28. März 2025

FDS2 - Übung 4

SS 2025

Tim Peko

Inhaltsverzeichnis

28. März 2025

1. Beispiel 1: Permutation

1.1. Lösungsansatz

Die rekursive Funktion `permute` generiert alle möglichen Permutationen eines Vektors durch systematisches Vertauschen der Elemente. Der Algorithmus verwendet einen rekursiven Ansatz, bei dem in jedem Schritt ein Element fixiert und die restlichen Elemente permutiert werden.

1.2. Testfälle

Die Testfälle werden automatisch in der `main` Funktion ausgeführt.

1.2.1. Testfall 1: [1, 2, 3]

Input:

```
{1, 2, 3}
```

Output:

```
{1, 2, 3}
{1, 3, 2}
{2, 1, 3}
{2, 3, 1}
{3, 2, 1}
{3, 1, 2}
```

Ergebnis: success

1.2.2. Testfall 2: [1, 2]

Input:

```
{1, 2}
```

Output:

```
{1, 2}
{2, 1}
```

Ergebnis: success

2. Beispiel 2: Mischen

2.1. Lösungsansatz

Die rekursive Funktion `random_shuffle` implementiert den Fisher-Yates Shuffle Algorithmus in einer rekursiven Variante. Der Algorithmus wählt in jedem Schritt zufällig ein Element aus dem verbleibenden Array und tauscht es mit dem aktuellen Element.

2.2. Testfälle

2.2.1. Testfall 1: [1, 2, 3, 4, 5]

Input:

```
{1, 2, 3, 4, 5}
```

Output:

```
{2, 1, 3, 5, 4}
```

Ergebnis: success

2.2.2. Testfall 2: [1, 2]

Input:

```
{1, 2}
```

Output:

```
{2, 1}
```

Ergebnis: success

3. Beispiel 3: Primzahlen erkennen

3.1. Lösungsansatz

Die indirekt rekursive Funktion `is_prime` prüft die Primzahleigenschaft durch rekursive Division mit allen möglichen Teilern bis zur Quadratwurzel der Zahl.

3.2. Testfälle

3.2.1. Testfall 1: 17

Input:

```
17
```

Output:

```
true
```

Ergebnis: success

3.2.2. Testfall 2: 4

Input:

```
4
```

Output:

```
false
```

Ergebnis: success

3.2.3. Testfall 3: 2

Input:

```
2
```

Output:

```
true
```

Ergebnis: success

3.2.4. Testfall 4: 1

Input:

```
1
```

Output:

```
false
```

Ergebnis: success

4. Beispiel 4: Große Zahlen vergleichen

4.1. Lösungsansatz

Die rekursive Funktion `compare` vergleicht zwei Zahlen zur Basis 1000, die als einfachverkettete Listen dargestellt sind. Der Vergleich erfolgt rekursiv von links nach rechts.

4.2. Testfälle

4.2.1. Testfall 1: 1002003 vs 1002004

Input:

```
num1 = 1002003
num2 = 1002004
```

Output:

```
-1
```

Ergebnis: success

4.2.2. Testfall 2: 1002003 vs 1002003

Input:

```
num1 = 1002003
num2 = 1002003
```

Output:

```
0
```

Ergebnis: success

4.2.3. Testfall 3: 2000000000049 vs 5

Input:

```
num1 = 2000000000049
num2 = 5
```

Output:

```
1
```

Ergebnis: success

5. Beispiel 5: Bisektion

5.1. Lösungsansatz

Die rekursive Funktion `bisection` findet eine Nullstelle einer stetigen Funktion im gegebenen Intervall durch wiederholte Intervallhalbierung.

5.2. Testfälle

Die Testfälle wurden mithilfe von GeoGebra überprüft.

5.2.1. Function 1: $f(x) = 2x^3 - 3x^2 + 0.5$

Input:

```
f(x) = 2x^3 - 3x^2 + 0.5
interval = [-2, 0]
```

Output:

```
-0.365997
```

Ergebnis: success

Input:

```
f(x) = 2x^3 - 3x^2 + 0.5
interval = [-2, 4]
```

Output:

```
1.36598
```

Ergebnis: success

5.2.2. Function 2: $f(x) = x + e^x$

Input:

```
f(x) = x + e^x
interval = [-1, 1]
```

Output:

```
-0.567169
```

Ergebnis: success

Input:

```
f(x) = x + e^x
interval = [0, 1]
```

Output:

```
Error: Function has same signs at interval bounds
```

Ergebnis: success

5.2.3. Function 3: $f(x) = \frac{5}{2}x^2 - 7x + 4$

Input:

```
f(x) = 2.5x^2 - 7x + 4
interval = [-1, 4]
```

Output:

```
Error: Function has same signs at interval bounds
```

Ergebnis: success

Input:

```
f(x) = 2.5x^2 - 7x + 4
interval = [1, 4]
```

Output:

```
1.99998
```

Ergebnis: success

Input:

```
f(x) = 2.5x^2 - 7x + 4
interval = [-1, 1]
```

Output:

28. März 2025

`0.800018`

Ergebnis: success

Aufwand in h: 8