

- 10a) **Was passiert bei Sortieren durch Einfügen, wenn die gegebene Folge bereits aufsteigendsortiert ist. Was passiert, wenn alle Elemente gleich sind? Ist es möglich, dass man bloß $O(n)$ Zeit benötigt?**

Gegebene sei eine Reihenfolge von n aufsteigend sortierten Elementen.

$$x_1 < x_2 < x_3 < \dots < x_{n-1} < x_n$$

Das erste Element wird in die leere Liste eingefügt. Dafür benötigt man keine Vergleiche.

$$x_1$$

Das zweite Element wird in die sortierte Liste eingefügt. Da $x_1 < x_2$ gilt, muss die Liste folgende Reihenfolge haben und man benötigt 1 Vergleich.

$$x_1, x_2$$

Das dritte Element wird eingefügt: $x_3 < x_2 < x_1$. Hier benötigt man 2 Vergleiche, um das Element in die sortierte Liste einzufügen. x_1, x_2, x_3

Das $(n-1)$ 'te Element wird eingefügt. Man benötigt $n-2$ Vergleiche, da $x_1 < x_2 < x_3 < \dots < x_{n-1}$ ist und die Liste ist dann $x_1, x_2, x_3, \dots, x_{n-1}$.

Das n 'te Element wird eingefügt. Hier braucht man $n-1$ Vergleiche, denn $x_1 < x_2 < x_3 < \dots < x_{n-1} < x_n$. Die Liste ist dann $x_1, x_2, x_3, \dots, x_{n-1}, x_n$.

Gesamtvergleiche: $0, 1, 2, \dots, n-2, n-1$ bzw. $O(n \cdot (n-1))$

Laufzeit: $O(n^2)$

Die Reihenfolge lässt sich auch in $O(n)$ sortieren, wenn man in der Liste das einzufügende Element von hinten sucht. Warum das so ist, wird in Aufgabenteil b) gezeigt.

- 10b) **Was passiert, wenn die Folge abwärts sortiert ist?**

Gegeben sei eine Reihenfolge von n absteigend sortierten Elementen.

$$x_1 > x_2 > x_3 > \dots > x_{n-1} > x_n$$

Das erste Element wird in die leere Liste eingefügt. Dafür benötigt man keine Vergleiche.

$$x_1$$

Das zweite Element wird in die sortierte Liste eingefügt. Da $x_1 > x_2$ gilt, muss die Liste folgende Reihenfolge haben und man benötigt 1 Vergleich. x_2, x_1

Das dritte Element wird eingefügt. Da $x_3 > x_2 > x_1$ ist, benötigt man 1 Vergleich, um das Element in die sortierte Liste einzufügen. x_3, x_2, x_1

Das $(n-1)$ 'te Element wird eingefügt. Man benötigt 1 Vergleich, weil $x_1 > x_2 > x_3 > \dots > x_{n-1}$ ist. Die Liste ist dann $x_{n-1}, \dots, x_3, x_2, x_1$.

Musterlösung: Alp 3 Übung 2, 6. November 2003

Sebastian Charlet

Bernadett Smolibocki

Das n 'te Element wird eingefügt. Hier braucht man 1 Vergleich, denn $x_1 > x_2 > x_3 > \dots > x_{n-1} > x_n$. Die Liste ist dann $x_{n-1}, x_n, \dots, x_3, x_2, x_1$.

Gesamtvergleiche: $0, 1, 1, \dots, 1, 1$ bzw. $O(n * (1))$

Laufzeit: $O(n)$

Das Einfügeelement wird von vorne gesucht.

10c) **Was passiert, wenn alle Elemente gleich sind?**

Gegeben sei wieder eine Reihenfolge von n gleichen Elementen.

$$x_1 = x_2 = \dots = x_{n-1} = x_n$$

Hier muss man unterscheiden, ob der Algorithmus mit „ \leq “ oder „ $<$ “ vergleicht.

Operation: absolut kleiner	Operation: kleiner gleich
Das erste Element wird in die leere Liste eingefügt. Dafür benötigt man keinen Vergleich. x_1	Das erste Element wird in die leere Liste eingefügt. Dafür benötigt man keinen Vergleich. x_1
Das zweite Element wird eingefügt, da x_2 nicht kleiner als x_1 ist, benötigt man nur 1 Vergleich und die Liste hat dann die Form. x_1, x_2	Das zweite Element wird eingefügt, da $x_1 = x_2$ ist, benötigt man nur 1 Vergleich und die Liste hat dann die Form. x_2, x_1
Das $n-1$ 'te Element wird eingefügt, da x_{n-1} nicht absolut kleiner als seine Vorgänger ist, benötigt man $n-2$ Vergleiche. x_1, x_2, \dots, x_{n-1}	Das $n-1$ 'te Element ist gleich seinem Vorgänger. Man benötigt also einen Vergleich. Die Liste sieht dann so aus. x_{n-1}, \dots, x_2, x_1
...	...
<u>Laufzeit:</u> $O(n^2)$	<u>Laufzeit:</u> $O(n)$

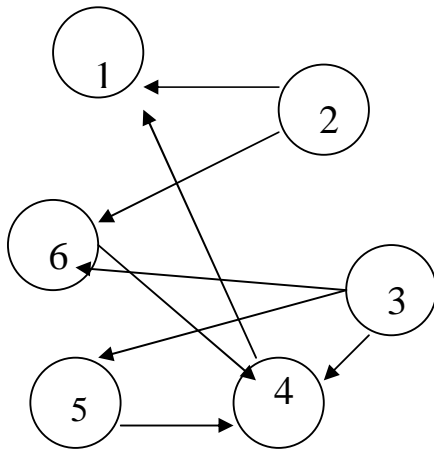
Das Einfügeelement wird von vorne gesucht.

11) Es wird immer $O(n^2)$ Zeit benötigt, denn nach der Definition des Algorithmus' wird die Liste in jedem Fall durchsucht, oder genauer der unsortierte Teil der Liste wird nach dem Auswahlelement durchsucht.

Also: $\sum_{i=0}^{n-1} n$ für jeden inneren Schleifendurchlauf (Auswahlelement finden) * n

(Elemente) ergibt: $n * (0 + \dots + (n-1))$ bzw. $O(n)$

15) **Geben Sie alle möglichen topologischen Sortierungen für folgende Eingabe an: $n = 6$ und $\{(6, 4), (3, 5), (4, 1), (3, 4), (5, 4), (2, 1), (2, 6), (3, 6)\}$.**



Elemente ohne Vorgänger: 2, 3

mögliche Sortierungen:

- 1) 2, 3, 5, 6, 4, 1
- 2) 2, 3, 6, 5, 4, 1
- 3) 3, 2, 5, 6, 4, 1
- 4) 3, 5, 2, 6, 4, 1
- 5) 3, 2, 6, 5, 4, 1

18) **Erweitern Sie den Algorithmus zum topologischen Sortieren aus der Vorlesung, sodass bei der Existenz eines Kreises nicht einfach mit einer Meldung abgebrochen wird, sondern auch ein Kreis (als "Beweis") ausgegeben wird.**

Algorithmus zur topologischen Sortierung:

Suche ein Element x ohne Vorgänger.

Setze x an die erste Stelle. Lösche x und alle Paare, die x enthalten.

Wiederhole das Ganze.

Pseudo-Code für die Kreisausgabe:

Für alle Knoten v Element V

Falls v einen Vorgänger hat

Für alle ausgehenden Kanten (v, w)

Setze v als einen Vorgänger von w

Wähle einen beliebigen Knoten mit Vorgänger

Solange der Knoten nicht markiert ist

Markiere den Knoten

Gehe zum Vorgänger

Solange der Knoten markiert ist

Gib den Knoten aus

Entferne die Markierung

Gehe zum Vorgänger