Chapter 3
Dynamic Programming

Algorithm Theory
WS 2015/16

Fabian Kuhn

Edit Distance

Given: Two strings $A = \underline{a_1 a_2 \dots a_m}$ and $B = \underline{b_1 b_2 \dots b_n}$

Goal: Determine the minimum number $D(A, B)$ of edit operations required to transform A into B

Edit operations:

a) **Replace** a character from string A by a character from B

b) **Delete** a character from string A

c) **Insert** a character from string B into A

$$\begin{array}{c|cccccccccccccccc}
\downarrow & \mathtt{m} & \mathtt{a} & \mathtt{-} & \mathtt{t} & \mathtt{h} & \mathtt{e} & \mathtt{m} & \mathtt{-} & \mathtt{-} & \mathtt{a} & \mathtt{t} & \mathtt{i} & \mathtt{c} & \cancel{\mathtt{i}} & \mathtt{a} & \mathtt{n} \\
 & \mathtt{m} & \mathtt{u} & \bigcirc\!\!\!\!\mathtt{l} & \mathtt{t} & \mathtt{i} & \mathtt{p} & \mathtt{l} & \mathtt{i} & \mathtt{c} & \mathtt{a} & \mathtt{t} & \mathtt{i} & \mathtt{o} & \mathtt{-} & \mathtt{-} & \mathtt{n}
\end{array}$$

Edit Distance – Cost Model $c(a,a)=0$

- Cost for **replacing** character a by b: $\boldsymbol{c(\underline{a},\underline{b}) \geq 0}$

- Capture insert, delete by allowing $a = \underline{\varepsilon}$ or $b = \varepsilon$:
  - Cost for **deleting** character a: $\underline{\boldsymbol{c(a,\varepsilon)}}$
  - Cost for **inserting** character b: $\underline{\boldsymbol{c(\varepsilon,b)}}$

- **Triangle inequality**:

$$c(a,c) \leq c(a,b) + c(b,c)$$

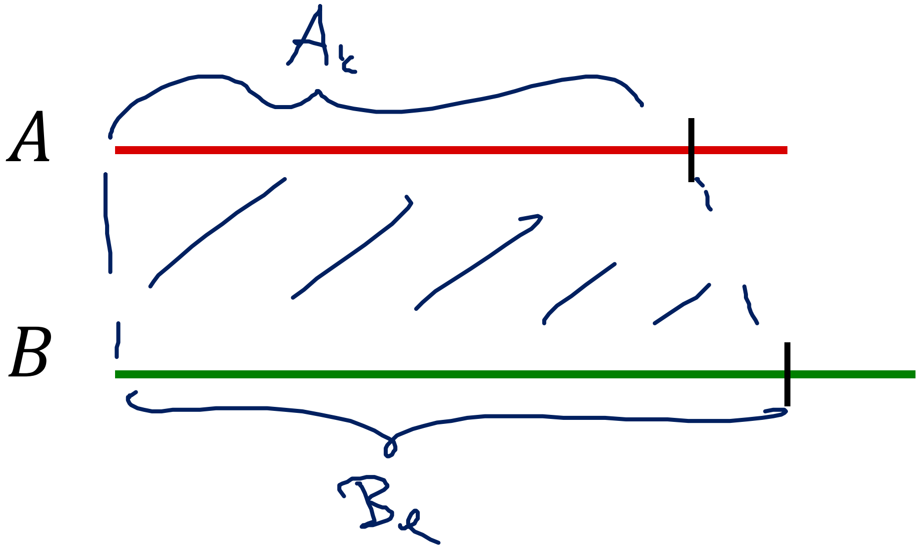
  → each character is changed at most once!

- **Unit cost model**: $c(a,b) = \begin{cases} 1, & \text{if } a \neq b \\ 0, & \text{if } a = b \end{cases}$

Computation of the Edit Distance

Let $A_k := \underline{\underline{a_1 \dots a_k}}$, $B_\ell := \underline{\underline{b_1 \dots b_\ell}}$, and

$$\underline{D_{k,\ell}} := D\underline{(A_k, B_\ell)}$$



Computing the Edit Distance

- Recurrence relation (for $k, \ell \geq 1$)

$$D_{k,\ell} = \min \begin{Bmatrix} D_{k-1,\ell-1} & + c(a_k, b_\ell) \\ D_{k-1,\ell} & + c(a_k, \varepsilon) \\ D_{k,\ell-1} & + c(\varepsilon, b_\ell) \end{Bmatrix} = \underbrace{\min \begin{Bmatrix} D_{k-1,\ell-1} & + 1 \,/\, 0 \\ D_{k-1,\ell} & + 1 \\ D_{k,\ell-1} & + 1 \end{Bmatrix}}_{\textbf{unit cost model}}$$

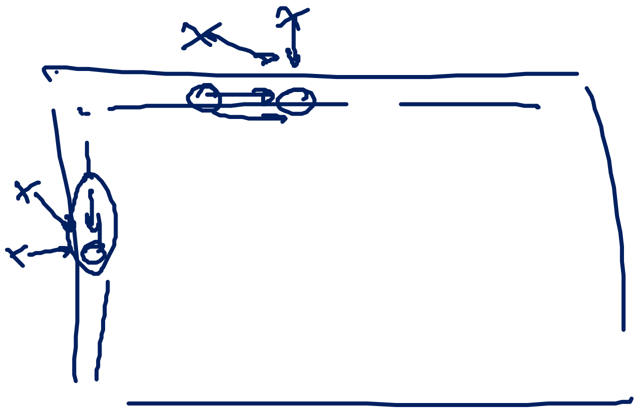
- Need to compute $D_{i,j}$ for all $0 \leq i \leq k$, $0 \leq j \leq \ell$:

$$\begin{tikzcd}
\boldsymbol{D_{k-1,\ell-1}} \arrow[dr, "+1 \,/\, 0"] & \boldsymbol{D_{k-1,\ell}} \arrow[d, "+1"] \\
\boldsymbol{D_{k,\ell-1}} \arrow[r, "+1"'] & \boldsymbol{D_{k,\ell}}
\end{tikzcd}$$

Recurrence Relation for the Edit Distance

Base cases:

$$
\begin{aligned}
\underline{D_{0,0}} &= D(\varepsilon,\varepsilon) = \underline{0}\\
D_{0,j} &= D\left(\underline{\varepsilon}, \underline{B_j}\right) = D_{0,j-1} + c\underline{\left(\varepsilon, b_j\right)}\\
D_{i,0} &= D(\underline{A_i}, \varepsilon) = D_{i-1,0} + c(a_i, \varepsilon)
\end{aligned}
$$



Recurrence relation:

$$
D_{i,j} = \min \begin{Bmatrix} D_{k-1,\ell-1} + c(a_k, b_\ell) \\ D_{k-1,\ell} + c(a_k, \varepsilon) \\ D_{k,\ell-1} + c(\varepsilon, b_\ell) \end{Bmatrix}
$$

Example

$\xrightarrow{A}$

B $\downarrow$

		a	b	c	c	a
$\rightarrow$	0	1	2	3	4	5
b	1	1	1	2	3	4
a	2	1	2	2	3	3
b	3					
d	4					
a	5					$\bigcirc$

+1 +1 +1

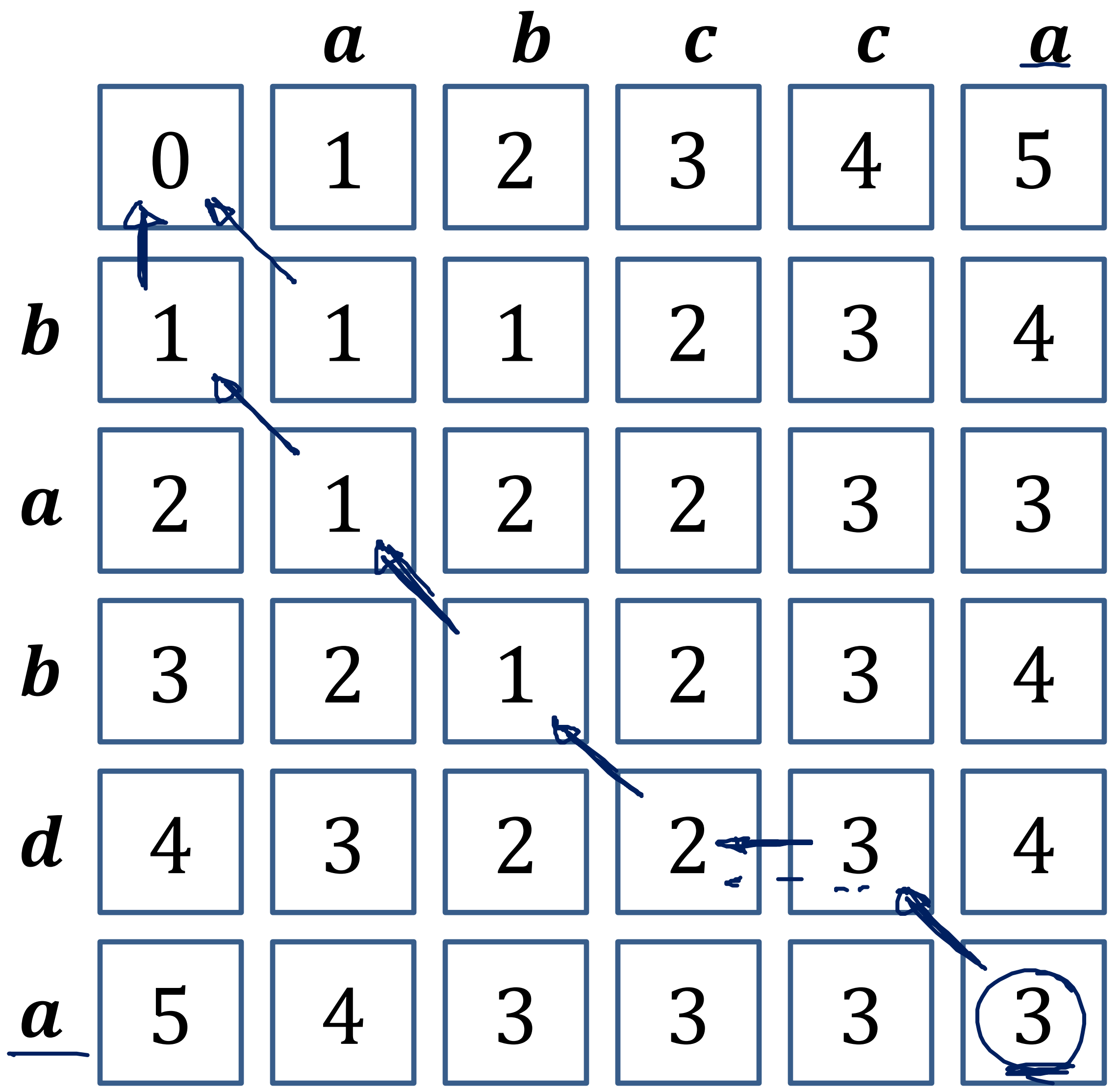
Computing the Edit Operations

Algorithm *Edit-Operations*(i, j)
Input: matrix D (already computed)
Output: list of edit operations

```
1  if $i = 0$ and $j = 0$ then return empty list

2  if $i \neq 0$ and $D[i,j] = D[i-1,j] + 1$ then
3      return Edit-Operations$(i-1, j) \circ$ „delete $a_i$“

4  else if $j \neq 0$ and $D[i,j] = D[i,j-1] + 1$ then
5      return Edit-Operations$(i, j-1) \circ$ „insert $b_j$“

6  else  // $D[i,j] = D[i-1,j-1] + c(a_i, b_j)$
7      if $a_i = b_i$ then return Edit-Operations$(i-1, j-1)$
8      else return Edit-Operations$(i-1, j-1) \circ$ „replace $a_i$ by $b_j$“
```

Initial call: *Edit-Operations*(*m*,*n*)

Edit Operations



		a	b	c	c	a
	0	1	2	3	4	5
b	1	1	1	2	3	4
a	2	1	2	2	3	3
b	3	2	1	2	3	4
d	4	3	2	2	3	4
a	5	4	3	3	3	3

$$\begin{array}{cccccc} - & a & b & c & c & a \\ b & a & b & d & - & a \\ \uparrow & & & \uparrow & \uparrow & \end{array}$$

Edit Distance: Summary

- Edit distance between two strings of length m and n can be computed in $O(\underline{mn})$ time.

- Obtain the edit operations:
  - for each cell, store which rule(s) apply to fill the cell
  - track path backwards from cell $(\underline{m,n})$
  - can also be used to get all optimal "alignments"

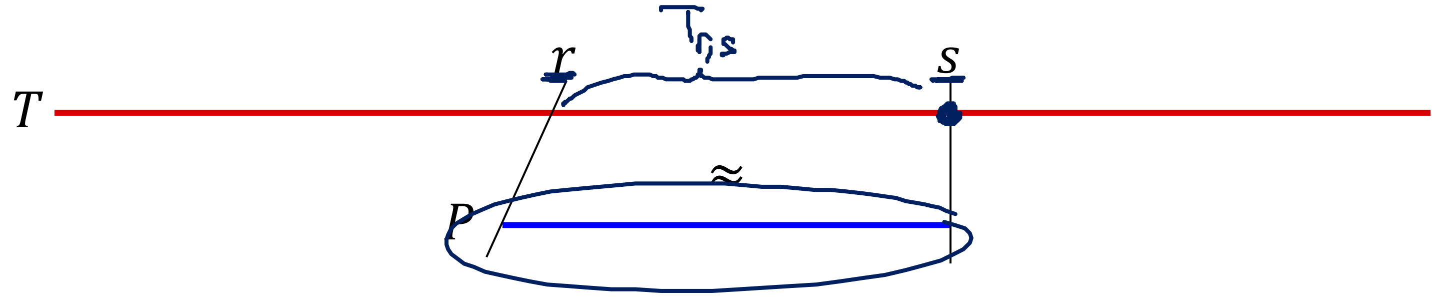
- Unit cost model:
  - interesting special case
  - each edit operation costs 1

Approximate String Matching $m \ll n$

Given: strings $\underline{T = t_1 t_2 \dots t_n}$ (text) and $\underline{P = p_1 p_2 \dots p_m}$ (pattern).

Goal: Find an interval $[\underline{r, s}]$, $1 \le \underline{r} \le \underline{s} \le n$ such that the sub-string $\underline{T_{r,s}} := t_r \dots t_s$ is the one with highest similarity to the pattern P:

$$\underset{1 \le r \le s \le n}{\arg\min}\ D\left(\underline{\underline{T_{r,s}, P}}\right)$$



Approximate String Matching

Naive Solution:

```
for all $1 \leq \underline{r} \leq \underline{s} \leq n$ do
    compute $\underline{D(T_{r,s}, P)}$
choose the minimum
```

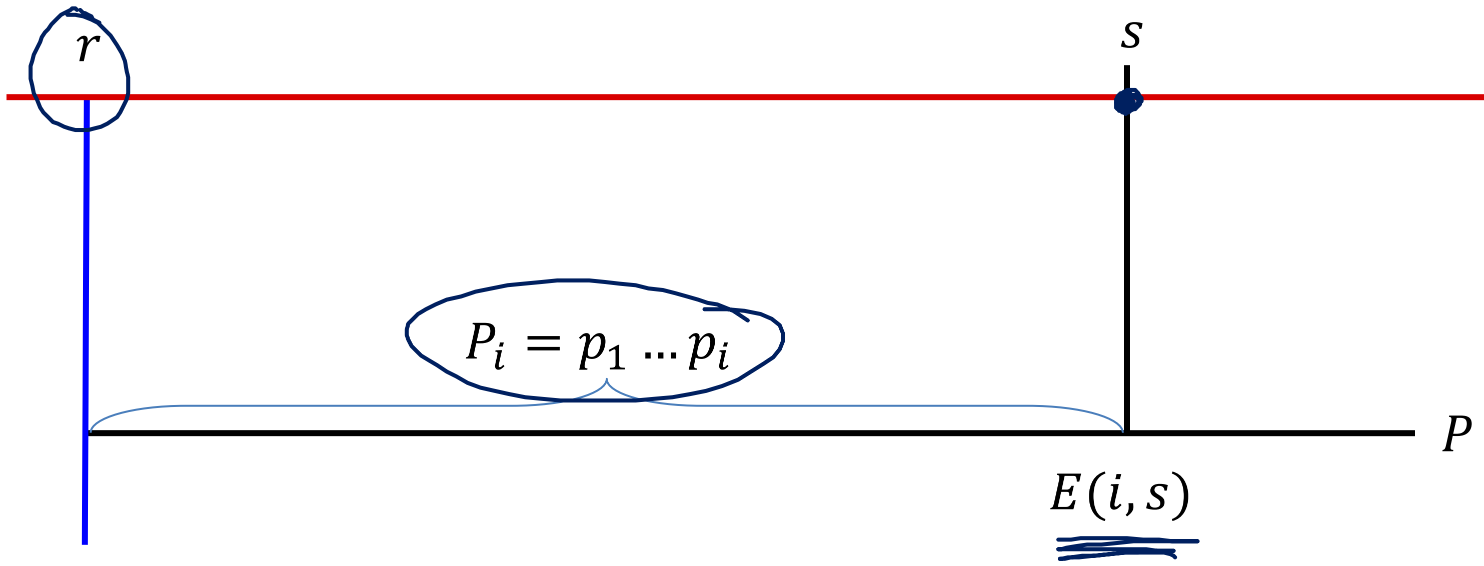
time: $\underline{O(n^3 \cdot m)}$

$O((s-r) \cdot m) = O(n \cdot m)$

Approximate String Matching

A related problem:

- For each position s in the text and each position i in the pattern compute the minimum edit distance $E(i,s)$ between $P_i = \underline{p_1 \dots p_i}$ and any substring $T_{r,s}$ of T that ends at position s.

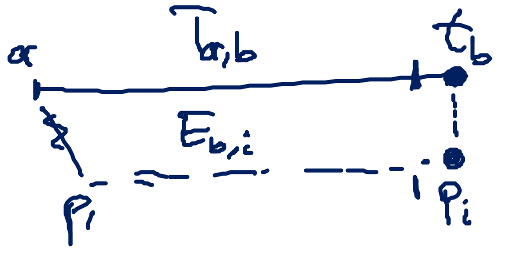


Approximate String Matching

Three ways of ending optimal alignment between T_b and P_i:

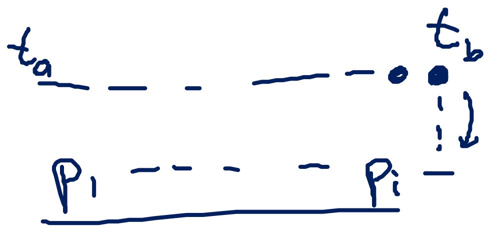
1. t_b is replaced by p_i:

   $$E_{b,i} = \underline{E_{b-1,i-1}} + \underline{\underline{c(t_b, p_i)}}$$



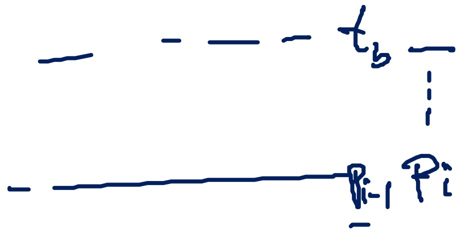
2. t_b is deleted:

   $$E_{b,i} = \underline{E_{b-1,i} + c(t_b, \varepsilon)}$$



3. p_i is inserted:

   $$E_{b,i} = \underline{\underline{E_{b,i-1}}} + c(\varepsilon, p_i\ )$$



Approximate String Matching

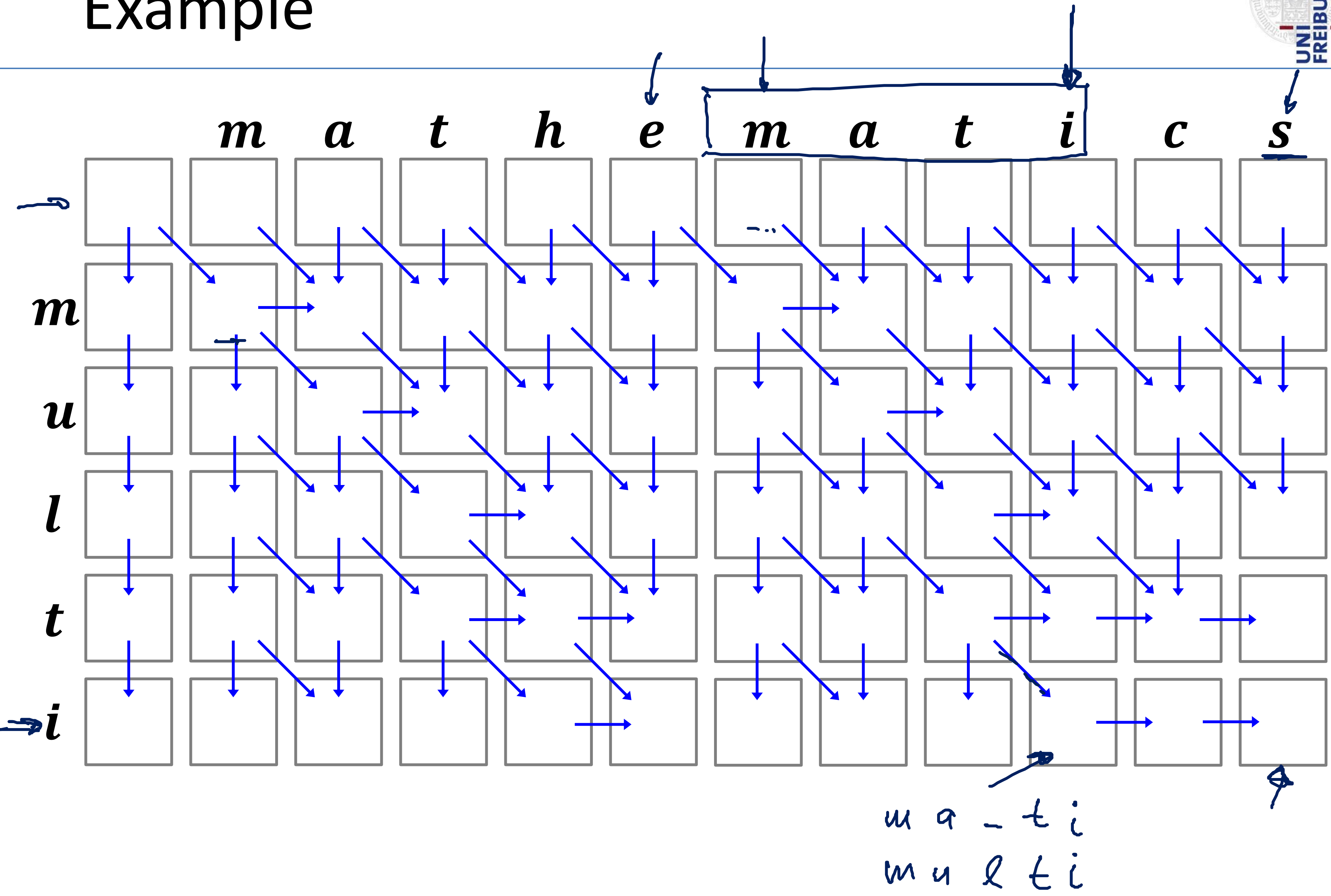
Recurrence relation (unit cost model):

$$E_{b,i} = \min \begin{cases} E_{b-1,i-1} + \cancel{1}\ 0 \quad \text{if } t_b = p_i \\ E_{b-1,i} \quad\ + 1 \\ E_{b,i-1} \quad\ + 1 \end{cases}$$

Base cases:

$$\underline{E_{0,0}} = \underline{0}$$
$$E_{\underline{0},\underline{i}} = \underline{i}$$
$$\underline{\underline{E_{i,0} = 0}}$$

Example



Approximate String Matching

- Optimal matching consists of optimal sub-matchings

- Optimal matching can be computed in $O(mn)$ time

- Get matching(s):
  - Start from minimum entry/entries in bottom row
  - Follow path(s) to top row

- Algorithm to compute $E(b, i)$ identical to edit distance algorithm, except for the initialization of $\underline{\underline{E(b, 0)}}$

Related Problems in Bioinformatics

Sequence Alignment:

Find optimal alignment of two given DNA, RNA, or amino acid sequences.

```
G A - C G G A T T A G
G A T C G G A A T - G
```

Global vs. Local Alignment:

- *Global alignment*: find optimal alignment of 2 sequences
- *Local alignment*: find optimal alignment of sequence 1 (patter) with sub-sequence of sequence 2 (text)