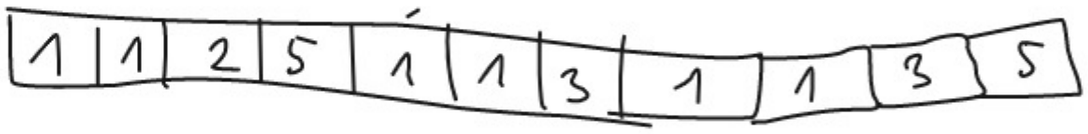
c) ... im Feld nur eine konstante Anzahl an unterschiedlichen Elementen existiert.

d) ... 99% aller Elemente den gleichen Wert besitzen.

Skizzieren Sie jeweils einen solchen Sortieralgorithmus oder beweisen Sie, warum es einen solchen nicht geben kann.

Tipp: Für die Nicht-Existenz eines Sortieralgorithmus bietet sich ein Widerspruchsbeweis an: Nehmen Sie an, es gäbe einen solchen Algorithmus und zeigen Sie, wie Sie damit *beliebige* Felder (also Felder ohne diese besonderen Voraussetzungen) in Linearzeit sortieren können.

c) C = 4




```
void sort (int[] A, int C){
  Map<Integer, Integer> map = new HashMap<Integer, Integer>();
  for (int i=0; i< A.length; ++i) {
     map[A[i]]++;
  }
  int i=0;
  for(int key : map.keySet()) {
    for(int j=0; j< map[key]; ++j){
      A[i] = key; ++i;
  } }
```

map

key	1	3	5	2
value	6	2	2	1

map[key] = value;

$\in \mathcal{O}(n)$

d) wie bei a)

(50% oder 99%
ist dabei egal)

Konstanten spielen bei asymptotischer Laufzeit keine Rolle!