- Up until now we have seen how to compute optimal global alignments and optimal local alignments for a pair of sequences.
- In the former, both sequences are aligned completely, while in the latter, implicitly all pairs of substrings of both sequences are aligned to obtain an optimal local alignment.
- The common technical concept underlying both methods, i.e. the computation of $(m+1) \times (n+1)$-matrices can adapted to other problems.
- Here we consider the approximate string matching problem as one such problem.

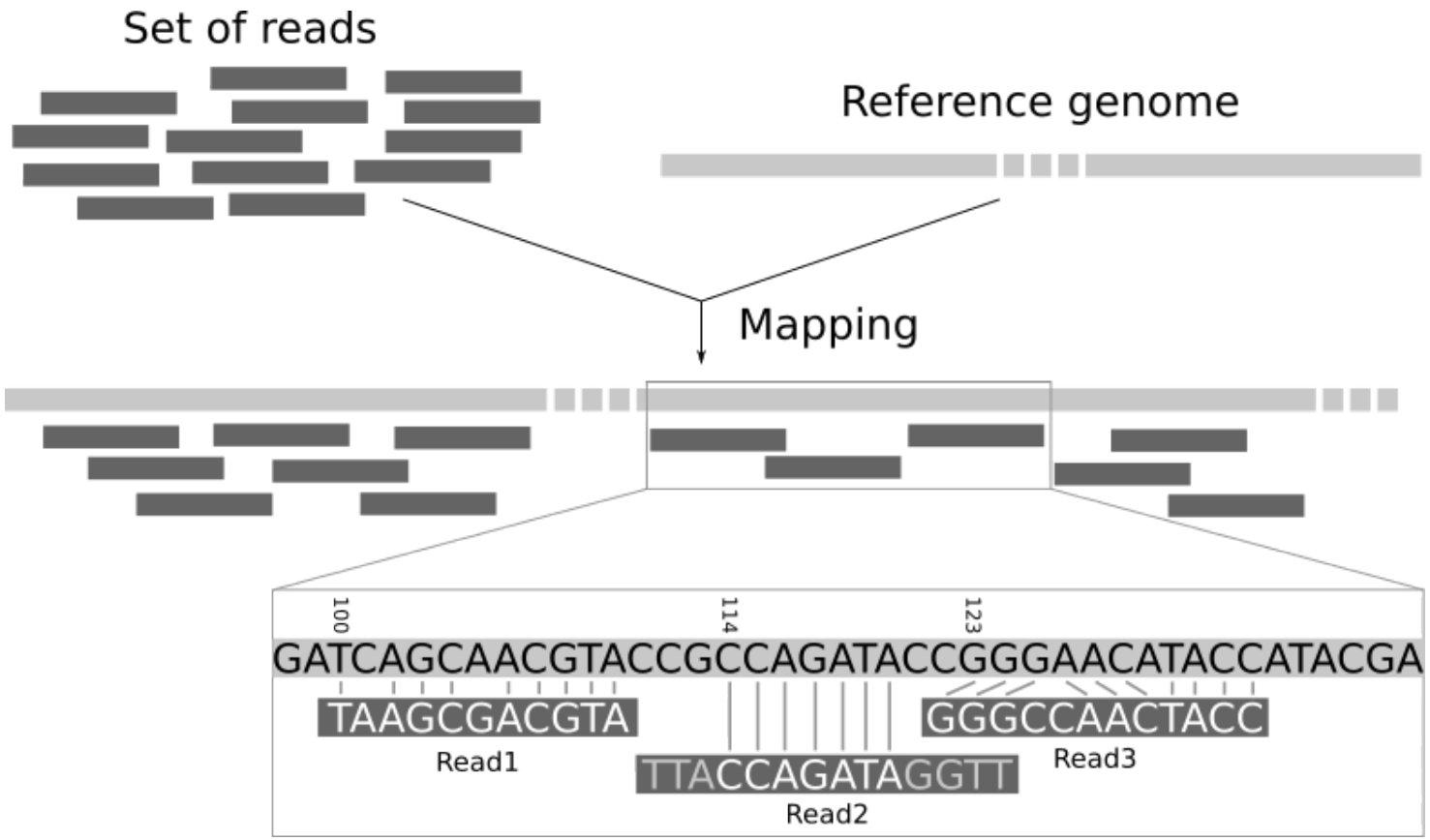
The approximate string matching problem

- Given a pattern $p \in \mathcal{A}^*$ of length m and an input string $t \in \mathcal{A}^*$ of length n, *the approximate string matching problem* (APSMP, for short) consists of finding the positions in t where an approximate match ends.
- These positions are referred to as solutions of the approximate string matching problem.
- Let $k \in \mathbb{R}^+$ be a threshold value.
- An *approximate match* is a substring w of t such that $edist_\delta(p, w) \leq k$.

The approximate string matching problem

- The approximate string matching problem is of special interest in biological sequence analysis.
- For instance, when searching a DNA database (the input string) for a pattern (e.g. a sequence read), few errors must be allowed, to take into account experimental inaccuracies as well as small differences in DNA among individuals of the same or related species.
- As the number of patterns has grown very fast, due to establishing novel sequencing technologies, the APSMP is one of the most well-studied problems (often named the read-mapping problem) in algorithmic bioinformatics.

Figure 1: Schematic explanation of the readmapping problem as a special instance of the approximate string matching problem. Each of 10^5-10^8 reads, i.e. a DNA sequence of $10^2 - 10^5$bp. must be matched against a reference genome, allowing indels and mismatches. The assignment of the reads to the positions where they match is called read mapping.



https://training.galaxyproject.org/training-material/topics/sequence-analysis/tutorials/mapping/tutorial.html

The approximate string matching problem

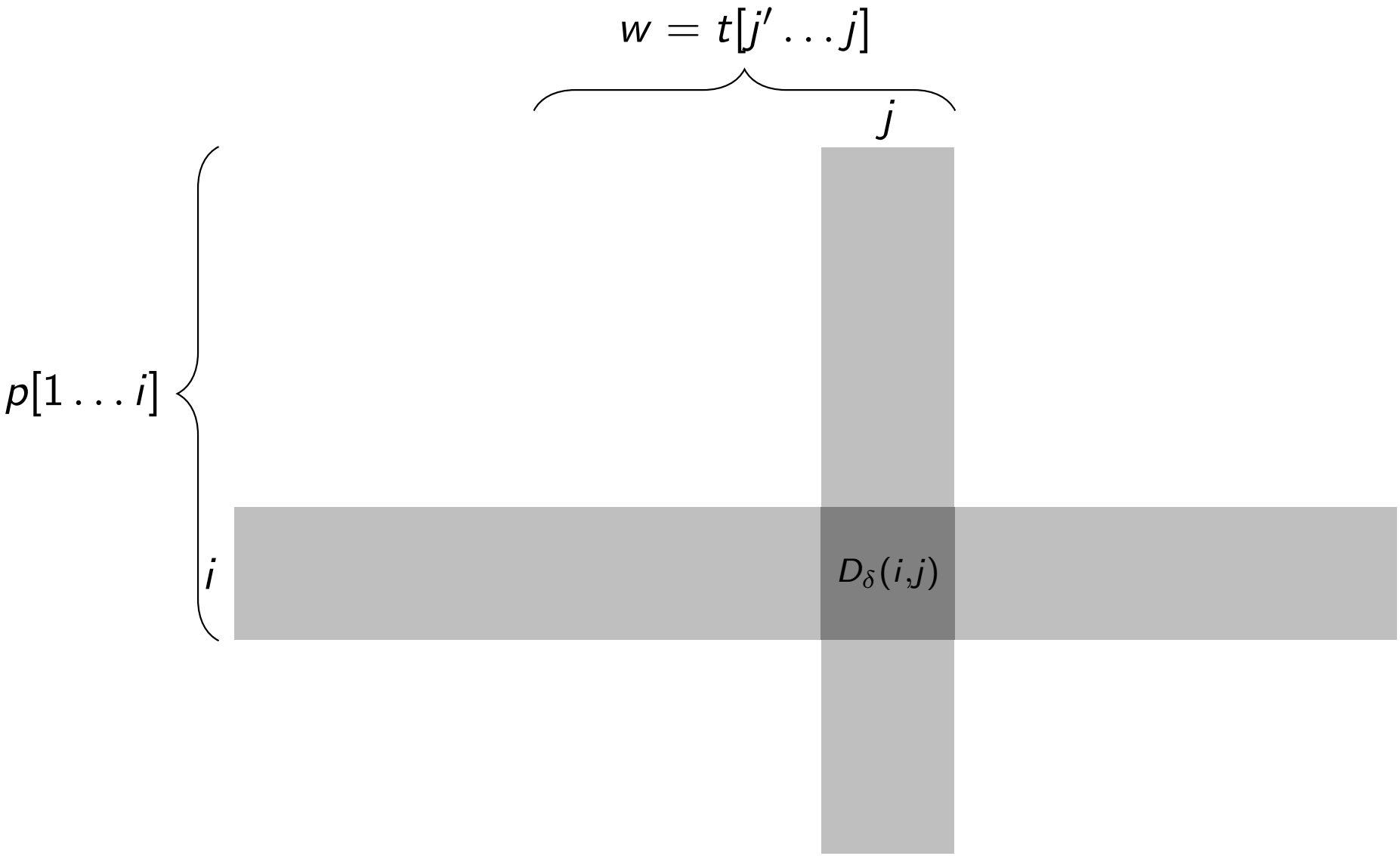
By a slight modification of the previous dynamic programming algorithms, one obtains a simple method to solve the approximate string matching problem.

Algorithm Compute an $(m+1) \times (n+1)$-table D_δ such that for all (i,j), $0 \leq i \leq m$, $0 \leq j \leq n$, the following holds:

$$D_\delta(i,j) = \min\{edist_\delta(p[1 \ldots i], w) \mid w \text{ is a suffix of } t[1 \ldots j]\} \quad \square$$

- Thus entry $D_\delta(i,j)$ states a property of the prefix of $p[1 \ldots i]$ of p (just like $E_\delta(i,j)$ states a property of the prefix $u[1 \ldots i]$ of u).
- Moreover, $D_\delta(i,j)$ states a property of the suffixes of $t[1 \ldots j]$ (just like $L_\sigma(i,j)$ states a property of the suffixes of $v[1 \ldots j]$).
- So table D_δ is a hybrid between E_δ and L_σ mixing global and local sequence alignment, see Figure 2 for an illustration.
- One often states that the APSMP is solved by semi-global alignment.

Figure 2: The curly brackets mark the substrings of p and t an entry $D_\delta(i, j)$ refers to.



The approximate string matching problem

To understand the relation of D_δ with the APSMP recall the definition of the notion *approximate match*:

> *there is an approximate match ending at position* j, $0 \leq j \leq n$, *if and only if* $edist_\delta(p, w) \leq k$ *for some suffix* w *of* $t[1 \ldots j]$.

The approximate string matching problem

To understand the relation of D_δ with the APSMP recall the definition of the notion *approximate match*:

> *there is an approximate match ending at position* j, $0 \leq j \leq n$, *if and only if* $edist_\delta(p, w) \leq k$ *for some suffix* w *of* $t[1 \ldots j]$.

Observe that the latter is equivalent to

$$\min \{edist_\delta(p, w) \mid w \text{ is a suffix of } t[1 \ldots j]\} \leq k$$

The approximate string matching problem

To understand the relation of D_δ with the APSMP recall the definition of the notion *approximate match*:

> *there is an approximate match ending at position j, $0 \leq j \leq n$, if and only if $edist_\delta(p, w) \leq k$ for some suffix w of $t[1 \ldots j]$.*

Observe that the latter is equivalent to

$$\min \{edist_\delta(p[1 \ldots m], w) \mid w \text{ is a suffix of } t[1 \ldots j]\} \leq k$$

The approximate string matching problem

To understand the relation of D_δ with the APSMP recall the definition of the notion *approximate match*:

> *there is an approximate match ending at position j, $0 \leq j \leq n$, if and only if $edist_\delta(p, w) \leq k$ for some suffix w of $t[1 \ldots j]$.*

Observe that the latter is equivalent to

$$\underbrace{\min\{edist_\delta(p[1 \ldots m], w) \mid w \text{ is a suffix of } t[1 \ldots j]\}}_{D_\delta(m,j)} \leq k$$

The approximate string matching problem

To understand the relation of D_δ with the APSMP recall the definition of the notion *approximate match*:

> *there is an approximate match ending at position* j, $0 \leq j \leq n$, *if and only if* $edist_\delta(p, w) \leq k$ *for some suffix* w *of* $t[1 \ldots j]$.

Observe that the latter is equivalent to

$$\underbrace{\min\{edist_\delta(p[1 \ldots m], w) \mid w \text{ is a suffix of } t[1 \ldots j]\}}_{D_\delta(m,j)} \leq k$$

which is equivalent to

$$D_\delta(m, j) \leq k$$

The approximate string matching problem

To understand the relation of D_δ with the APSMP recall the definition of the notion *approximate match*:

> *there is an approximate match ending at position j, $0 \leq j \leq n$, if and only if $edist_\delta(p, w) \leq k$ for some suffix w of $t[1 \ldots j]$.*

Observe that the latter is equivalent to

$$\underbrace{\min\{edist_\delta(p[1 \ldots m], w) \mid w \text{ is a suffix of } t[1 \ldots j]\}}_{D_\delta(m,j)} \leq k$$

which is equivalent to

$$D_\delta(m, j) \leq k$$

Thus, to solve the approximate string matching problem, one computes table D_δ and outputs all j satisfying $D_\delta(m, j) \leq k$. See Figure 3, for an example.

Figure 3: Table D_δ for $p = \texttt{atggc}$, $t = \texttt{aggtatcgc}$ and the unit cost function δ. Let $k = 2$. Then in the last row we find the boxed values $\leq k$ for $j \in \{3, 4, 7, 8, 9\}$. Hence approximate matches end at these positions.

							j				
				a	g	g	t	a	t	c	g	c
		D_δ	0	1	2	3	4	5	6	7	8	9
		0	0	0	0	0	0	0	0	0	0	0
	a	1	1	0	1	1	1	0	1	1	1	1
i	t	2	2	1	1	2	1	1	0	1	2	2
	g	3	3	2	1	1	2	2	1	1	1	2
	g	4	4	3	2	1	2	3	2	2	1	2
	c	5	5	4	3	**2**	**2**	3	3	**2**	**2**	**1**

The approximate string matching problem

– The approximate matches w ending at position j (and an optimal alignment of p and w, if necessary) can be output, by a traceback from $D_\delta(m, j)$ to an entry $D_\delta(0, j')$, $0 \leq j' \leq j - 1$ in the first row of table D_δ.

Recurrences for D_δ are easy to derive. For $i = 0$ we have

$$\begin{aligned} D_\delta(i, j) &= \min\{edist_\delta(p[1 \ldots i], w) \mid w \text{ is a suffix of } t[1 \ldots j]\} \\ &= \min\{edist_\delta(p[1 \ldots 0], w) \mid w \text{ is a suffix of } t[1 \ldots j]\} \\ &= \min\{edist_\delta(\varepsilon, w) \mid w \text{ is a suffix of } t[1 \ldots j]\} \\ &= \min\{edist_\delta(\varepsilon, \varepsilon)\} \\ &= 0 \end{aligned}$$

The approximate string matching problem

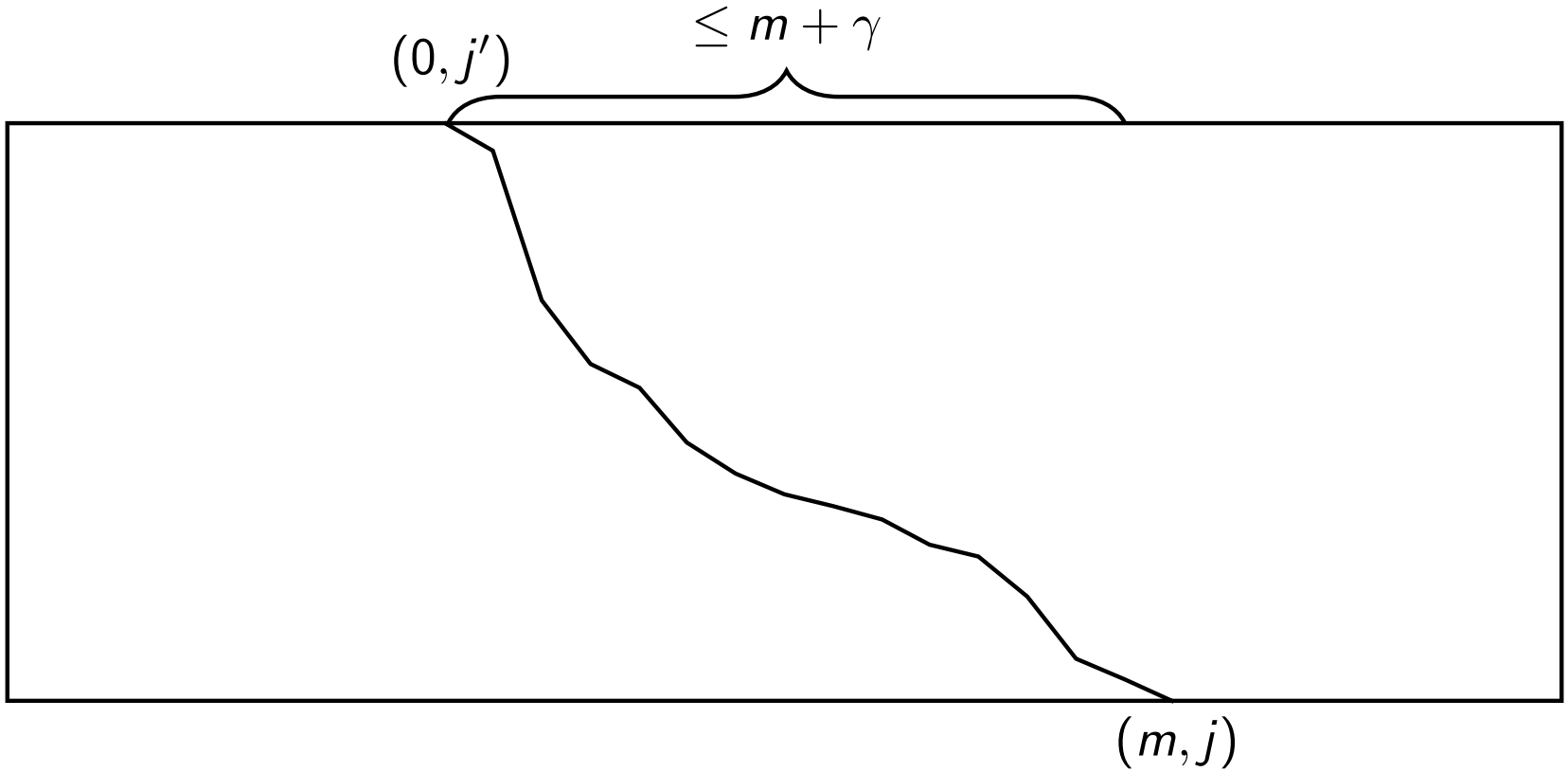
- That is, the values in the first row of D_δ are all 0.
- Intuitively, this means that any prefix of $t[1 \ldots j]$ (i.e. $t[1 \ldots j' - 1]$ if w begins at position j'), can be inserted at no cost.
- The remaining values of D_δ (i.e. for $i > 0$) are computed in the same way as one computes E_δ:

$$D_\delta(i,j) = \begin{cases} 0 & \text{if } i = 0 \\ D_\delta(i-1,0) + \delta(p[i] \to \varepsilon) & \text{if } i > 0 \text{ and } j = 0 \\ \min \left\{ \begin{array}{l} D_\delta(i-1,j) + \delta(p[i] \to \varepsilon) \\ D_\delta(i,j-1) + \delta(\varepsilon \to t[j]) \\ D_\delta(i-1,j-1) + \delta(p[i] \to t[j]) \end{array} \right\} & \text{if } i > 0 \text{ and } j > 0 \end{cases}$$

The approximate string matching problem

- Obviously, each entry in table D_δ can be evaluated in constant time.
- D_δ can be evaluated column by column from left to right, in the same way as E_δ, see the DP-Algorithm for computing the edit distance.
- Hence D_δ can be computed in $O(mn)$ time.
- If only the positions where an approximate match ends are to be enumerated, then $O(m)$ space suffices.
- To additionally compute approximate matches using a traceback of minimizing paths one does not have to store the entire table D_δ.
- This is because the threshold k determines an upper bound on the number of insertions and replacements allowed in an alignment of p with a suffix of $t[1 \ldots j]$, see Figure 4.

Figure 4: A minimizing path from $D_\delta(m, j)$ to $D_\delta(0, j')$ has a maximum length of $m + \gamma$ where $\gamma = \left\lfloor \frac{k}{\min\{\delta(\varepsilon \to b) | b \in \mathcal{A}\}} \right\rfloor$.



Each replacement (diagonal edge) and insertion (horizontal edge) increases the length of the suffix of $t[1 \ldots j]$ matching p by 1. A deletion (vertical edge) does not affect the length of the matching suffix. So the larger the number of replacements and insertions, the larger the distance of j' and j.

The approximate string matching problem

- Each replacement involves a different position of p and so the number of replacements is at most m.
- Each replacement has cost ≥ 0 and each insertion has cost $\geq \min\{\delta(\varepsilon \to b) \mid b \in \mathcal{A}\} > 0$.
- Thus γ insertions have at least total costs of

$$\gamma \cdot \min\{\delta(\varepsilon \to b) \mid b \in \mathcal{A}\}$$

This must be $\leq k$, and from $\gamma \cdot \min\{\delta(\varepsilon \to b) \mid b \in \mathcal{A}\} \leq k$ we conclude

$$\gamma \leq \left\lfloor \frac{k}{\min\{\delta(\varepsilon \to b) \mid b \in \mathcal{A}\}} \right\rfloor .$$

- So we have up to $m + \gamma$ replacements and insertions, and thus only have to store $\leq m + \gamma$ columns of D_δ at any time of the computation.
- So the space requirement is $O(m(m + \gamma))$.

The approximate string matching problem

If δ is the unit cost function, then $\min\{\delta(\varepsilon \to b) \mid b \in \mathcal{A}\} = 1$ and so $\gamma = k$. As a consequence, the space requirement is $O(m(m+k))$.