Die palindromische Ordnung

In[380]:=
```
Remove["Global`*"]
```

Ist $a := a_1\, a_2 \ldots a_n$ eine natürliche Zahl mit Ziffern a_i, so heißt a Palindrom, genau dann, wenn $a = a_1\, a_2 \ldots a_n = a_n\, a_{n-1} \ldots a_1$ gilt. So ist beispielsweise die Zahl 1771 ein Palindrom, 1771 jedoch nicht.
Für jede natürliche Zahl n definieren wir folgende iterative Vorschrift:

$a_0 := n$. Falls a_n ein Palindrom ist stopt der Algorithmus, andernfalls setze $a_{n+1} := a_n + R[a_n]$, wobei die Zahl $R[a_n]$ dadurch entsteht, daß die Ziffernfolge von a_n umgekehrt wird. Die Frage ist nun, nach wievielen Iterationsschritten, wenn überhaupt, dieser Algorithmus terminiert. Die Anzahl n der Iterationen die benötigt werden, bis a_n ein Palindrom ist heißt dann *palindromische Ordnung* von a

Um dieses Problem im *Mathematica* zu untersuchen benötigen wir zunächst eine Funktion die die Ziffernfolge einer natürlichen Zahl umdreht. Da man die Ziffernfolge einer Zahl a einfach mit **IntegerDigits[a]** erhält ist diese Aufgabe einfach zu lösen.

In[381]:=
```
a = 1234567
```

Out[381]=
```
1234567
```

In[382]:=
```
liste = IntegerDigits[a]
```

Out[382]=
```
{1, 2, 3, 4, 5, 6, 7}
```

In[383]:=
```
rliste = Reverse[liste]
```

Out[383]=
```
{7, 6, 5, 4, 3, 2, 1}
```

Um nun aus der umgekehrten Liste der Ziffernfolge der Zahl a wieder eine Zahl zu machen bedienen wir uns der Funktion **Fold[...]** um eine Funktion zu definieren, die eine Zahl liefert die die umgekehrte Ziffernfolge hat, wie die ursprümgliche Zahl a.

In[384]:=
```
reverseInteger[n_] := Fold[10 #1 + #2 &, 0, Reverse[IntegerDigits[n]]]
```

In[385]:=
```
reverseInteger[1234567]
```

Out[385]=
```
7654321
```

Nun ist es einfach eine Funktion zu definieren, die testet, ob eine natürliche Zahl ein Palindrom ist oder nicht

In[386]:=
```
testPalindrom[n_] := n == reverseInteger[n]
```

In[387]:=
```
testPalindrom[121]
```

Out[387]=
```
True
```

In[388]:=
```
testPalindrom[1772]
```

Out[388]=
```
False
```

Jetzt können wir recht einfach die palindromische Ordnung natürlicher Zahlen bestimmen. Dazu definieren wir noch eine Funkton, die die Anzahl der Iterationen ausgibt, die nötig sind bis die Iterationsfolge ein Palindrom ergibt. Vorsichtshalber geben wir auch eine Maximalzahl von durzuführenden Iterationsschritten an.Die Funktion **palindromOrdnung[n,maxiter]** bestimmt die palindromische Ordnung der Zahl **n** mt einer Obergrenze von **maxiter**

In[389]:=
```
palindromOrdnung[n_, maxiter_] := Module[{temp = n, i = 1},
  While[testPalindrom[temp] == False && i < maxiter,
   temp = temp + reverseInteger[temp]; i++;
  ]; (* End while *)
  If[i >= maxiter,
   Print["*** Maximum der Iterationen erreicht: ", n, ": ", temp]; , i]
 ]
```

In[390]:=
```
palindromOrdnung[169, 100]
```

Out[390]= 3

In[391]:=
```
palindromOrdnung[196, 100]
```

```
*** Maximum der Iterationen erreicht: 196
 : 13 158 897 331 226 320 592 562 872 742 059 146 230 247 889 513
```

Als nächstes suchen wir all jene Zahlen in einem Intervall, die zu einer vorgegebenen, maximalen Iterationstiefe **nicht** auf ein Palindrom führen. Dazu

In[392]:=
```
palindromfunc[n_] := n + reverseInteger[n]
```

Dann

In[393]:=
```
palindromOrdnungTest[n_, max_] :=
 Length[NestWhileList[palindromfunc, n, Not[testPalindrom[#]] &, 1, max]]
```

In[394]:=
```
palindromOrdnungTest[196, 200]
```

Out[394]= 201

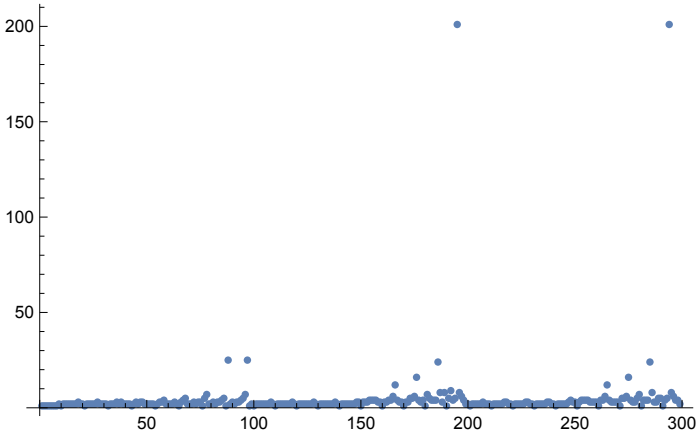
Die Zahl 196 ist der erste Kandidat, der anscheinend nicht auf ein Palindrom führt

In[395]:=
```
temp = Table[palindromOrdnungTest[n, 200], {n, 2, 300}]
```

Out[395]=
```
{1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2,
 2, 2, 1, 2, 2, 2, 3, 2, 3, 2, 2, 2, 2, 1, 2, 3, 2, 3, 3, 2, 2, 2, 2, 2, 1, 2, 3, 3,
 4, 2, 2, 2, 2, 3, 2, 1, 3, 4, 5, 2, 2, 2, 3, 2, 3, 3, 1, 5, 7, 2, 2, 3, 2, 3, 3,
 4, 5, 1, 25, 2, 3, 2, 3, 3, 4, 5, 7, 25, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 3, 2, 1, 2,
 2, 2, 2, 2, 2, 2, 3, 2, 1, 2, 2, 2, 2, 2, 2, 2, 3, 2, 1, 2, 2, 2, 2, 2, 2, 2, 3,
 2, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 1, 3, 3, 3, 4, 4, 4, 4, 3, 3, 1, 3, 3, 4, 4, 6,
 12, 4, 3, 3, 1, 3, 3, 5, 5, 6, 16, 4, 3, 4, 1, 7, 5, 4, 4, 4, 24, 8, 3, 8, 1, 5,
 9, 4, 5, 201, 8, 6, 4, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 1, 2, 2, 2, 2, 2, 3, 3,
 2, 2, 1, 2, 2, 2, 2, 2, 3, 3, 2, 2, 1, 2, 2, 2, 2, 2, 3, 3, 2, 2, 1, 2, 2, 2, 2, 2,
 3, 4, 3, 3, 1, 3, 4, 4, 4, 4, 3, 3, 3, 3, 1, 4, 4, 6, 12, 4, 3, 3, 3, 3, 1, 5, 5,
 6, 16, 4, 3, 3, 5, 7, 1, 4, 4, 4, 24, 8, 3, 3, 5, 5, 1, 4, 5, 201, 8, 6, 4, 4, 2}
```

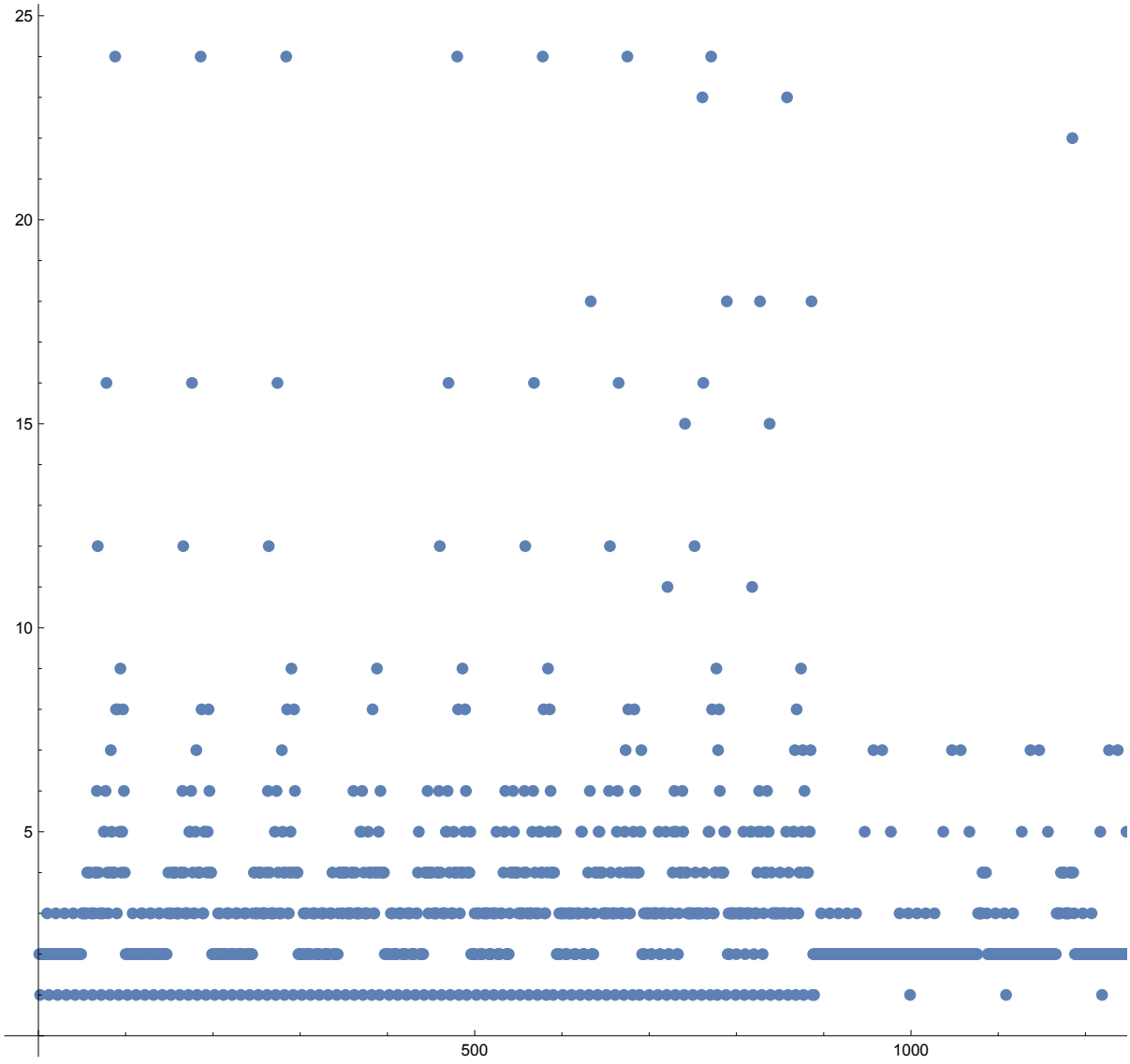
In[396]:=
```
ListPlot[temp, PlotRange → All]
```

Out[396]=



In[397]:=
```
ListPlot[DeleteCases[Table[palindromOrdnungTest[n, 200], {n, 100, 2000}], 201],
 PlotRange → All, AxesOrigin → {0, 0}]
```

Out[397]=



In[398]:=
```
ListPlot[Table[palindromOrdnungTest[n, 200], {n, 100, 2000}], PlotRange → All,
 AxesOrigin → {0, 0}, PlotLabel → Style["Palindromische Ordnung bis 2000", 16],
 AxesLabel → {"n", "Ordnung(n)"}]
```

Out[398]=