Grundzüge der Informatik 1

Vorlesung 12 - flipped classroom

Dynamische Programmierung

Dynamische Programmierung für Optimierungsprobleme

1. Bestimme rekursive Struktur einer optimalen Lösung durch Zurückführen auf optimale Teillösungen

2. Entwerfe rekursive Methode zur Bestimmung des *Wertes* einer optimalen Lösung.

3. Transformiere rekursive Methode in eine iterative (bottom-up) Methode zur Bestimmung des Wertes einer optimalen Lösung.

4. Bestimmen aus dem Wert einer optimalen Lösung und in 3. ebenfalls berechneten Zusatzinformationen eine optimale Lösung.

Dynamische Programmierung

Aufgabe 1

- Betrachten Sie die Modifikation von Jump, bei der auch Sprünge über 3 Felder erlaubt sind
- Stellen Sie zunächst eine Rekursionsgleichung auf

Dynamische Programmierung

Rekursionsgleichung

- Score(n) = A[n] + max(Score(n-1), Score(n-2), Score(n-3))
- Score(1) = A[1]
- Score(2) = A[1] + A[2]
- Score(3) = max(A[1]+A[3], A[1]+A[2]+A[3])

Dynamische Programmierung

Aufgabe 2

- Betrachten Sie die Modifikation von Jump, bei der auch Sprünge über 3 Felder erlaubt sind
- Entwickeln Sie mit Hilfe von dynamischer Programmierung einen Algorithmus der für diese Variante den Wert einer optimalen Lösung berechnet

Dynamische Programmierung

Score3DP(A,n)

1. Score = new array [1..n]
2. Score[1] = A[1]
3. Score[2] = A[1] + A[2]
4. Score[3] = max(A[1]+A[3], A[1] + A[2] + A[3])
5. **for** i=4 **to** n **do**
6.   Score[i] = A[i] + max(Score[i-1],Score[i-2], Score[i-3])
7. **return** Score[n]

Dynamische Programmierung

Aufgabe 3

- Betrachten Sie die Modifikation von Jump, bei der auch Sprünge über 3 Felder erlaubt sind
- Entwickeln Sie dann einen Algorithmus, der eine optimale Lösung berechnet

Dynamische Programmierung

```
ScoreLösung(A,Score,n)
if n=1 then return {1}
if n=2 then return {1, 2}
if n=3 then
  if A[1]+A[2]+A[3] > A[1]+A[3] return {1, 2, 3}
  else return {1,3}
if Score[n] = A[n] + Score[n-1] then return {n} ∪ ScoreLösung(A, Score, n-1)
else
  if Score[n] = A[n] + Score[n-2] then
    return {n} ∪ ScoreLösung(A, Score, n-2)
  else return {n} ∪ ScoreLösung(A, Score, n-3)
```

Dynamische Programmierung

Aufgabe 4

- Welche der folgenden Formeln sind gültige Rekursionen? Geben Sie (wenn möglich) eine iterative Implementierung an, die F(n) bzw. F(n,m) berechnet.

(a) $F(n) = n + \min\{F(n-1), F(n+1)\}$, wenn $n>1$
$F(1) = 1$

(b) $F(n) = 100 + F(n-2)$, wenn $n>2$
$F(2) = 1$
$F(1) = 1$

(c) $F(n,m) = 1 + \min\{F(n-1,m-1), F(n-1,m)\}$, wenn $n,m>1$
$F(n,1) = 20$, wenn $n>1$
$F(1,m) = 10$

(d) $F(n,m) = nm + \min\{F(n-1,m+1), F(n,m-1)\}$, wenn $n,m>1$
$F(1,m) = 1$, wenn $m>1$
$F(n,1) = 0$

Dynamische Programmierung

Aufgabe 4

(a) $F(n) = n + \min\{F(n-1), F(n+1)\}$, wenn $n>1$
$F(1) = 1$

- Dies ist keine gültige Rekursion, da man für die Berechnung von F(n) den Wert von F(n+1) benötigt und dies nie zu einem Rekursionsabbruch führt

Dynamische Programmierung

Aufgabe 4

(b) $F(n) = 100 + F(n-2)$, wenn $n>2$
$F(2) = 1$
$F(1) = 1$

Rekursion(n)

1. F = **new** array[1...n]
2. F[1] = 1
3. F[2] = 1
4. **For** i=3 **to** n **do**
5.   F[i] = 100+F[i-2]
6. **return** F[n]

Dynamische Programmierung

Aufgabe 4

(c) $F(n,m) = 1 + \min\{F(n-1,m-1), F(n-1,m)\}$, wenn $n,m>1$
$F(n,1) = 20$, wenn $n>1$
$F(1,m) = 10$

Rekursion2(n,m)

```
F = new array[1..n][1..m]
for i=2 to n do
   F[i][1] = 20
for j=1 to m do
   F[1][j] = 10
for i=2 to n do
   for j=2 to m do
      F[i][j] = 1 + min(F[i-1][j-1], F[i-1][j])
return F[n][m]
```

Dynamische Programmierung

Aufgabe 4

(d) $F(n,m) = nm + \min\{F(n-1,m+1), F(n,m-1)\}$, wenn $n,m>1$
$F(1,m) = 1$, wenn $m>1$
$F(n,1) = 0$

Rekursion2(n,m)

```
F = new array[1..n][1..m+n]
for i=2 to m+n do
  F[1][i] = 1
for i=1 to n do
  F[i][1] = 0
for i=2 to n do
  for j=2 to m+n-i do
    F[i][j] = ij + min(F[i-1][j+1], F[i][j-1])
return F[n][m]
```