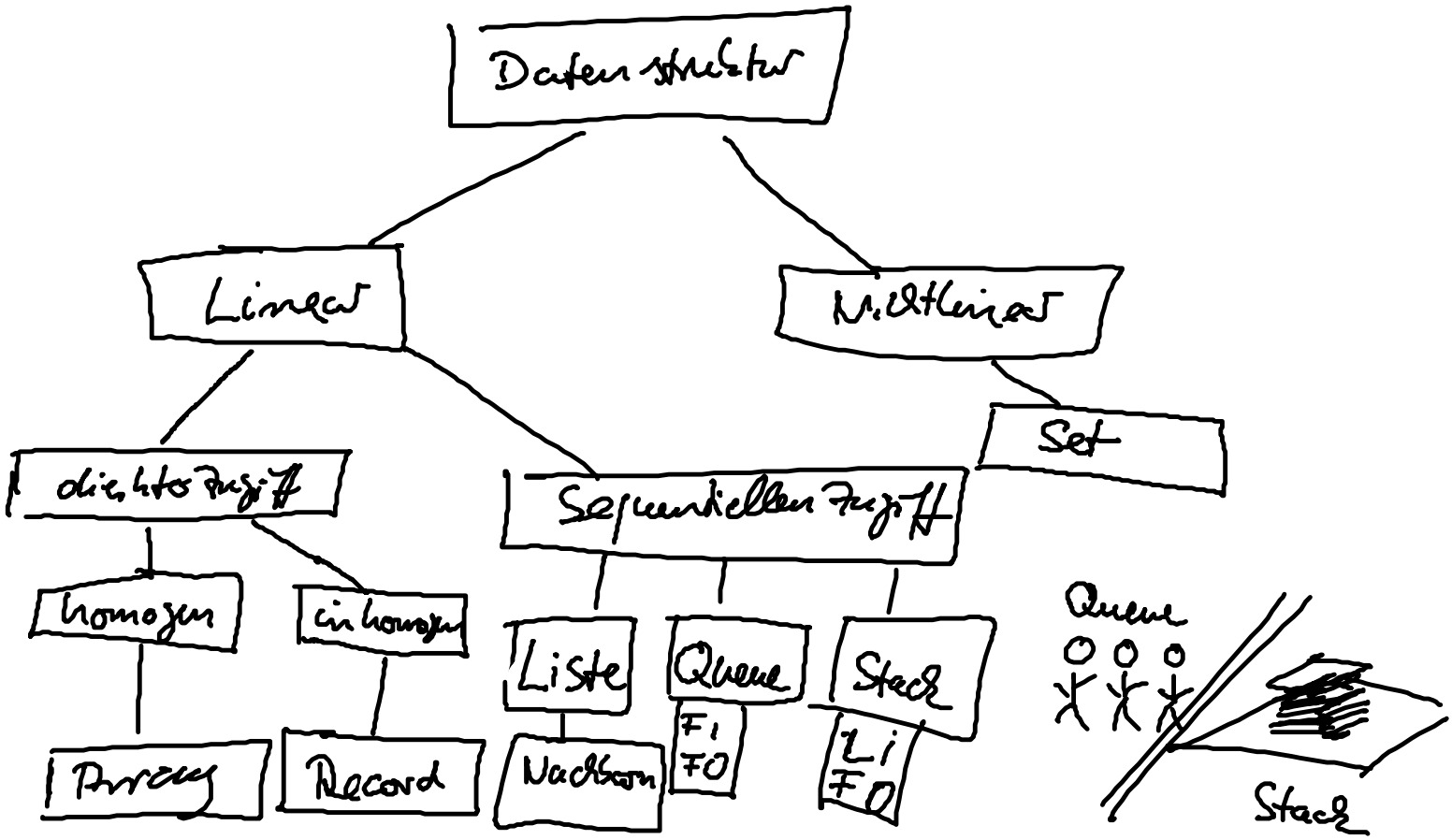
Datenstrukturen



Direkter Zugriff: Vorteile

(Algorithmen auf Arrays)

Sequentielle Suche:

Eingabe: Unsortiertes Array a, Wert x

Ausgabe: Index von dem x in Array steht oder -1 falls x nicht in Array.

S. Suche des Algorithmus Array bis gefunden oder Ende erreicht.
Aufwand im schlechtesten Fall: Länge des Arrays (n).

Binäre Suche:

Eingabe: Sortiertes Array a, Wert x
Ausgabe: wie oben

- Wähle mittleren Index $k = \lfloor \frac{n}{2} \rfloor$ vergleiche a[k] mit x.
- Falls x gefunden gib k zurück.
- Andernfalls:
  - a[k] < x => Suche in k+1 bis n
  - a[k] > x => Suche in 1 bis k-1

  (Hinweis: hier Array von 1 bis n indiziert! In Java immer 0 bis n-1)
- Wende das Verfahren auf die entsprechende Hälfte an. Falls diese leer ist gib -1 zurück.

3 | 5 | 6 | 8 | 10 | 12 | 13 | 16 x = 10
↑ 8 ≠ 10, 8 < 10

10 | 12 | 13 | 16
↑ 12 > 10

10
10 = x => k = 5 zurück geben

Java Methode:

```
/*
 * ....
 */
public int binarySearch (int[] a, int x) {

  int k; // variable for the index or -1

  int i, j // lower & upper bound on current search
           //   region in array
  int i = 0;
  int j = a.length - 1
        ~~this.a.length~~ - 1

  while ( i <= j ) {
    k = (i+j)/2;
    if (a[k] == x) return k; // found x hurray!
      if (x < a[k]) j = k-1;
      else   i = k+1;
  }
  return -1;

}
```

Beweis der Korrektheit:

Mittels Schleifeninvariante:

$a[i] \leq x \leq a[j]$ und $0 \leq i \leq j \leq n-1$

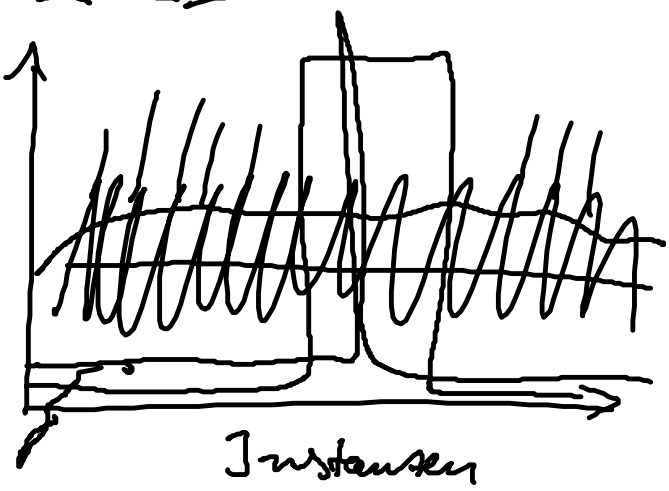
$\Longleftrightarrow$ Korrektheit mit „Array sortiert".

Satz (Aufwand der binären Suche)

Sei $V(n)$ die maximale Anzahl von Vergleichen ($a[k] == x$), die die b.S. auf einem Array der Länge n durchführen muss.

Dann gilt:

$$V(n) \leq \lfloor \log_2 n \rfloor + 1$$



Beweis:

durch Induktion nach n:

zu zeigen $V(n) \leq \lfloor \log n \rfloor + 1$

IA: $n = 1$

$1 + \lfloor \log_2 1 \rfloor = 0 + 1 = 1$

IVor: für Länge des Arrays $< n$ gilt die Beh.!

IS: Nach dem 1. Vergleich muss

entweder - nicht

oder - eine Hälfte betrachtet werden

$$V(n) \leq 1 + V\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$$ (weil Element in der Mitte erledigt)

IVor: $V(\lfloor n/2 \rfloor) \leq \lfloor \log(\lfloor \frac{n}{2} \rfloor) \rfloor + 1$

$$V(n) \leq 1 + \left\lfloor \log_2 \left(\left\lfloor \frac{n}{2} \right\rfloor \right) \right\rfloor + 1$$

$\leq 1 + \log_2\left(\frac{n}{2}\right) + 1$ (IV, log monoton)

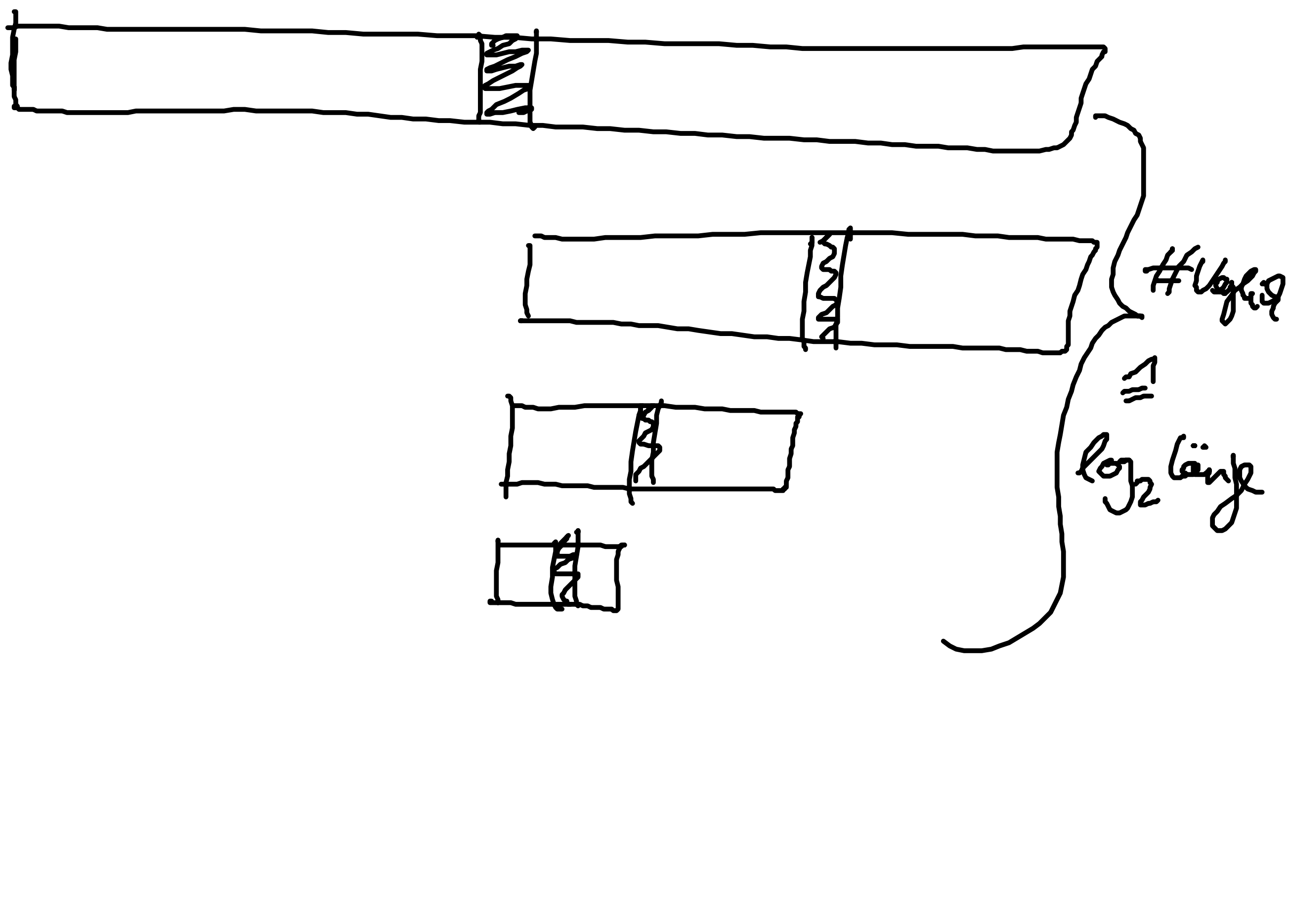
$= 1 + \log_2\left(\frac{n}{2}\right) + \log_2 2$

$= 1 + \log\left(\frac{n}{2} \cdot 2\right)$

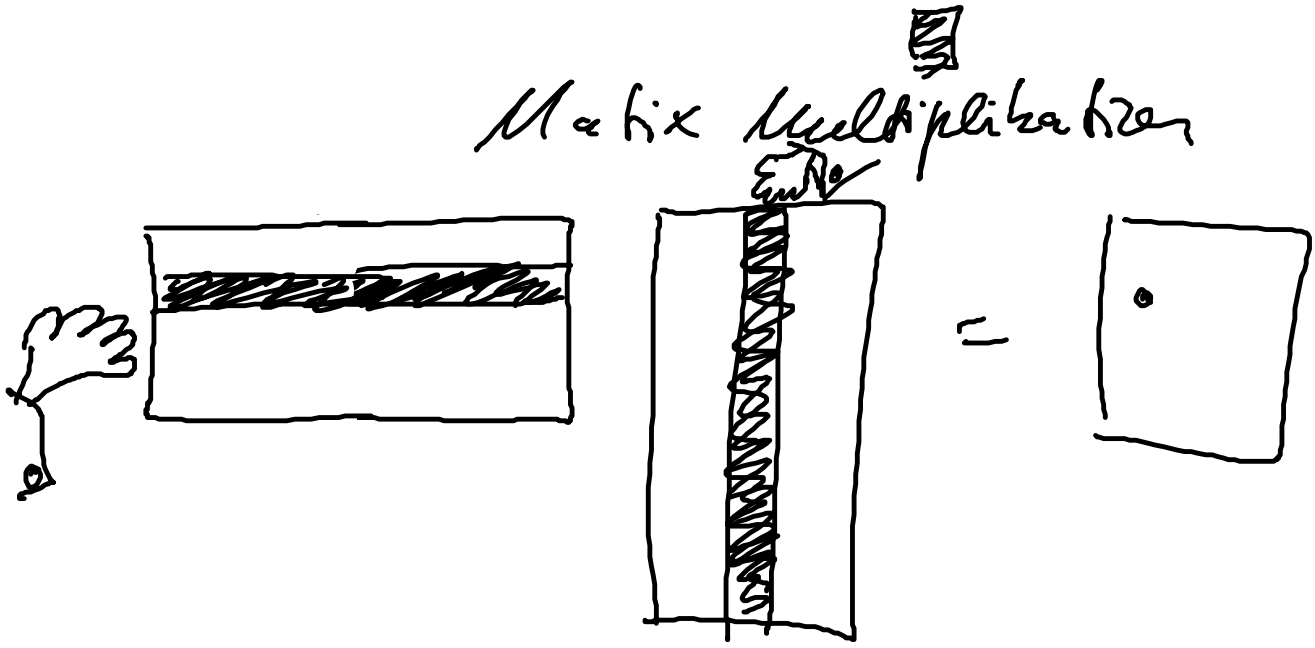
$= 1 + \log(n)$

$V(n) \leq 1 + \lfloor \log(n) \rfloor$ //weil $V(n)$ ganzz.

$\square$



Matrix Multiplikation



```
double [][] a = new double[m][p];
"      "   "  b = "      "[p][n];
"      "   "  c = "      "[m][n];
   ⋮
for( int i=0; i<m; i++){
    for(int j=0; j<n; j++){
     c[i][j]=0;
     for( int k=0; k<p; k++){
      c[i][j] += a[i][k] * b[k][j];
     }
    }
}
```

$\sum_{k=1}^{p} a_{ik} \cdot b_{kj}$

Skalarpr. zweier p dimensionaler Vektoren

Sequentieller Zugriff

Listen Objekte :

Arrayfeld :