Implementiere zunächst die rekursive Methode. Welches Laufzeitverhalten ist zu erwarten? Wie könnte es bestätigt werden?

```
public double samrek(int n) {
      if (n==0) {
         return w_0;
       }
       else {
          if (n==1) {
             return w_1;
          }
          else {
             return 1.8*samrek(n-1)-0.9*samrek(n-2)+1;
          }
       }
   }
```

Zeitmessung:

```
public long zeitMessen(int n){
      long time = System.nanoTime();
        samrek(n);
      return System.nanoTime() - time;
   }
```

Fehlerquellen:

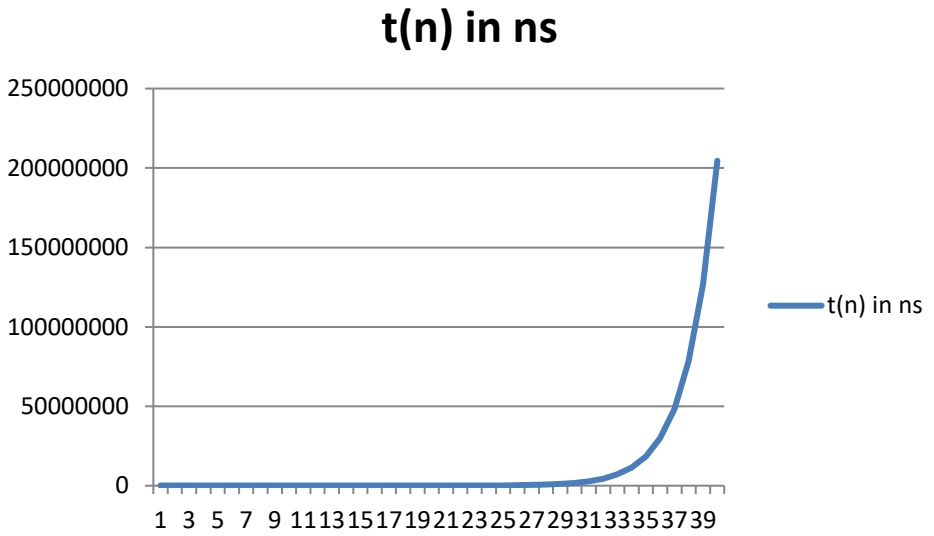
- Zeitmessung erfordert auch Zeit
  (kann mit leerem Anweisungsblock gemessen werden.)
- Prozess kann während der Ausführung z.B. durch Garbage Collection unterbrochen werden.
  ➔ mehrmals messen und **Minimum** nehmen

Miss für verschiedene n-Werte die Laufzeit von samrek und trage sie in eine Tabelle ein. Führe jeweils mehrere Messungen durch!

```
public void Messung(int og){
        for (int i=0;i<og; i++) {
            long minZeit = zeitMessen(i);
            for (int j=0;j<9;j++){
                long zeit = zeitMessen(i);
                if (zeit<minZeit){minZeit = zeit;}
            }
            System.out.println(i +";" +minZeit);
      }
   }
```

Miss für verschiedene n-Werte die Laufzeit von samrek und trage sie in eine Tabelle ein. Führe jeweils mehrere Messungen durch!

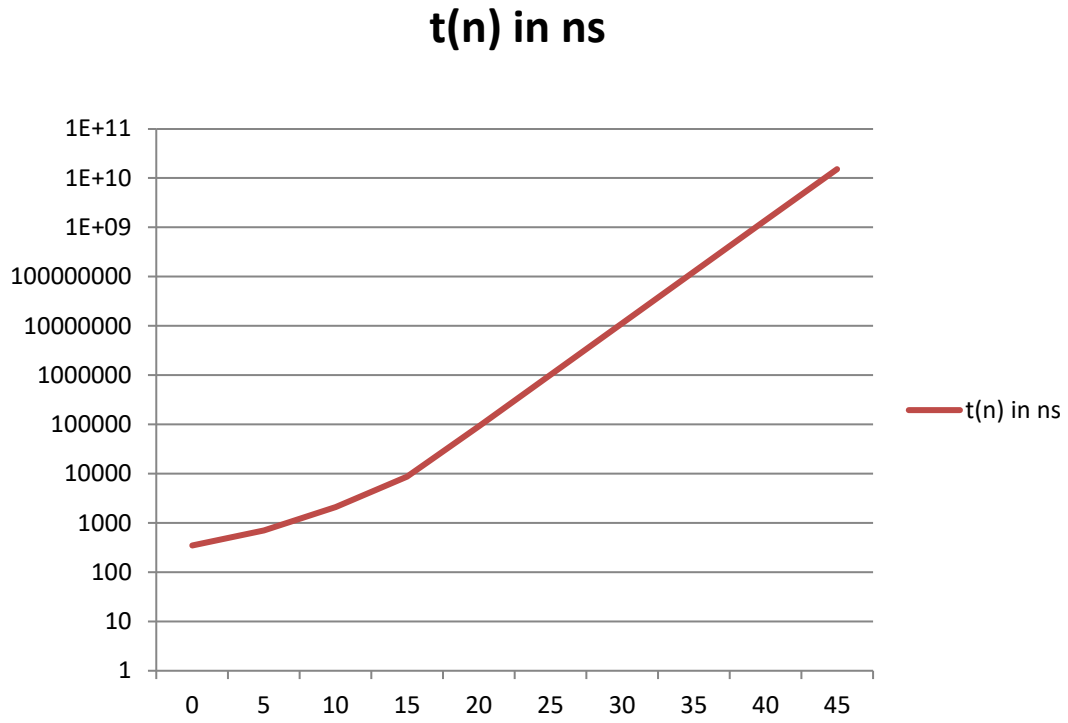
n	t(n)
0	100
1	200
2	400
3	700
4	1200
5	1500
6	3200
7	4500
8	5700
9	9000
10	1100
11	1600
12	2700
13	4100
14	7000
15	10900
16	17200
17	29700



Halblogarithmische Auftragung

Ordinate hat logarithmische Skalierung

samrek(n)



log 1000 = 3
log 100 = 2
log 10 = 1
log 1 = 0

Der Graph einer Exponentialfunktion ist halblogarithmisch aufgetragen eine Gerade.

Schritte zählen

Nach jeder Anweisung wird das Attribut schrittzaehler um eins (evtl. auch gewichtet) erhöht.

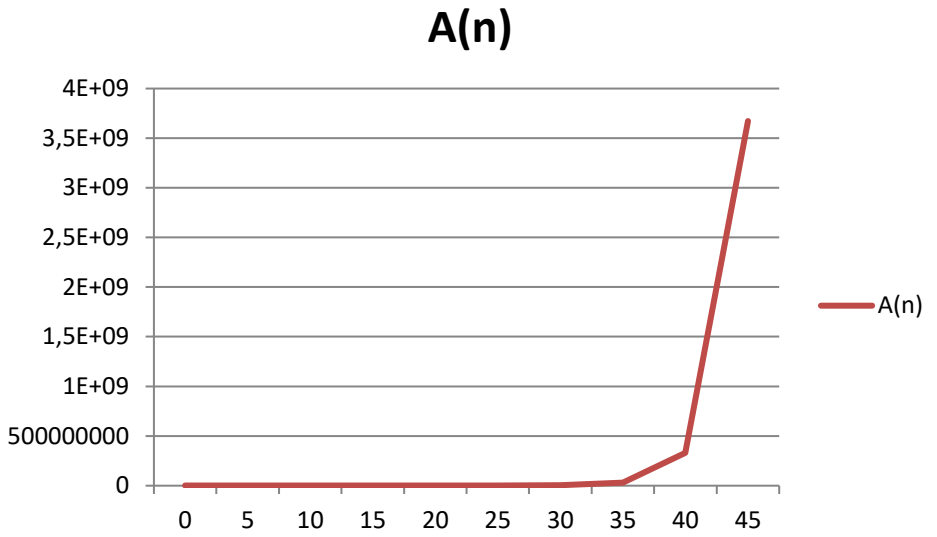
- sagt nichts über den absoluten Zeitverbrauch
- unabhängig von konkretem Rechner und Unterbrechungen durch andere Prozesse

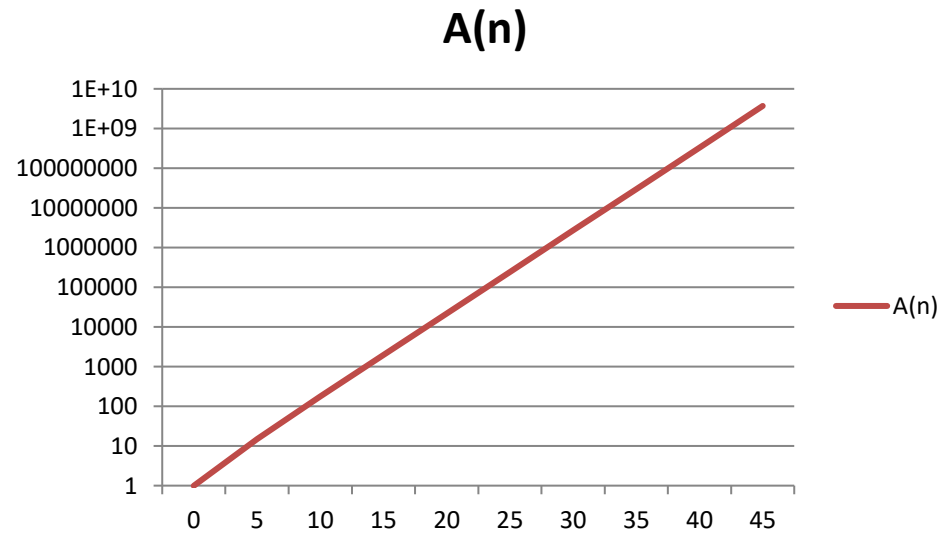
```
public long schritteZaehlen(int n) {
        schrittzaehler = 0;
        samrek(n);
        return schrittzaehler;
    }
```

```
public double samrek(int n) {
      schrittzaehler++;
      if (n==0) {
         return w_0;
       }
       else {
          if (n==1) {
             return w_1;
          }
          else {
             return 1.8*samrek(n-1)-0.9*samrek(n-2)+1;
          }
       }
   }
```

Zähle für verschiedene n-Werte die Schritte von samrek und trage sie in eine Tabelle ein.

n	A(n)
0	1
1	1
2	3
3	5
5	15
10	177
15	1973
20	21891
25	242785
30	2692537
35	29860703
40	331160281
45	3672623805





Ordnung: $O(e^n)$

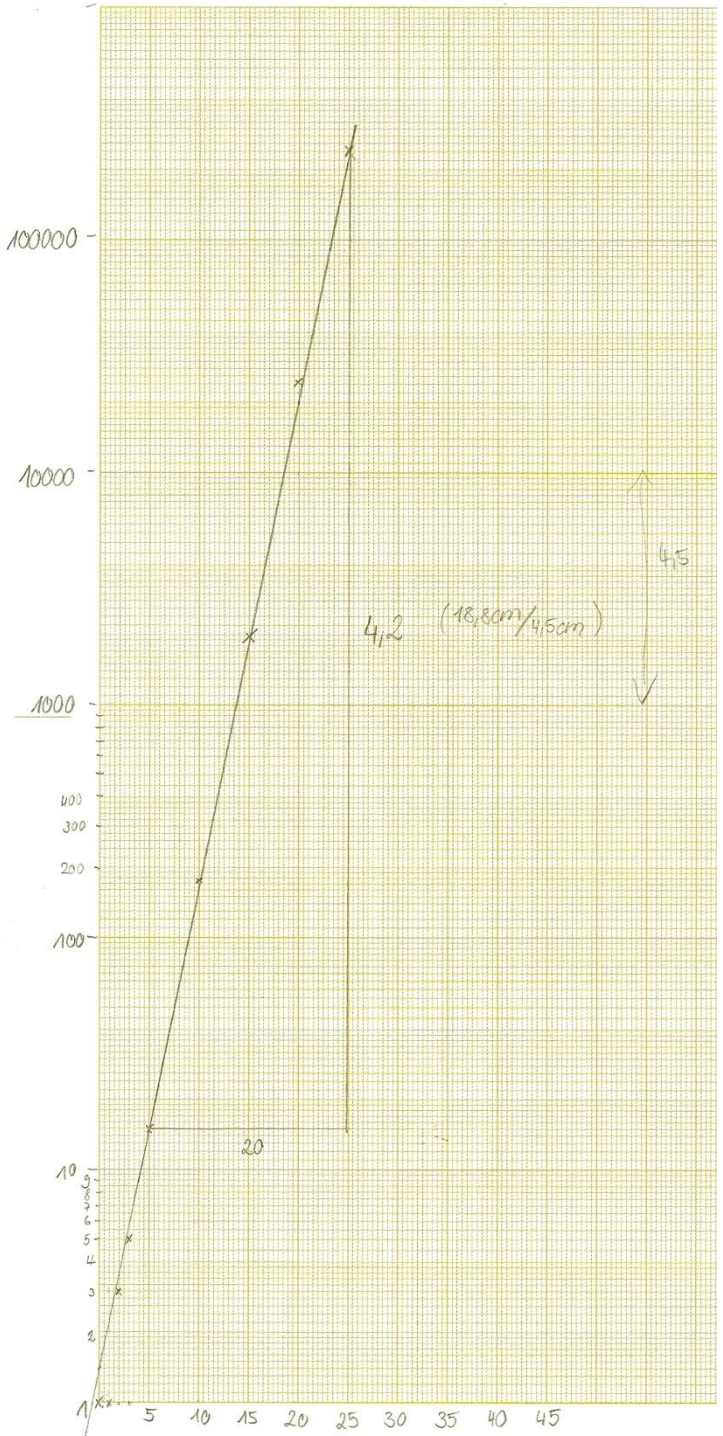
Landau-Symbol

http://www.papersnake.de/logarithmenpapier/einfachlogarithmisch/

Ermittle rechnerisch oder graphisch die Exponentialfunktion:

- rechnerisch:
  Ansatz: $f(x) = b \cdot a^x$
  2 Wertepaare einsetzen und a, b bestimmen
- graphisch:
  Geradengleichung ablesen: $y = mx + t$
  d.h. $\log f(x) = mx + t$,
  d.h. $f(x) = 10^{mx+t} = 10^t \cdot (10^m)^x$

Ergebnis: $f(x) \approx 1{,}446 \cdot 1{,}618^x$


100000
10000
1000
4,5
4,2 (18,8cm/4,5cm)
20
1
5
10
15
20
25
30
35
40
45

- rechnerisch:

$$f(x) = b \cdot a^x$$

$$f(15) = 1973 = b \cdot a^{15} \qquad I$$

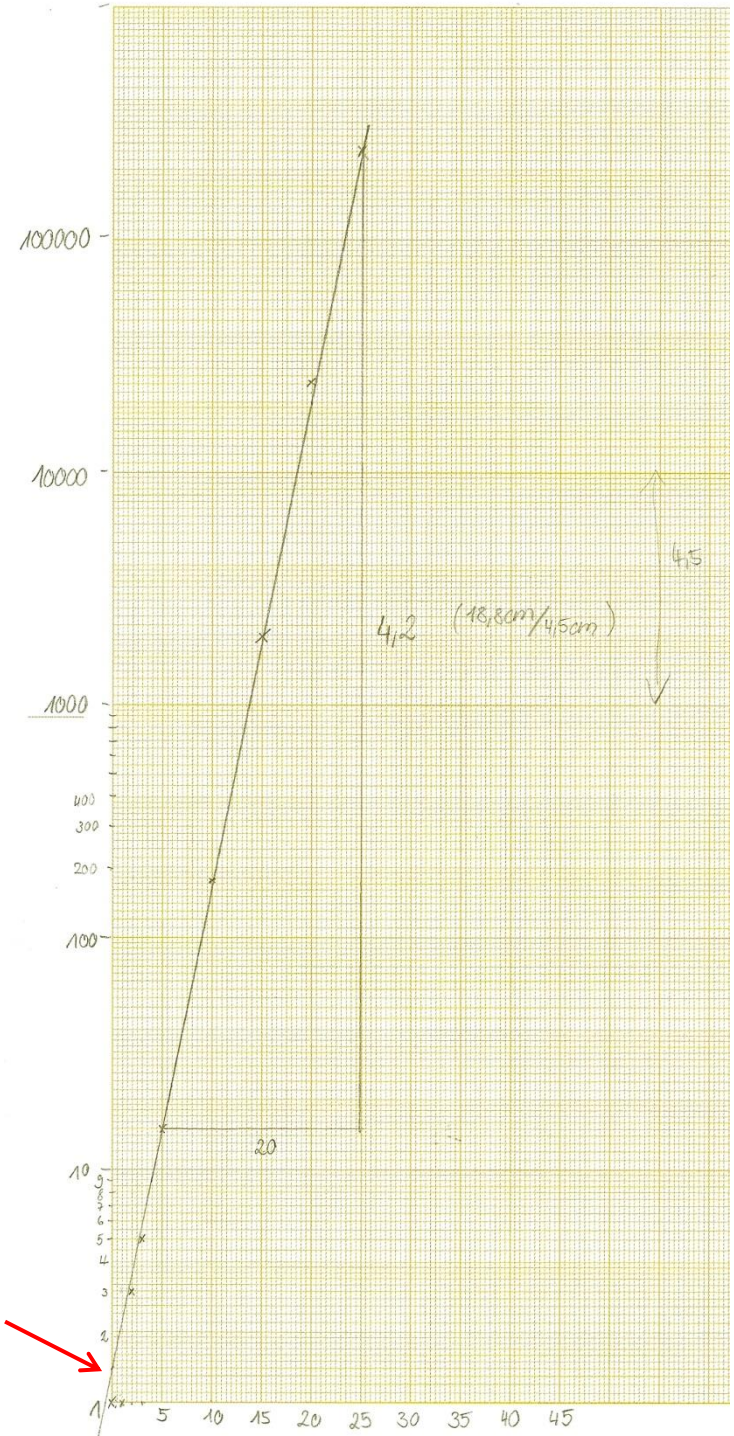
$$f(40) = 331160281 = b \cdot a^{40} \qquad II$$

$$II / I: \quad a^{25} = \frac{331160281}{1973} \approx 167846$$

$$a \approx 1,62$$

$$in\ I: \quad b = \frac{1973}{a^{15}} \approx 1,45$$

- graphisch:



y-Abschnitt messen:
entweder lineare Skala

$$t \approx \frac{0,7}{4,5} \approx 0,16$$

oder logarithmische Skala

$$t \approx \log 1,45 \approx 0,16$$

Steigung:

$$m = \frac{4,2}{20} = 0,21$$

Geradengleichung:

$$y = 0,21 \cdot x + 0,16$$

$$y = 0,21 \cdot x + 0,16$$

Da logarithmische Skala, gilt für die betrachtete Funktion:

$$f(x) = 10^{0,21 \cdot x + 0,16}$$

$$f(x) = \left(10^{0,21}\right)^{x} \cdot 10^{0,16} = 1,62^{x} \cdot 1,45$$

d.h. a=1,62, b=1,45

Insb:
Je größer die Steigung m der Geraden, desto größer die Basis a der Exponentialfunktion.

Implementiere die iterative Methode.
Welches Laufzeitverhalten ist zu erwarten?

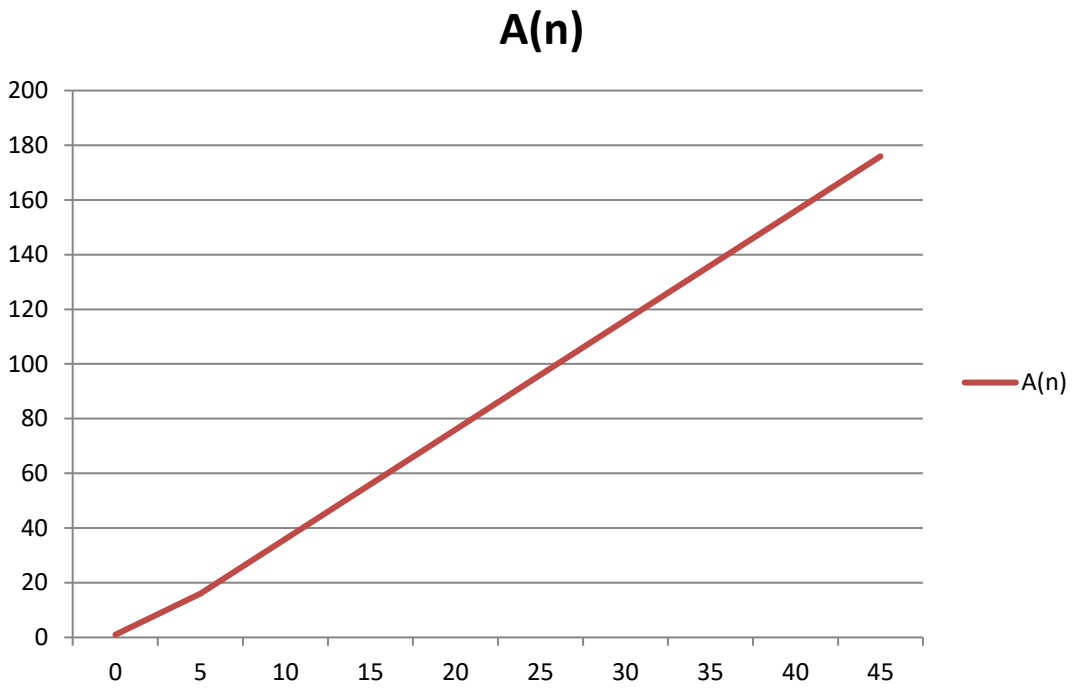
```
public double samit(int n){
    if (n==0) { return w_0;  }
    else {
        if (n==1){ return w_1;}
        else {
            double w_n = -1;
            for (int i=0; i<n-1; i++){
                w_n = 1.8 * w_1 - 0.9 * w_0 +1;
                w_0 = w_1;
                w_1 = w_n;
            }
            return w_n;
        }
     }
  }
```

samit(n)

```
public double samit(int n){
    if (n==0) {
        schrittzaehler++;
        return w_0;
    }
    else {
        if (n==1){
            schrittzaehler++;
            return w_1;
        }
        else {
            double w_n = -1;
            for (int i=0; i<n-1; i++){
                schrittzaehler++;
                w_n = 1.8 * w_1 - 0.9 * w_0 +1;
                schrittzaehler++; //eventuell stärker gewichten
                w_0 = w_1;
                schrittzaehler++;
                w_1 = w_n;
                schrittzaehler++;
            }
            return w_n;
        }
    }
}
```

Zähle für verschiedene n-Werte die Schritte von samit und trage sie in eine Tabelle ein.

n	A(n)
0	1
1	1
2	4
3	8
5	16
10	36
15	56
20	76
25	96
30	116
35	136
40	156
45	176



Ordnung: $O(n)$

Prüfe, dass eine evtl. Gewichtung der Rechenschritte, nichts an der Laufzeitordnung ändert.

…oder eine eigene Methode, die nur Schritte zählt und den gleichen Aufbau hat, wie die zu messende Methode:

```
public double samrek(int n) {
      schrittzaehler++;
      if (n==0) {
         return w_0;
       }
       else {
          if (n==1) {
             return w_1;
          }
          else {
             return 1.8*samrek(n-1)
               -0.9*samrek(n-2)+1;
          }
       }
   }
```

```
public long nurSchritte(int n) {

      if (n==0) {
         return 1;
       }
       else {
          if (n==1) {
             return 1;
          }
          else {
             return 1+nurSchritte(n-1)
               + nurSchritte(n-2);
          }
       }
   }
```

HA: S. 126 – 128 (knapp Mitte) lesen!
S. 130 / 1 Kombinationen

```
public class Kombinationen {
   public long fakultaet (int n) {
      if (n==1){
         return 1;
      }
      else {
         return n*fakultaet(n-1);
      }
   }

   public long schritteZaehlen(int n){
      if (n==1){
         return 1;
      }
      else {
         return 1+schritteZaehlen(n-1);
      }
   }
```

```
   public void tabelle(int weite){
      for (int i = 1; i< 10000; i = i+weite){
         System.out.println(i + ": " +
                  schritteZaehlen(i));
      }
   }
}
```

Lineare Funktion: O(n)

iterative Lösung:

```
public long fakit(int n){
        long erg = 1;
        for (int i = 1; i<=n; i++){
            erg = erg * i;
        }
        return erg;
    }
```

Vorteil: kein stack overflow