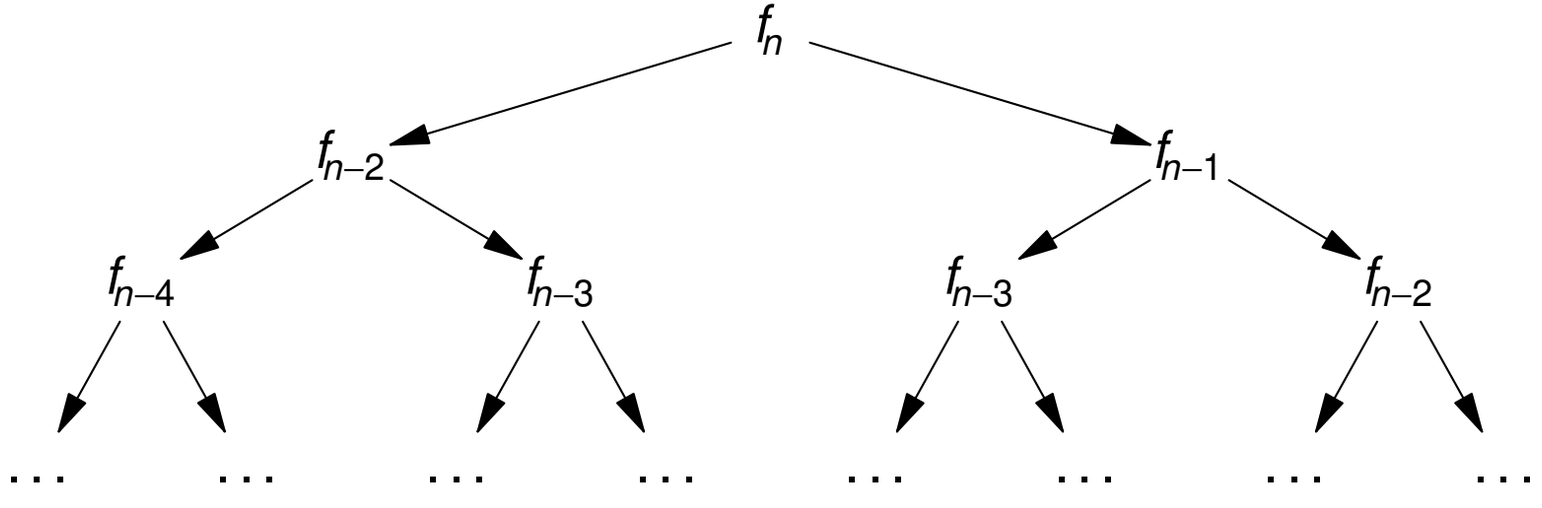
6 Tabellengestützte Programmierung (dynamic programming)

6.1 Einführende Beispiele

6.1.1 Fibonacci-Zahlen

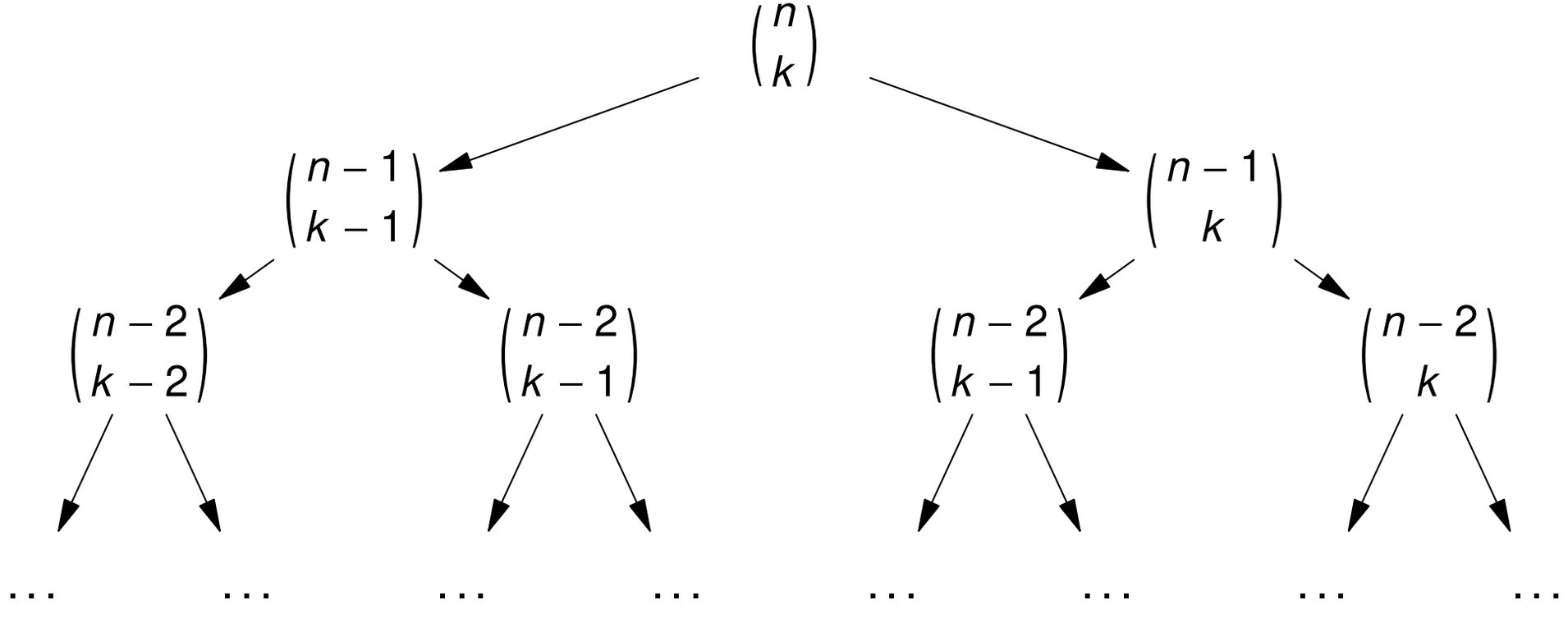
❒ Rekursive Definition für $n \in \mathbb{N}_0$: $f_n = \begin{cases} n & \text{für } n \leq 1 \\ f_{n-2} + f_{n-1} & \text{für } n \geq 2 \end{cases}$



6.1.2 Binomialkoeffizienten

❒ Rekursive Berechnung für $n \in \mathbb{N}_0$ und $k = 0, \ldots, n$ (vgl. Pascalsches Dreieck):

$$\binom{n}{k} = \begin{cases} 1 & \text{für } k = 0 \text{ oder } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{sonst} \end{cases}$$



6.1.3 Allgemeines Prinzip

Problem

- Bei der rekursiven Berechnung werden Teilergebnisse immer wieder – mit sehr hohem Aufwand – neu berechnet.

Lösungsidee

- Einmal berechnete Teilergebnisse werden in einer Tabelle (beispielsweise in einer Streuwerttabelle) gespeichert.

- Vor der Berechnung eines Teilergebnisses wird anhand der Tabelle überprüft, ob es bereits vorhanden ist.

Effekt

- Berechnungen, die ohne diese Optimierung extrem hohe (oft exponentielle) Laufzeit besitzen, können wesentlich effizienter (normalerweise mit polynomieller Laufzeit) durchgeführt werden.

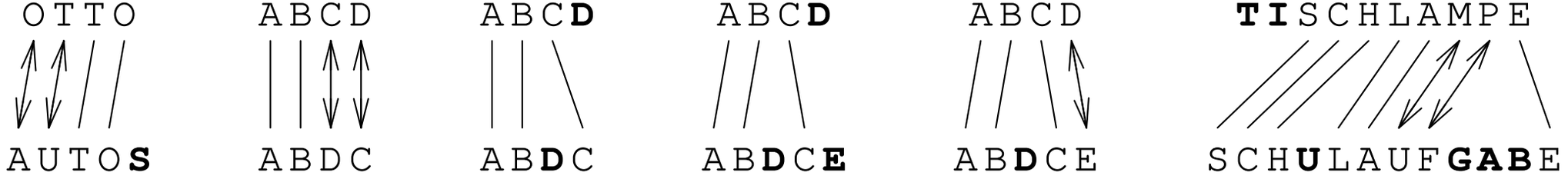
6.2 Editierdistanz

6.2.1 Definition

- Die *Editierdistanz* (oder Levenshtein-Distanz) $D(s, t)$ zweier Zeichenketten s und t ist die minimale Anzahl elementarer *Editieroperationen*, mit denen s in t überführt werden kann.

- Elementare Editieroperationen sind:
  - Entferne das Zeichen ... an Position ...
  - Füge das Zeichen ... nach Position ... ein
  - Ersetze das Zeichen ... an Position ... durch das Zeichen ...

6.2.2 Beispiele



6.2.3 Beobachtungen

- Die Editierdistanz ist offenbar symmetrisch, d. h. es gilt $D(s, t) = D(t, s)$ für alle Zeichenketten s und t.

- Folgende Operationen sind offensichtlich unnötig:
  - Löschen eines zuvor eingefügten Zeichens
    $\rightarrow$ Zeichen gar nicht einfügen
  - Löschen eines zuvor ersetzten Zeichens
    $\rightarrow$ Gleich das ursprüngliche Zeichen löschen
  - Ersetzen eines zuvor eingefügten Zeichens
    $\rightarrow$ Gleich das endgültige Zeichen einfügen
  - Mehrmaliges Ersetzen eines Zeichens
    $\rightarrow$ Zeichen einmalig durch das endgültige Zeichen ersetzen

- Ohne offensichtlich unnötige Operationen gibt es für jedes Zeichen aus *s* genau drei Möglichkeiten:
  - Es bleibt unverändert.
  - Es wird durch ein anderes Zeichen ersetzt, das anschließend unverändert bleibt.
  - Es wird entfernt.
- Umgekehrt gibt es für jedes Zeichen aus *t* genau drei Möglichkeiten:
  - Es wurde unverändert aus *s* übernommen.
  - Es ist durch Ersetzung eines Zeichens aus *s* entstanden.
  - Es wurde neu eingefügt.

6.2.4 Rekursionsgleichung

Satz 1

- Für eine beliebige Zeichenkette s und die leere Zeichenkette ε gilt:
  $D(s, \varepsilon) = D(\varepsilon, s) = |s|$ = Länge von s.

Beweis

- Um s in die leere Zeichenkette zu überführen, sind offenbar genau $|s|$ Löschoperationen nötig.

- Um die leere Zeichenkette in s zu überführen, sind offenbar genau $|s|$ Einfügeoperationen nötig.

Satz 2

- Für nichtleere Zeichenketten s und t gilt: $D(s,t) = \min \left\{ \begin{array}{l} D(s',t)+1 \\ D(s,t')+1 \\ D(s',t')+\delta \end{array} \right\}$.
- Dabei bezeichne s' bzw. t' die Zeichenkette s bzw. t ohne ihr letztes Zeichen.
- δ ist 0, wenn die letzten Zeichen von s und t gleich sind, sonst 1.

Beweis

- Zu zeigen:
  Die minimale Anzahl elementarer Editieroperationen, um s in t zu überführen, ist gleich dem durch das Minimum definierten Wert M.
- Das heißt:
  a) Es gibt eine Folge von M Editieroperationen, mit denen s in t überführt werden kann.
  b) Es gibt keine kürzere Folge, d. h. für jede Folge E, die s in t überführt, gilt $M \leq |E|$.

Teil a:

- Nach Definition von D gibt es eine Folge mit $D(s', t)$ Editieroperationen, die s' in t überführt.
  Wenn man diese Operationen auf s statt auf s' anwendet und anschließend noch das letzte Zeichen der resultierenden Zeichenkette entfernt, hat man s mit $D(s', t) + 1$ Operationen in t überführt.

- Ebenso gibt es eine Folge mit $D(s, t')$ Editieroperationen, die s in t' überführt.
  Wenn man diese Operationen auf s anwendet und anschließend noch das letzte Zeichen von t am Ende der resultierenden Zeichenkette anfügt, hat man s mit $D(s, t') + 1$ Operationen in t überführt.

- Ebenso gibt es eine Folge mit $D(s', t')$ Editieroperationen, die s' in t' überführt.
  Wenn man diese Operationen auf s statt auf s' anwendet und anschließend, wenn nötig, noch das letzte Zeichen der resultierenden Zeichenkette durch das letzte Zeichen von t ersetzt, hat man s mit $D(s', t') + \delta$ Operationen in t überführt.

- Somit gibt es Folgen zur Überführung von s in t mit $D(s', t) + 1$, $D(s, t') + 1$ und $D(s', t') + \delta$ Operationen.
  Da M einer dieser drei Werte ist, gibt es somit auch eine Folge mit M Operationen.

Teil b:

- Sei E eine beliebige Folge von Editieroperationen, die s in t überführt.
- O. B. d. A. enthält E keine offensichtlich unnötigen Operationen.
  (Andernfalls werden diese entfernt. Wenn dann für die verkürzte Folge $M \leq |E|$ gilt, gilt dies auch für die ursprüngliche Folge.)
- Fallunterscheidung nach der Herkunft des letzten Zeichens von t:
- Fall 1: Das letzte Zeichen von t ist das letzte Zeichen von s (wie im 6. Beispiel):
  - Dann kann man mit derselben Folge von Editieroperationen s' in t' überführen.
  - Daher gilt: $M \underset{\text{(i)}}{\leq} D(s', t') + \delta \underset{\text{(ii)}}{=} D(s', t') \underset{\text{(iii)}}{\leq} |E|$.
    (i) $D(s', t') + \delta$ ist einer der drei Werte, deren Minimum M bezeichnet.
    (ii) $\delta = 0$, da das letzte Zeichen von t gleich dem letzten Zeichen von s ist.
    (iii) $D(s', t')$ ist gemäß Definition von D die *minimale* Anzahl von Editieroperationen, mit denen s' in t' überführt werden kann.

- Fall 2:
  Das letzte Zeichen von t ist durch Ersetzung des letzten Zeichens von s entstanden (wie im 5. Beispiel):
  - Wenn man in der Folge E die entsprechende Ersetzungsoperation weglässt, erhält man eine Folge zur Überführung von s' in t' mit $|E| - 1$ Operationen.
  - Daher gilt (mit ähnlichen Begründungen wie in Fall 1):
    $M \leq D(s', t') + \delta = D(s', t') + 1 \leq (|E| - 1) + 1 = |E|$.

- Fall 3: Das letzte Zeichen von t wurde während der Überführung hinzugefügt (wie im 1. Beispiel):
  - Wenn man in der Folge E die entsprechende Einfügeoperation weglässt, erhält man eine Folge zur Überführung von s in t' mit $|E| - 1$ Operationen.
  - Daher gilt: $M \leq D(s, t') + 1 \leq (|E| - 1) + 1 = |E|$.

- Fall 4: Keiner der obigen Fälle ist zutreffend, das heißt:
  Das letzte Zeichen von t ist entweder ein Zeichen aus s, oder es ist durch eine Ersetzung eines Zeichens aus s entstanden, aber dieses Zeichen ist nicht das letzte Zeichen von s (wie im 3. Beispiel):
  - Dann müssen während der Überführung alle Zeichen von s hinter diesem Zeichen entfernt worden sein.
    Insbesondere muss das letzte Zeichen von s irgendwann entfernt worden sein.
  - Wenn man in der Folge E die entsprechende Löschoperation weglässt, erhält man eine Folge zur Überführung von s' in t mit $|E| - 1$ Operationen.
  - Daher gilt: $M \leq D(s', t) + 1 \leq (|E| - 1) + 1 = |E|$.
- Somit gilt in allen Fällen: $M \leq |E|$.

6.2.5 Praktische Berechnung

- Für zwei Zeichenketten s und t sei $D_{i,\,j} = D_{i,\,j}(s, t)$ die Editierdistanz zwischen dem Präfix der Länge i von s und dem Präfix der Länge j von t.

- Aufgrund der obigen Sätze können die Werte $D_{i,\,j}$ wie folgt berechnet werden:
  - $D_{0,\,0} = 0$
  - $D_{i,\,0} = i$ für $i = 1, \ldots, |s|$
  - $D_{0,\,j} = j$ für $j = 1, \ldots, |t|$
  - $$D_{i,\,j} = \min \left\{ \begin{array}{l} D_{i-1,\,j} + 1 \\ D_{i,\,j-1} + 1 \\ D_{i-1,\,j-1} + \delta \end{array} \right\} \text{ für } i = 1, \ldots, |s| \text{ und } j = 1, \ldots, |t| \text{ mit } \delta = \begin{cases} 0 \text{ für } s_i = t_j \\ 1 \text{ sonst} \end{cases}$$

- $D(s, t)$ ist gleich $D_{|s|,\,|t|}(s, t)$.

6.2.6 Ermittlung der tatsächlichen Editieroperationen

- Bei der Berechnung eines Werts $D_{i,\,j}$ kann man notieren, aus welchem seiner drei „Nachbarn“ $D_{i-1,\,j}$, $D_{i,\,j-1}$ oder $D_{i-1,\,j-1}$ er „entstanden“ ist bzw. entstanden sein kann. Diese Information kann aber auch nachträglich durch Vergleich eines Werts mit seinen drei Nachbarn ermittelt werden.
- Wenn $D_{i,\,j}$ aus $D_{i-1,\,j}$ entstanden ist (oben), dann muss an dieser Stelle die Operation
  - Entferne das Zeichen s_i an Position i (bei Überführung von s in t)
  - Füge das Zeichen s_i nach Position j ein (bei Überführung von t in s)

  ausgeführt werden.
- Wenn $D_{i,\,j}$ aus $D_{i,\,j-1}$ entstanden ist (links), dann muss an dieser Stelle die Operation
  - Füge das Zeichen t_j nach Position i ein (bei Überführung von s in t)
  - Entferne das Zeichen t_j an Position j (bei Überführung von t in s)

  ausgeführt werden.
- Wenn $D_{i,\,j}$ aus $D_{i-1,\,j-1}$ entstanden ist (diagonal), dann muss an dieser Stelle entweder nichts gemacht werden (wenn $s_i = t_j$) oder die Operation
  - Ersetze das Zeichen s_i an Position i durch das Zeichen t_j (bei Überf. von s in t)
  - Ersetze das Zeichen t_j an Position j durch das Zeichen s_i (bei Überf. von t in s)

  ausgeführt werden (wenn $s_i \neq t_j$).

- Um die Folge aller Editieroperationen auszugeben, beginnt man mit $D_{|s|,\,|t|}$, ermittelt den passenden Nachbarn, gibt die zugehörige Operation aus und wiederholt den Vorgang mit diesem Nachbarn, bis man bei $D_{0,\,0}$ angekommen ist.
- Auf diese Weise entsteht die Folge der Editieroperationen automatisch von hinten nach vorn.
  Will man die Operationen in umgekehrter Reihenfolge ausführen, muss man beachten, dass Einfügungen und Entfernungen die Positionen der nachfolgenden Zeichen bzw. Operationen verändern.

6.2.7 Laufzeit

- $O(|s| \cdot |t|)$

6.2.8 Beispiel

- s = ABCD, t = ABDCE

	j	0	1	2	3	4	5
i		t	A	B	D	C	E
0	s	**0**	1	2	3	4	5
1	A	1	**0**	1	2	3	4
2	B	2	1	**0**	**1**	2	3
3	C	3	2	1	1	**1**	2
4	D	4	3	2	1	2	**2**

- Ergebnis: $D(s, t) = D_{4,5} = 2$
- Die fettgedruckten Zahlen zeigen den „Weg“ von $D_{4,5}$ zurück zu $D_{0,0}$.
- Die Editieroperationen zur Überführung von s in t lauten demnach:
  - Ersetze das Zeichen D an Position 4 durch das Zeichen E.
  - Füge das Zeichen D nach Position 2 ein.
- Die Editieroperationen zur Überführung von t in s lauten:
  - Ersetze das Zeichen E an Position 5 durch das Zeichen D.
  - Entferne das Zeichen D an Position 3.

6.3 Blocksatz

6.3.1 Problemstellung

- Gegeben sei eine Folge von Wörtern $w_1, \ldots, w_n$ mit Breiten $\lambda_1, \ldots, \lambda_n$, die in dieser Reihenfolge auf Zeilen der Breite λ verteilt werden sollen, sodass der zum Auffüllen der Zeilen zusätzlich benötigte Zwischenraum möglichst gleichmäßig verteilt ist (vgl. § 4.1 und § 4.2).

- Frage: Was bedeutet „möglichst gleichmäßig verteilt“, d. h. wie lautet eine sinnvolle Bewertungsfunktion für das Optimierungsproblem?

- Mögliche Antwort: Um einen großen Zwischenraum härter zu „bestrafen“ als mehrere kleine, soll die Summe der *Quadrate* aller Zwischenraumbreiten möglichst klein sein.

6.3.2 Malus einer einzelnen Zeile

- Wenn eine Zeile der Breite λ die Wörter $w_i, \ldots, w_j$ enthält und zwischen je zwei Wörtern ein Mindestabstand σ_0 ist, ist die Gesamtbreite des zusätzlich vorhandenen Zwischenraums in dieser Zeile: $\sigma = \lambda - \sum_{k=i}^{j} \lambda_k - m\,\sigma_0$ mit $m = j - i$ = Anzahl der Wortzwischenräume in dieser Zeile (d. h. 1 weniger als die Anzahl der Wörter).

- Wenn dieser Zwischenraum gleichmäßig verteilt wird, ergibt sich als Summe der Quadrate der Einzelzwischenräume: $\mu(i, j) = \begin{cases} m\left(\frac{\sigma}{m}\right)^2 = \frac{\sigma^2}{m} & \text{für } m > 0 \quad (*) \\ \sigma^2 & \text{für } m = 0 \quad (**) \end{cases}$

  (*) Bei ganzzahliger Rechnung: $(m - r) \cdot q^2 + r \cdot (q + 1)^2$ mit $q = \left\lfloor \frac{\sigma}{m} \right\rfloor$ und $r = \sigma \bmod m$

  (**) Wenn die Zeile nur ein Wort enthält ($i = j \Leftrightarrow m = 0$), befindet sich der gesamte Zwischenraum σ am Ende der Zeile.

- Sonderfall: Übervolle Zeile
  - Wenn $\sigma < 0$ ist, ist die Zeile *übervoll*.
  - Im Fall $i = j$ lässt sich dies nicht vermeiden, weil die Breite λ_i des einzigen Worts auf der Zeile dann größer als die Zeilenbreite λ ist. In diesem Fall sei $\mu(i, j) = 0$.
  - Im Fall $i < j$ ist die Zeile *unnötig übervoll*, d. h. die Anzahl ihrer Wörter sollte reduziert werden. In diesem Fall sei $\mu(i, j) = \infty$.

- Sonderfall: Letzte Zeile
  Da Zwischenraum in der letzten Zeile (normalerweise) nicht bestraft werden soll, sei $\mu(i, j) = 0$ für $j = n$, sofern $\sigma \geq 0$ ist.

- $\mu(i, j)$ heißt *Malus* der Zeile mit den Wörtern $w_i, \ldots, w_j$.

6.3.3 Malus eines gesamten Absatzes

- Für $0 < i_1 < \ldots < i_k = j \leq n$ ist $\mu(\{ i_1, \ldots, i_k \}) = \mu(1, i_1) + \mu(i_1 + 1, i_2) + \ldots + \mu(i_{k-1} + 1, i_k)$ der Gesamtmalus aller Zeilen, der sich ergibt, wenn man die Wörter $w_1, \ldots, w_j$ so auf Zeilen verteilt, dass nach den Wörtern $i_1, \ldots, i_k$ jeweils ein Zeilenumbruch erfolgt.

- Für $j = 1, \ldots, n$ sei $\mu(j) = \min \left\{ \mu(\{ i_1, \ldots, i_k \}) \;\middle|\; 0 < i_1 < \ldots < i_k = j \right\}$,
d. h. $\mu(j)$ ist der kleinste Malus, den man erreichen kann, wenn man die Wörter $w_1, \ldots, w_j$ irgendwie auf Zeilen verteilt.

- Zusätzlich sei $\mu(0) = 0$.

- Gesucht ist somit $\mu(n)$ sowie die Indizes $0 < i_1 < \ldots < i_k = n$ der Wörter, nach denen jeweils ein Zeilenumbruch erfolgen soll, um diesen minimalen Malus zu erreichen.

6.3.4 Rekursionsgleichung

- Um die ersten j Wörter $w_1, \ldots, w_j$ beliebig auf Zeilen zu verteilen,
  - wählt man eine beliebige Stelle $i \in \{0, \ldots, j-1\}$ für den letzten Zeilenumbruch (d. h. dieser soll nach Wort i erfolgen),
  - verteilt die ersten i Wörter beliebig auf Zeilen
  - und setzt die restlichen Wörter $i+1, \ldots, j$ auf die letzte Zeile.

- Um die ersten j Wörter *optimal* (d. h. mit minimalem Malus) auf Zeilen zu verteilen,
  - wählt man für die Verteilung der ersten i Wörter jeweils eine Verteilung mit minimalem Malus $\mu(i)$,
  - addiert dazu den Malus $\mu(i+1, j)$ der resultierenden letzten Zeile
  - und wählt von diesen Möglichkeiten mit Gesamtmalus $\mu(i) + \mu(i+1, j)$ für $i = 0, \ldots, j-1$ das Minimum.

- Somit gilt für $j = 1, \ldots, n$:
  $\mu(j) = \min\{\mu(i) + \mu(i+1, j) \mid i = 0, \ldots, j-1\}$.

6.3.5 Praktische Berechnung

- Für die praktische Berechnung des Minimums $\mu(j)$ lässt man i rückwärts von $j-1$ bis 0 laufen und beendet die Schleife, sobald $\mu(i+1,\ j) = \infty$ ist, d. h. wenn die letzte Zeile unnötig übervoll wäre, weil $\mu(i+1,\ j)$ dann für alle weiteren Werte von i ebenfalls ∞ wäre und sich somit kein kleinerer Gesamtwert mehr ergeben würde.

- Damit kann $\mu(j)$ nacheinander für $j = 0, \ldots, n$ wie folgt berechnet und in einer Tabelle gespeichert werden:

  1 Setze $\mu(0) = 0$.

  2 Für $j = 1, \ldots, n$:

    1 Setze $\mu(j) = \infty$.

    2 Für $i = j-1, \ldots, 0$:

      1 Berechne $\mu = \mu(i+1,\ j)$.

      2 Wenn $\mu = \infty$ ist, beende die Schleife.

      3 Setze $\mu = \mu(i) + \mu$.

      4 Wenn $\mu < \mu(j)$ ist: Setze $\mu(j) = \mu$ und $\pi(j) = i$.

- Die optimalen Umbruchstellen $i_k, \ldots, i_1$ erhält man „rückwärts“ als n, $\pi(n)$, $\pi(\pi(n))$, …

6.3.6 Beispiel

- Wenn man den Text `a b c d e f ghij` in „Schreibmaschinenschrift“ (alle Zeichen einschließlich Leerzeichen besitzen Breite 1) bei Zeilenbreite 7 mit dem in § 4.2 beschriebenen Nächstbest-Algorithmus formatiert, ergibt sich:

```
a b c d
e     f
ghij
```

- Die Anwendung des zuvor beschriebenen Algorithmus ergibt:

j	i $\pi(j)$	Letzte Zeile (Wort $i+1$ bis j)	$\mu(i+1,\ j)$	$\mu(i)$	$\mu(i)+\mu(i+1,\ j)$ $\mu(j)$
0					**0**
1	**0**	`a++++++`	$6^2 = 36$	0	**36**
2	1	`b++++++`	$6^2 = 36$	36	72
	0	`a-++++b`	$4^2 = 16$	0	**16**
3	2	`c++++++`	$6^2 = 36$	16	52
	1	`b-++++c`	$4^2 = 16$	36	52
	0	`a-+b-+c`	$1^2 + 1^2 = 2$	0	**2**

j	i $\pi(j)$	Letzte Zeile (Wort $i+1$ bis j)	$\mu(i+1,\ j)$	$\mu(i)$	$\mu(i)+\mu(i+1,\ j)$ $\mu(j)$
4	3	`d++++++`	$6^2 = 36$	2	38
	2	`c-++++d`	$4^2 = 16$	16	32
	1	`b-+c-+d`	$1^2 + 1^2 = 2$	36	38
	0	`a-b-c-d`	0	0	**0**
5	4	`e++++++`	$6^2 = 36$	0	36
	3	`d-++++e`	$4^2 = 16$	2	**18**
	2	`c-+d-+e`	$1^2 + 1^2 = 2$	16	18
	1	`b-c-d-e`	0	36	36
	0	`a-b-c-d-e`	∞		
6	5	`f++++++`	$6^2 = 36$	18	54
	4	`e-++++f`	$4^2 = 16$	0	16
	3	`d-+e-+f`	$1^2 + 1^2 = 2$	2	**4**
	2	`c-d-e-f`	0	16	16
	1	`b-c-d-e-f`	∞		
7	**6**	`ghij`	0	4	**4**
	5	`f-ghij`	0	18	18
	4	`e-f-ghij`	∞		

- Erläuterung zu den Tabellen:
  - Für jeden Wert von j läuft i rückwärts von $j-1$ bis 0, solange $\mu(i+1,\ j) < \infty$ ist.
  - Die fettgedruckten Werte in der zweiten bzw. letzten Spalte entsprechen $\pi(j)$ bzw. $\mu(j)$.
  - In der dritten Spalte symbolisieren Minuszeichen den Mindestabstand zwischen zwei Wörtern und Pluszeichen den zusätzlich benötigten Zwischenraum, der für die Berechnung von $\mu(i+1,\ j)$ verwendet wird.
- Die optimalen Umbruchstellen sind: $n = 7$, $\pi(7) = 6$, $\pi(6) = 3$, $\pi(3) = 0$
- Damit ergibt sich folgende optimale Formatierung:

```
a  b  c
d  e  f
ghij
```

6.3.7 Laufzeit

- Da zwischen je zwei Wörtern ein Mindestabstand der Breite σ_0 sein soll, haben auf einer Zeile der Breite λ maximal $m = \left\lfloor \frac{\lambda}{\sigma_0} \right\rfloor + 1$ Wörter Platz.

- Deshalb wird die innere Schleife zur Berechnung von $\mu(j)$ nach höchstens $m = \mathrm{O}(1)$ Durchläufen beendet.

- Somit ist die Gesamtlaufzeit zur Berechnung von $\mu(0), \ldots, \mu(n)$ gleich $\mathrm{O}(n)$ (und nicht $\mathrm{O}(n^2)$, wie in https://en.wikipedia.org/wiki/Line_wrap_and_word_wrap behauptet wird).

- Zum Vergleich: Die Gesamtzahl der Möglichkeiten, n Wörter auf Zeilen zu verteilen, ist 2^{n-1}: Nach jedem Wort außer dem letzten kann entweder ein Zeilenumbruch erfolgen oder nicht.

- Also führt die tabellengestützte Programmierung zu einer immensen Laufzeitverbesserung.

6.3.8 Erweiterungsmöglichkeiten

- Trennung langer Wörter
  - Silbentrennung (mit unterschiedlich „guten“ Trennstellen)
  - Trennung vor oder nach bestimmten Zeichen (z. B. nach Schrägstrichen in URLs)
  - Trennung mathematischer Formeln
- Zusätzliche Strafpunkte für einzelne Trennungen
  oder mehrere aufeinanderfolgende Zeilen mit Trennungen
- Zeilen mit einem einzigen Wort noch „härter bestrafen“
- Verbotene und bevorzugte Zeilenumbrüche
- Einen Absatz künstlich um eine oder mehrere Zeilen verlängern, um
  - „Schusterjungen“ und „Hurenkinder“ zu vermeiden
  - alle Spalten auf der letzten Seite eines mehrspaltigen Texts gleich lang zu machen
- Aber: Auch der beste Algorithmus kann nicht „zaubern“.
  Manchmal muss man einen Text einfach umformulieren,
  um einen störenden Umbruch zu vermeiden.

derit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Diese Zeile ist ein Schusterjunge, aber es kommt noch

schlimmer: Diese ist ein Hurenkind.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehen-