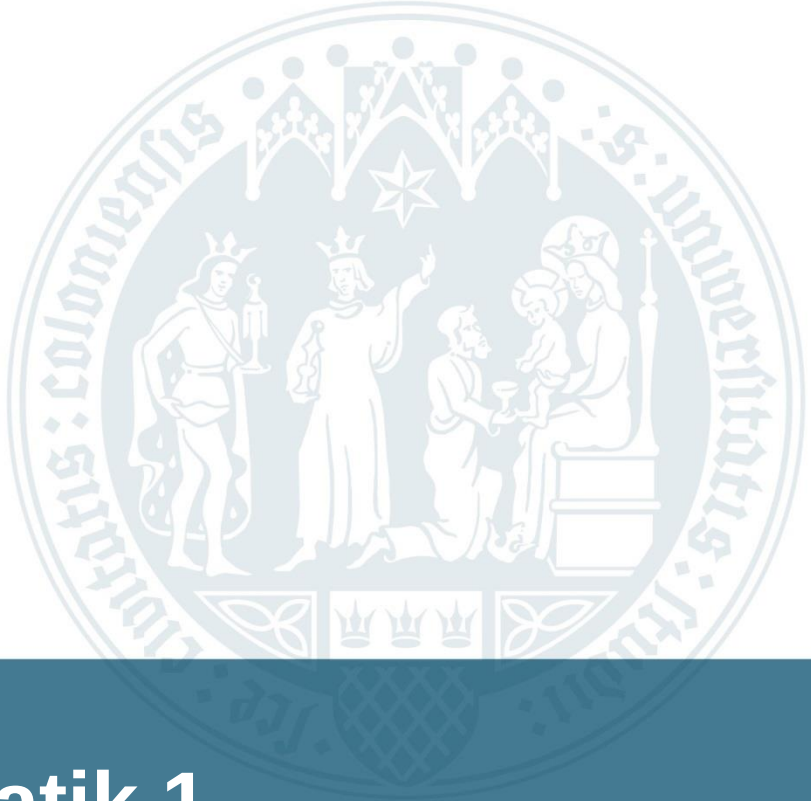
Universität zu Köln



Grundzüge der Informatik 1

Vorlesung 13 - flipped classroom

Dynamische Programmierung

Aufgabe 1

- Eine Firma kann unterschiedliche Produkte aus zwei Ressourcen herstellen. Produkt i benötigt eine Menge r[i] von Ressource 1 und t[i] von Ressource 2 und hat einen Wert von w[i]. Die Einträge in r und t sind nichtnegativ und ganzzahlig. Einer der beiden Einträge r[i] und t[i] ist positiv.
- Angenommen, die Firma kann n unterschiedliche Produkttypen herstellen, deren Werte und Ressourcenbedarf in den Feldern r[1..n], t[1..n] und w[1..n] gegeben sind und sie verfügt über T Einheiten von Ressource 1 und S Einheiten von Ressource 2. Welche Produkte sollte die Firma herstellen, um ihren Gewinn zu maximieren?
- Finden Sie eine Rekursionsgleichung für den Erlös Opt(k,i,j), den die Firma mit der Produkttypen 1 bis k mit vorhandenen Ressourcen i und j erzielen kann.
- Entwickeln Sie einen Algorithmus für die Berechnung des optimalen Erlöses.

Dynamische Programmierung

Rekursion

- Opt(k,i,j) = max {w[k] + Opt(k,i-r[k],j-t[k]), Opt(k-1,i,j)}, wenn r[k]≤i und t[k]≤j
- Opt(k,i,j) = Opt(k-1,i,j), sonst
- Opt(0,i,j) = 0

Dynamische Programmierung

Erlös(n,R,T)

```
1.  Opt = new array [0,..,n][0..R][0,..,T]
2.  for i = 0 to R do                    \* Rekursionsabbruch
3.     for j=0 to T do
4.        Opt[0][i][j] = 0
5.  for k = 1 to n do
6.     for i = 0 to R do
7.        for j=0 to T do
8.           if r[k] ≤i und t[k] ≤j then
                Opt[k][i][j] = max{Opt[k-1][i][j], w[k] + Opt[k][i-r[k]][j-t[k]]}
9.           else Opt[k][i][j] = Opt[k-1][i][j]
10. return Opt[n][R][T]
```

Gierige Algorithmen

Entwurfsprinzip „Gierige Algorithmen“

- Ziel: Lösung eines Optimierungsproblems
- Herangehensweise: Konstruiere Lösung Schritt für Schritt, indem immer ein einfaches „lokales“ Kriterium optimiert wird
- Vorteil: Typischerweise einfache, schnelle und leicht zu implementierende Algorithmen

Beobachtungen

- Gierige Algorithmen optimieren einfaches lokales Kriterium
- Dadurch werden nicht alle möglichen Lösungen betrachtet
- Dies macht die Algorithmen oft schnell
- Je nach Problem und Algorithmus kann die optimale Lösung übersehen werden

Gierige Algorithmen

Aufgabe 2

- Gegeben: Menge von reellwertigen Punkten $\{x_1,\ldots,x_n\}$ im 1D (abgespeichert in einem Feld A)
- Gesucht: Die minimale Anzahl Intervalle der Länge 1, die ausreicht, um die Punkte zu überdecken
- Was ist die Laufzeit Ihres Algorithmus?

Gierige Algorithmen

Algorithmus GierigeIntervalle (Array A[1…n])

- Sortiere die Zahlen in A aufsteigend
- i=1; Anz=0
- **while** i ≤ n **do**
- L=A[i]; i=i+1; Anz=Anz+1
- **while** A[i]≤L+1 und i ≤ n **do**
- i=i+1

Laufzeit: O(n log n)