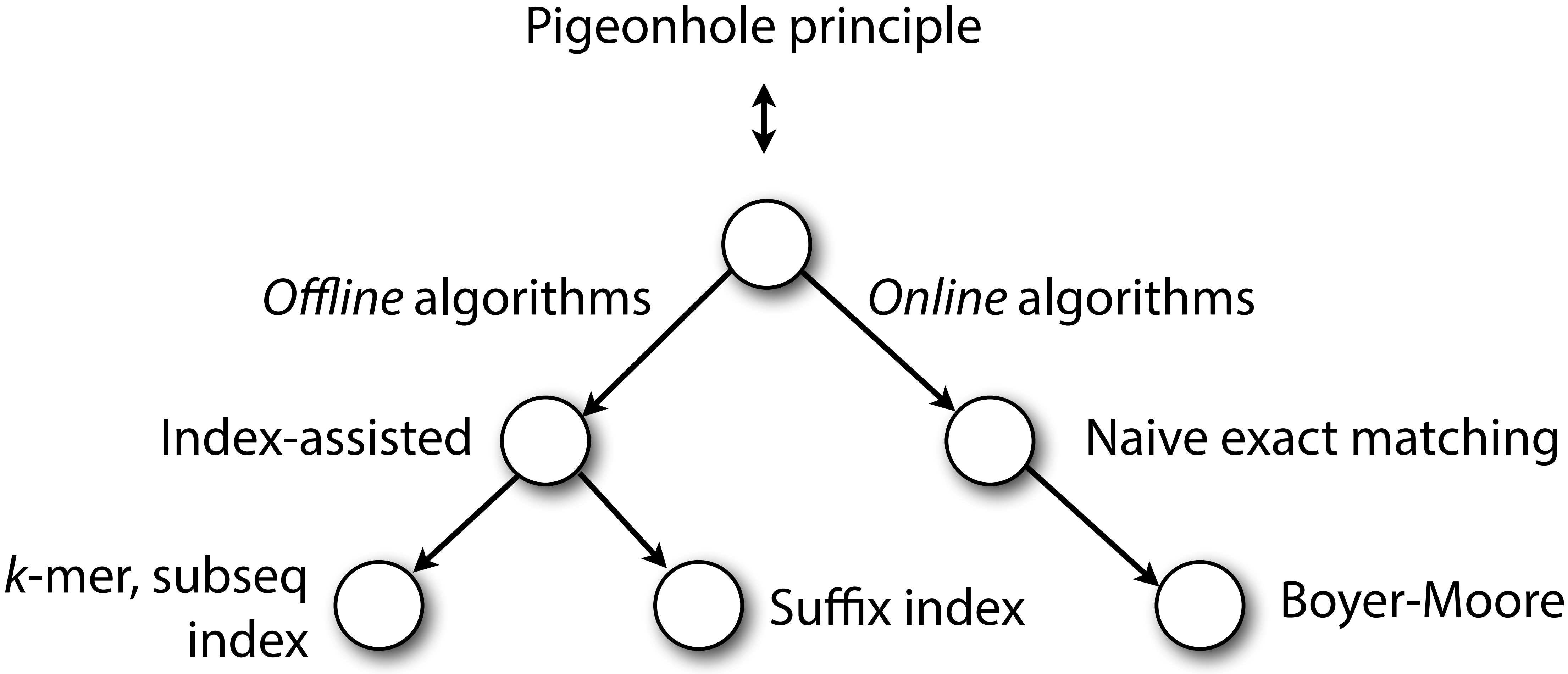
Edit distance



Pigeonhole principle

Offline algorithms

Online algorithms

Index-assisted

Naive exact matching

k-mer, subseq index

Suffix index

Boyer-Moore

Dynamic programming and edit distance

Edit distance

For X, Y where $|X| = |Y|$, *hamming distance* = minimum # substitutions needed to turn one into the other

For X, Y, *edit distance* = minimum # edits (substitutions, insertions, deletions) needed to turn one into the other

Finding distances

```
def hammingDistance(x, y):
    nmm = 0
    for i in xrange(0, len(x)):
        if x[i] != y[i]:
            nmm += 1
    return nmm


def editDistance(x, y):

    ????
```

Edit distance

If $|X| = |Y|$ what can we say about the relationship between `editDistance`(X, Y) and `hammingDistance`(X, Y)?

`editDistance`(X, Y) ≤ `hammingDistance`(X, Y)

```
X: G C G T A T G C G G C T A - A C G C
   | |   | | | | | | | | | |   | | |
Y: G C - T A T G C G G C T A T A C G C
```

Edit distance

If *x* and *y* are different lengths, what can we say about `editDistance`(*X*, *Y*)?

$$\texttt{editDistance}(X, Y) \geq ||X| - |Y||$$

X: ? ?

Y: ? ? ? ?

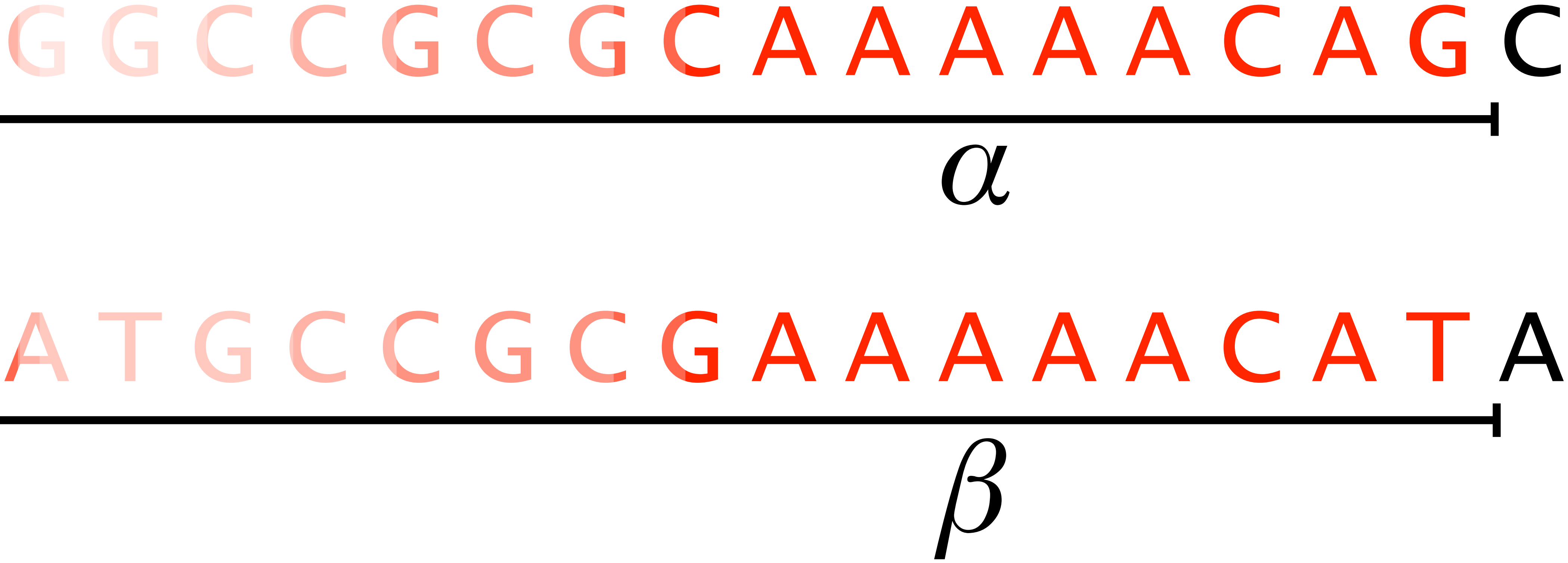
X

GGCCGCGCAAAAACAGC

Y

ATGCCGCGAAAAACATA

`editDistance`(*X*[:-1], *Y*[:-1]) = 147



GGCCGCGCAAAAACAGC

α

ATGCCGCGAAAAACATA

β

α C

β A

$$\texttt{edist}(\alpha\text{C}, \beta\text{A}) = \min\begin{cases} \texttt{edist}(\alpha, \beta) + 1 \\ \texttt{edist}(\alpha\text{C}, \beta) + 1 \\ \texttt{edist}(\alpha, \beta\text{A}) + 1 \end{cases}$$

$$\boxed{\alpha}\,\mathrm{C}$$

$$\boxed{\beta}\,\mathrm{A}$$

$$\texttt{edist}(\alpha\mathrm{C}, \beta\mathrm{A}) = \min \begin{cases} \boxed{\texttt{edist}(\alpha, \beta) + 1} \\ \texttt{edist}(\alpha\mathrm{C}, \beta) + 1 \\ \texttt{edist}(\alpha, \beta\mathrm{A}) + 1 \end{cases}$$

$$\boxed{\alpha\ \mathrm{C}}$$

$$\boxed{\beta}\ \mathrm{A}$$

$$\texttt{edist}(\alpha\mathrm{C}, \beta\mathrm{A}) = \min \begin{cases} \texttt{edist}(\alpha, \beta) + 1 \\ \boxed{\texttt{edist}(\alpha\mathrm{C}, \beta) + 1} \\ \texttt{edist}(\alpha, \beta\mathrm{A}) + 1 \end{cases}$$

$$\boxed{\alpha}\ \mathrm{C}$$

$$\boxed{\beta\ \mathrm{A}}$$

$$\texttt{edist}(\alpha\mathrm{C}, \beta\mathrm{A}) = \min \begin{cases} \texttt{edist}(\alpha, \beta) + 1 \\ \texttt{edist}(\alpha\mathrm{C}, \beta) + 1 \\ \boxed{\texttt{edist}(\alpha, \beta\mathrm{A}) + 1} \end{cases}$$

$$\alpha \; \mathtt{x}$$

$$\beta \; \mathtt{y}$$

$$\texttt{edist}(\alpha \mathrm{x}, \beta \mathrm{y}) = \min \begin{cases} \texttt{edist}(\alpha, \beta) + \delta(\mathrm{x}, \mathrm{y}) \\ \texttt{edist}(\alpha \mathrm{x}, \beta) + 1 \\ \texttt{edist}(\alpha, \beta \mathrm{y}) + 1 \end{cases}$$

$\delta(\mathrm{x}, \mathrm{y}) = 0$ if $\mathrm{x} = \mathrm{y}$, or 1 otherwise

```
delt = 1 if a[-1] != b[-1] else 0
return min(edDistRecursive(a[:-1], b[:-1]) + delt,
           edDistRecursive(a, b[:-1]) + 1,
           edDistRecursive(a[:-1], b) + 1)
```

$$\mathtt{edist}(\alpha x, \beta y) = \min \begin{cases} \mathtt{edist}(\alpha, \beta) + \delta(x, y) \\ \mathtt{edist}(\alpha x, \beta) + 1 \\ \mathtt{edist}(\alpha, \beta y) + 1 \end{cases}$$

$\delta(x, y) = 0$ if $x = y$, or 1 otherwise

```python
def edDistRecursive(a, b):

    delt = 1 if a[-1] != b[-1] else 0
    return min(edDistRecursive(a[:-1], b[:-1]) + delt,
               edDistRecursive(a[:-1], b) + 1,
               edDistRecursive(a, b[:-1]) + 1)
```

```python
def edDistRecursive(a, b):
    if len(a) == 0:
        return len(b)
    if len(b) == 0:
        return len(a)
    delt = 1 if a[-1] != b[-1] else 0
    return min(edDistRecursive(a[:-1], b[:-1]) + delt,
               edDistRecursive(a[:-1], b) + 1,
               edDistRecursive(a, b[:-1]) + 1)
```