There are many variations and extension of the edit distance, for example:

- Hamming distance allows only the subtitution operation.

- Damerau–Levenshtein distance adds an edit operation:

  T Transposition swaps two adjacent characters.

- With weighted edit distance, each operation has a cost or weight, which can be other than one.

- Allow insertions and deletions (indels) of factors at a cost that is lower than the sum of character indels.

We will focus on the basic Levenshtein distance.

Computing Edit Distance

Given two strings $A[1..m]$ and $B[1..n]$, define the values d_{ij} with the recurrence:

$$
\begin{aligned}
d_{00} &= 0, \\
d_{i0} &= i, \ \ 1 \le i \le m, \\
d_{0j} &= j, \ \ 1 \le j \le n, \ \text{and} \\
d_{ij} &= \min \begin{cases} d_{i-1,j-1} + \delta(A[i], B[j]) \\ \quad d_{i-1,j} + 1 \\ \quad d_{i,j-1} + 1 \end{cases} \qquad 1 \le i \le m, 1 \le j \le n,
\end{aligned}
$$

where

$$
\delta(A[i], B[j]) = \begin{cases} 1 & \text{if } A[i] \neq B[j] \\ 0 & \text{if } A[i] = B[j] \end{cases}
$$

Theorem 4.2: $d_{ij} = ed(A[1..i], B[1..j])$ for all $0 \le i \le m$, $0 \le j \le n$. In particular, $d_{mn} = ed(A, B)$.

Example 4.3: $A =$ ballad, $B =$ handball

d		h	a	n	d	b	a	l	l
	0	1	2	3	4	5	6	7	8
b	1	1	2	3	4	4	5	6	7
a	2	2	1	2	3	4	4	5	6
l	3	3	2	2	3	4	5	4	5
l	4	4	3	3	3	4	5	5	4
a	5	5	4	4	4	4	4	5	5
d	6	6	5	5	4	5	5	5	6

$ed(A, B) = d_{mn} = d_{6,8} = 6$.

Proof of Theorem 4.2. We use induction with respect to $i+j$. For brevity, write $A_i = A[1..i]$ and $B_j = B[1..j]$.

Basis:

$$d_{00} = 0 = ed(\epsilon, \epsilon)$$
$$d_{i0} = i = ed(A_i, \epsilon) \quad (i \text{ deletions})$$
$$d_{0j} = j = ed(\epsilon, B_j) \quad (j \text{ insertions})$$

Induction step: We show that the claim holds for d_{ij}, $1 \le i \le m, 1 \le j \le n$. By induction assumption, $d_{pq} = ed(A_p, B_q)$ when $p+q < i+j$.

The value $ed(A_i, B_j)$ is based on an optimal edit sequence. We have three cases depending on what the last edit operation is:

N or S: $ed(A_i, B_j) = ed(A_{i-1}, B_{j-1}) + \delta(A[i], B[j]) = d_{i-1,j-1} + \delta(A[i], B[j])$.

I: $ed(A_i, B_j) = ed(A_i, B_{j-1}) + 1 = d_{i,j-1} + 1$.

D: $ed(A_i, B_j) = ed(A_{i-1}, B_j) + 1 = d_{i-1,j} + 1$.

Since the edit sequence is optimal, the correct value is the minimum of the three cases, which agrees with the definition of d_{ij}. □

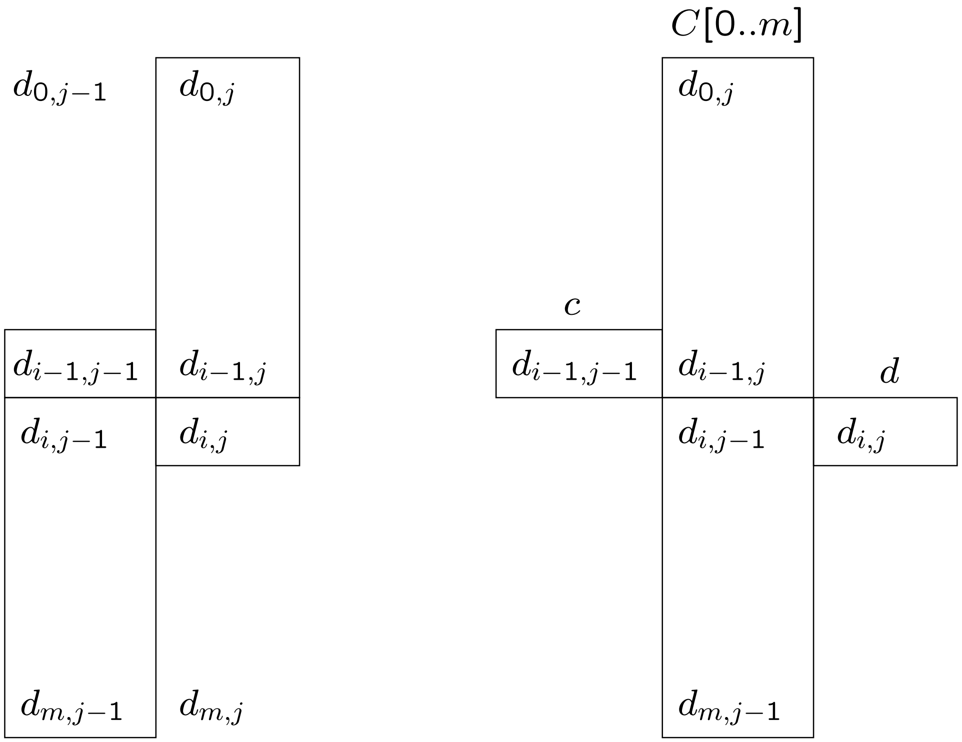
The recurrence gives directly a dynamic programming algorithm for computing the edit distance.

Algorithm 4.4: Edit distance
Input: strings $A[1..m]$ and $B[1..n]$
Output: $ed(A,B)$
(1) for $i \leftarrow 0$ to m do $d_{i0} \leftarrow i$
(2) for $j \leftarrow 1$ to n do $d_{0j} \leftarrow j$
(3) for $j \leftarrow 1$ to n do
(4) for $i \leftarrow 1$ to m do
(5) $d_{ij} \leftarrow \min\{d_{i-1,j-1} + \delta(A[i], B[j]), d_{i-1,j} + 1, d_{i,j-1} + 1\}$
(6) return d_{mn}

The time and space complexity is $\mathcal{O}(mn)$.

The space complexity can be reduced by noticing that each column of the matrix (d_{ij}) depends only on the previous column. We do not need to store older columns.

A more careful look reveals that, when computing d_{ij}, we only need to store the bottom part of column $j-1$ and the already computed top part of column j. We store these in an array $C[0..m]$ and variables c and d as shown below:



Algorithm 4.5: Edit distance in $\mathcal{O}(m)$ space
Input: strings $A[1..m]$ and $B[1..n]$
Output: $ed(A, B)$

(1) **for** $i \leftarrow 0$ **to** m **do** $C[i] \leftarrow i$
(2) **for** $j \leftarrow 1$ **to** n **do**
(3) $\quad c \leftarrow C[0]$; $C[0] \leftarrow j$
(4) $\quad$ **for** $i \leftarrow 1$ **to** m **do**
(5) $\qquad d \leftarrow \min\{c + \delta(A[i], B[j]), C[i-1] + 1, C[i] + 1\}$
(6) $\qquad c \leftarrow C[i]$
(7) $\qquad C[i] \leftarrow d$
(8) return $C[m]$

- Note that because $ed(A, B) = ed(B, A)$ (exercise), we can assume that $m \leq n$.

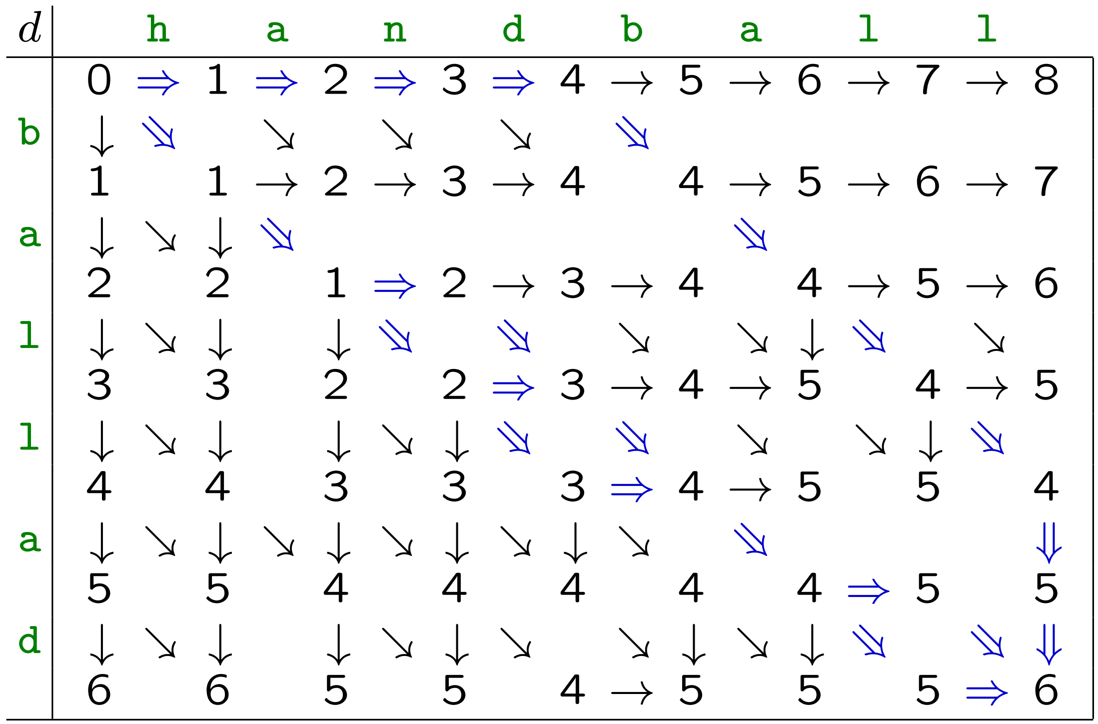
It is also possible to find optimal edit sequences and alignments from the matrix d_{ij}.

An edit graph is a directed graph, where the nodes are the cells of the edit distance matrix, and the edges are as follows:

- If $A[i] = B[j]$ and $d_{ij} = d_{i-1,j-1}$, there is an edge $(i-1, j-1) \to (i, j)$ labelled with N.
- If $A[i] \neq B[j]$ and $d_{ij} = d_{i-1,j-1} + 1$, there is an edge $(i-1, j-1) \to (i, j)$ labelled with S.
- If $d_{ij} = d_{i,j-1} + 1$, there is an edge $(i, j-1) \to (i, j)$ labelled with I.
- If $d_{ij} = d_{i-1,j} + 1$, there is an edge $(i-1, j) \to (i, j)$ labelled with D.

Any path from $(0, 0)$ to (m, n) is labelled with an optimal edit sequence.

Example 4.6: $A =$ ballad, $B =$ handball



There are 7 paths from $(0, 0)$ to $(6, 8)$ corresponding to, for example, the following edit sequences and alignments:

```
IIIINNNNDD   SNISSNIS   SNSSINSI
----ballad   ba-lla-d   ball-ad-
handball--   handball   handball
```

Approximate String Matching

Now we are ready to tackle the main problem of this part: approximate string matching.

Problem 4.7: Given a text $T[1..n]$, a pattern $P[1..m]$ and an integer $k \geq 0$, report all positions $j \in [1..m]$ such that $ed(P, T(j - \ell...j]) \leq k$ for some $\ell \geq 0$.

The factor $T(j - \ell...j]$ is called an approximate occurrence of P.

There can be multiple occurrences of different lengths ending at the same position j, but usually it is enough to report just the end positions.
We ask for the end position rather than the start position because that is more natural for the algorithms.

Define the values g_{ij} with the recurrence:

$$
\begin{aligned}
&g_{0j} = 0,\ \ 0 \le j \le n, \\
&g_{i0} = i,\ \ 1 \le i \le m,\ \text{and} \\
&g_{ij} = \min \begin{cases} g_{i-1,j-1} + \delta(P[i], T[j]) \\ \quad g_{i-1,j} + 1 \\ \quad g_{i,j-1} + 1 \end{cases} \qquad 1 \le i \le m, 1 \le j \le n.
\end{aligned}
$$

Theorem 4.8: For all $0 \le i \le m$, $0 \le j \le n$:

$$g_{ij} = \min\{ed(P[1..i], T(j-\ell...j]) \mid 0 \le \ell \le j\}\ .$$

In particular, j is an ending position of an approximate occurrence if and only if $g_{mj} \le k$.

Proof. We use induction with respect to $i + j$.

Basis:

$$g_{00} = 0 = ed(\epsilon, \epsilon)$$
$$g_{0j} = 0 = ed(\epsilon, \epsilon) = ed(\epsilon, T(j - 0..j]) \qquad \text{(min at } \ell = 0\text{)}$$
$$g_{i0} = i = ed(P[1..i], \epsilon) = ed(P[1..i], T(0 - 0..0]) \quad (0 \le \ell \le j = 0)$$

Induction step: Essentially the same as in the proof of Theorem 4.2.

Example 4.9: $P = \texttt{match}$, $T = \texttt{remachine}$, $k = 1$

g		r	e	m	a	c	h	i	n	e
	0	0	0	0	0	0	0	0	0	0
m				⇘						
	1	1	1	0	1	1	1	1	1	1
a					⇘					
	2	2	2	1	0	1	2	2	2	2
t					⇓					
	3	3	3	2	1	1	2	3	3	3
c						⇘				
	4	4	4	3	2	1	2	3	4	4
h							⇘			
	5	5	5	4	3	2	1	2	3	4

One occurrence ending at position 6.

Algorithm 4.10: Approximate string matching
Input: text $T[1..n]$, pattern $P[1..m]$, and integer k
Output: end positions of all approximate occurrences of P

for $i \leftarrow 0$ to m do $g_{i0} \leftarrow i$
for $j \leftarrow 1$ to n do $g_{0j} \leftarrow 0$
for $j \leftarrow 1$ to n do
    for $i \leftarrow 1$ to m do
        $g_{ij} \leftarrow \min\{g_{i-1,j-1} + \delta(A[i], B[j]), g_{i-1,j} + 1, g_{i,j-1} + 1\}$
    if $q_{mj} \leq k$ then output j

- Time and space complexity is $\mathcal{O}(mn)$.

- The space complexity can be reduced to $\mathcal{O}(m)$ by storing only one column as in Algorithm 4.5.

Ukkonen’s Cut-off Heuristic

We can speed up the algorithm using the diagonal monotonicity of the matrix (g_{ij}):

> A diagonal d, $-m \leq d \leq n$, consists of the cells g_{ij} with $j - i = d$.
> Every diagonal in (g_{ij}) is monotonically increasing.

Example 4.11: Diagonals -3 and 2.

g		r	e	m	a	c	h	i	n	e
	0	0	0	0	0	0	0	0	0	0
m	1	1	1	0	1	1	1	1	1	1
a	2	2	2	1	0	1	2	2	2	2
t	3	3	3	2	1	1	2	3	3	3
c	4	4	4	3	2	1	2	3	4	4
h	5	5	5	4	3	2	1	2	3	4

More specifically, we have the following property.

Lemma 4.12: For every $i \in [1..m]$ and every $j \in [1..n]$, $g_{ij} = g_{i-1,j-1}$ or $g_{ij} = g_{i-1,j-1} + 1$.

Proof. By definition, $g_{ij} \leq g_{i-1,j-1} + \delta(P[i], T[j]) \leq g_{i-1,j-1} + 1$. We show that $g_{ij} \geq g_{i-1,j-1}$ by induction on $i + j$.

The induction assumption is that $g_{pq} \geq g_{p-1,q-1}$ when $p \in [1..m]$, $q \in [1..n]$ and $p + q < i + j$. At least one of the following holds:

1. $g_{ij} = g_{i-1,j-1} + \delta(P[i], T[j])$. Then $g_{ij} \geq g_{i-1,j-1}$.
2. $g_{ij} = g_{i-1,j} + 1$ and $i > 1$. Then

$$g_{ij} = g_{i-1,j} + 1 \overset{\text{ind. assump.}}{\geq} g_{i-2,j-1} + 1 \overset{\text{definition}}{\geq} g_{i-1,j-1}$$

3. $g_{ij} = g_{i,j-1} + 1$ and $j > 1$. Then

$$g_{ij} = g_{i,j-1} + 1 \overset{\text{ind. assump.}}{\geq} g_{i-1,j-2} + 1 \overset{\text{definition}}{\geq} g_{i-1,j-1}$$

4. $i = 1$. Then $g_{ij} \geq 0 = g_{i-1,j-1}$.

$g_{ij} = g_{i,j-1} + 1$ and $j = 1$ is not possible because $g_{i,1} \leq g_{i-1,0} + 1 < g_{i,0} + 1$. □