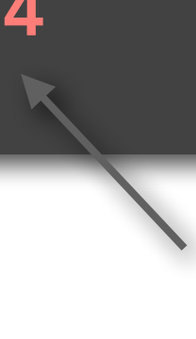
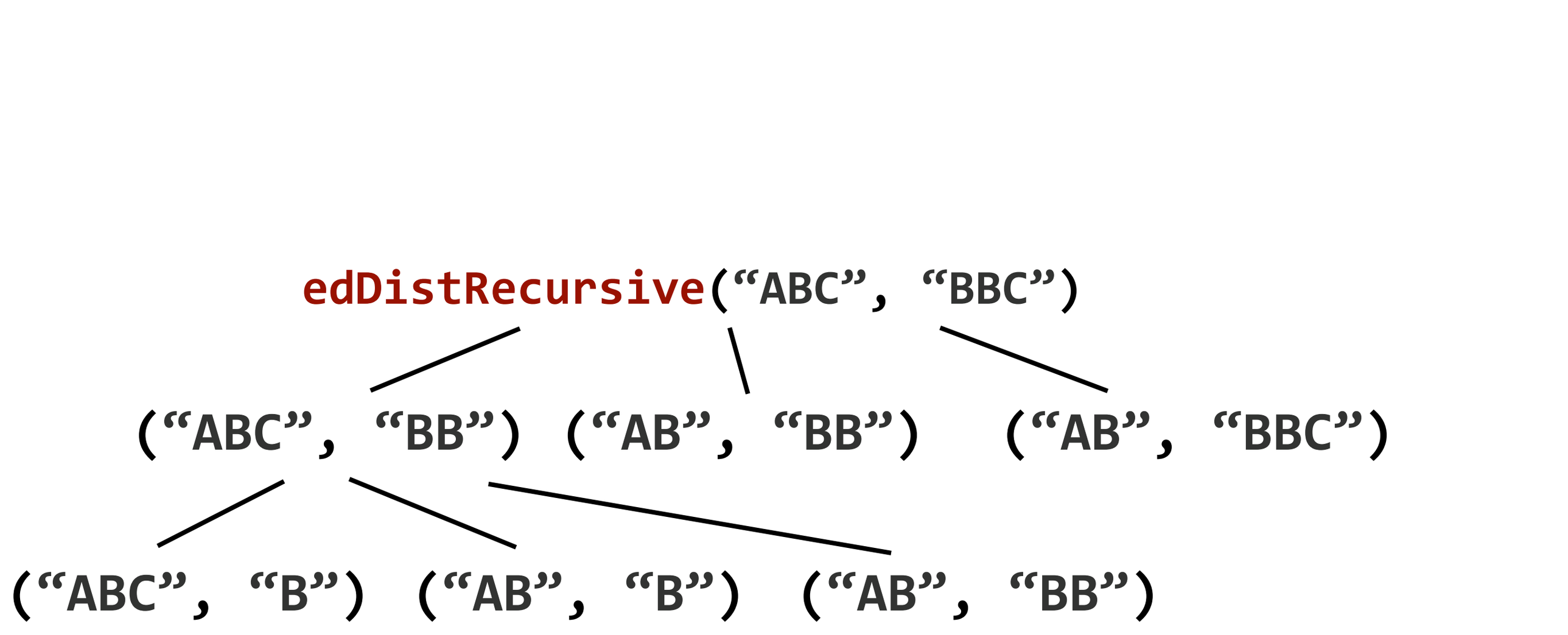
Edit distance

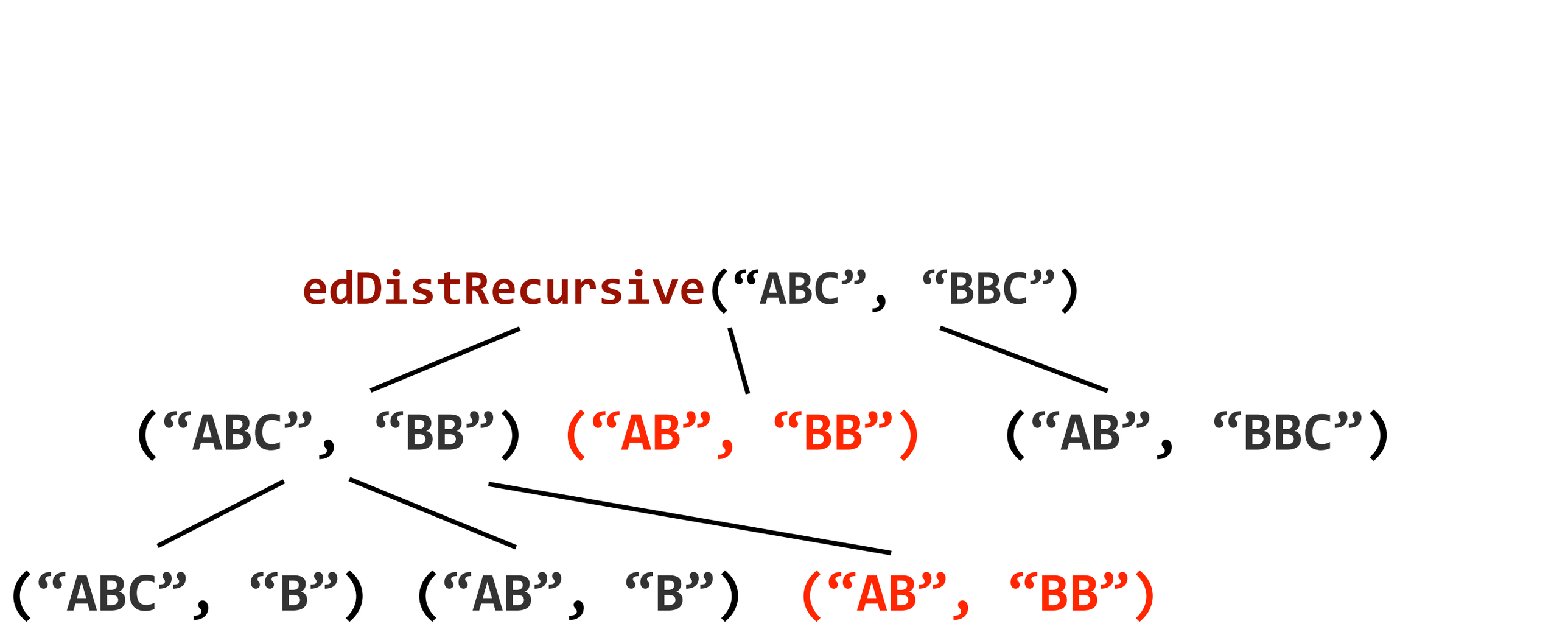
```
def edDistRecursive(a, b):
    if len(a) == 0:
        return len(b)
    if len(b) == 0:
        return len(a)
    delt = 1 if a[-1] != b[-1] else 0
    return min(edDistRecursive(a[:-1], b[:-1]) + delt,
               edDistRecursive(a[:-1], b) + 1,
               edDistRecursive(a, b[:-1]) + 1)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

```
>>> import datetime as d
>>> st = d.datetime.now(); \
... edDistRecursive("Shakespeare", "shake spear"); \
... print (d.datetime.now()-st).total_seconds()
3
31.498284
```



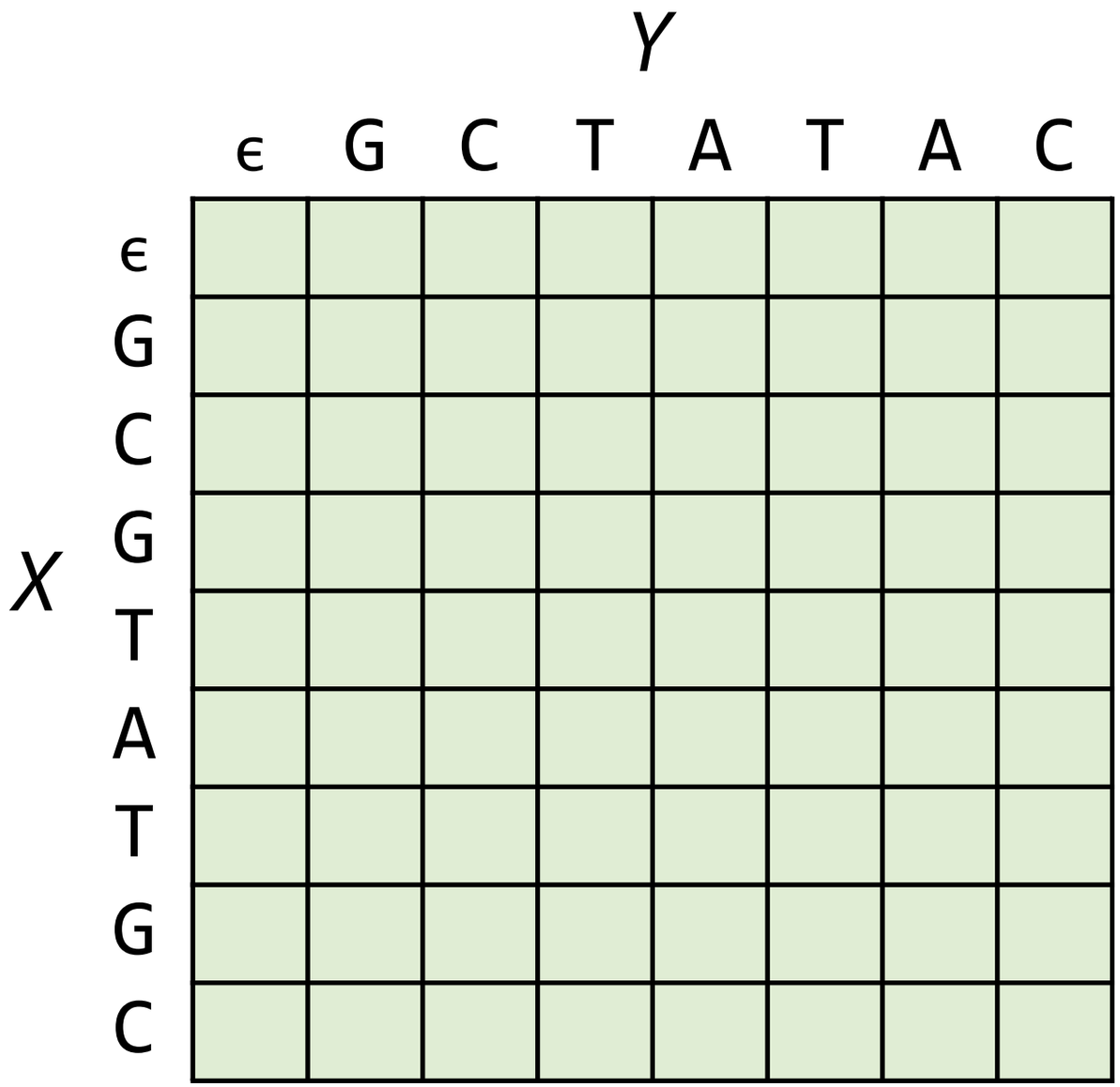
31.5 seconds!

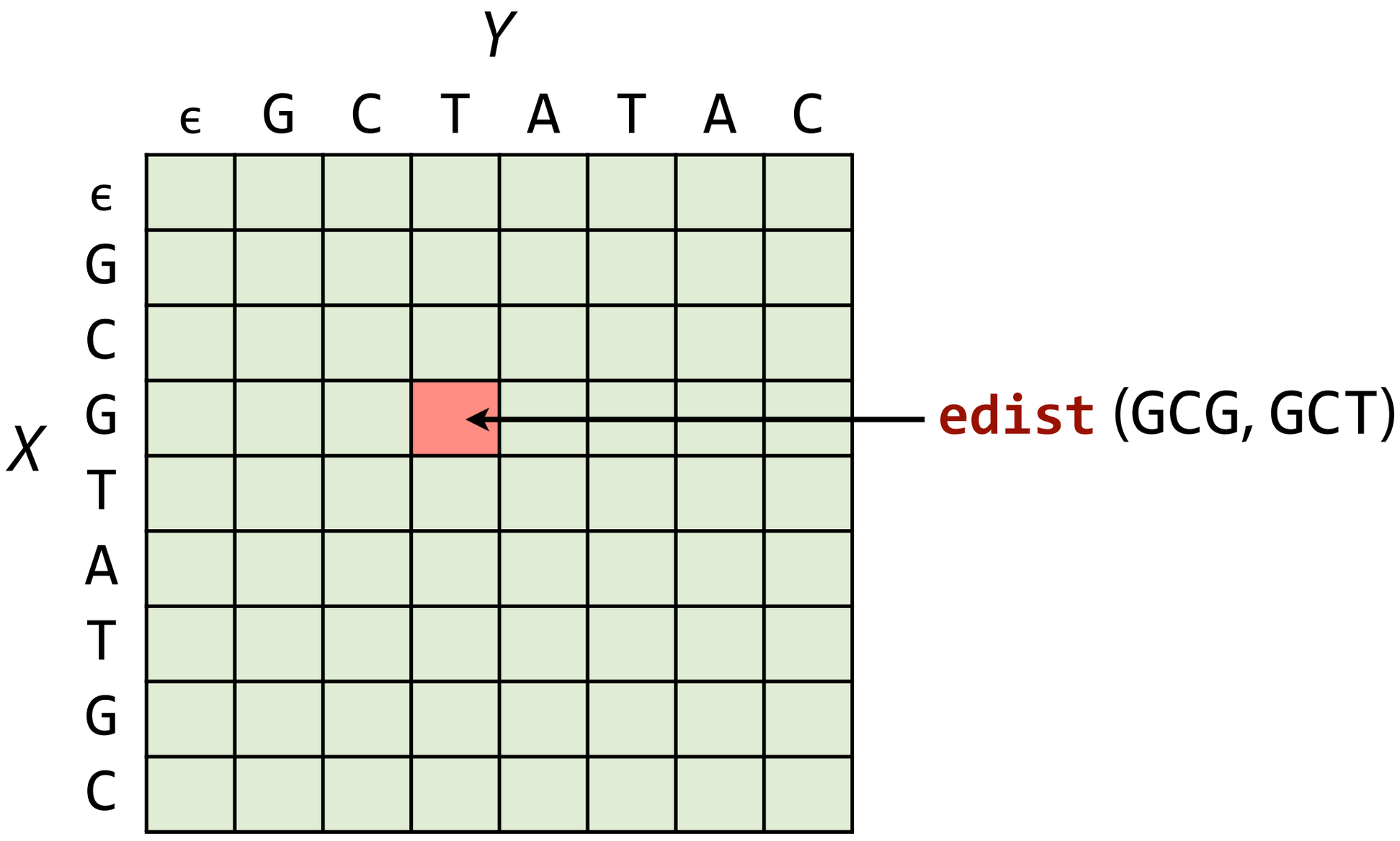

editDistRecursive("ABC", "BBC")
("ABC", "BB") ("AB", "BB") ("AB", "BBC")
("ABC", "B") ("AB", "B") ("AB", "BB")

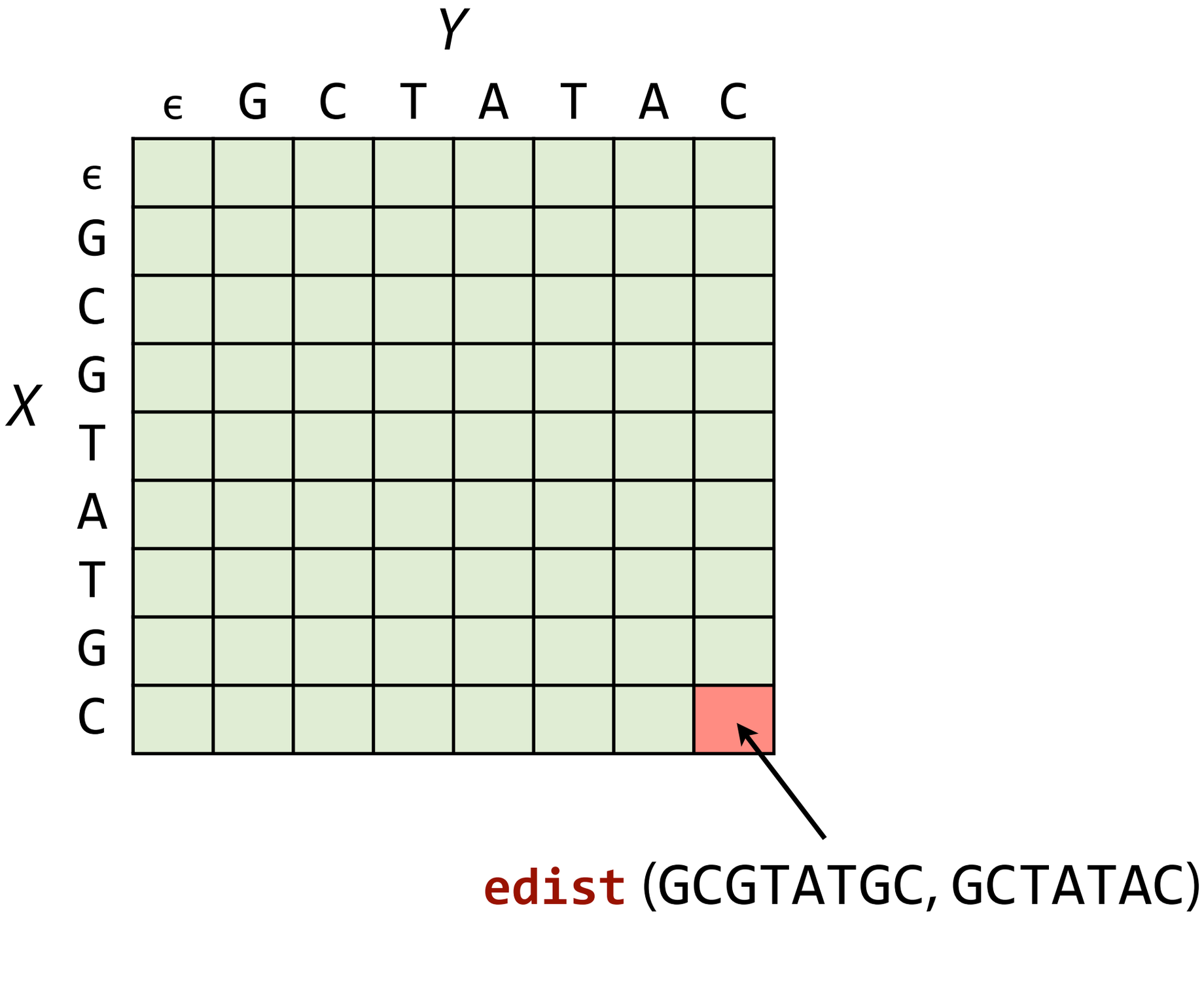

edDistRecursive("ABC", "BBC")
("ABC", "BB") ("AB", "BB") ("AB", "BBC")
("ABC", "B") ("AB", "B") ("AB", "BB")

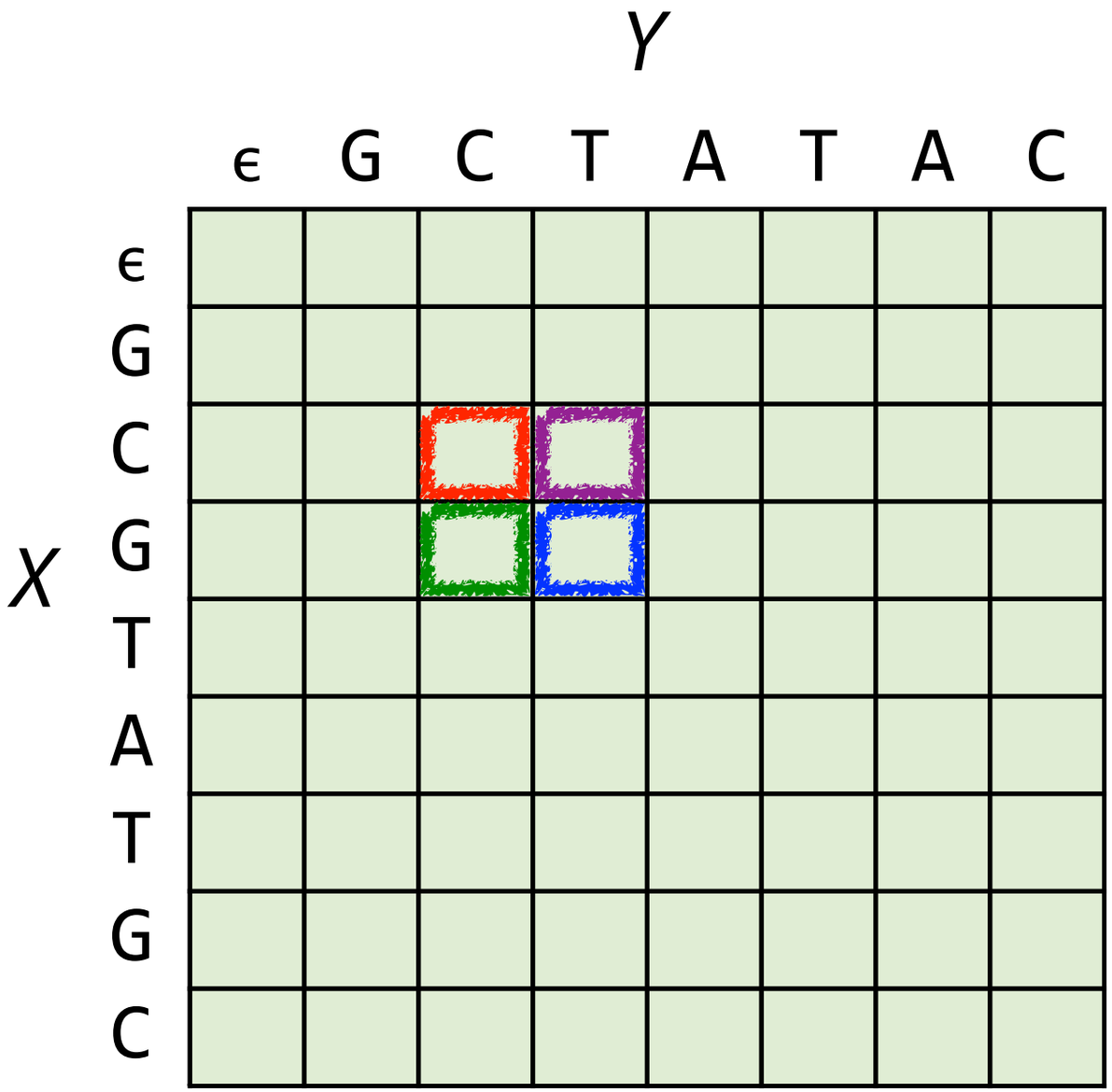
```
n = 0
def edDistRecursive(a, b):
    global n
    if len(a) == 0:
        return len(b)
    if len(b) == 0:
        return len(a)
    if a == 'Shake' and b == 'shake':
        n += 1
    delt = 1 if a[-1] != b[-1] else 0
    return min(edDistRecursive(a[:-1], b[:-1]) + delt,
               edDistRecursive(a[:-1], b) + 1,
               edDistRecursive(a, b[:-1]) + 1)
```

```
>>> edDistRecursive("Shakespeare", "shake spear")
3
>>> n
8989
```

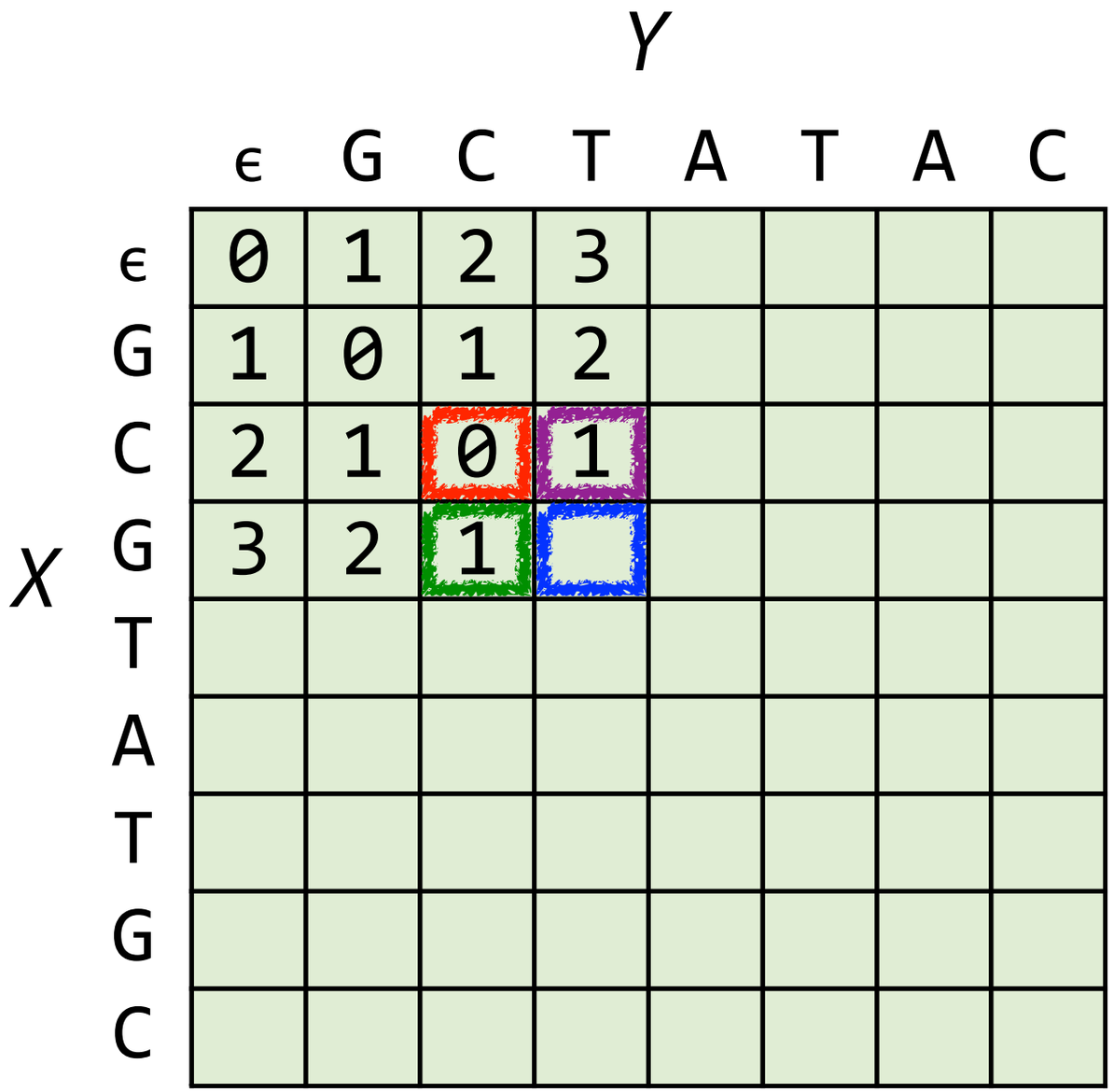

Y
ϵ G C T A T A C
X
ϵ
G
C
G
T
A
T
G
C


Y
ϵ G C T A T A C
X
ϵ
G
C
G
T
A
T
G
C
edist (GCG, GCT)


Y
ϵ G C T A T A C
X
ϵ
G
C
G
T
A
T
G
C
edist (GCGTATGC, GCTATAC)



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



$$\mathtt{edist}(\alpha x, \beta y) = \min \begin{cases} \mathtt{edist}(\alpha, \beta) + \delta(x, y) & = 0 + 1 = 1 \\ \mathtt{edist}(\alpha x, \beta) + 1 & = 1 + 1 = 2 \\ \mathtt{edist}(\alpha, \beta y) + 1 & = 1 + 1 = 2 \end{cases}$$

		Y							
		ϵ	G	C	T	A	T	A	C
	ϵ	0	1	2	3				
	G	1	0	1	2				
	C	2	1	0	1				
X	G	3	2	1	1				
	T								
	A								
	T								
	G								
	C								

$$\texttt{edist}(\alpha x, \beta y) = \min \begin{cases} \texttt{edist}(\alpha, \beta) + \delta(x, y) & = 0 + 1 = 1 \\ \texttt{edist}(\alpha x, \beta) + 1 & = 1 + 1 = 2 \\ \texttt{edist}(\alpha, \beta y) + 1 & = 1 + 1 = 2 \end{cases}$$

		Y						
		ϵ	G	C	T	A	T	A	C
	ϵ	0	1	2	3	4	5	6	7
	G	1	0	1	2	3	4	5	6
	C	2	1	0	1	2	3	4	5
X	G	3	2	1	1	2	3	4	5
	T	4	3	2	1	2	2	3	4
	A	5	4	3	2	1	2	2	3
	T	6	5	4	3	2	1	2	3
	G	7	6	5	4	3	2	2	3
	C	8	7	6	5	4	3	3	2

← Final result

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

```
>>> import datetime as d
>>> st = d.datetime.now(); \
... edDistMatrix("Shakespeare", "shake spear"); \
... print (d.datetime.now()-st).total_seconds()
3
0.000266
```

		Y							
		ϵ	G	C	T	A	T	A	C
X	ϵ	0	1	2	3	4	5	6	7
	G	1	0	1	2	3	4	5	6
	C	2	1	0	1	2	3	4	5
	G	3	2	1	1	2	3	4	5
	T	4	3	2	1	2	2	3	4
	A	5	4	3	2	1	2	2	3
	T	6	5	4	3	2	1	2	3
	G	7	6	5	4	3	2	2	3
	C	8	7	6	5	4	3	3	2

For any pair of prefixes from *X* & *Y*, edit distance is calculated *once*

Dynamic programming