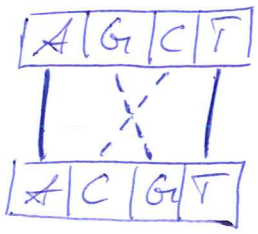
- gr. Übung AuD2 20.5.2015

- dyn. Progr. ⟷ „Memoisation"

- Am Bsp. LCS (Longest Common Subsequence)
  → Ähnlichkeit von Strings
  - Alphabet Σ
  - Sequenzen $X = x_1 \dots x_n$ mit $x_i \in \Sigma$ für $i \in \{1, \dots, n\}$
    $Y = y_1 \dots y_m$ " $y_i \in \Sigma$ " $i \in \{1, \dots, m\}$
  - Teilsequenz (TS): entsteht durch Weglassen von Buchstaben
    Bsp. ACTG ist TS von $\underline{A}C\underline{C}T\underline{A}T\underline{A}\underline{T}\underline{G}TT$
  - LCS(X,Y) := TS von X und Y mit max. Länge
    Bsp. X = A C C T A T A T G T T
    Y = C A T G A C A T T G A
    Vorschläge?
  - LCS nicht eindeutig:
    Gegenbsp.:
    $X = AGCT \Rightarrow LCS_1(X,Y) = ACT$
    $Y = ACGT \quad\quad LCS_2(X,Y) = AGT$



  - Dyn. Progr. für LCS:
    Es sei $X^i := x_1 \dots x_i$ und $Y^j := y_1 \dots y_j$ für $i \in \{1, \dots, n\}$ und $j \in \{1, \dots, m\}$
    und $X^0 = Y^0 = \emptyset$
    $LCS(X^i, Y^j) = 0$, falls $i = 0 \vee j = 0$ Initialisierung

$$LCS(X^i, Y^j) = \max\{\underbrace{LCS(X^{i-1}, Y^j)}_{\text{ignoriere } x^i}, \underbrace{LCS(X^i, Y^{j-1})}_{\text{ignoriere } y^j}, \underbrace{LCS(X^{i-1}, Y^{j-1}) + \mu(x_i, y_j)}_{\text{matche } x^i, y^j}\}$$

~~falls $x_i = y_j$ sonst~~

~~$LCS(X^i, Y^j) = \max$~~

mit $\mu(B, B') = \begin{cases} 1 & \text{falls } B = B' \\ 0 & \text{sonst} \end{cases}$ für $B, B' \in A$

- Algorithmus:

```
int LCS(X,Y){
    n := |X|
    m := |Y|
    init LCS[n][m+1]

    1. for (i∈{0,...,n}){
            LCS[i][0] = 0}
    2. for (j∈{0,...,m}){
            LCS[0][j] = 0}
    3. for (i∈{1,...,n}){
            for (j∈{1,...,m})
                max = LCS[i-1][j]
                if(max < LCS[i][j-1]){
                        max = LCS[i][j-1]}
                if(max < LCS[i][j] + u(i,j)){
                        max = LCS[i][j] + u(x_i, y_j)}
            }
    }

int u(x_i, y_j){
    if (x_i == y_i){ return 1}
    else {return 0}
}
```

Initialisierung

Hilfsfunktion

Laufzeit:

1. $\Theta(n)$ 2. $\Theta(m)$ 3. $\Theta(n \cdot m)$

~~Mittwoch: 9:45 – 11:15~~
~~Freitag: 13:15 – 14:45~~

ACCTATATGTT
CATGACATTGA

		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	1	2	3	3	3	4	4	4
A	0	1	1	1	1	2	3	4	4	4	4	4
C	0	1	2	2	2	2	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

- Dyno Progr. „bottom-up"
  - Lösung von Teilproblemen
    => Lösung für nächst größere Probleme

$LCS(X^i, Y^j)$ ist

$$\underbrace{LCS(X^{i-1}, Y^j)}_{1.},\ \underbrace{LCS(X^i, Y^{j-1})}_{2.} \text{ oder } \underbrace{LCS(X^{i-1}, Y^{j-1}) + 1_{x_i = y_j}}_{3.}$$

Bsp.:

1. $X^i = A\ D\ C$, $Y^j = B\ A\ D$
2. $X^i = A\ D\ C$, $Y^j = D\ C\ B$
3. $X^i = A\ D\ C$, $Y^j = A\ B\ C$

- Dyn. Progr.: rel. Iht. (D&C) NEIN
  -> Divide & Conquer
  („top-down")

- Allg.: D&C
  zerteile Problem klug -> löse Teilprobleme -> setze zu großer Lösung zusammen
  top -> down

- LCS mit D&C:

```
int LCSmitD&C(X^i, Y^j){
    if (X == ∅){ return 0}
    else if (Y == ∅){ return 0}
    else {
        z1 = LCSmitD&C(X^{i-1}, Y^j)
        z2 = LCSmitD&C(X^i, Y^{j-1})
        z3 = LCSmitD&C(X^{i-1}, Y^{j-1}) + M(X_i, Y_j)
        return max{z1, z2, z3}
    }
}

int M(A X_i, A Y_j){
    if (X_n == Y_m){ return 1}
    else return 0
}
```

} Rekursionsanker

} rel. Aufrufe

} Hilfsfkt.

- Korrektheit, siehe DP
- Laufzeit: $\Omega\left(2^{\max\{n+m\}}\right)$ -> allein Aufrufe 1., 2.
  (rel. Pfadlänge = ~~n+m~~ $\max\{n+m\}$)

– Verbesserung v. LCS mit D&C:

„erinnere Dich an bereits berechnetes"
(Memoisation ← lat. Memorandum ≙ „das zu Erinnernde")

→ $\forall i \in \{0,\dots,n\}, j \in \{0,\dots,m\}$ berechne LCS mit D&C nur einmal

→ „Gedächtnis": mem[n+1][m+1]
mit mem[i][j] = LCS mit D&C(i,j) falls schon berechnet

– Alg. D&C mit Memoisation:

```
init mem[n+1][m+1]
for (i∈{0,...,n}){
    for (j∈{0,...,m}){
        mem[i][j] = -∞
    }
}
LCS mit D&C und Mem(x^i, y^j, mem){
    if(x==∅){return 0}
    else if(y==∅){return 0}
    else
    {
        if(mem[i][j] = -∞){ mem[i][j] = LCS mit D&C und Mem(x^{i-1}, y^j, mem)}
        if(mem[i][j-1] = -∞){ mem[i][j-1] = LCS mit D&C und Mem(x^i, y^{j-1}, mem)}
        if(mem[i-1][j-1] = -∞){ mem[i-1][j-1] = LCS mit D&C und Mem(x^{i-1}, y^{j-1}, mem)}
        return max{mem[i-1][j], mem[i][j-1], mem[i-1][j-1]
                                    + M(x_i, x_j)}
    }
}
```

siehe LCS mit D&C

– Korrektheit siehe LCS mit D&C
– Laufzeit: Jeder Eintrag von mem wird ~~nur~~ genau einmal berechnet
⇒ $\Theta(n \cdot m)$