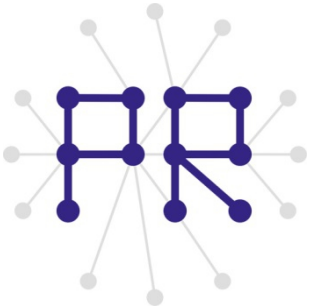
UNIVERSITÄT SIEGEN



Einführung in die Informatik I

Kapitel II.2: Spezielle Suchalgorithmen

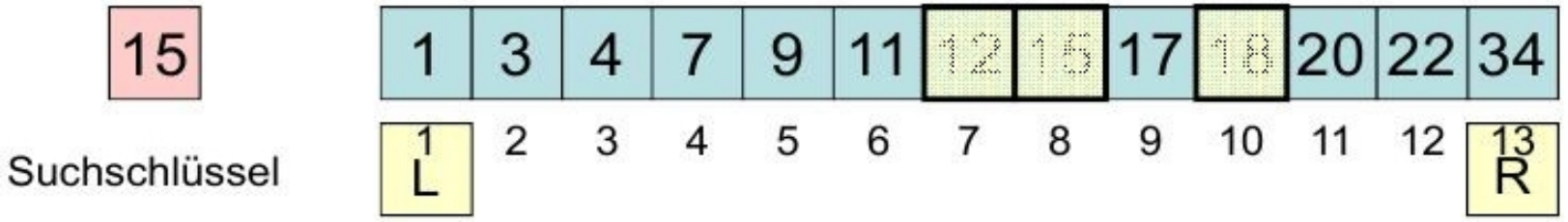
Prof. Dr.-Ing. Marcin Grzegorzek
Juniorprofessur für Mustererkennung im Institut für Bildinformatik
Department Elektrotechnik und Informatik
Fakultät IV der Universität Siegen

23.01.2013

Inhaltsverzeichnis

Binäre Suche (Bi-Sektion)

- Eine effiziente Suchmethode bei **sortierten** Daten ist das Bi-Sektionsverfahren (Divide-and-Conquer)
- Vorgehensweise:
  - Erstelle zwei **Grenzen** (Links=1 , Rechts=n)
  - Prüfe das Element in der **Mitte** des Feldes, ob es größer oder kleiner als das gesuchte Element ist
  - Verschiebe die linke bzw rechte Grenze um eins über die Mitte
  - Abbruch :
    - Rechts < Links
    - Mitte == Suchschlüssel



Binäre Suche: Details

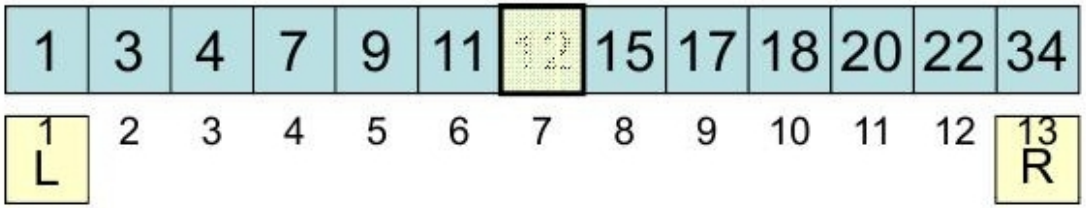
- **`Mitte=floor((Links+Rechts)/2);`**

15

Suchschlüssel

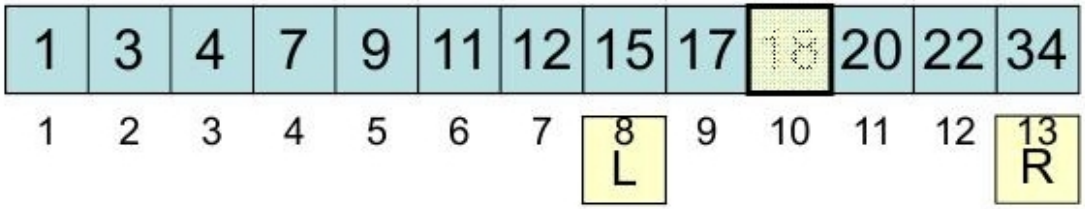
M=(13+1)/2=7

15<12 → L=M+1



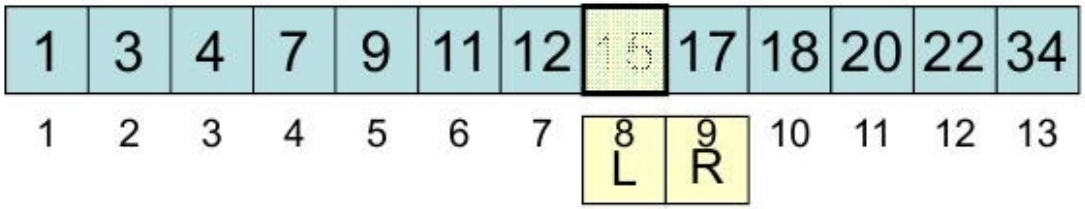
M=(13+8)/2=10.5

15<18 → R=M-1



M=(9+8)/2=8.5

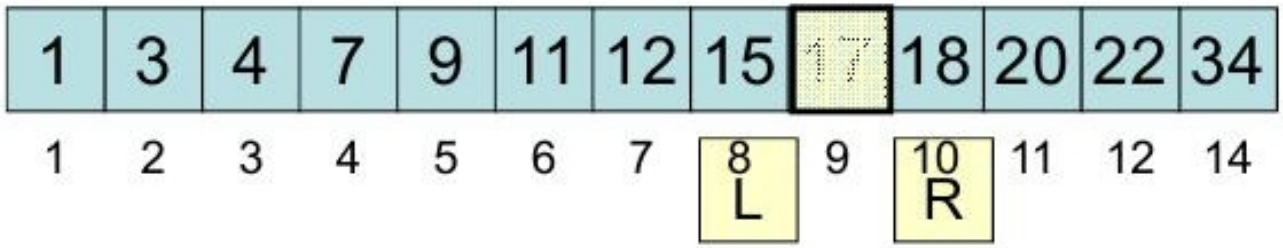
15==15 → **Gefunden!**



Algorithmus: Binäre Suche

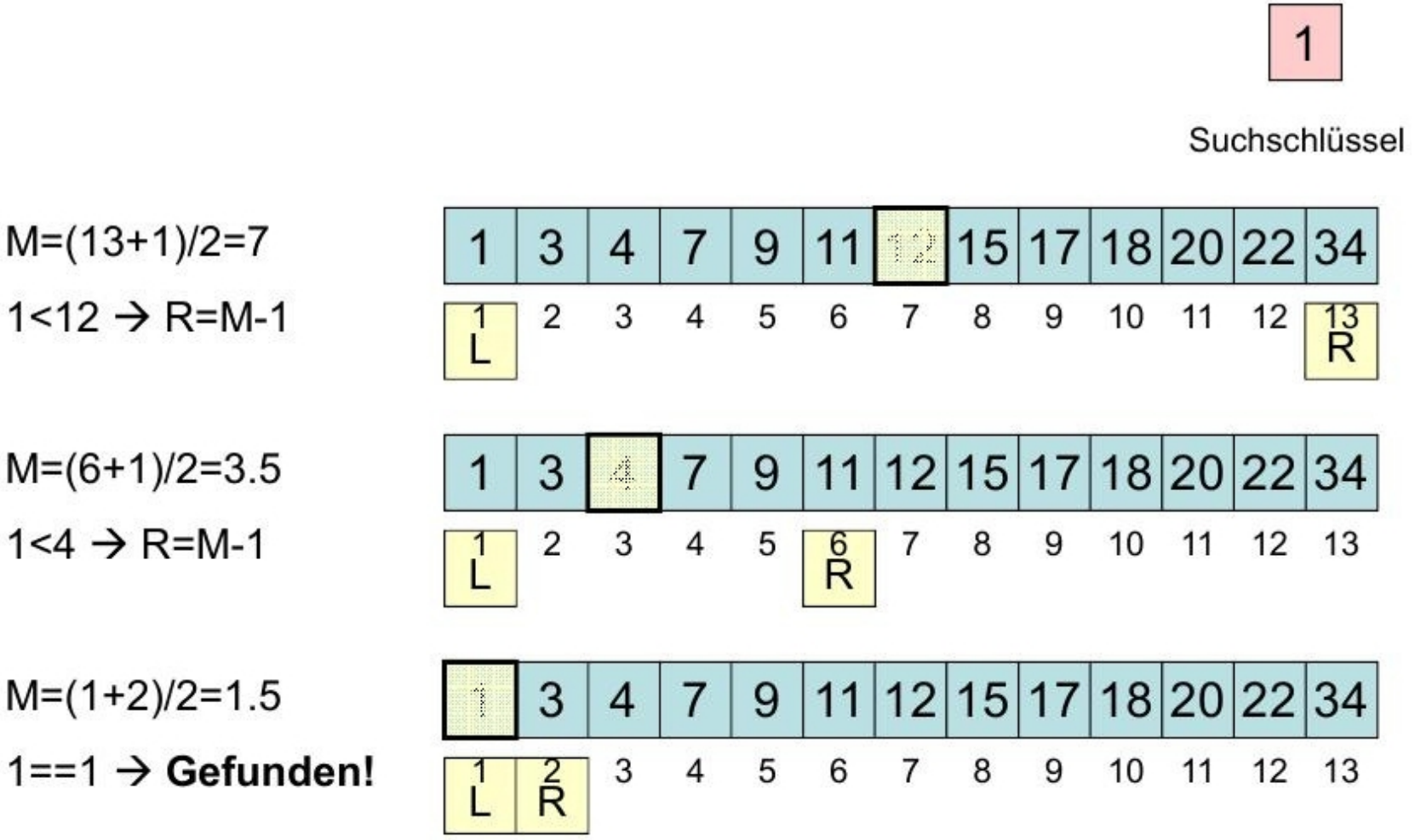
- Die **Mitte** zwischen der linken und rechten Grenze wird durch folgende **Formel** berechnet: (Mittelwert)

```
Mitte=floor((Rechts+Links)/2;
```

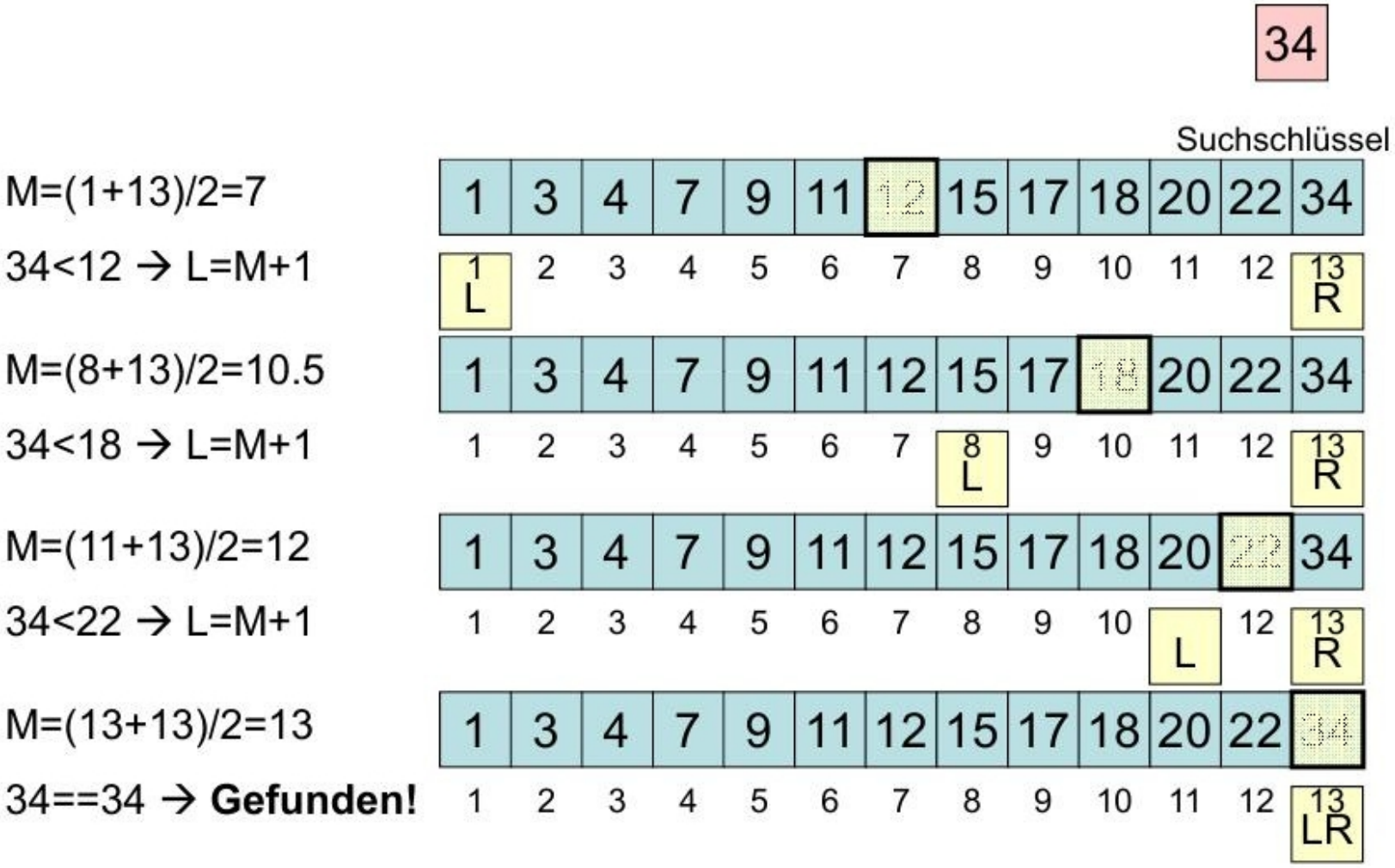


- **Problematisch** bei diesem Algorithmus sind folgende Punkte:
  - Was passiert an der linken Grenze?
  - Was passiert an der rechten Grenze?
  - Was passiert wenn der Suchschlüssel gar nicht in dem Feld vorhanden ist?

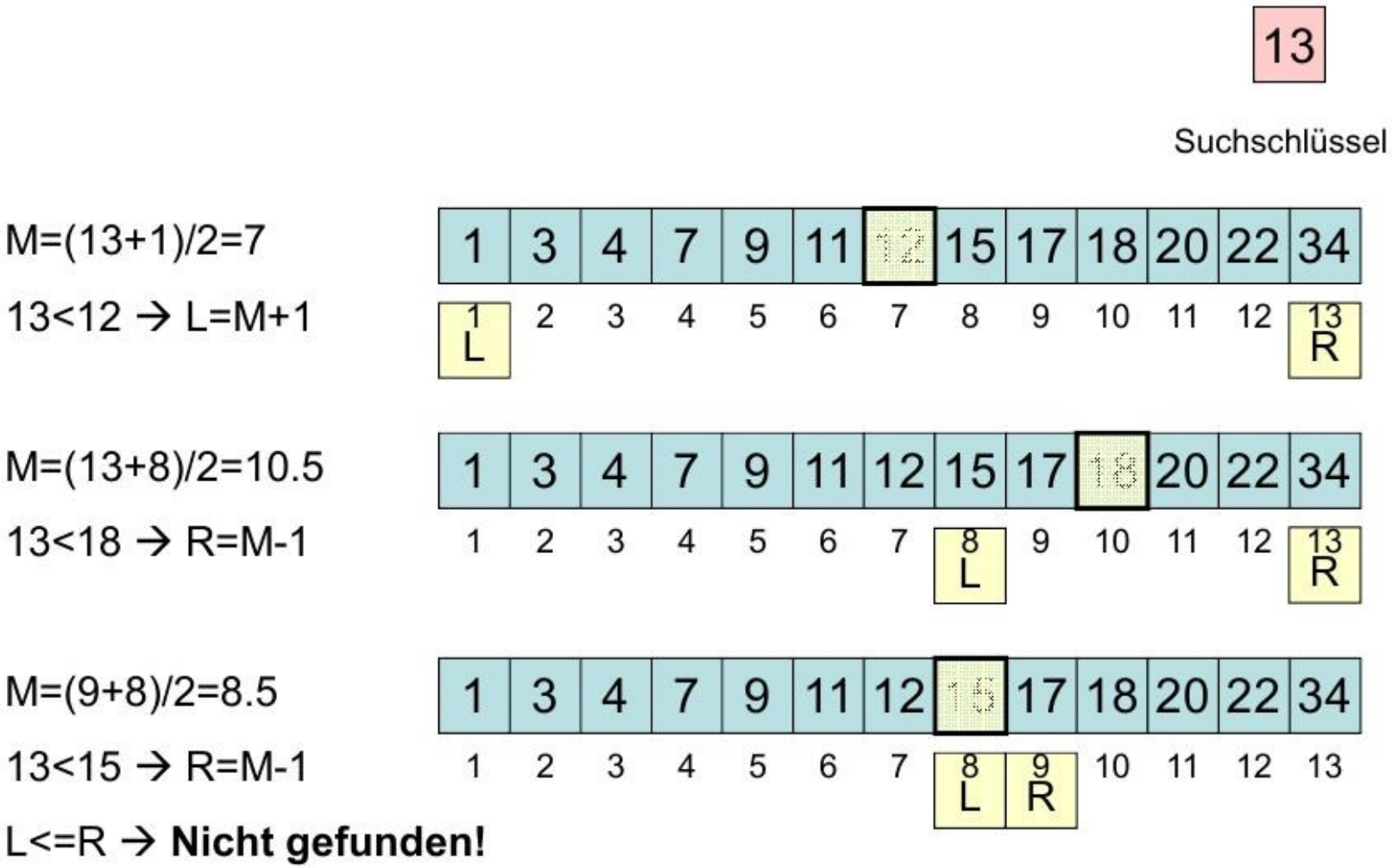
Binäre Suche: linke Grenze



Binäre Suche: rechte Grenze



Binäre Suche: erfolglos

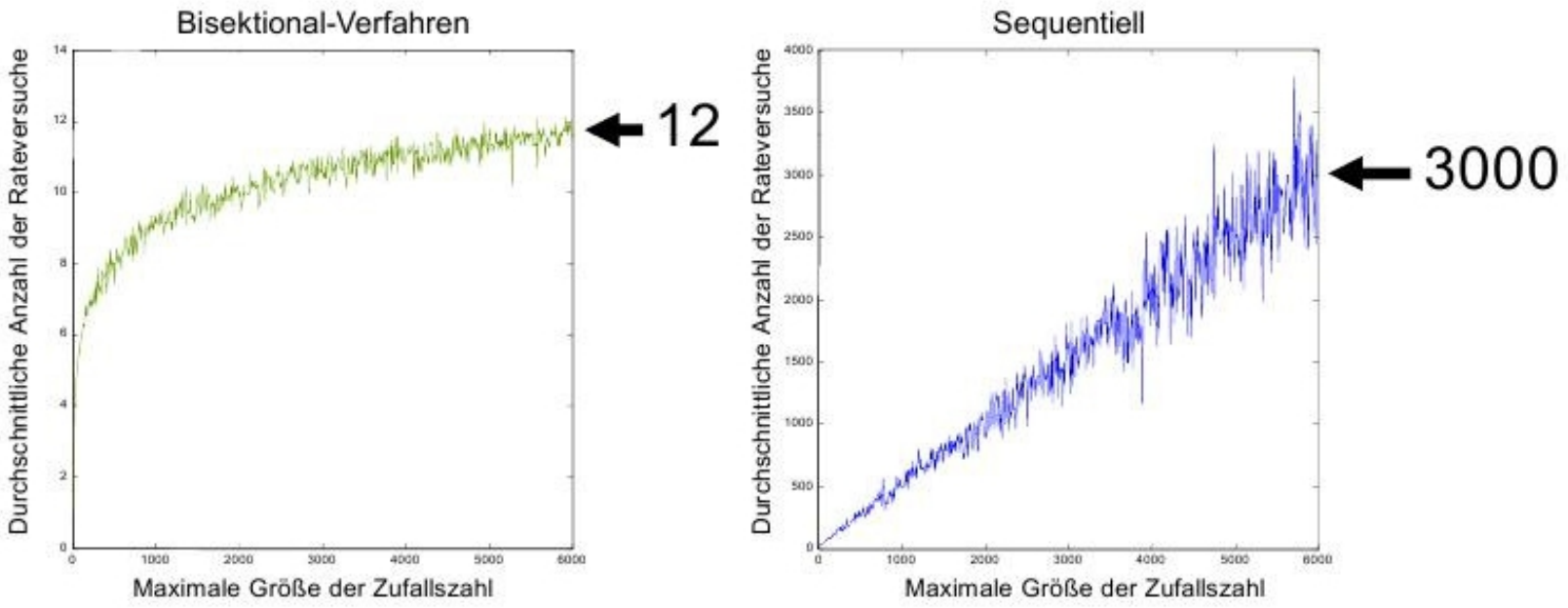


Binäre Suche: Algorithmus

```
function Position=Suche_BiSec(x,Key)
Links    = 1;                  % Erste Position
Rechts   = length(x);          % Letzte Position
Position = 0;                  % Default
while Links<=Rechts
    Mitte = floor((Links+Rechts)/2);
    if     x(Mitte)==Key       % gefunden..
        Position = Mitte;
        break;
    elseif x(Mitte)< Key       % nicht gefunden...
        Links    = Mitte+1;
    else
        Rechts   = Mitte-1;
    end;
end;
```

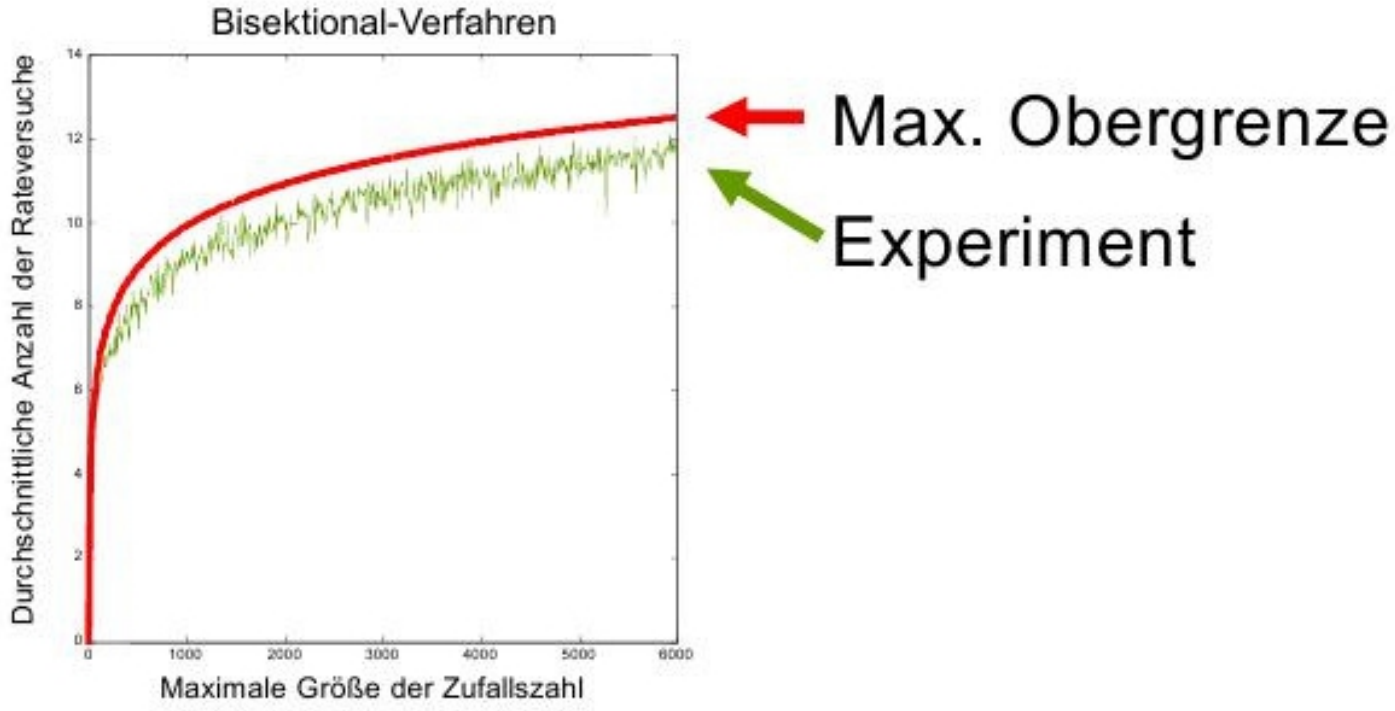
Analyse des Laufzeitenverhaltens

- Je 30 mal eine Zufallszahl im Intervall 1:n zu finden
- Der Mittelwert der 30 Versuche wird gebildet



Eigenschaften: Binäre Suche

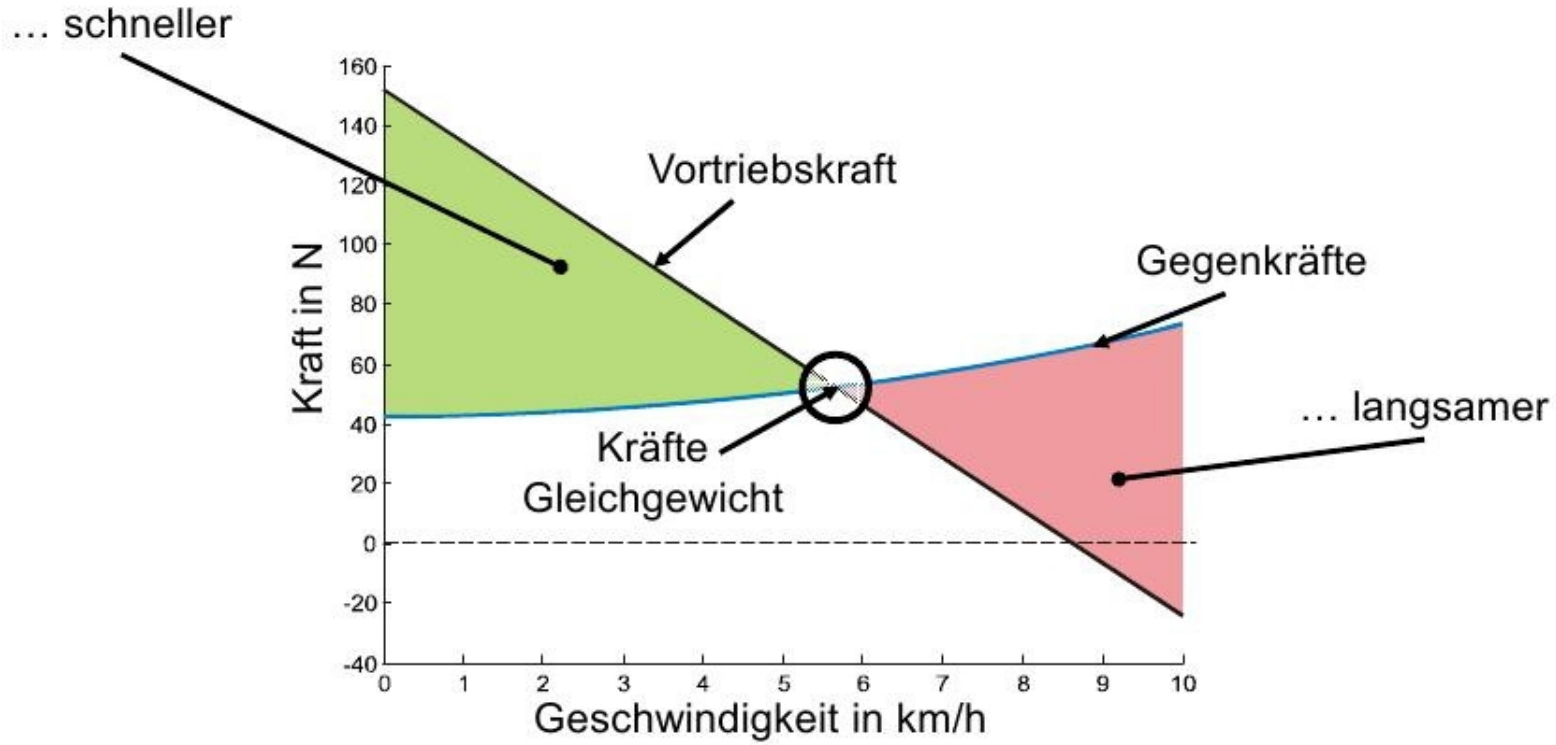
- Die Binäre Suche erfordert **niemals mehr** als $\log_2(n+1)$ Vergleiche
- Die Eigenschaft folgt unmittelbar aus der Tatsache, dass das Suchintervall **wenigstens halbiert** wird



Zusammenfassung: Suchen

Zugriff	Zufällig	Sequentiell	Bi-Sektion
Findet Schlüssel, wenn in der Liste	Ja	Ja	Ja
Abbruch, wenn nicht in der Liste	Nein	Ja	Ja
Voraussetzung: sortiert	Nein	Nein	Ja
Durchschnittliche Zahl der Versuche	n	n/2	$\log_2(n)$

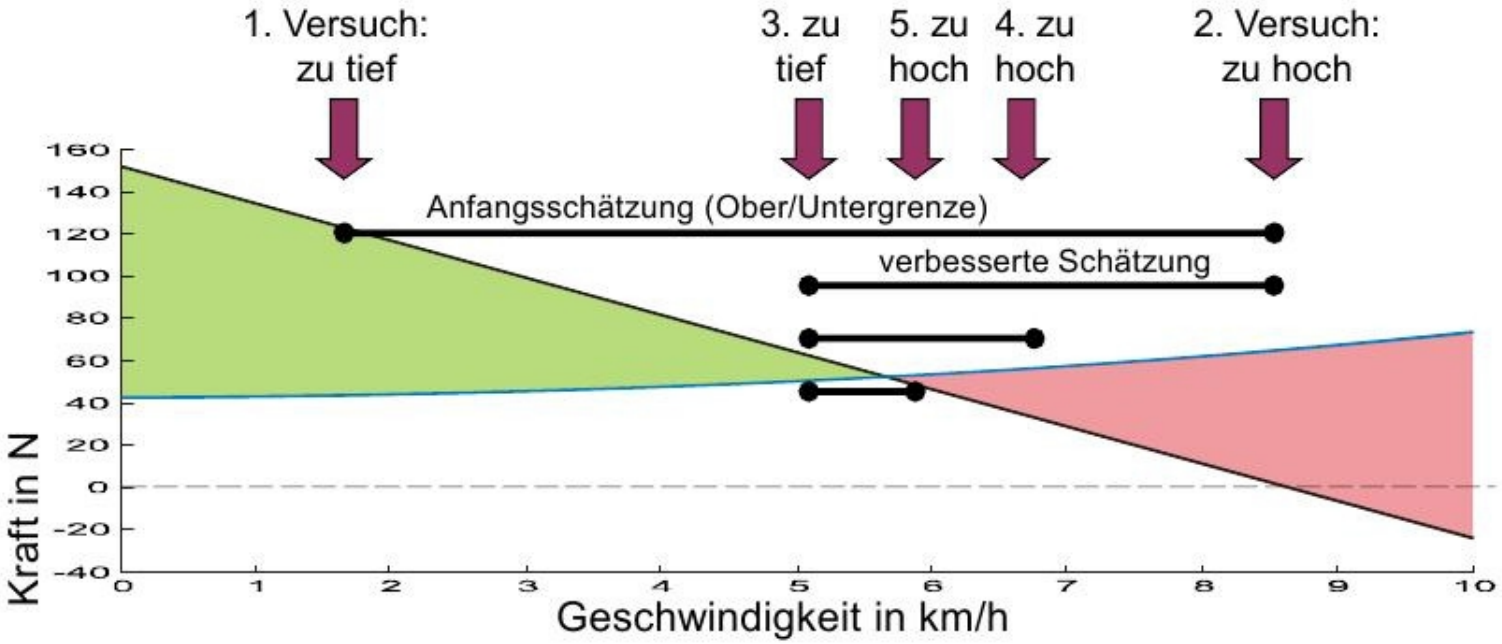
Fahrradbeispiel: Diagramm der Kräfte



- Am Kreuzungspunkt sind die Kräfte gleich groß:

$$F_{Vortrieb} = F_{Roll} + F_{Steigung} + F_{Luft}$$

Einschachteln des Gleichgewichts



- Das Kräftegleichgewicht findet man mit Hilfe einer Variante der binären Suche.
- Allerdings kann man das Ergebnis nur bis auf eine gewisse Genauigkeit ε bestimmen.

MATLAB Implementierung

```
% Eingaben
i  =input ('Übersetzung  :');
S  =input ('Steigung     :');
LG =input ('Linke  Grenze:');
RG =input ('Rechte Grenze:');
eps=input ('Genauigkeit  :');

% Anfangsberechnungen
F_LG=F_Ges(LG,S,i);
F_RG=F_Ges(RG,S,i);

% Anfangsberechnungen
F_LG=F_Ges(LG,Steig,i);
F_RG=F_Ges(RG,Steig,i);
```

```
% Fehlerabfrage
if F_LG<0 | F_RG>0
    error('Falsches Anfangsintervall');
else
    while RG-LG>epsilon
        % Mittelwert
        v_M=(RG+LG)/2 ;
        F_M=F_Ges(v_M,Steig,i);

        % Neue Intervallgrenzen
        if F_M>0
            LG  =v_M;  F_LG=F_M;
        else
            RG  =v_M;  F_RG=F_M;
        end
    end

    % Ergebnis
    disp (,Ergebnis zwischen'); LG
    disp ('und');               RG
end
```