Fachbereich Medienproduktion

- Herzlich willkommen zur Vorlesung im Studienfach:
  - Grundlagen der Informatik

Themenübersicht

- Grundlagen der Informatik
  - Grundlagen der Rechnertechnik
  - Algorithmen und Datenstrukturen
    - Primitive Datentypen
    - Abstrakte Datentypen
    - Algorithmen
  - Rechnernetze und das Internet
  - IT Sicherheit

Algorithmen

- Verfahren zur Lösung eines bestimmten Problems
- Müssen „endlich“ sein, um ein Computerprogramm zu nutzen

Beispiel „euklidischer Algorithmus“
2300 Jahre altes Verfahren zur Bestimmung des größten gemeinsamen Teilers zweier Zahlen

Nicht-negative Zahlen p & q	int **gcd**(int p, int q) {
Wenn q gleich 0, ist p die Antwort.	if (q== 0) return p;
Sonst, teile p durch q und bestimme Rest r.	int r = p % q;
Wiederholung mit q und r.	return gcd(q, r); } //Rekursion

Beispiele für Algorithmen

- Mathematische Bibliothek
  - Wurzel, Sinus-Funktionen, usw.
- Kryptografische Bibliothek
  - Verschlüsselung, Hashing, Signierung
- Sortieralgorithmen
  - Eine der häufigsten Aufgaben bei der Datendarstellung
  - Als Vorbereitung für weitere Algorithmen zur Datenverarbeitung
- Suchalgorithmen
  - Auffinden von Daten

Selectionsort: einfache Sortierung

Aufgabe: Sortierung eines INT-Arrays nach aufsteigender Wertigkeit

Lösung: Suche des nächst kleinstem Elements und Vertauschung mit der neue Stelle im Array

```
void sort( int[] a)
{
    int N = a.length;
    for (int i = 0;    i < N;    i++)
    {
        int min = i;   // Index des minimal-Wertes
        for (int j = i+1;    j < N;    j++)
        {
            if (a[j] < a[min])
                min= j;        // neuer minimal-Wert
        }
        // vertausche a[i] und a[min]
        int temp=a[i];
        a[i] = a[min];
        a[min]=temp;
    }
    return;
}
```

Insertionsort: Kartensortieren

Aufgabe: Sortierung eines INT-Arrays nach aufsteigender Wertigkeit

Lösung: Suche des nächst kleinstem Elements und Einfügen an neue Stelle im Array durch Verschieben aller anderen Elemente

```
void sort( int[] a)
{
    int N = a.length;
    for (int i = 1;    i < N;    i++)
    {
        for (int j = i;  j > 0 ;  j--)
        {
            if (  a[j] < a[j-1])  // prüfe vorherigen Wert
            {
               // Vertausche a[j] und a[j-1]
               int temp=a[j];
               a[j] = a[j-1];
                a[j-1]=temp;
            }
            else
                break;  //innere Schleife abbrechen
        }
    }
    return;
}
```

Bewertung der Sortieralgorithmen

Selectionsort

- Vergleichsoperationen: ~N²/2
- Tauschoperationen: N

- Ausführungszeit unabhängig von der Ausgangsordnung
- Minimale Datenverschiebung

Insertionsort

Worst-Case

- Vergleichsoperationen: ~N²/2
- Tauschoperationen: ~N²/2

Best-Case

- Vergleichsoperationen: N-1
- Tauschoperationen: 0

- Kurze Ausführungszeit bei vorsortierten Daten

Weitere Sortieralgorithmen

- Shellsort
- Mergesort
- Heapsort
- Quicksort

Sequential Search

Aufgabe: Auffinden eines Elements mit vorgebendem „key“

Lösung: Suche alle Elemente einer Datenmenge nach „key“ ab

```
int search(int[] a,   int key )
{
      int N = a.length;
      for (int i = 0;   i < N;   i++)
      {
            if ( a[i] == key ) // Prüfe element gleich key
                  return i;   // index als Rückgabewert
      }
      return -1;
}
```

Binary Search

Aufgabe: Auffinden eines Elements mit vorgebendem „key“

Vorrausetzung: Sortiertes Array liegt bereits vor

Lösung: Beginne Suche in der Mitte und prüfe ob key oberhalb oder unterhalb liegt, Wiederholung mit jeweils nächsten Abschnitt

```
int search(int[] a,   int key )
{
      int lo = 0;
      int hi = a.length -1;

      while ( lo <= hi )
      {
            int mid = lo + (hi – lo) / 2; //Mittleren Index
            if (key < a[mid])
                  hi = mid -1;
            else if (key > a[mid])
                  lo = mid + 1;
            else
               return mid;
         }
         return -1;
}
```

- Vielen Dank für Ihre Aufmerksamkeit!