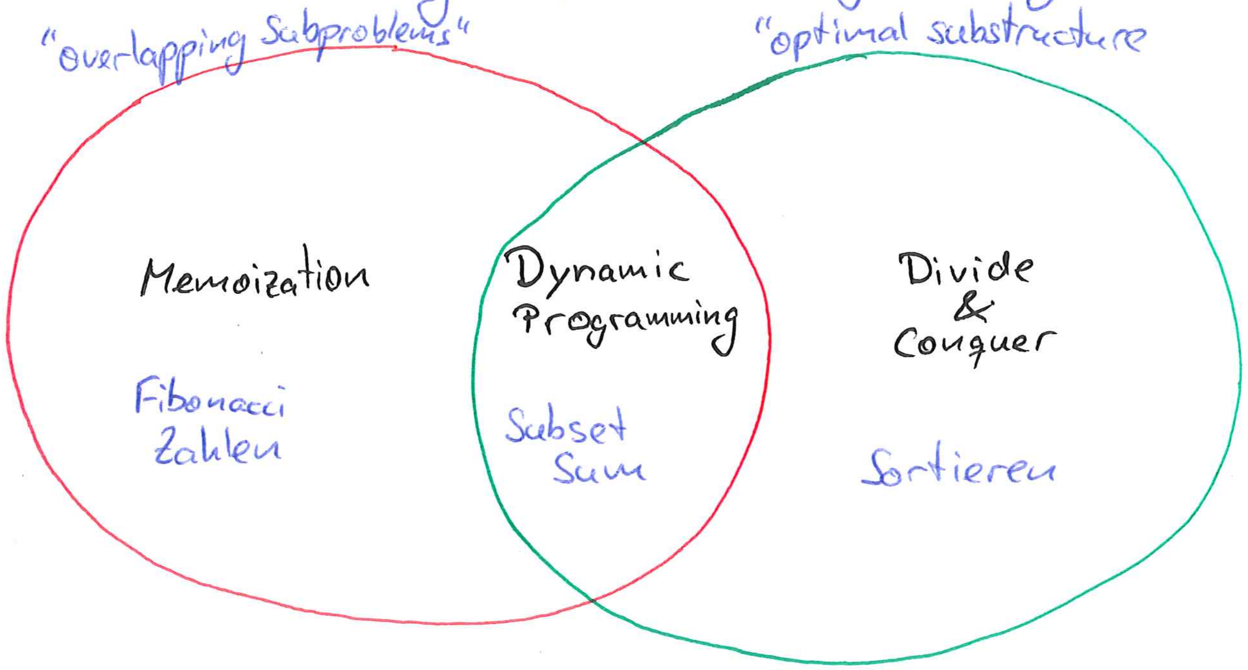
AuD II - Übung: Dynamic Programming



Overlapping Subproblems

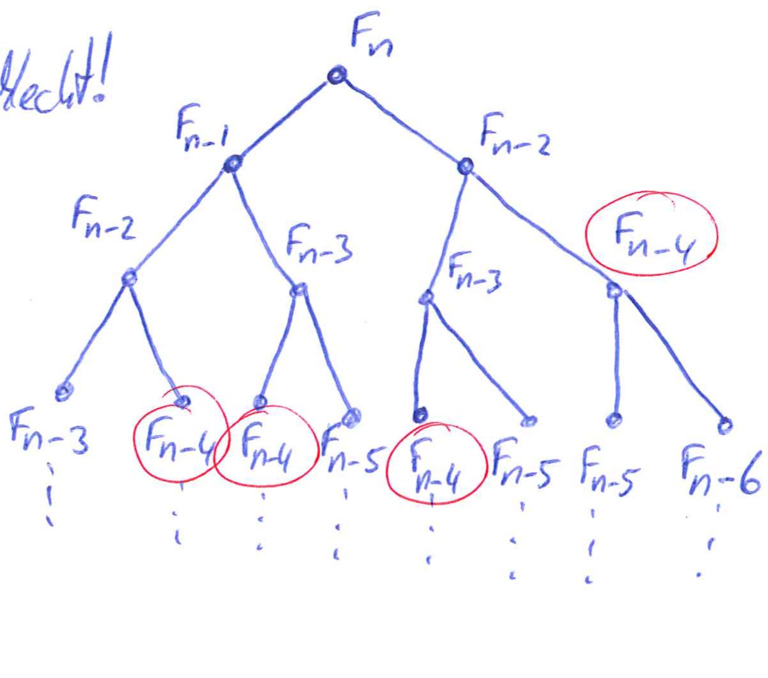
Subprobleme tauchen mehrfach auf.

Bsp: Fibonacci-Zahlen

$$F_n := F_{n-1} + F_{n-2}, \quad F_0 = 0, F_1 = 1$$

Rekursiver Ansatz ist schlecht!

↳ exponentielle Laufzeit...



Besser: "Memoization"

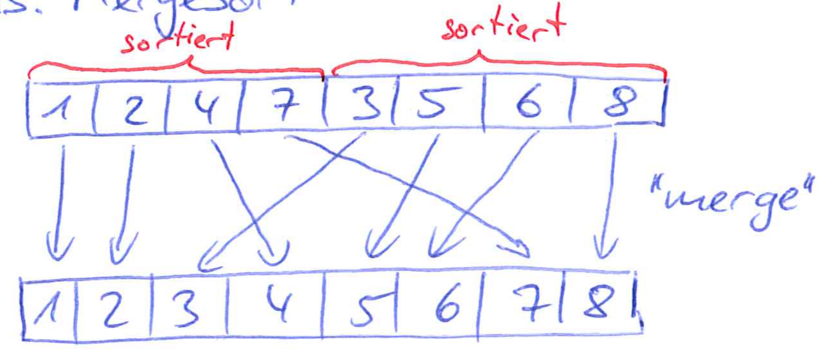
Merke Teilergebnisse!

z.B:

```
F[0] := 0
F[1] := 1
for i=2 to n do
   F[i] := F[i-1] + F[i-2]
return F[n]
```

Optimal substructure

Eine optimale Lösung kann aus optimalen Teillösungen effizient konstruiert werden.
Stichwort: "Divide & Conquer"
z.B. Mergesort



Besitzt ein Problem beide Eigenschaften, kann man DP anwenden!

Aus Vorlesung bekannt: Subset Sum, Knapsack

Wir betrachten nun ein neues Problem und konstruieren ein DP um es zu lösen.

Das Problem:

Longest Common Subsequence
(LCS)

Gegeben:
- Alphabet Σ
- Sequenzen
  $X = x_1, \dots, x_n$ mit $x_i \in \Sigma$ für $i \in \{1, \dots, n\}$
  $Y = y_1, \dots, y_m$ mit $y_i \in \Sigma$ für $i \in \{1, \dots, m\}$

Gesucht:

Eine längste <u>Teilsequenz</u> T, die in X und Y vorkommt

Eine Teilsequenz eines Wortes entsteht durch Weglassen von Buchstaben.
z.B. ist ACTG eine TS von ACCTATATGTT

Anwendung?

Ähnlichkeit von Wörtern/Strings, z.B. in der DNA

Sei $LCS(X,Y)$ die Länge einer längsten TS von X und Y.

Bsp: $X = ACCTATATGTT$, $Y = CATGACATTGA$ $\big\}\ LCS(X,Y) = 6$

Vorschläge?

Wie löst man LCS mit DP?

1. Überlegung: Irgendwas über die Länge der Wörter
↳ Betrachte Teilwörter bis zum i-ten/j-ten Buchstaben!

$X^i := x_1 \dots x_i \quad , \quad Y^j := y_1 \dots y_j$

2. Überlegung: Was ist der kleinste Input, den man sehr einfach lösen kann? ⇒ Wörter sind leer!
Gibt es dazu eine Verallgemeinerung?
Ja! Ein Wort ist leer.
D.h.: $LCS(X^i, Y^j) = 0$, falls $i = 0$ oder $j = 0$

Und nun? Welche Möglichkeiten existieren?

1. Ignoriere den letzten Buchstaben von X^i
2. Ignoriere den letzten Buchstaben von Y^j
3. Falls beide letzten Buchstaben übereinstimmen, erhöhe Zähler um eins und ignoriere die beiden Buchstaben.

Als Rekursionsgleichung:

$$LCS(X^i, Y^j) := \begin{cases} 0 & \text{, falls } i=0 \text{ oder } j=0 \\ \underbrace{LCS(X^{i-1}, Y^{j-1})}_{\text{Fall 3}}+1 & \text{, falls } x_i = y_j \\ \max(\underbrace{LCS(X^{i-1}, Y^j)}_{\text{Fall 1}}, \underbrace{LCS(X^i, Y^{j-1})}_{\text{Fall 2}}) & \text{, sonst} \end{cases}$$

Am Beispiel:

Y \ X	∅	A	C	C	T	A	T	A	T	G	T	T
∅	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	1	1	1	1	1
A	0	1	1	1	1	2	2	2	2	2	2	2
T	0	1	1	1	2	2	3	3	3	3	3	3
G	0	1	1	1	2	2	3	3	3	4	4	4
A	0	1	1	1	2	3	3	4	4	4	4	4
C	0	1	2	2	2	3	3	4	4	4	4	4
A	0	1	2	2	2	3	3	4	4	4	4	4
T	0	1	2	2	3	3	4	4	5	5	5	5
T	0	1	2	2	3	3	4	4	5	5	6	6
G	0	1	2	2	3	3	4	4	5	6	6	6
A	0	1	2	2	3	4	4	5	5	6	6	6

← init (row ∅)

↑ init (column ∅)

$LCS(X,Y)$ ist also 6.
Folgt man den Pfeilen rückwärts, können wir auch die Lösung sehen:

ACATTG