Dynamic Programming: String Comparison

Pavel Pevzner

Department of Computer Science and Engineering
University of California, San Diego

Algorithmic Design and Techniques
Algorithms and Data Structures at edX

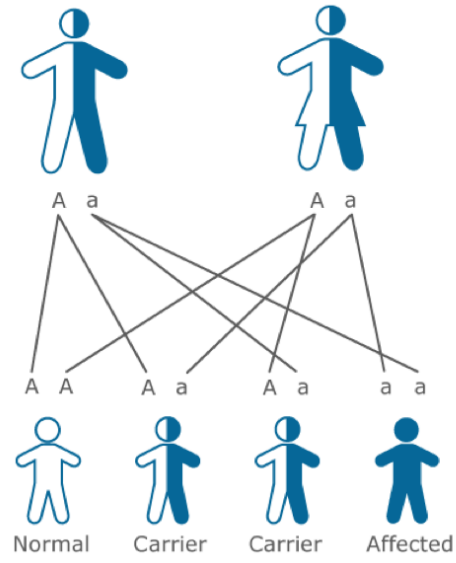
Cystic Fibrosis

Cystic fibrosis (CF): An often fatal disease which affects the respiratory system and produces an abnormally large amount of mucus.

- Mucus is a slimy material that coats epithelial surfaces and is secreted into fluids such as saliva.

Approximately 1 in 25 Humans Carry a Faulty CF Gene

- When BOTH parent carry a faulty gene, there is a 25% chance that their child will have cystic fibrosis.
- In the early 1980s biologists hypothesized that CF is caused by mutations in an unidentified gene.



Where Is the Cystic Fibrosis Gene?

- In the late 1980s, biologists narrowed the search for the CF gene to a million nucleotide long region on chromosome 7.
- However, this regions contained many genes and it was unclear which of them was responsible for CF.

chromosome 7

Where Is the Cystic Fibrosis Gene?

- In the late 1980s, biologists narrowed the search for the CF gene to a million nucleotide long region on chromosome 7.
- However, this regions contained many genes and it was unclear which of them was responsible for CF.

chromosome 7

Hint 1: Cystic fibrosis involves sweet secretion with abnormally high sodium levels

Where Is the Cystic Fibrosis Gene?

- In the late 1980s, biologists narrowed the search for the CF gene to a million nucleotide long region on chromosome 7.
- However, this regions contained many genes and it was unclear which of them was responsible for CF.

chromosome 7

Hint 1: Cystic fibrosis involves sweet secretion with abnormally high sodium levels

Hint 2: By that time biologists already knew the sequences of some genes responsible for secretion, e.g., **ATP binding proteins** act as transport channels responsible for secretion

Where Is the Cystic Fibrosis Gene?

- In the late 1980s, biologists narrowed the search for the CF gene to a million nucleotide long region on chromosome 7.
- However, this regions contained many genes and it was unclear which of them was responsible for CF.

chromosome 7

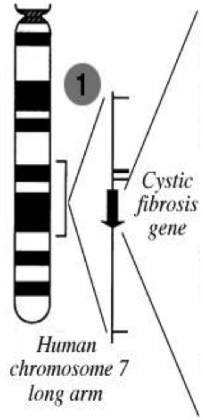
Hint 1: Cystic fibrosis involves sweet secretion with abnormally high sodium levels

Hint 2: By that time biologists already knew the sequences of some genes responsible for secretion, e.g., **ATP binding proteins** act as transport channels responsible for secretion

Hint 3: Should we search for genes in this region that are similar to known genes responsible for secretion?

Identifying the Cystic Fibrosis Gene

- BINGO: One of the genes in this region was similar to **ATP binding proteins** that act as transport channels responsible for secretion.



Hint 1: Cystic fibrosis involves sweet secretion with abnormally high sodium levels

Hint 2: By that time biologists already knew the sequences of some genes responsible for secretion, e.g., **ATP binding proteins** act as transport channels responsible for secretion

Hint 3: Should we search for genes in this region that are similar to known genes responsible for secretion?

Outline

1. The Alignment Game
2. Computing Edit Distance
3. Reconstructing an Optimal Alignment

The Alignment Game

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

Alignment game: remove all symbols from two strings in such a way that the number of points is maximized:

Remove the 1st symbol from **both** strings:

- 1 point if the symbols match,
- 0 points if they don't match

Remove the 1st symbol from **one** of the strings:

- 0 points

The Alignment Game

A	T	G	T	T	A	T	A
A	T	C	G	T	C	C	
+1							

The Alignment Game

A	T	G	T	T	A	T	A
A	T	C	G	T	C	C	
+1	+1						

The Alignment Game

```
A  T  -  G  T  T  A  T  A
A  T  C  G  T  C  C
+1 +1
```

The Alignment Game

```
A  T  -  G  T  T  A  T  A
A  T  C  G  T  C  C
+1 +1    +1
```

The Alignment Game

```
A T - G T T A T A
A T C G T C C
+1 +1   +1 +1
```

The Alignment Game

```
A T - G T T A T A
A T C G T - C C
+1 +1   +1 +1
```

The Alignment Game

```
A T - G T T A T A
A T C G T - C C
+1 +1   +1 +1
```

The Alignment Game

```
A T - G T T A T A
A T C G T - C - C
+1 +1   +1 +1
```

The Alignment Game

```
A T - G T T A T A
A T C G T - C - C
+1 +1   +1 +1
```

The Alignment Game

```
A T - G T T A T A
A T C G T - C - C
+1 +1   +1 +1       =4
```

Sequence Alignment

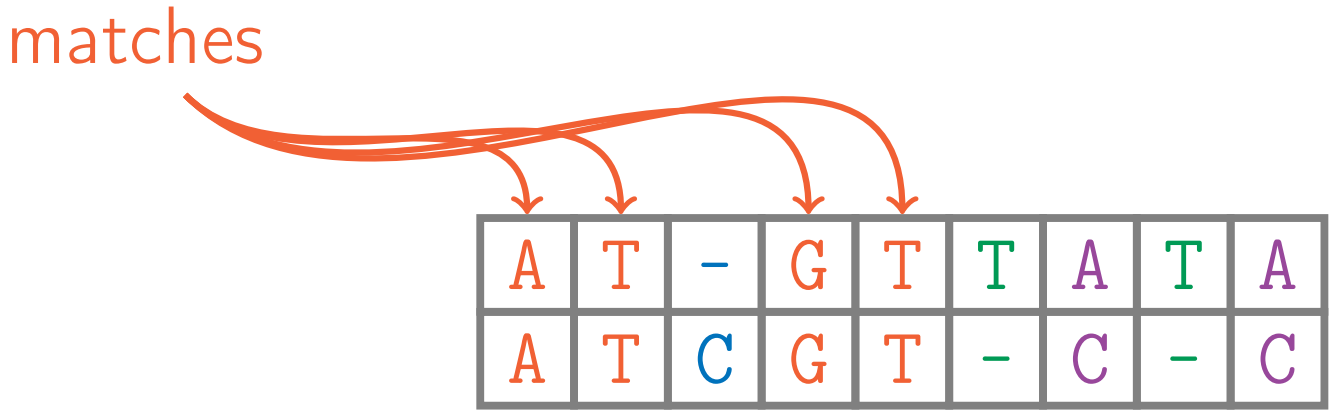
A	T	-	G	T	T	A	T	A
A	T	C	G	T	-	C	-	C

Alignment of two strings is a two-row matrix:

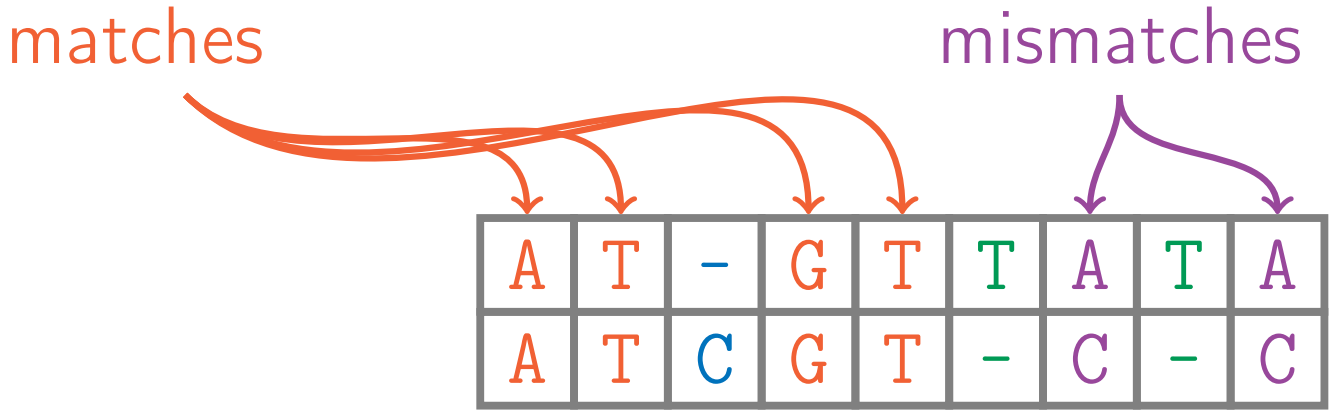
1st row: symbols of the 1st string (in order) interspersed by "–"

2nd row: symbols of the 2nd string (in order) interspersed by "–"

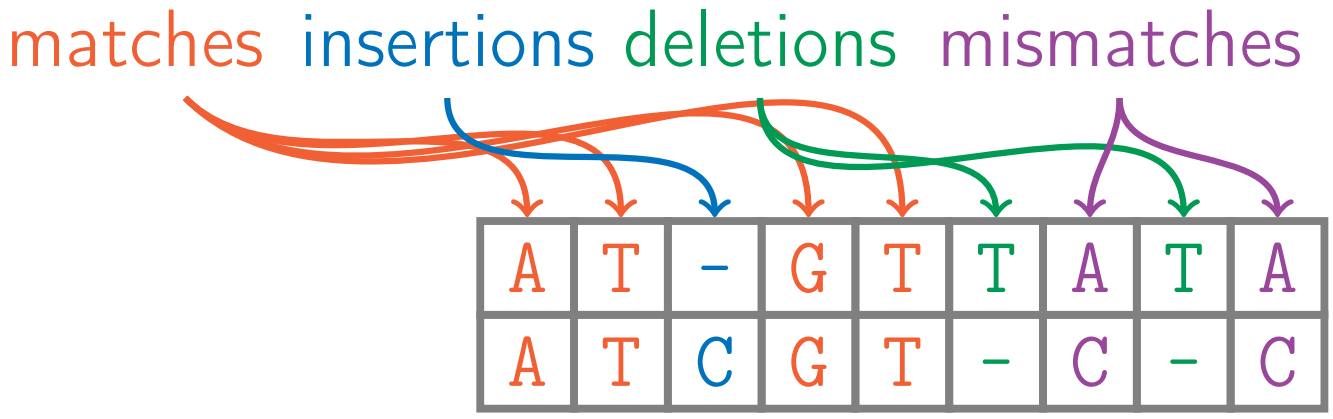
Sequence Alignment



Sequence Alignment



Sequence Alignment



Alignment Score

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

Alignment score: premium for every match ($+1$) and penalty for every mismatch ($-\mu$), indel ($-\sigma$).

Alignment Score

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

+1 +1 -1 +1 +1 -1 +0 -1 +0 =1

Alignment score: premium for every match ($+1$) and penalty for every mismatch ($-\mu$), indel ($-\sigma$).

Example: $\mu = 0$ and $\sigma = 1$

Alignment Score

$$\#\text{matches} - \mu \cdot \#\text{mismatches} - \sigma \cdot \#\text{indels}$$

Optimal alignment

Input: Two strings, mismatch penalty μ, and indel penalty σ.

Output: An alignment of the strings maximizing the score.

Common Subsequence

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

Matches in an alignment of two strings (ATGT) form their common subsequence

Longest common subsequence

Input: Two strings.

Output: A longest common subsequence of these strings.

Longest common subsequence

Input: Two strings.

Output: A longest common subsequence of these strings.

Maximizing the length of a common subsequence corresponds to maximizing the score of an alignment with $\mu = \sigma = 0$.

Edit distance

Input: Two strings.

Output: The minimum number of operations (insertions, deletions, and substitutions of symbols) to transform one string into another.

Edit distance

Input: Two strings.

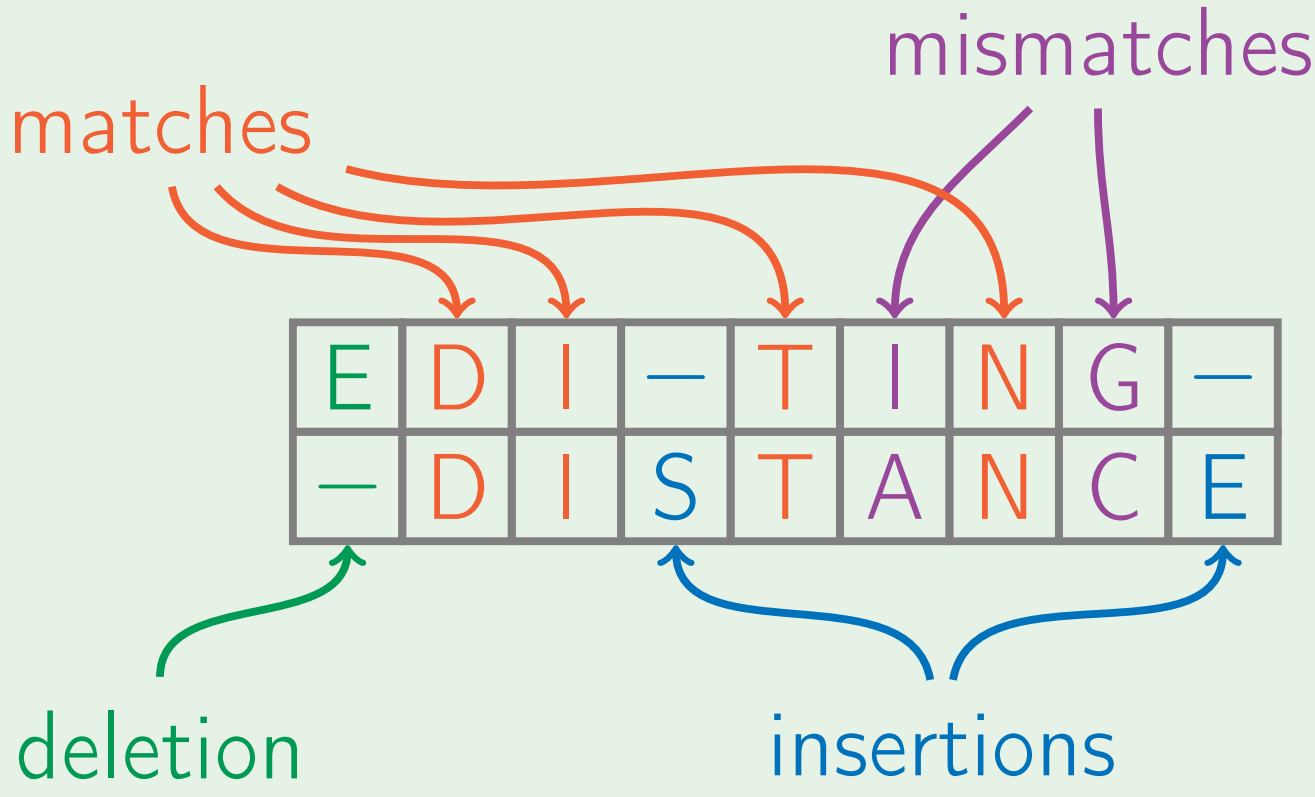
Output: The minimum number of operations (insertions, deletions, and substitutions of symbols) to transform one string into another.

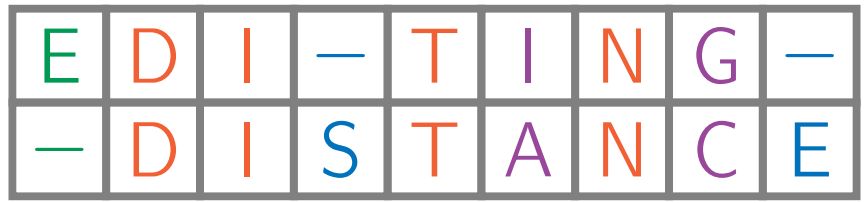
The minimum number of insertions, deletions and mismatches in an alignment of two strings (among all possible alignments).

Example

E	D	I	–	T	I	N	G	–
–	D	I	S	T	A	N	C	E

Example


mismatches
matches
E D I – T I N G –
– D I S T A N C E
deletion
insertions


E D I – T I N G –
– D I S T A N C E

E	D	I	–	T	I	N	G	–
–	D	I	S	T	A	N	C	E

the total number of symbols in two strings=

E	D	I	–	T	I	N	G	–
–	D	I	S	T	A	N	C	E

the total number of symbols in two strings=

+2·#matches
+2·#mismatches
+1·#insertions
+1·#deletions

E	D	I	−	T	I	N	G	−
−	D	I	S	T	A	N	C	E

the total number of symbols in two strings=

$$
\begin{aligned}
&+2\cdot\#\text{matches} \\
&+2\cdot\#\text{mismatches} \\
&+1\cdot\#\text{insertions} \\
&+1\cdot\#\text{deletions}
\end{aligned}
=
\begin{aligned}
&+2\cdot\#\text{matches} \\
&-1\cdot\#\text{insertions} \\
&-1\cdot\#\text{deletions} \\
&+2\cdot\#\text{mismatches} \\
&+2\cdot\#\text{insertions} \\
&+2\cdot\#\text{deletions}
\end{aligned}
$$

E	D	I	–	T	I	N	G	–
–	D	I	S	T	A	N	C	E

the total number of symbols in two strings=

$$
\begin{aligned}
&+2\cdot\#\text{matches} \\
&+2\cdot\#\text{mismatches} \\
&+1\cdot\#\text{insertions} \\
&+1\cdot\#\text{deletions}
\end{aligned}
=
\begin{aligned}
&\left.\begin{aligned}
&+2\cdot\#\text{matches} \\
&-1\cdot\#\text{insertions} \\
&-1\cdot\#\text{deletions}
\end{aligned}\right] \\
&\left.\begin{aligned}
&+2\cdot\#\text{mismatches} \\
&+2\cdot\#\text{insertions} \\
&+2\cdot\#\text{deletions}
\end{aligned}\right]
\end{aligned}
\qquad
\begin{aligned}
&2\cdot\text{AlignmentScore} \\
&(\mu = 0,\ \sigma = 1/2) \\
&+ \\
&2\cdot\text{EditDistance}
\end{aligned}
$$

E	D	I	–	T	I	N	G	–
–	D	I	S	T	A	N	C	E

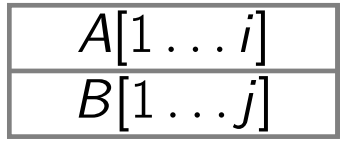
minimizing edit distance

=

maximizing alignment score

Outline

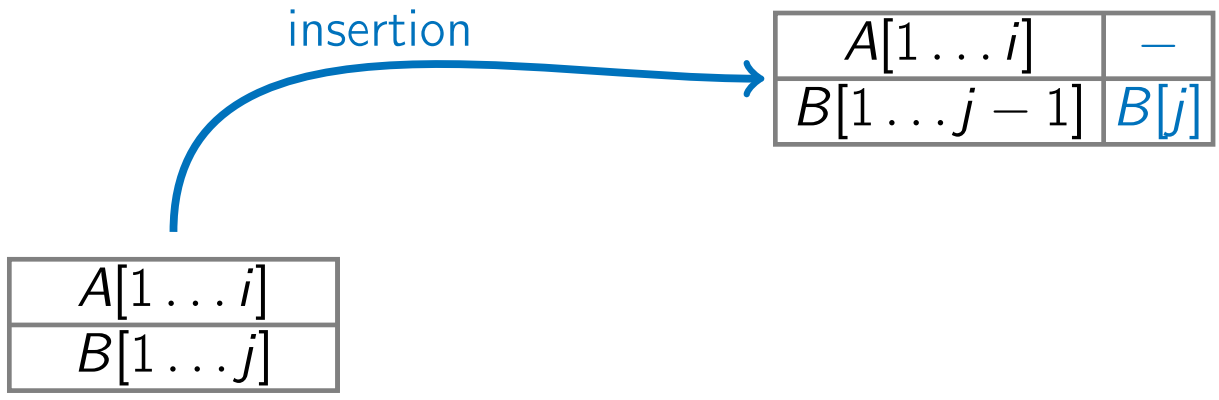
1. The Alignment Game
2. Computing Edit Distance
3. Reconstructing an Optimal Alignment



Given strings $A[1 \ldots n]$ and $B[1 \ldots m]$, what is an optimal alignment (an alignment that results in minimum edit distance) of an i-prefix $A[1 \ldots i]$ of the first string and a j-prefix $B[1 \ldots j]$ of the second string?

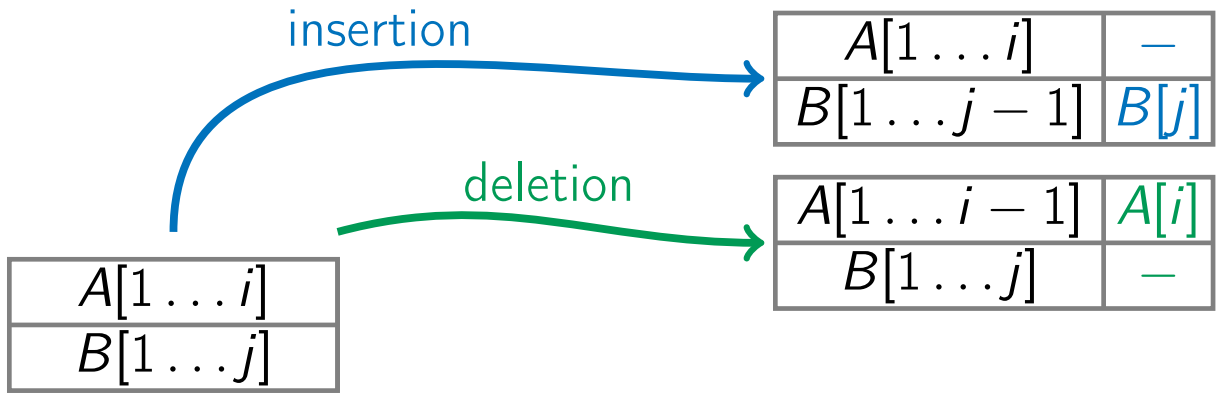
$A[1 \ldots i]$
$B[1 \ldots j]$

The last column of an optimal alignment is either



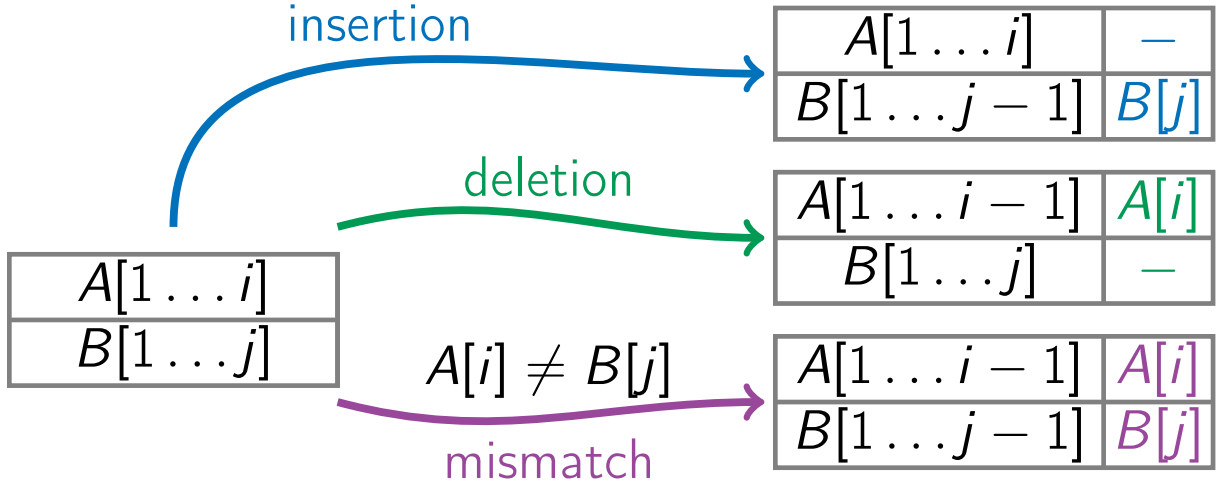
The last column of an optimal alignment is either

an insertion,



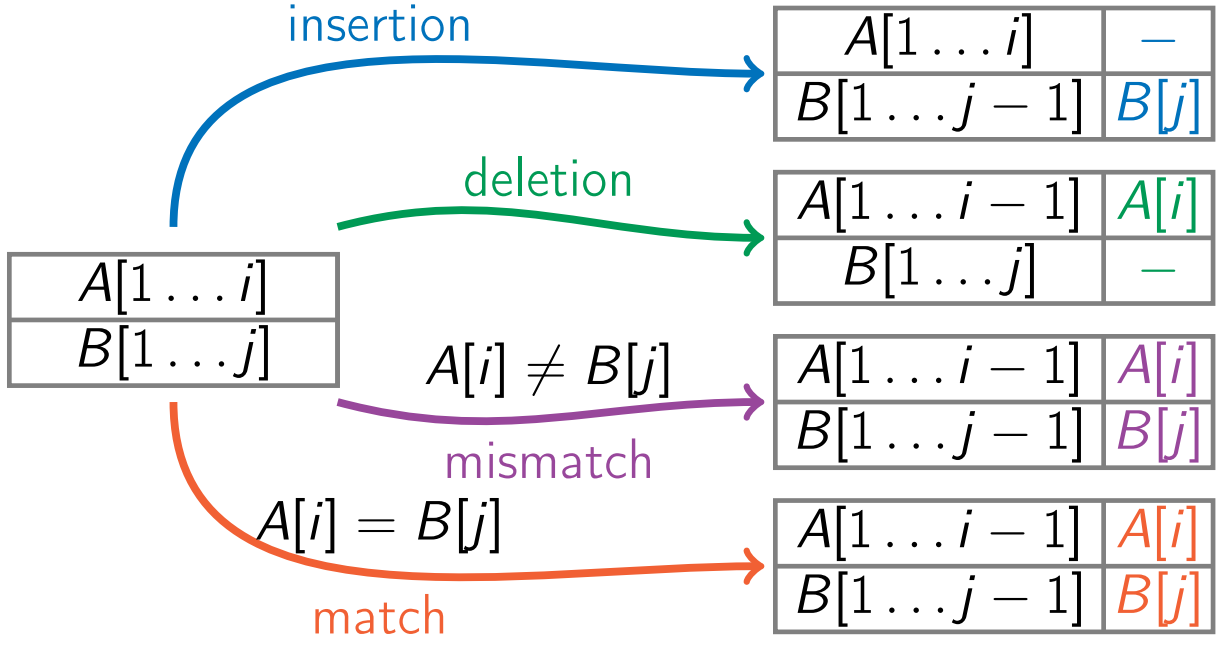
The last column of an optimal alignment is either

- an insertion,
- a deletion,



The last column of an optimal alignment is either

- an insertion,
- a deletion,
- a mismatch,



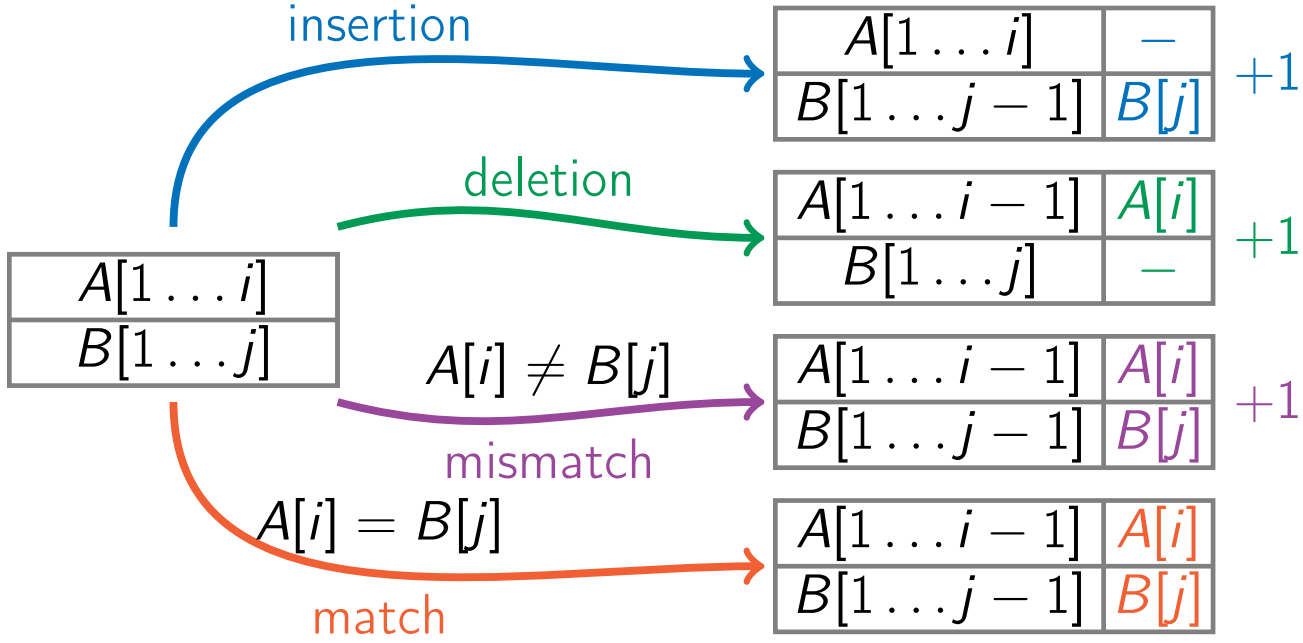
The last column of an optimal alignment is either

an insertion,

a deletion,

a mismatch,

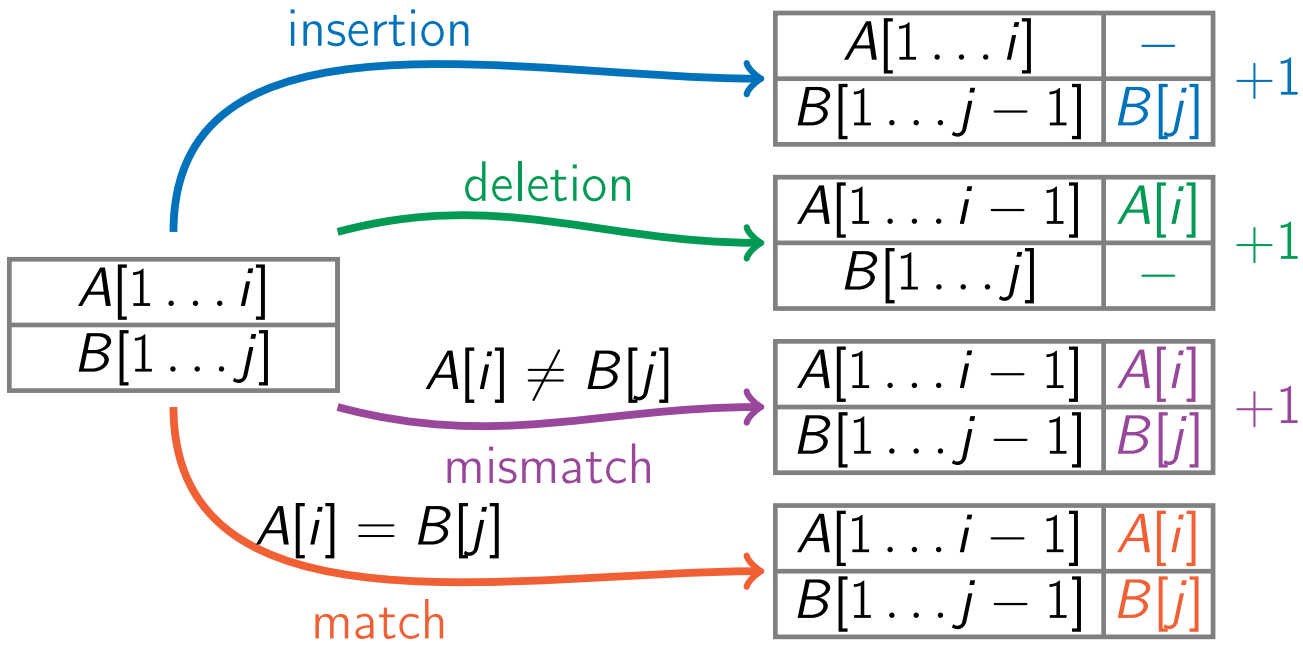
or a match.



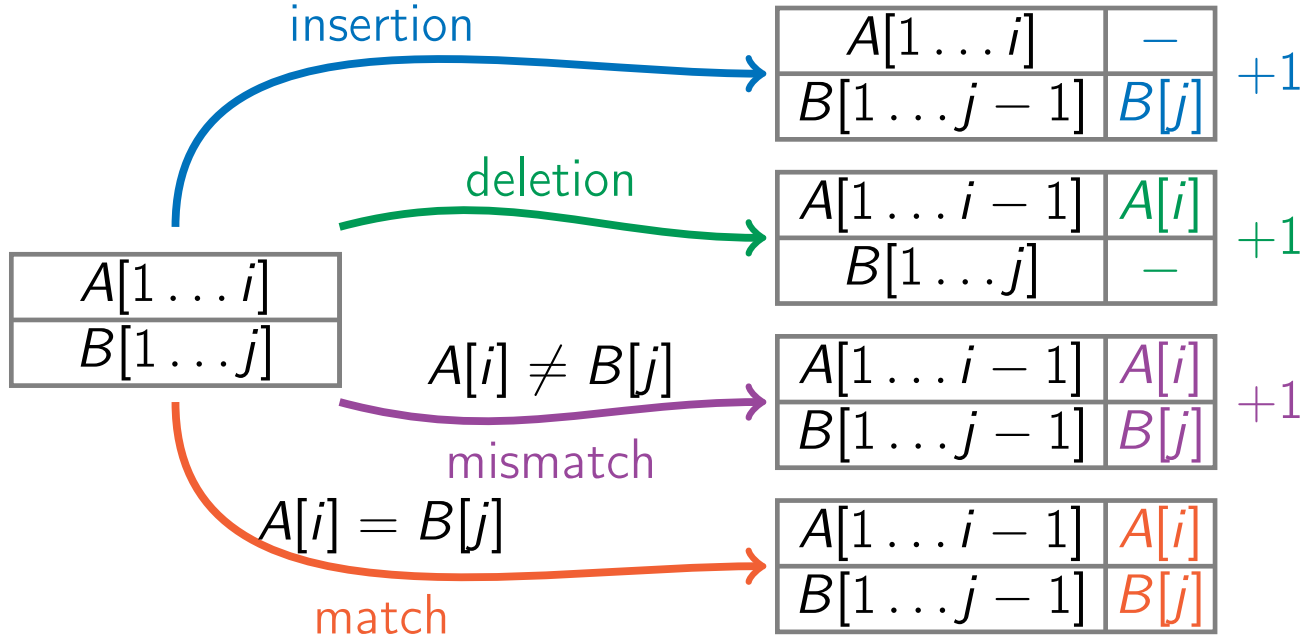
The last column of an optimal alignment is either

- an insertion,
- a deletion,
- a mismatch,
- or a match.

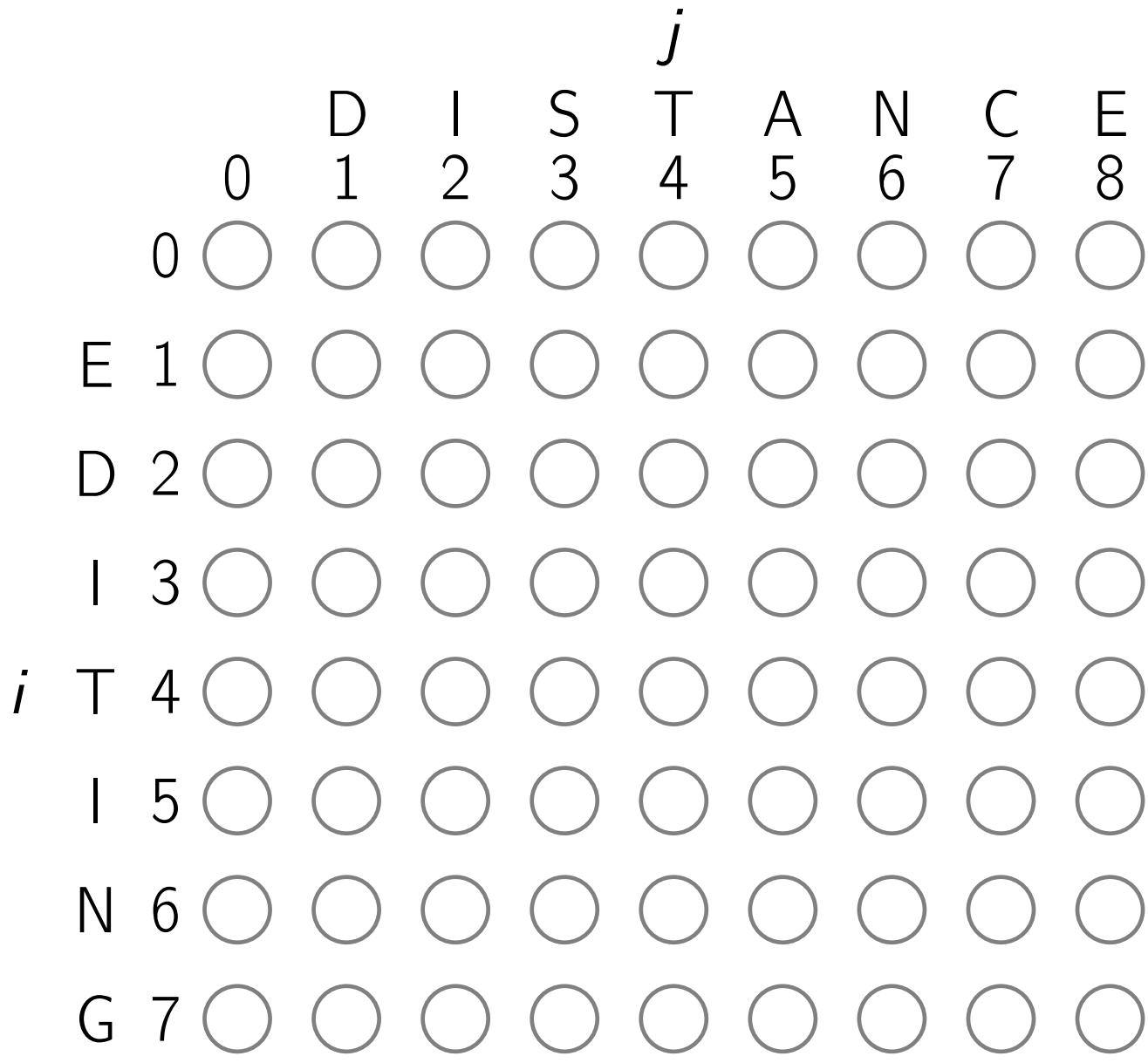
What is left (after the removal of the last column) is an **optimal** alignment of the corresponding two prefixes.



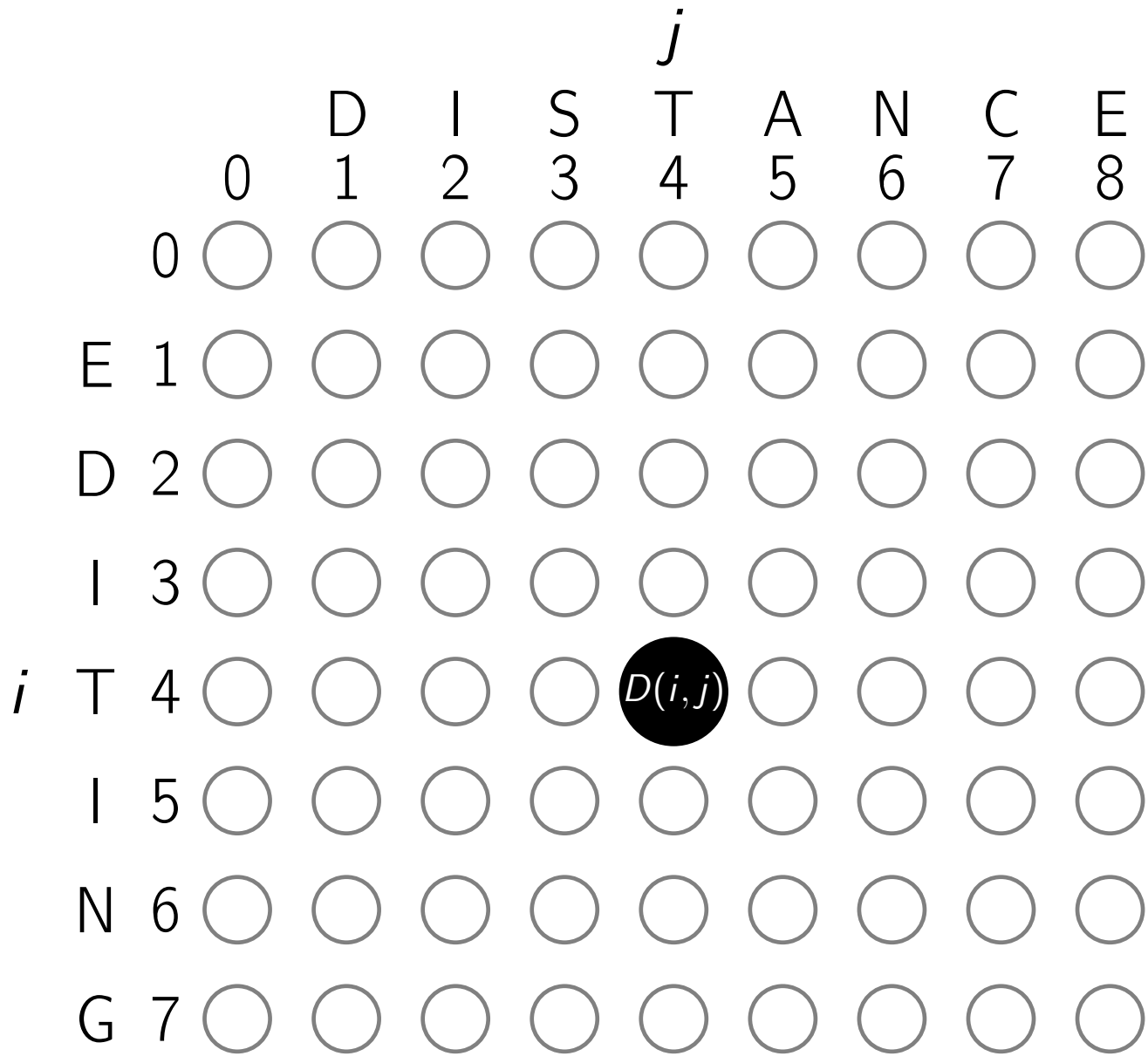
Let $D(i, j)$ be the edit distance of an i-prefix $A[1 \ldots i]$ and a j-prefix $B[1 \ldots j]$.



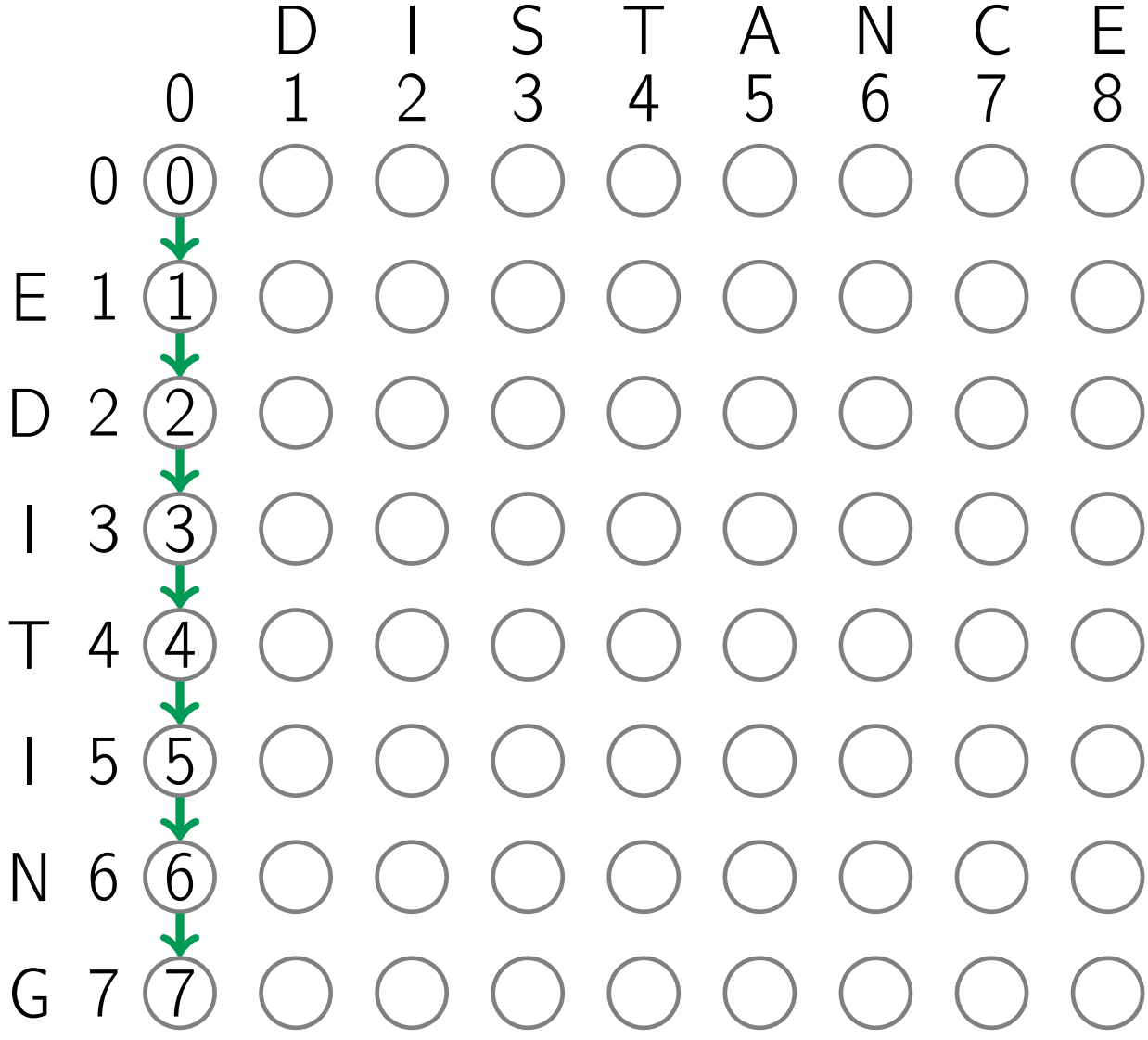
$$D(i,j) = \min \begin{cases} D(i,j-1)+1 & \\ D(i-1,j)+1 & \\ D(i-1,j-1)+1 & \text{if } A[i] \neq B[j] \\ D(i-1,j-1) & \text{if } A[i] = B[j] \end{cases}$$

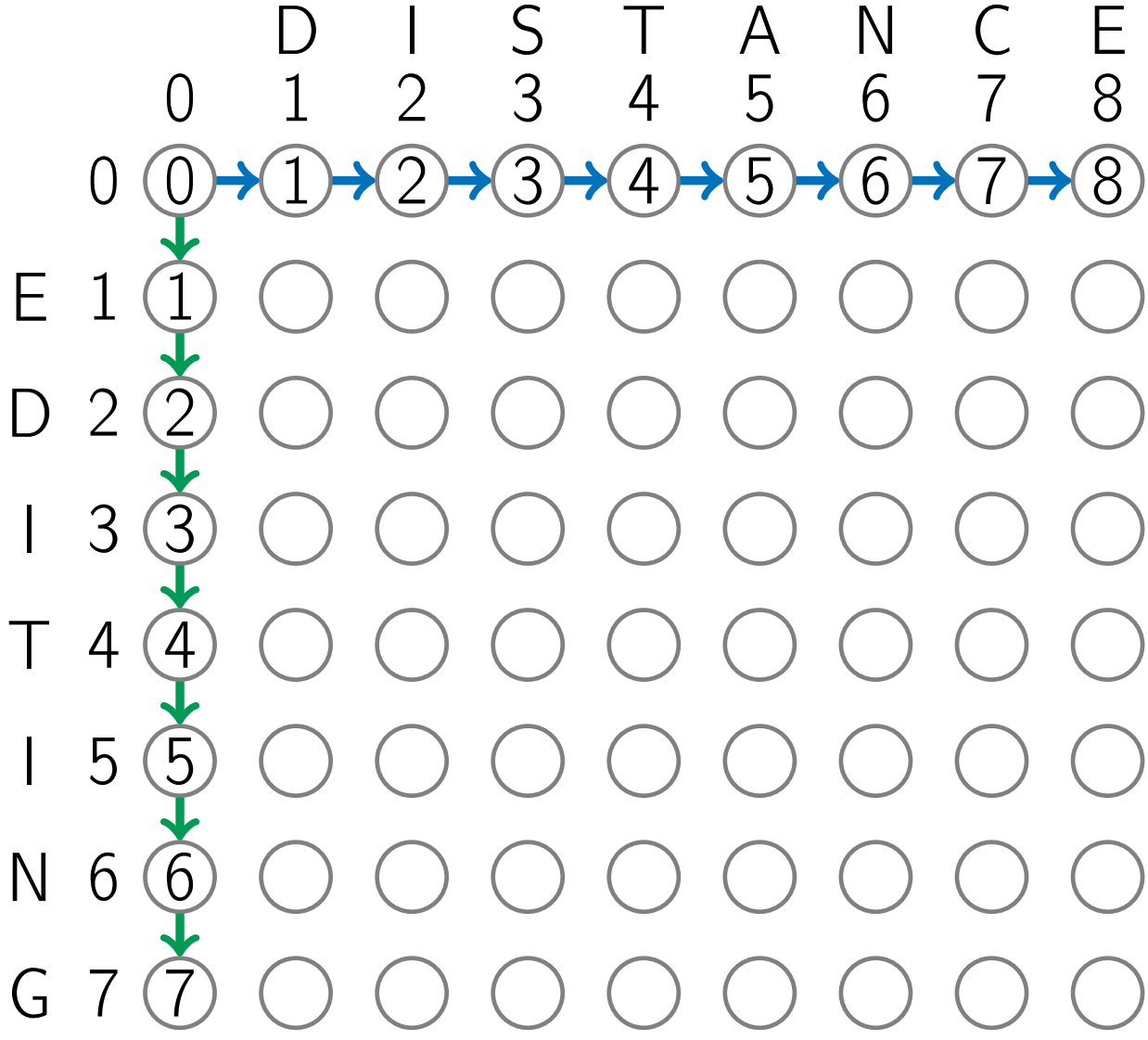


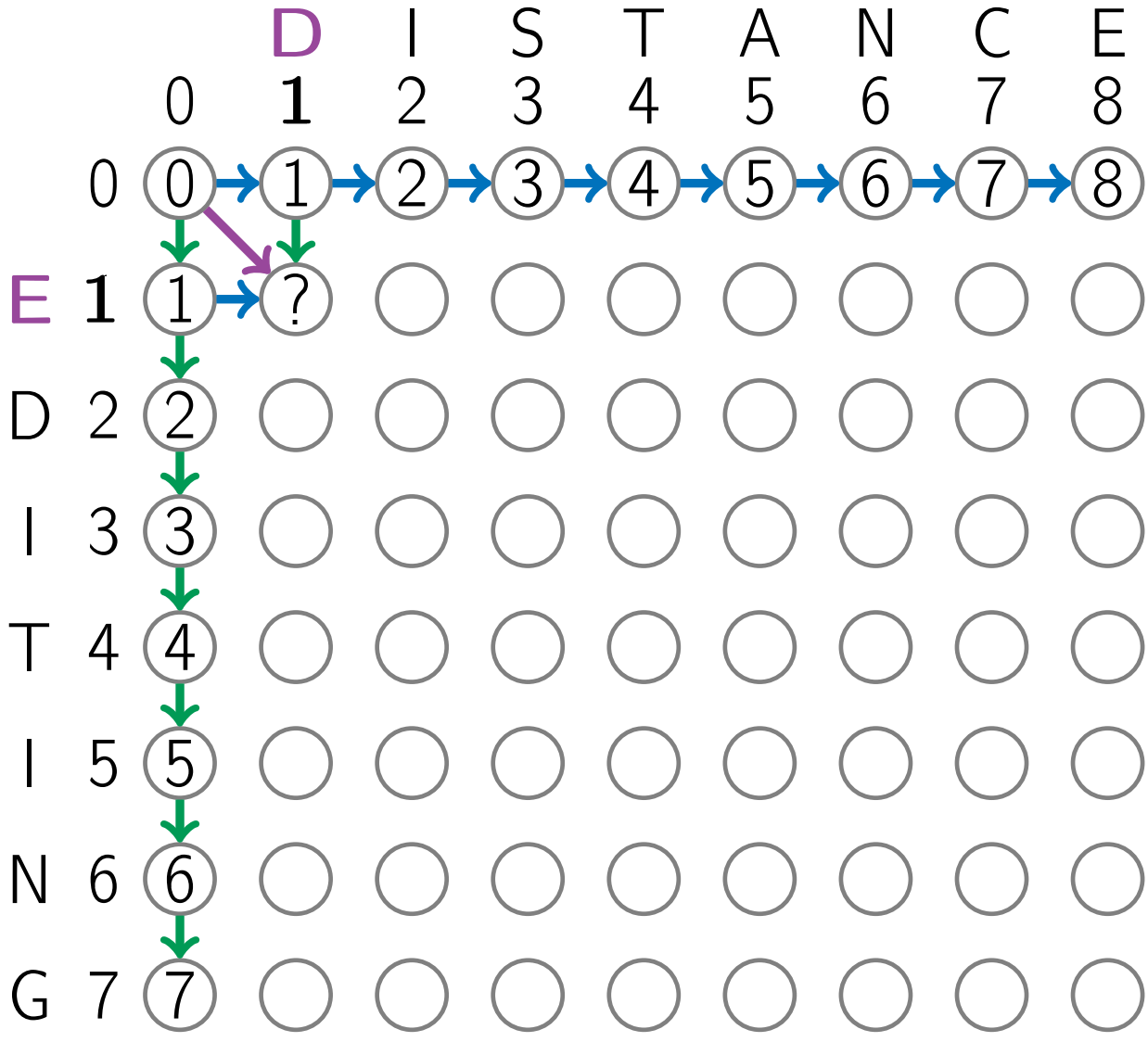
comparing $A[1 \ldots n] =$ EDITING
and $B[1 \ldots m] =$ DISTANCE



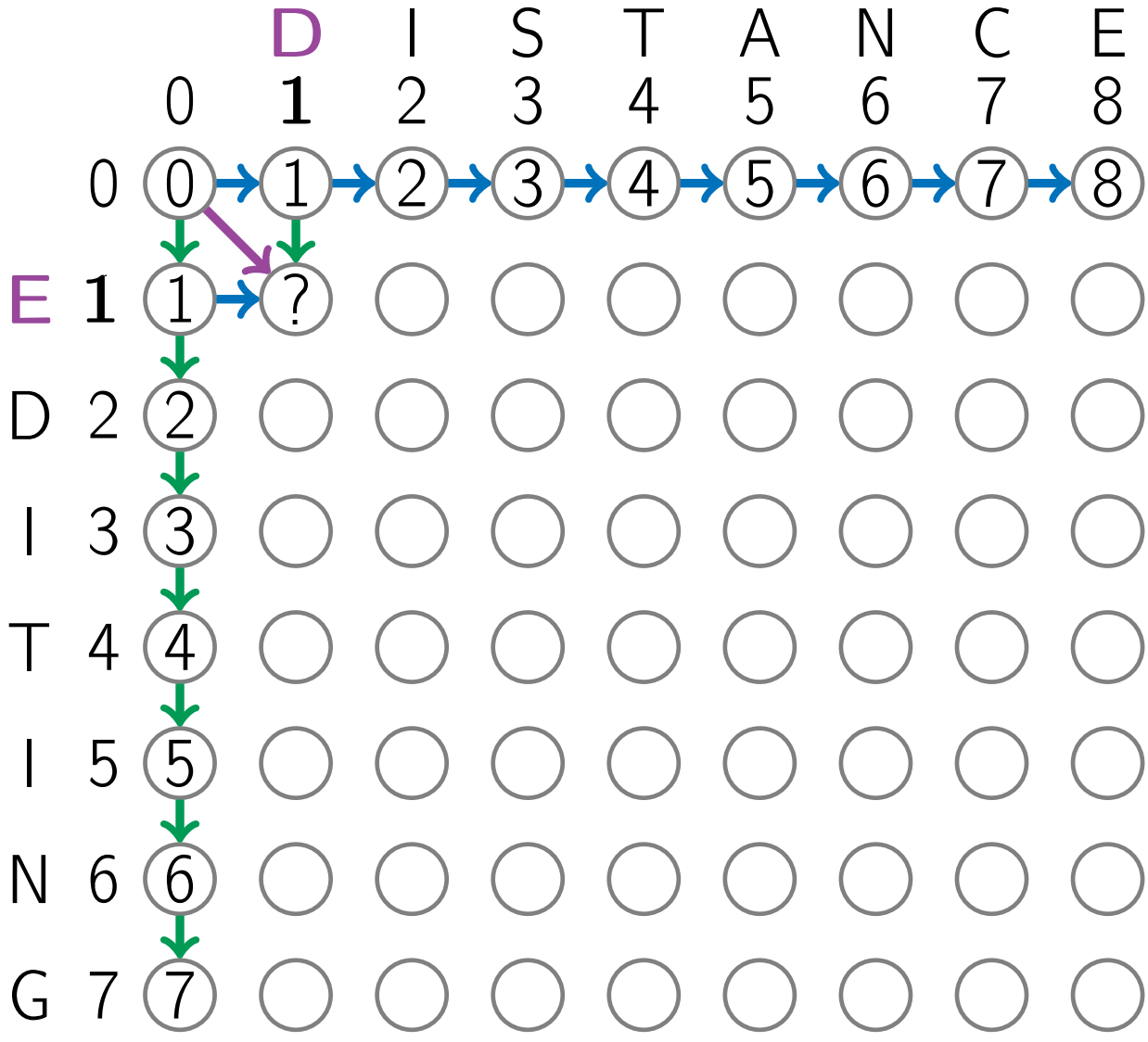
comparing $A[1 \ldots n] =$ EDITING
and $B[1 \ldots m] =$ DISTANCE


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0
E 1 1
D 2 2
I 3 3
T 4 4
I 5 5
N 6 6
G 7 7

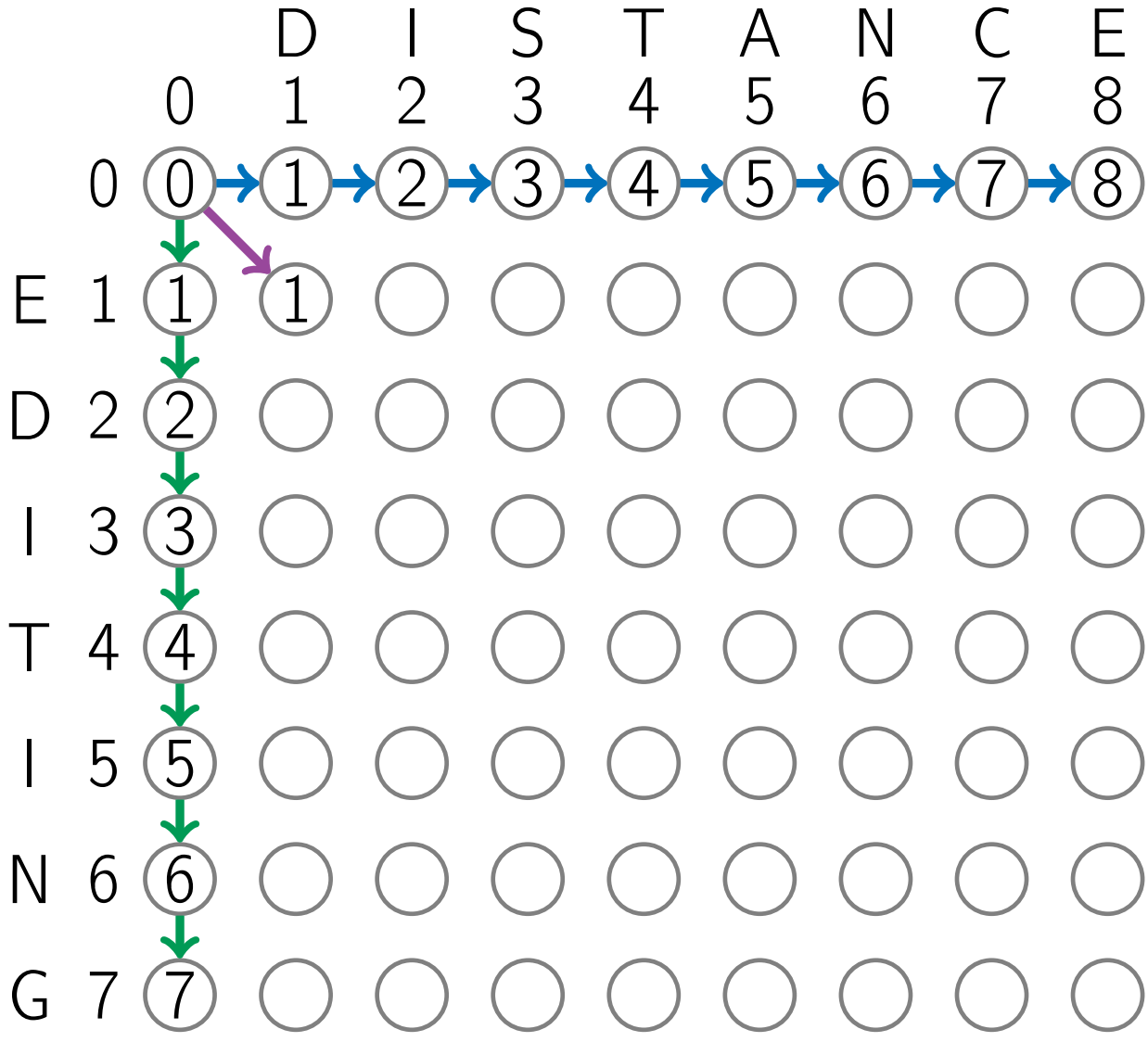

D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1
D 2 2
I 3 3
T 4 4
I 5 5
N 6 6
G 7 7

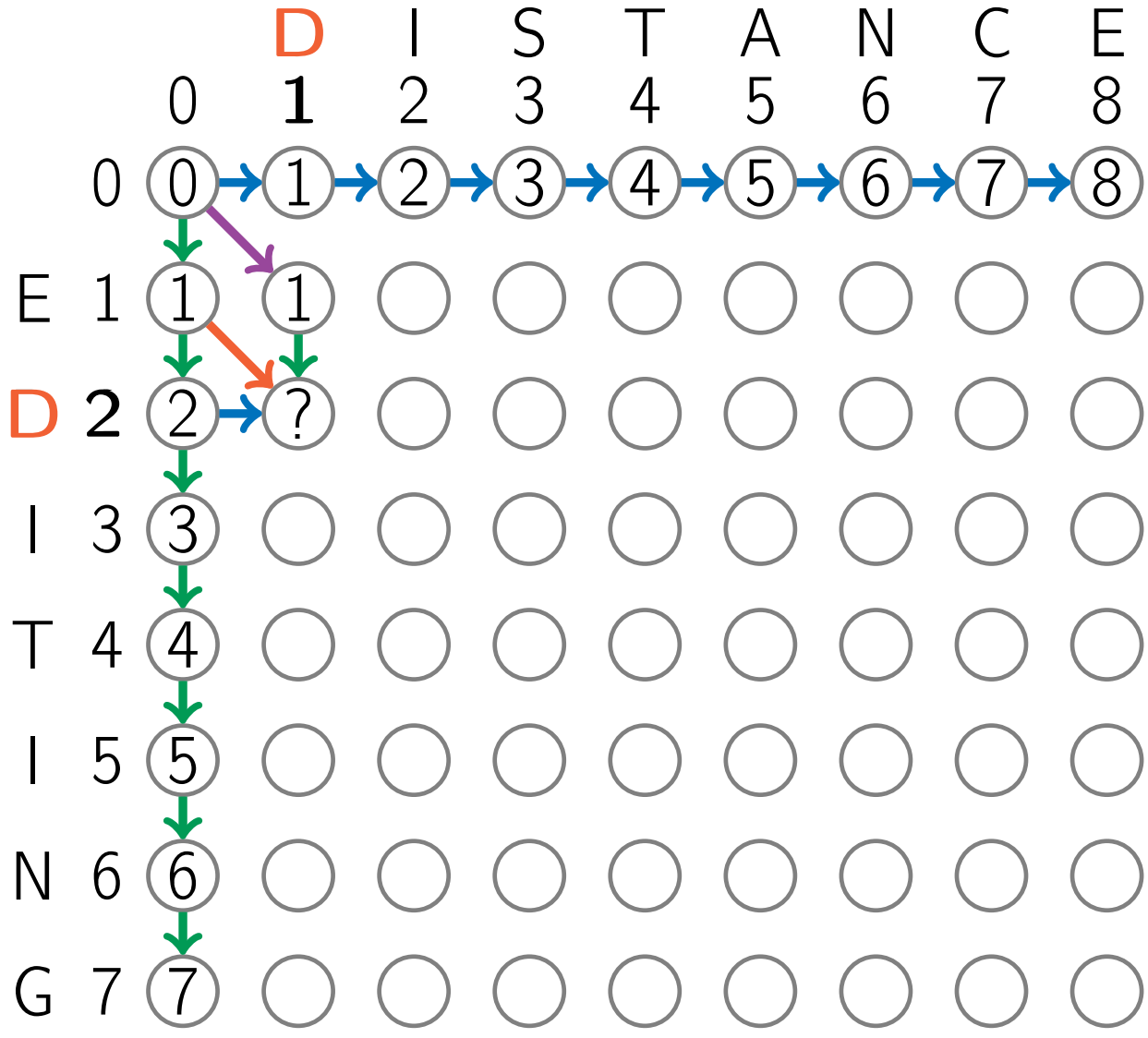


$$D(1,1) = \min\{D(1,0)+1, D(0,1)+1, D(0,0)+1\}$$

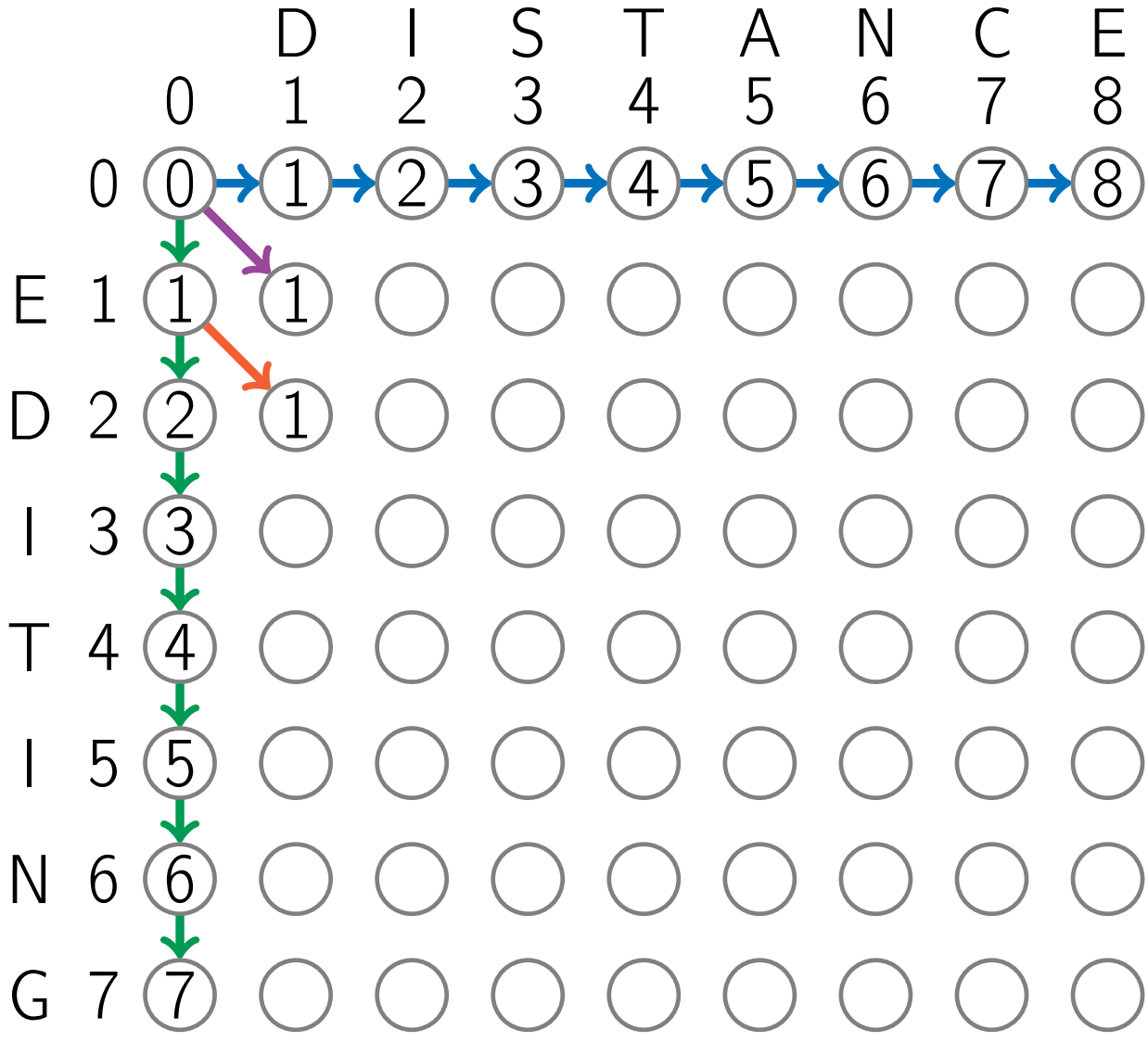


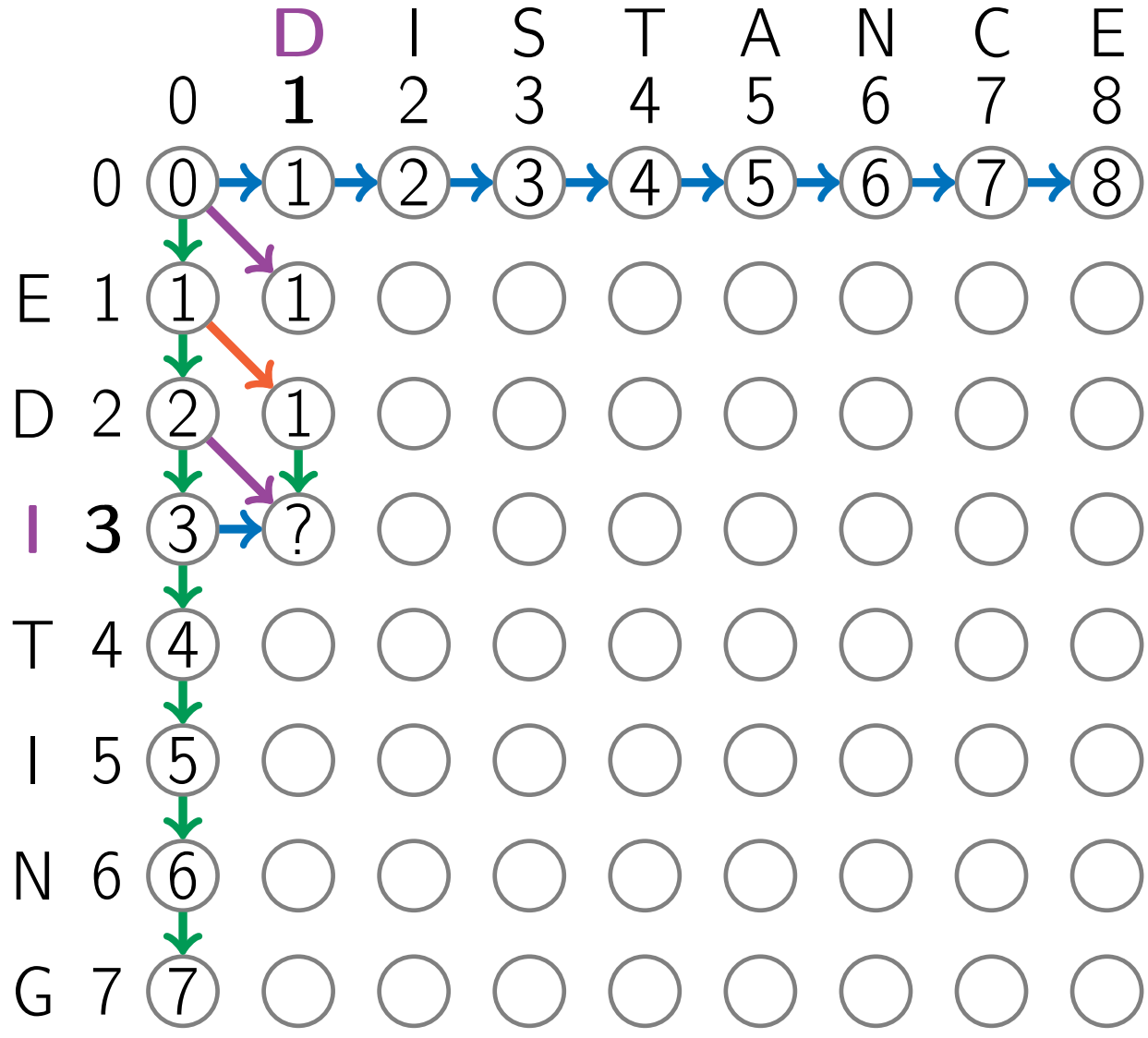
$$D(1,1) = \min\{2, 2, 1\}$$


D I S T A N C E
0 1 2 3 4 5 6 7 8
0
E 1
D 2
I 3
T 4
I 5
N 6
G 7

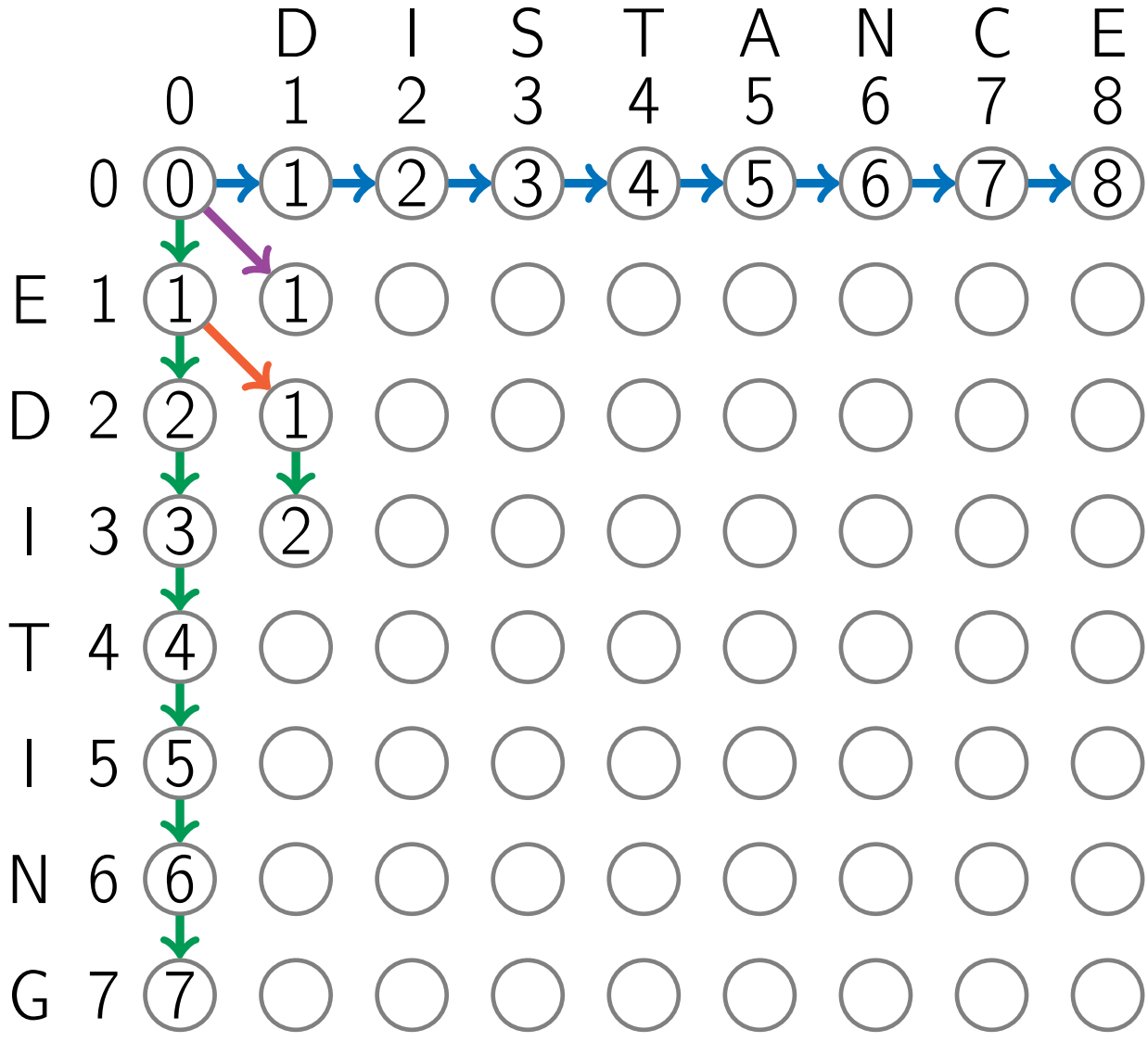


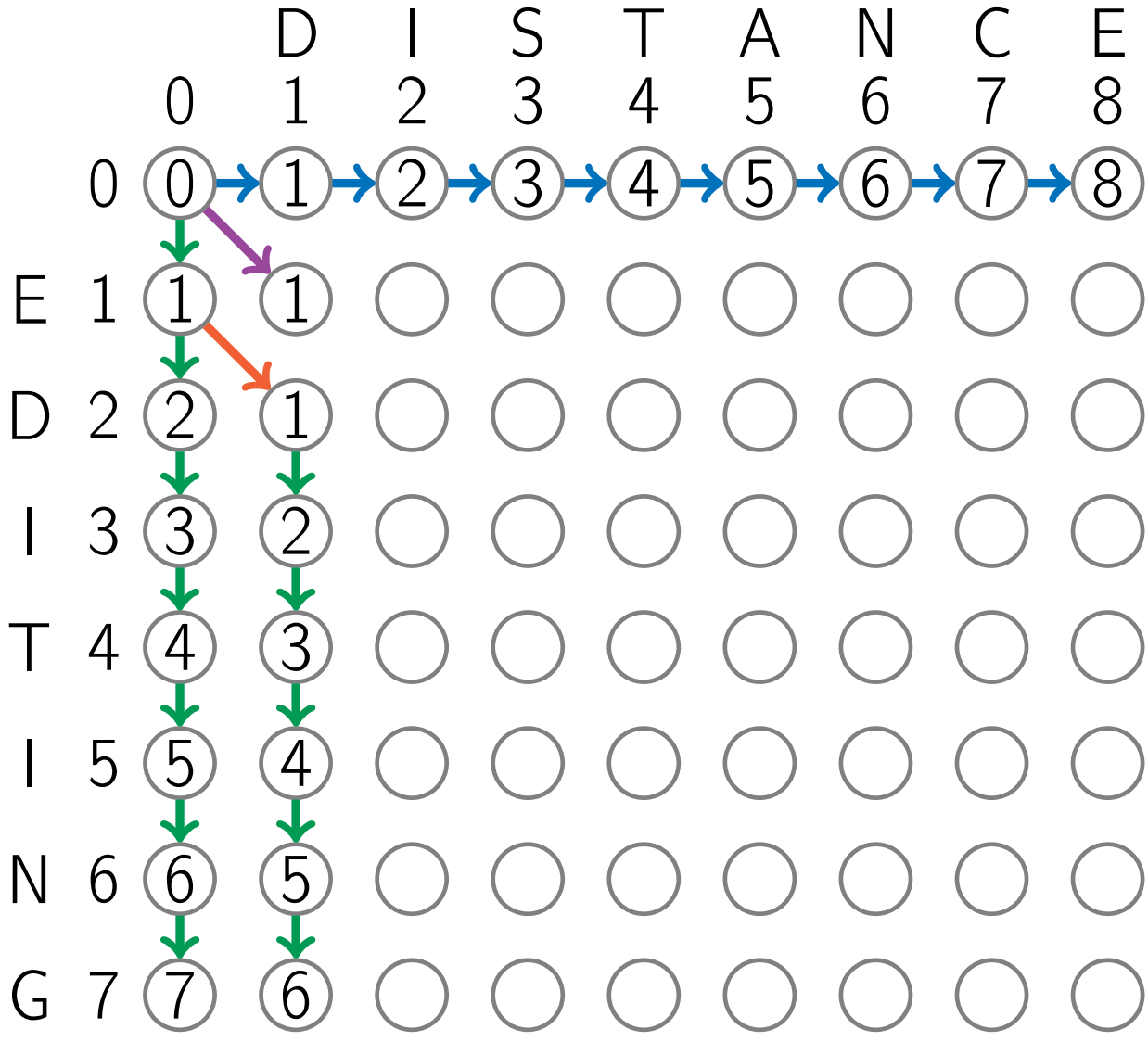
$$D(2,1) = \min\{D(2,0)+1, D(1,1)+1, D(1,0)\}$$

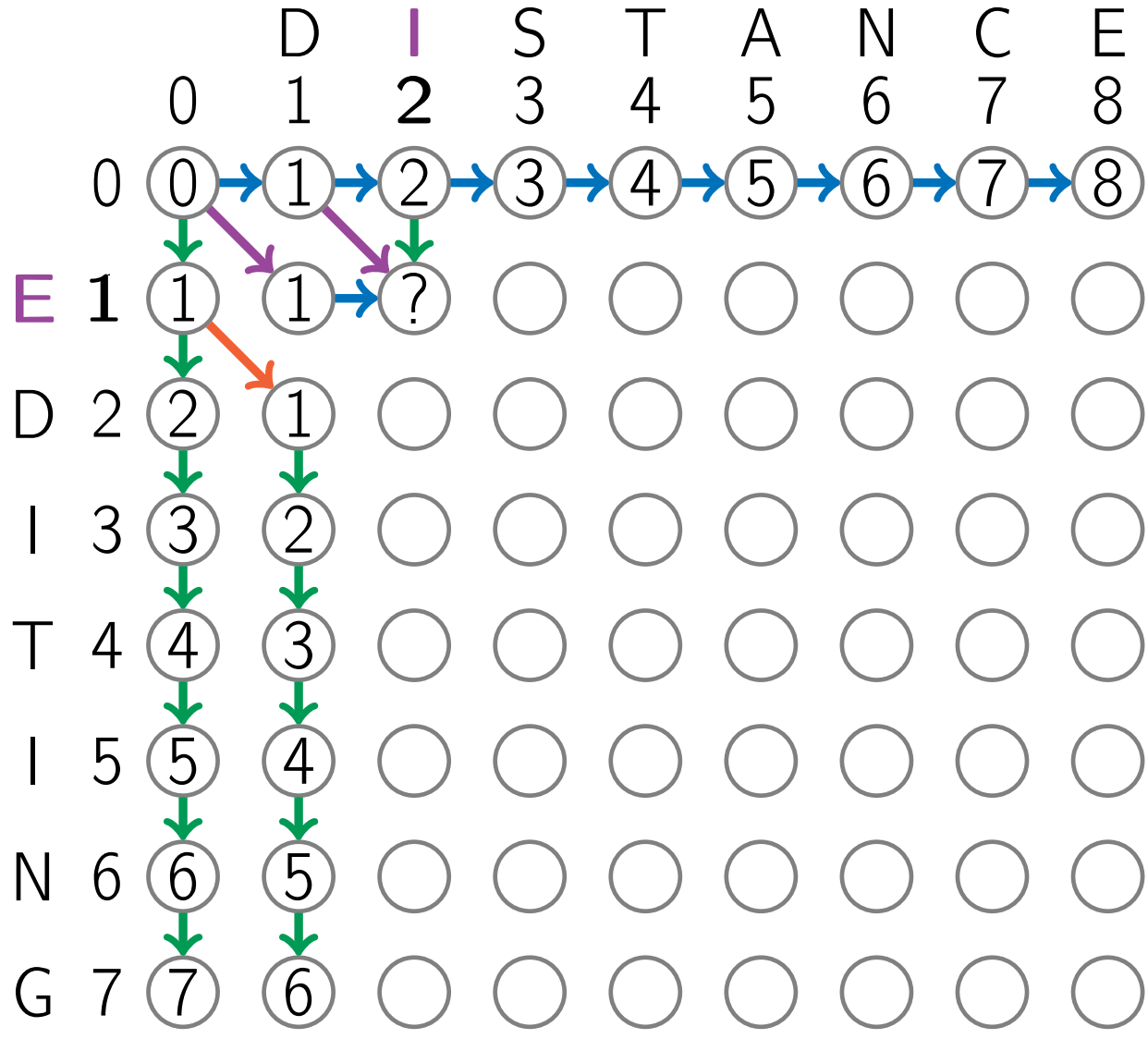

D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1
D 2 2 1
I 3 3
T 4 4
I 5 5
N 6 6
G 7 7



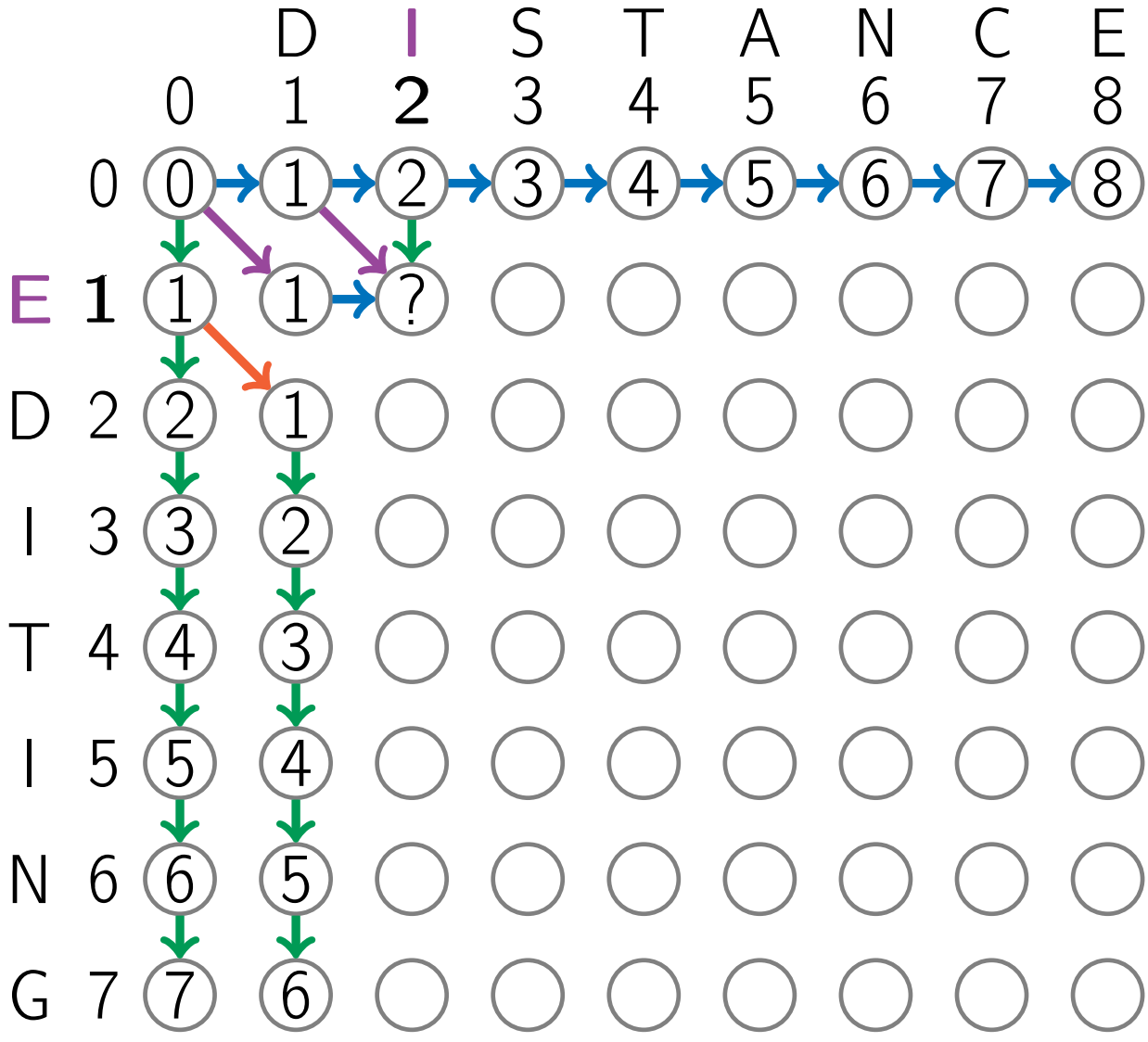
$$D(3,1) = \min\{D(3,0)+1, D(2,1)+1, D(2,0)+1\}$$


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1
D 2 2 1
I 3 3 2
T 4 4
I 5 5
N 6 6
G 7 7

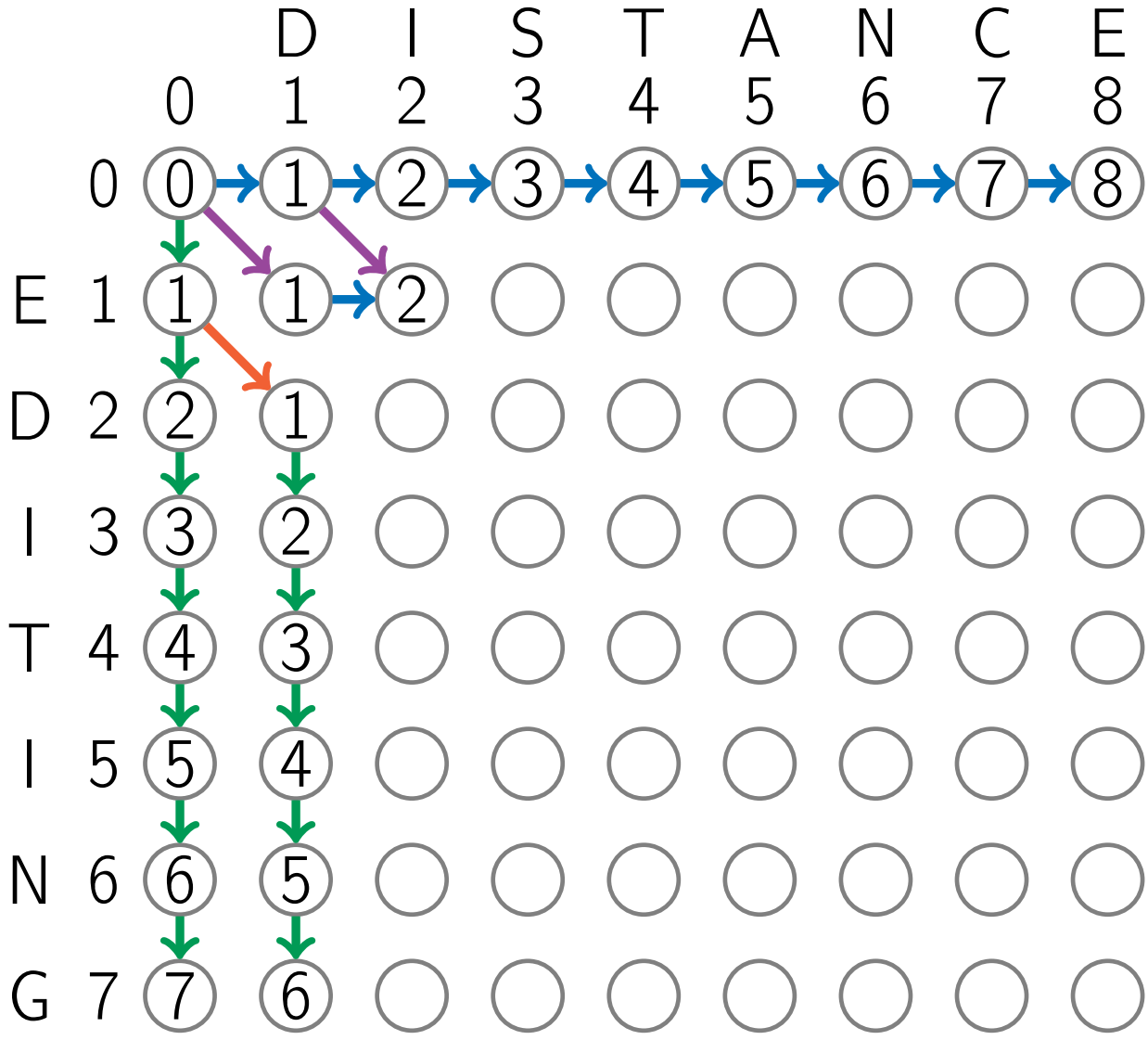

D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1
D 2 2 1
I 3 3 2
T 4 4 3
I 5 5 4
N 6 6 5
G 7 7 6

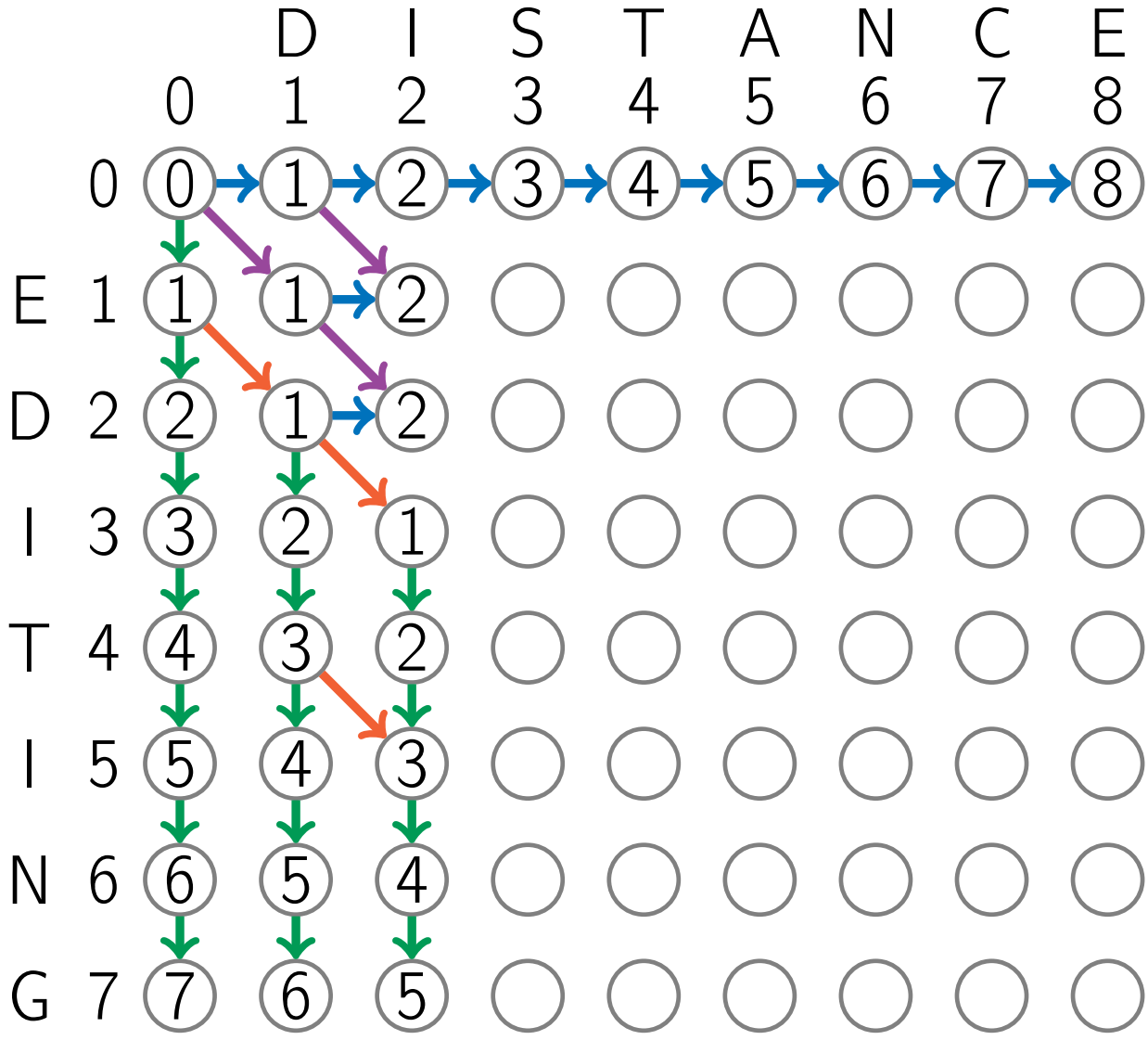


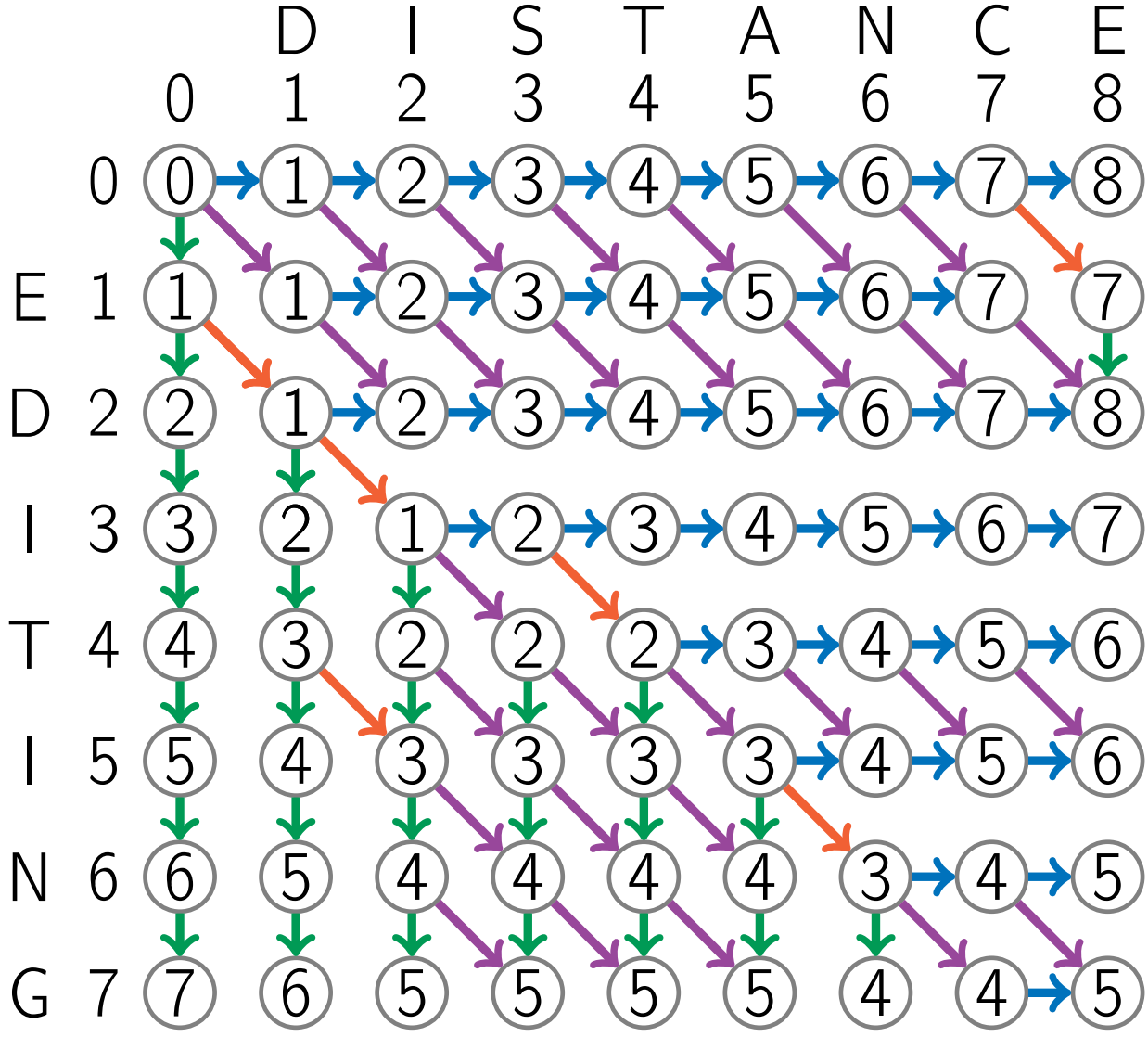
$$D(1,2) = \min\{D(1,1)+1, D(0,2)+1, D(0,1)+1\}$$


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 ?
D 2 2 1
I 3 3 2
T 4 4 3
I 5 5 4
N 6 6 5
G 7 7 6

$$D(1,1) = \min\{2, 3, 2\}$$


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 2
D 2 2 1
I 3 3 2
T 4 4 3
I 5 5 4
N 6 6 5
G 7 7 6


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 2
D 2 2 1 2
I 3 3 2 1
T 4 4 3 2
I 5 5 4 3
N 6 6 5 4
G 7 7 6 5


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 2 3 4 5 6 7 7
D 2 2 1 2 3 4 5 6 7 8
I 3 3 2 1 2 3 4 5 6 7
T 4 4 3 2 2 2 3 4 5 6
I 5 5 4 3 3 3 3 4 5 6
N 6 6 5 4 4 4 4 3 4 5
G 7 7 6 5 5 5 5 4 4 5

EditDistance($A[1 \ldots n], B[1 \ldots m]$)

$D(i, 0) \leftarrow i$ and $D(0, j) \leftarrow j$ for all i, j

```
for j from 1 to m:
  for i from 1 to n:
    insertion ← D(i, j − 1) + 1
    deletion ← D(i − 1, j) + 1
    match ← D(i − 1, j − 1)
    mismatch ← D(i − 1, j − 1) + 1
    if A[i] = B[j]:
      D(i, j) ← min(insertion, deletion, match)
    else:
      D(i, j) ← min(insertion, deletion, mismatch)
return D(n, m)
```

Outline

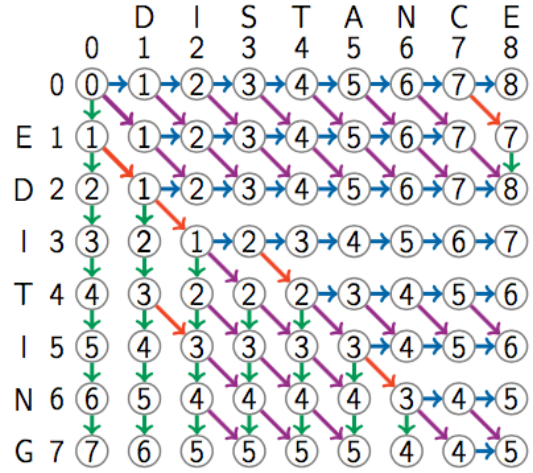
1. The Alignment Game
2. Computing Edit Distance
3. Reconstructing an Optimal Alignment

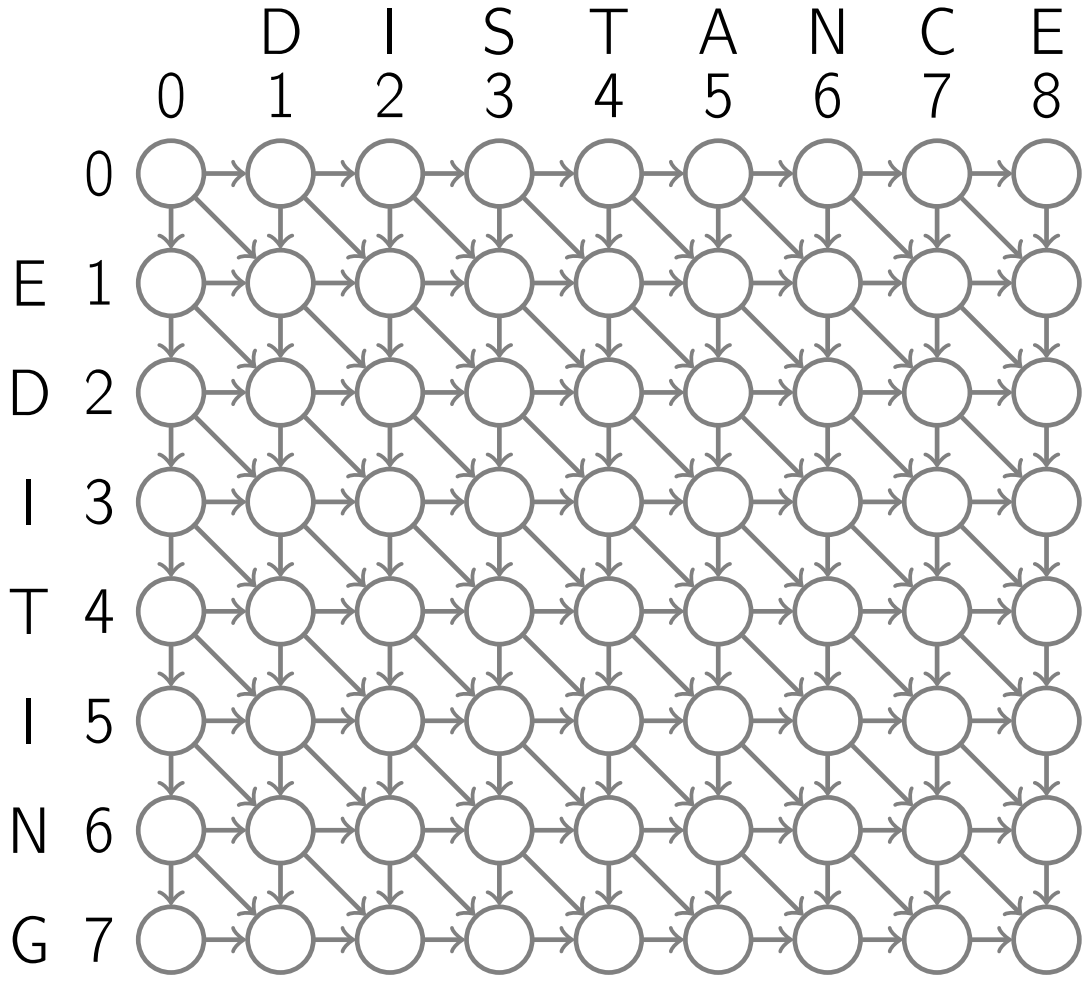
Optimal Alignment

- We have computed the edit distance, but how can we find an optimal alignment?

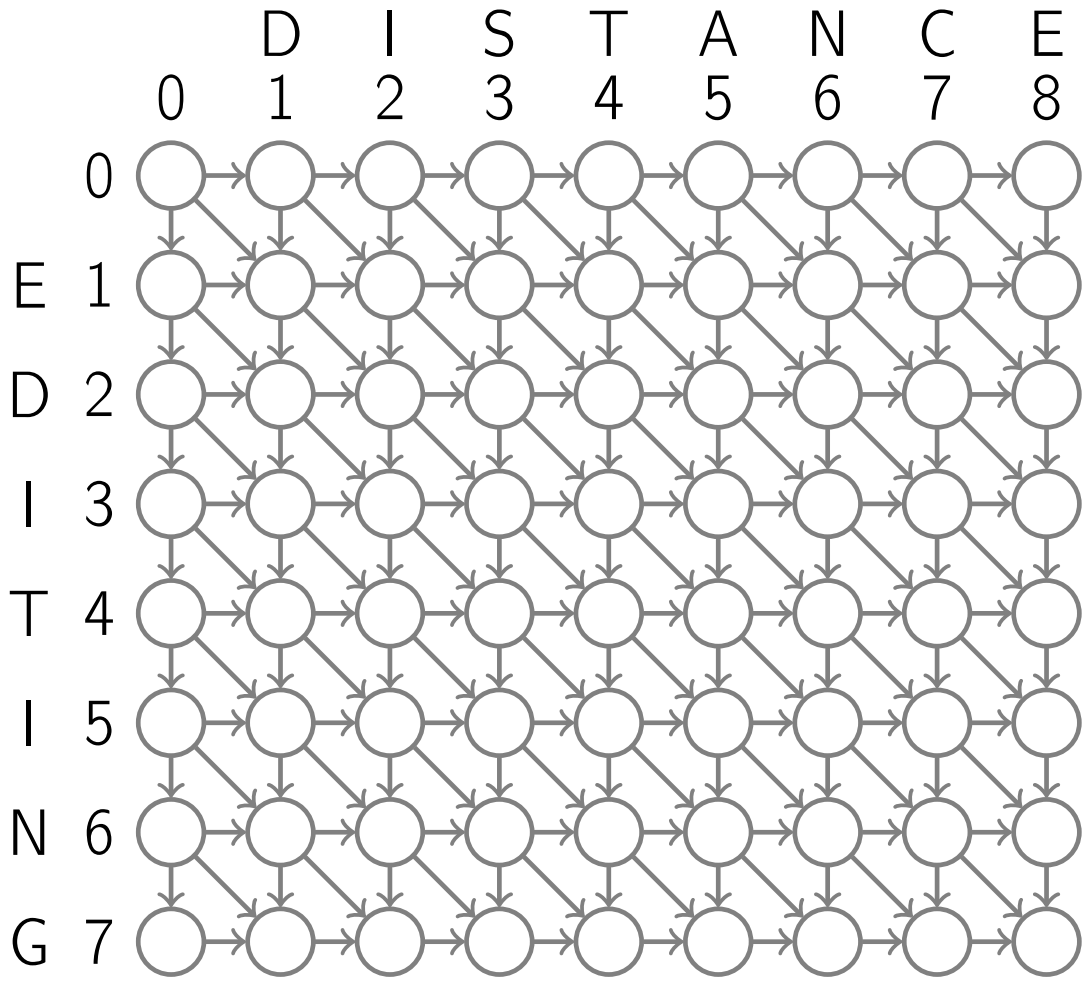
Optimal Alignment

- We have computed the edit distance, but how can we find an optimal alignment?
- The backtracking pointers that we stored will help us to reconstruct an optimal alignment.

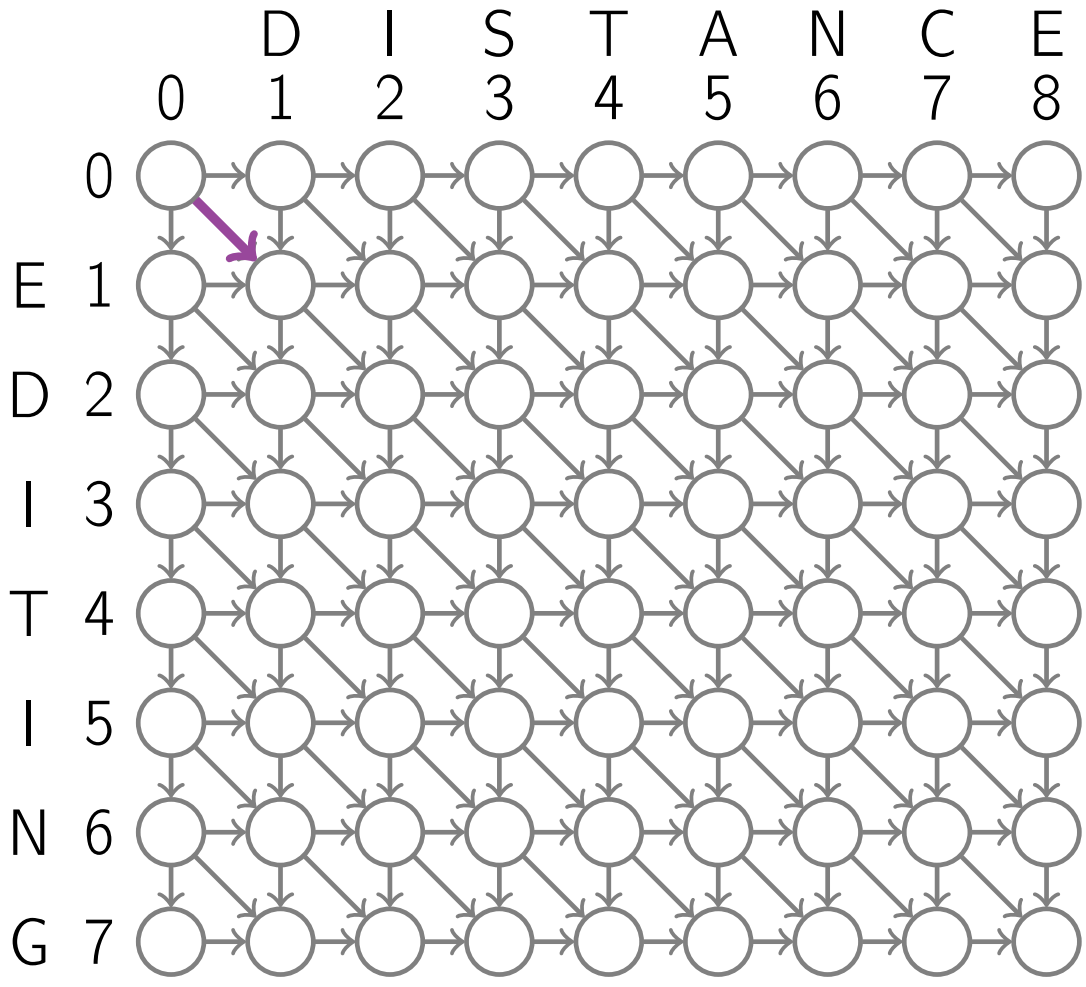



D I S T A N C E
0 1 2 3 4 5 6 7 8
0
E 1
D 2
I 3
T 4
I 5
N 6
G 7

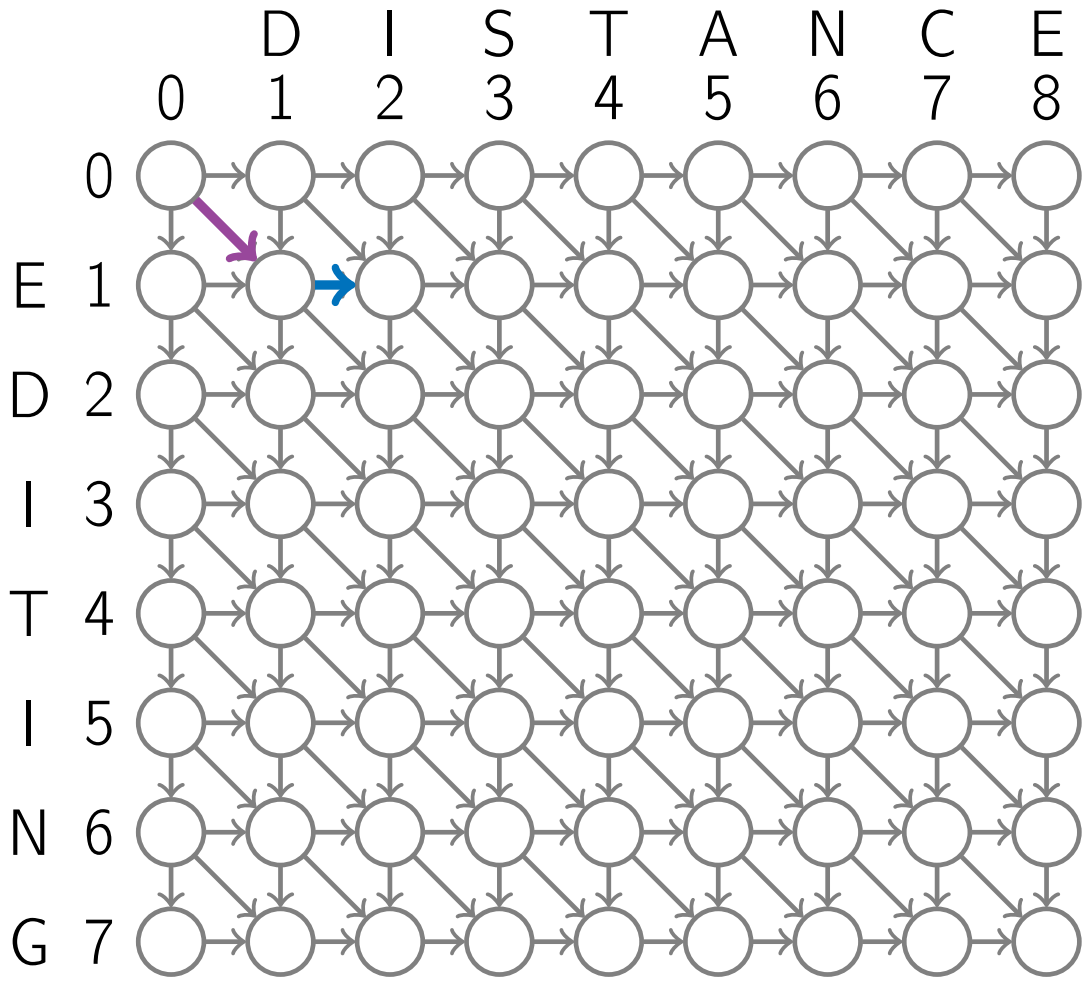
any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$

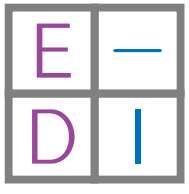


any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$

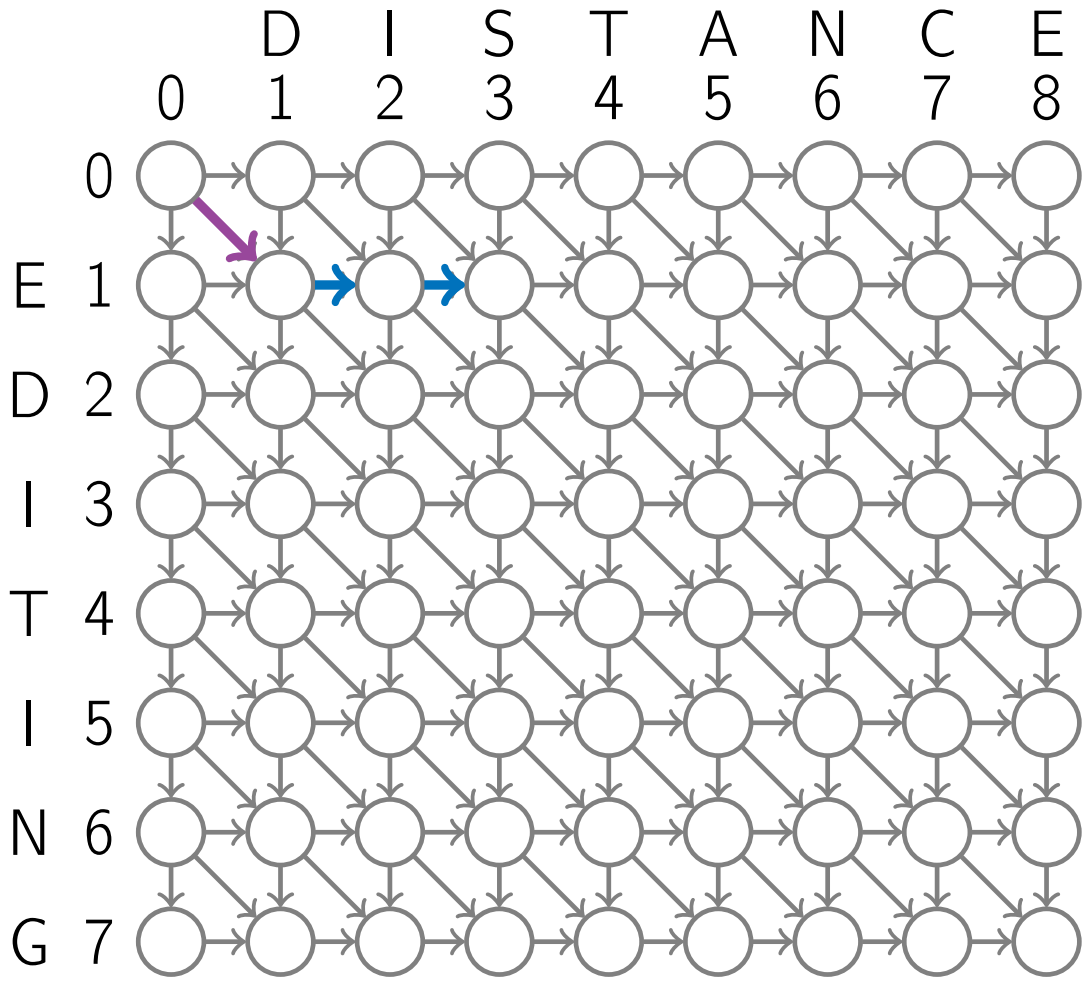


any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$



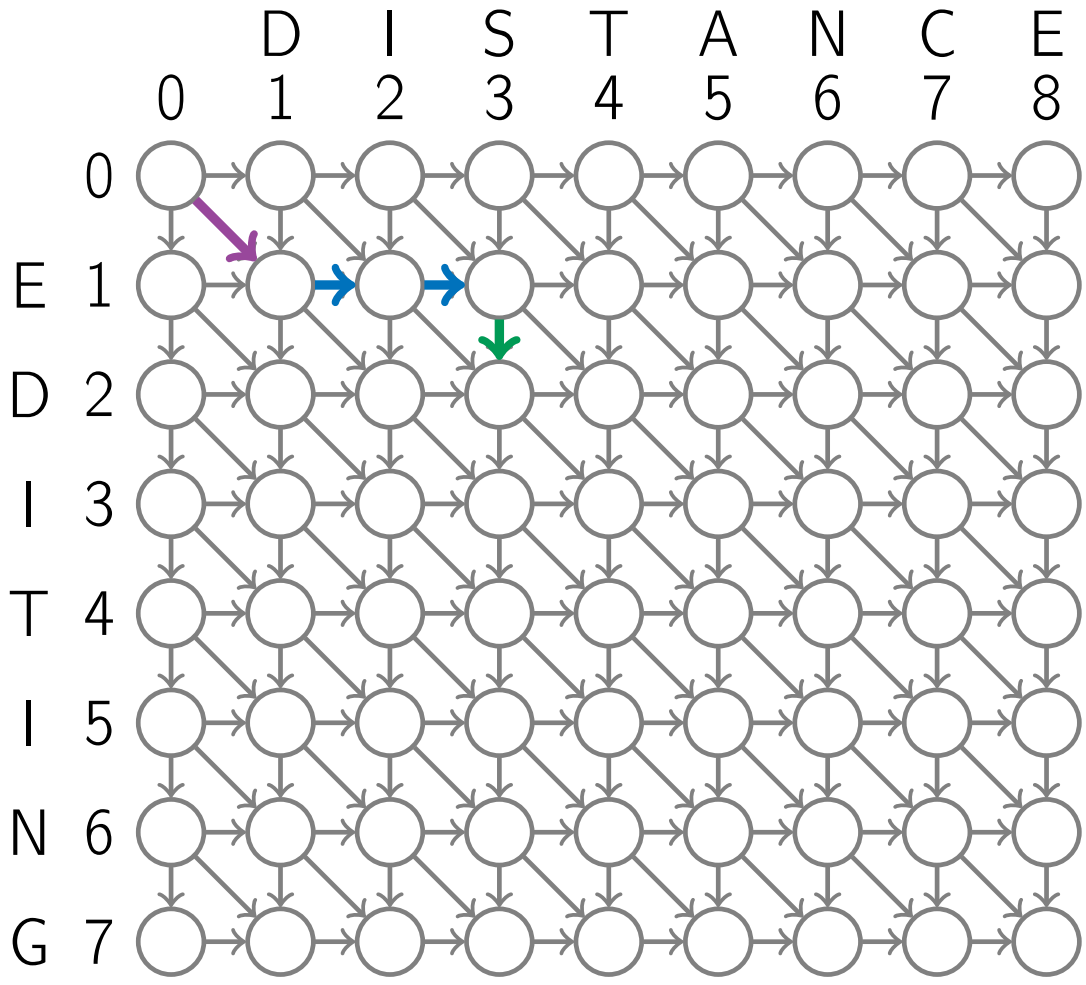


any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$



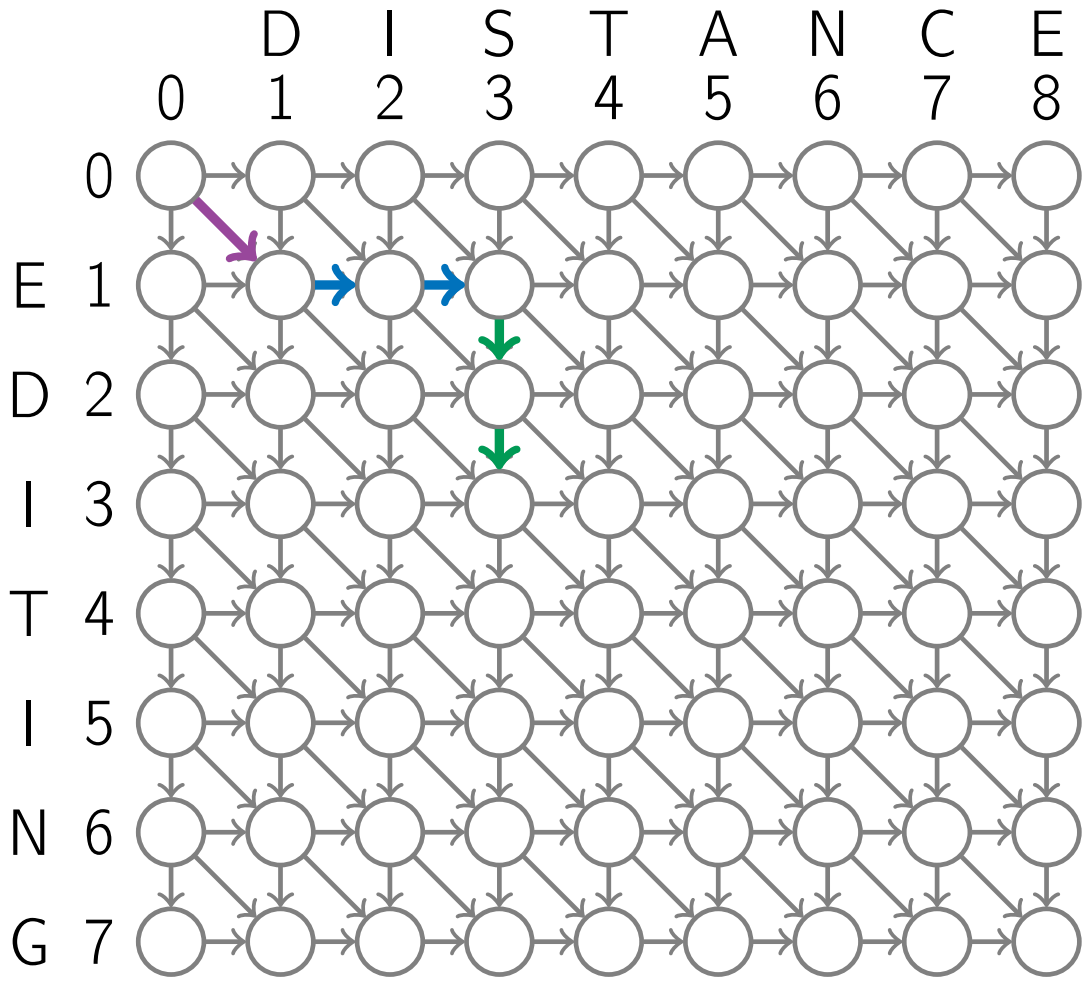
E	—	—
D	I	S

any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$



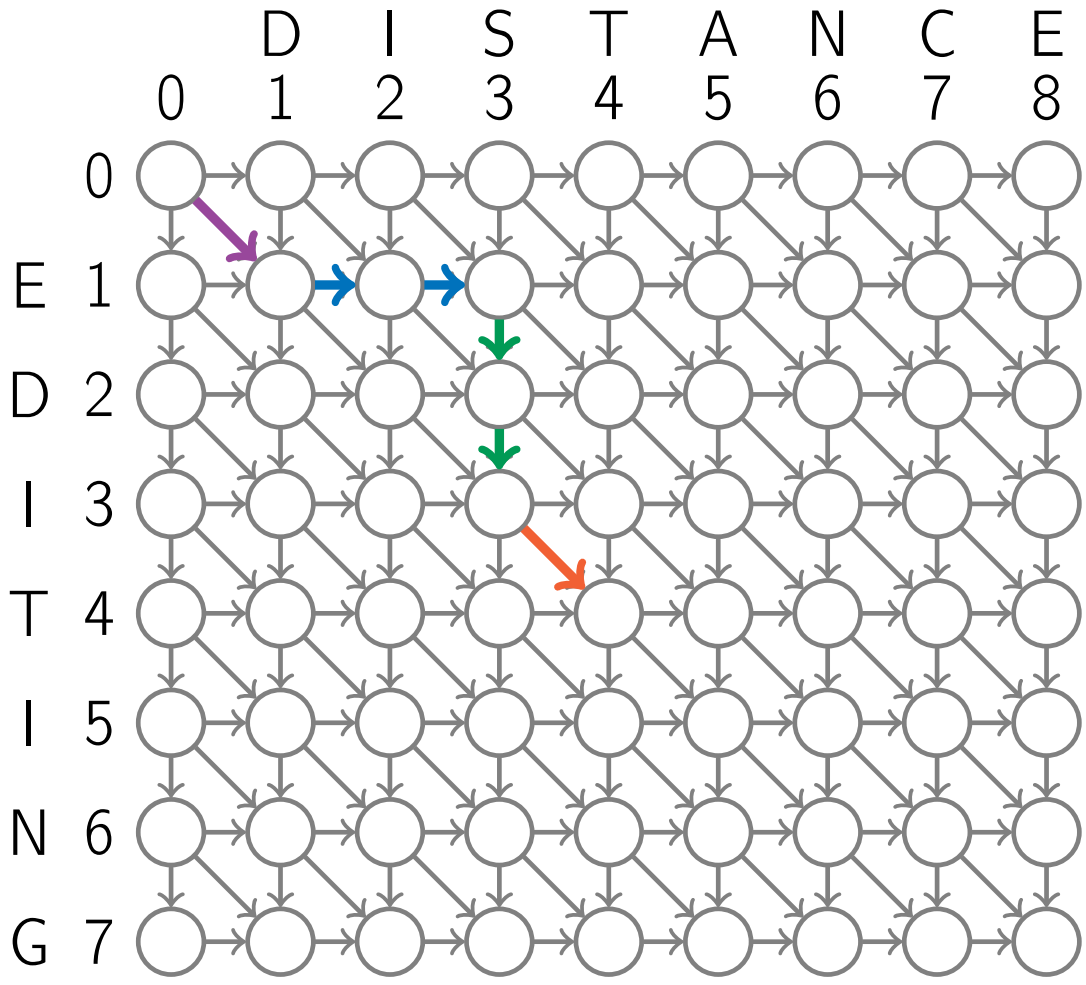
E	—	—	D
D	I	S	—

any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$



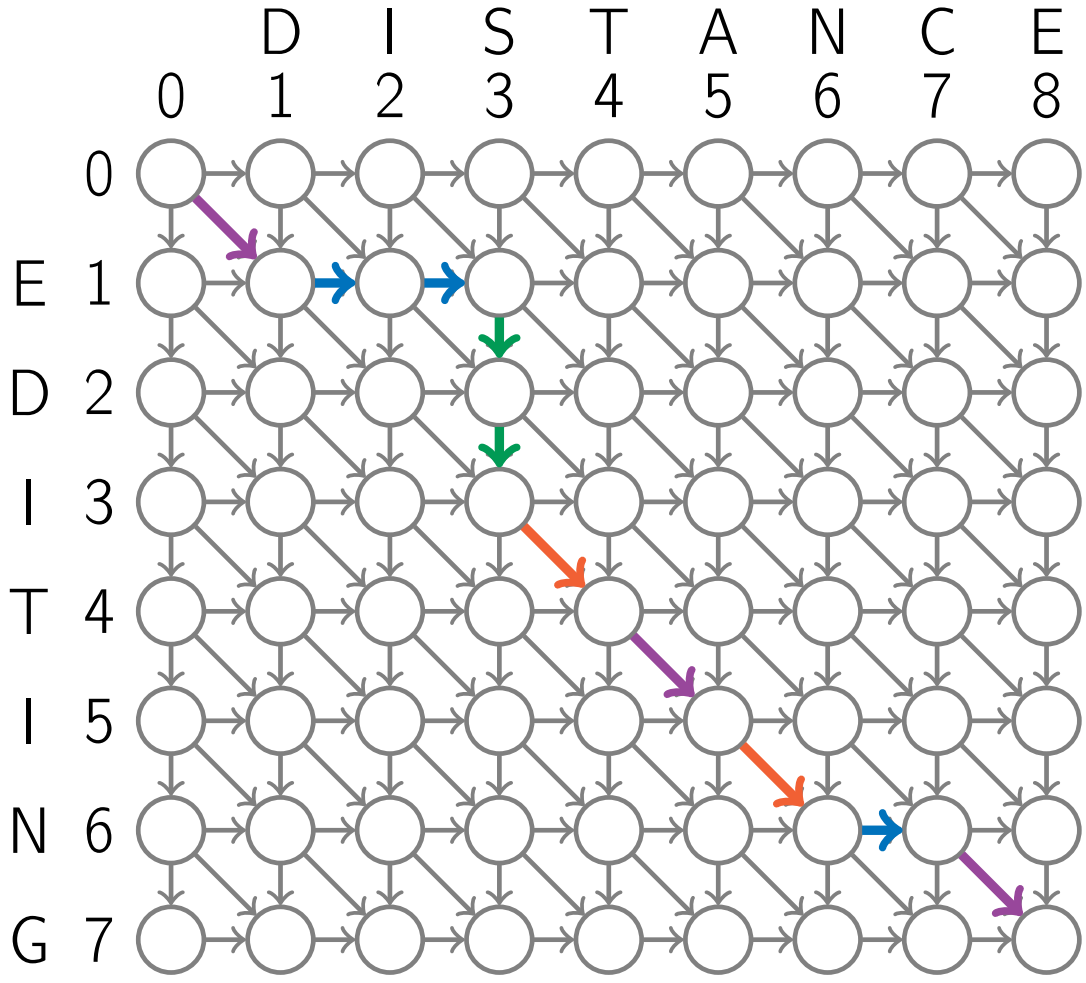
E	—	—	D	I
D	I	S	—	—

any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$



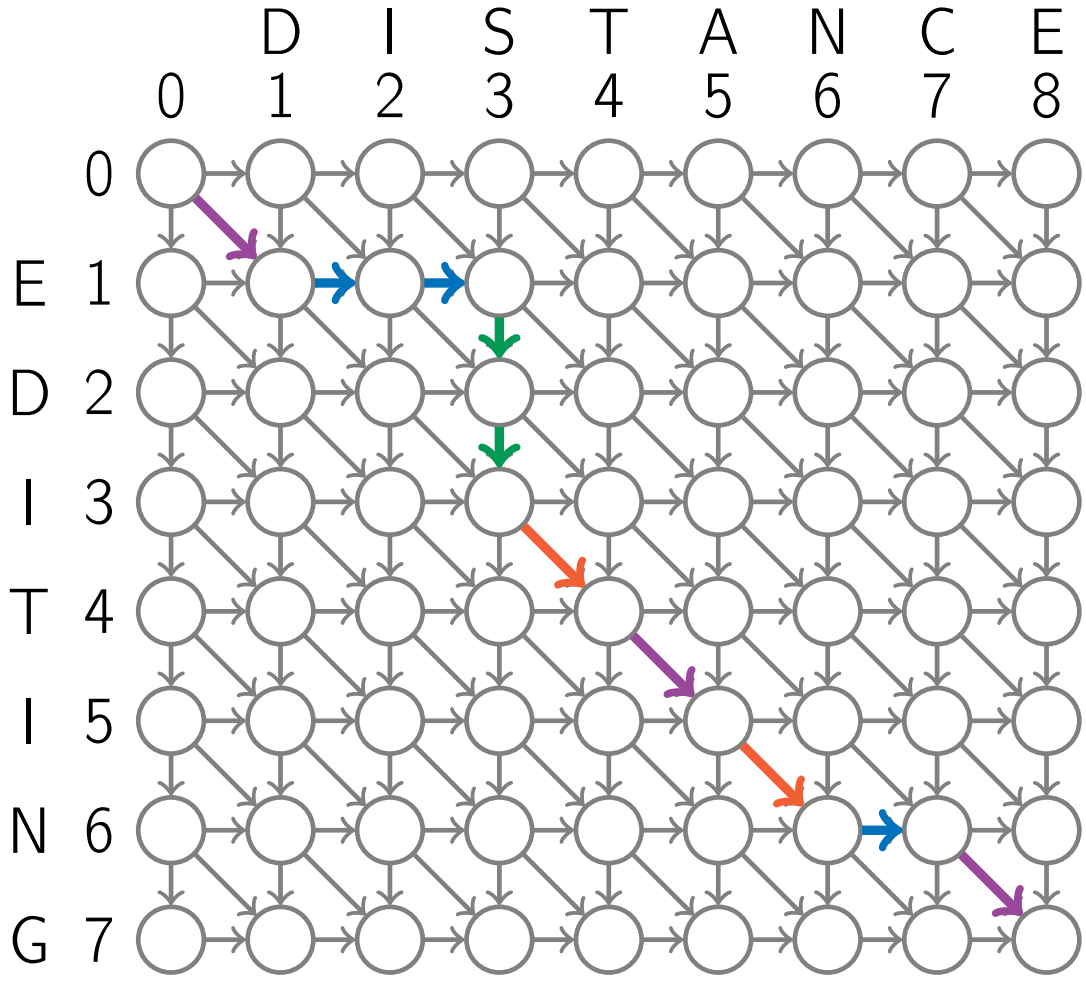
E	—	—	D	I	T
D	I	S	—	—	T

any path from $(0,0)$ to (i,j) spells an alignment of prefixes $A[1 \ldots i]$ and $B[1 \ldots j]$



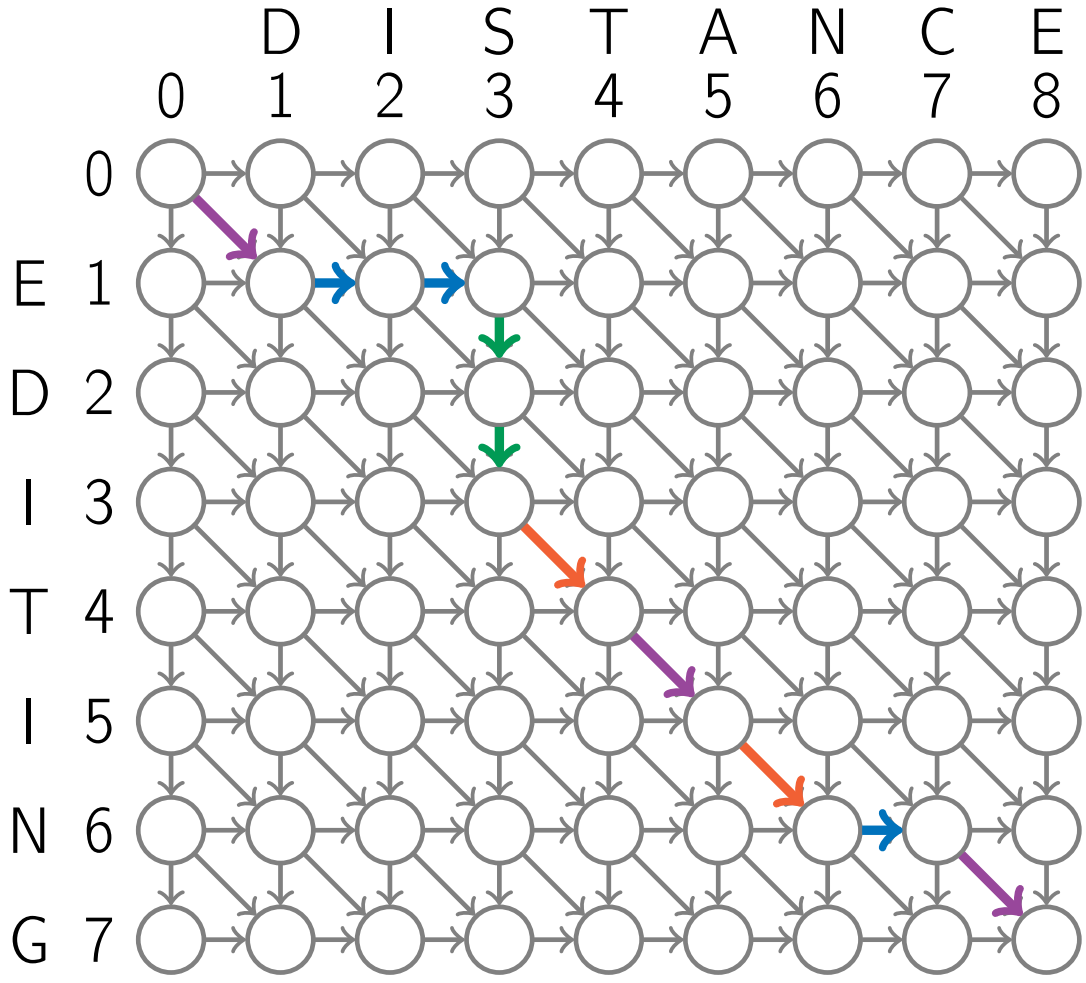
E	–	–	D	I	T	I	N	–	G
D	I	S	–	–	T	A	N	C	E

the constructed path corresponds to distance 8 and is not optimal (edit distance is 5)

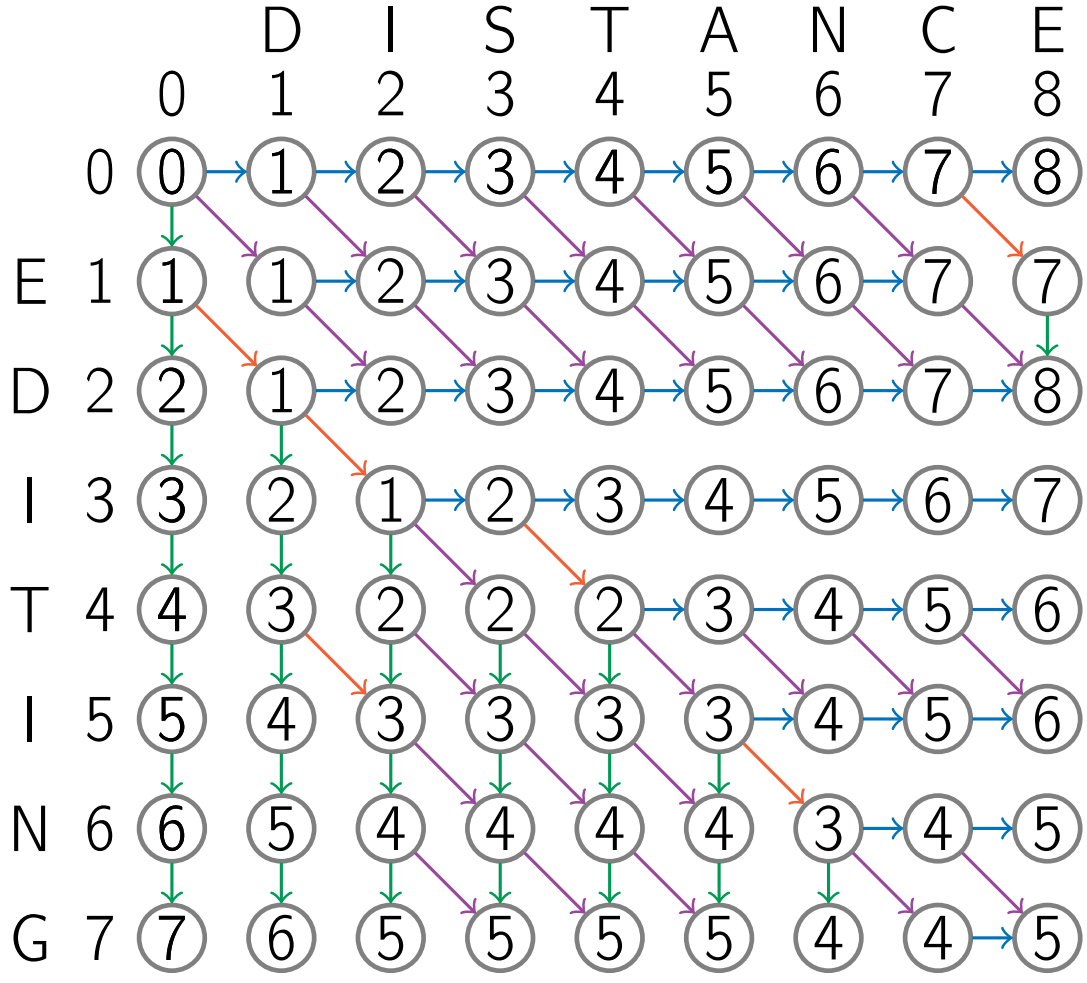


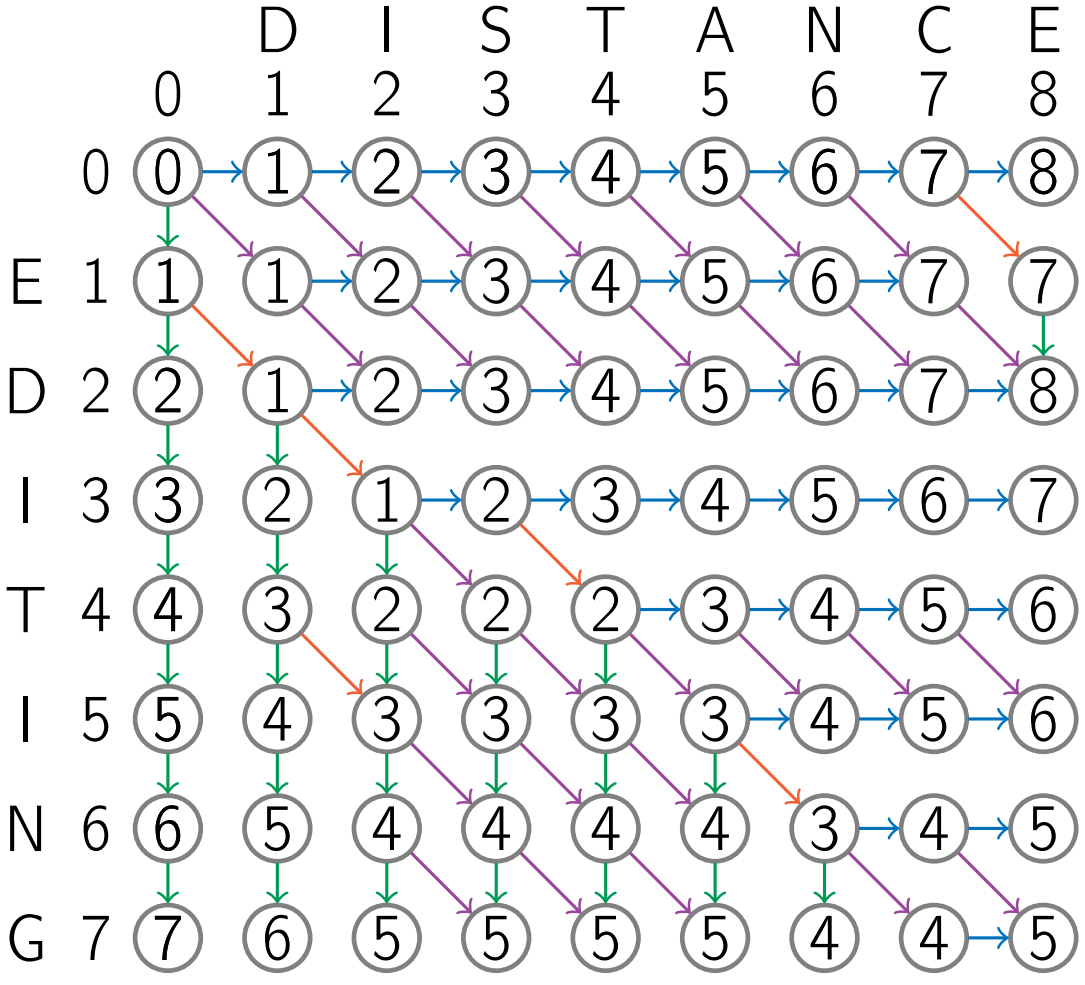
E	–	–	D	I	T	I	N	–	G
D	I	S	–	–	T	A	N	C	E

to construct an optimal alignment we will use the backtracking pointers

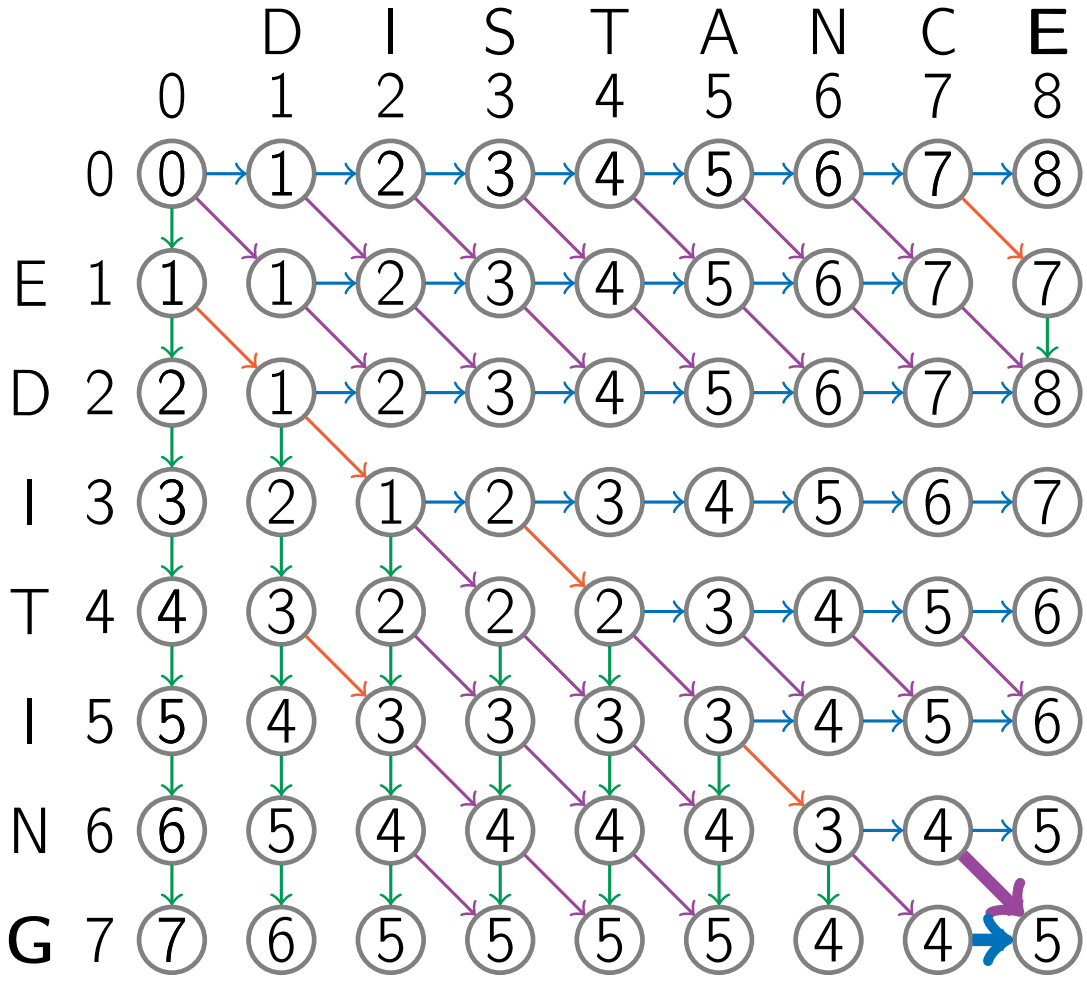


E	–	–	D	I	T	I	N	–	G
D	I	S	–	–	T	A	N	C	E

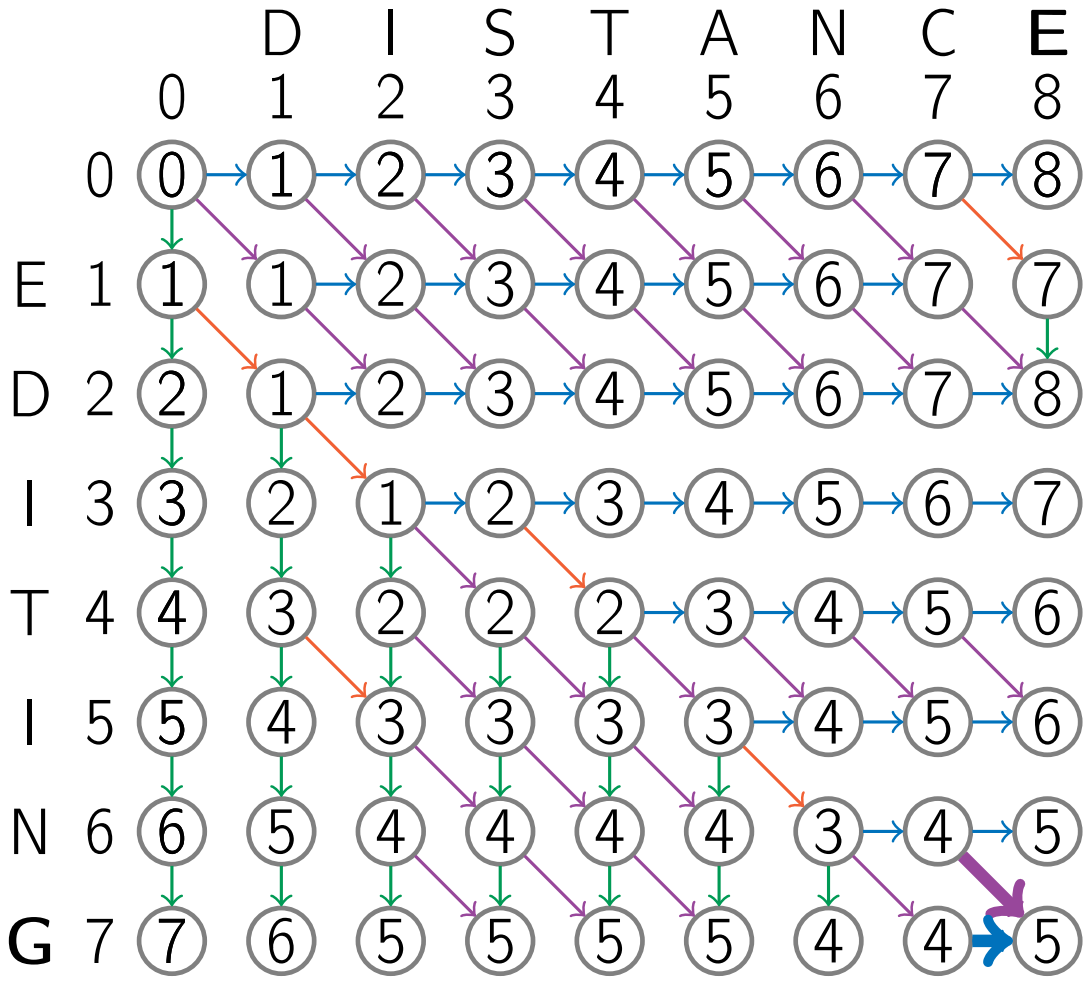

D I S T A N C E
0 1 2 3 4 5 6 7 8
0
E 1
D 2
I 3
T 4
I 5
N 6
G 7



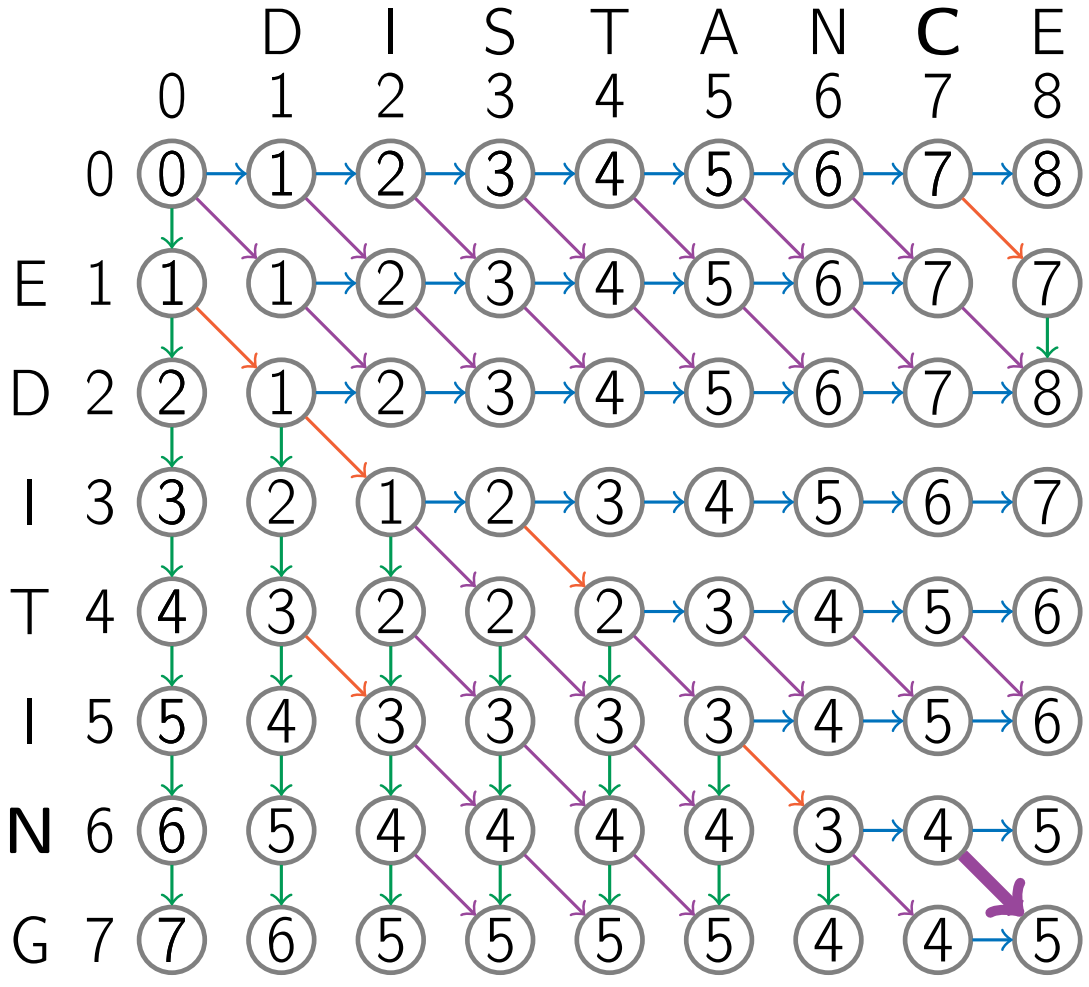
the edit distance is 5



we arrived to the bottom right cell by moving along the backtracking pointers shown below

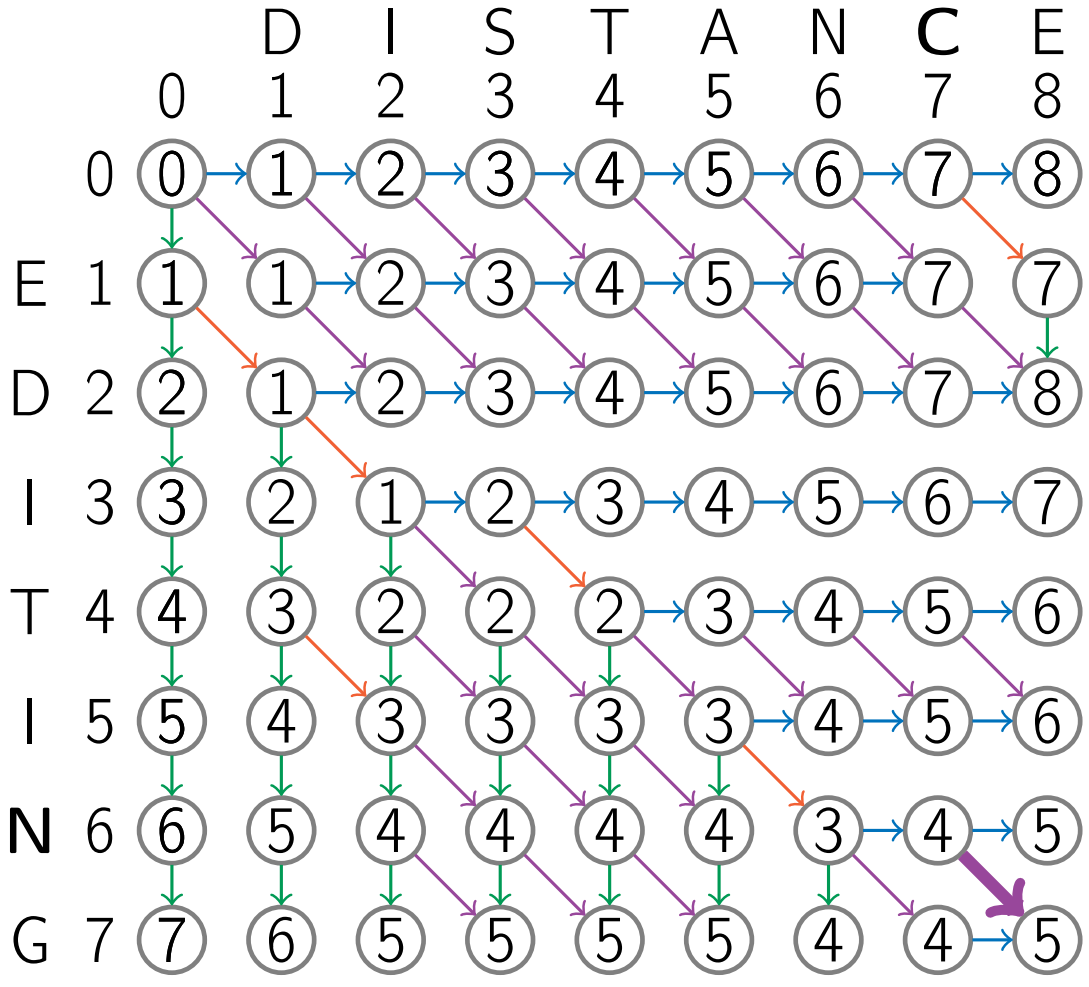


there exists an optimal alignment whose last column is a mismatch and an optimal alignment whose last column is an insertion



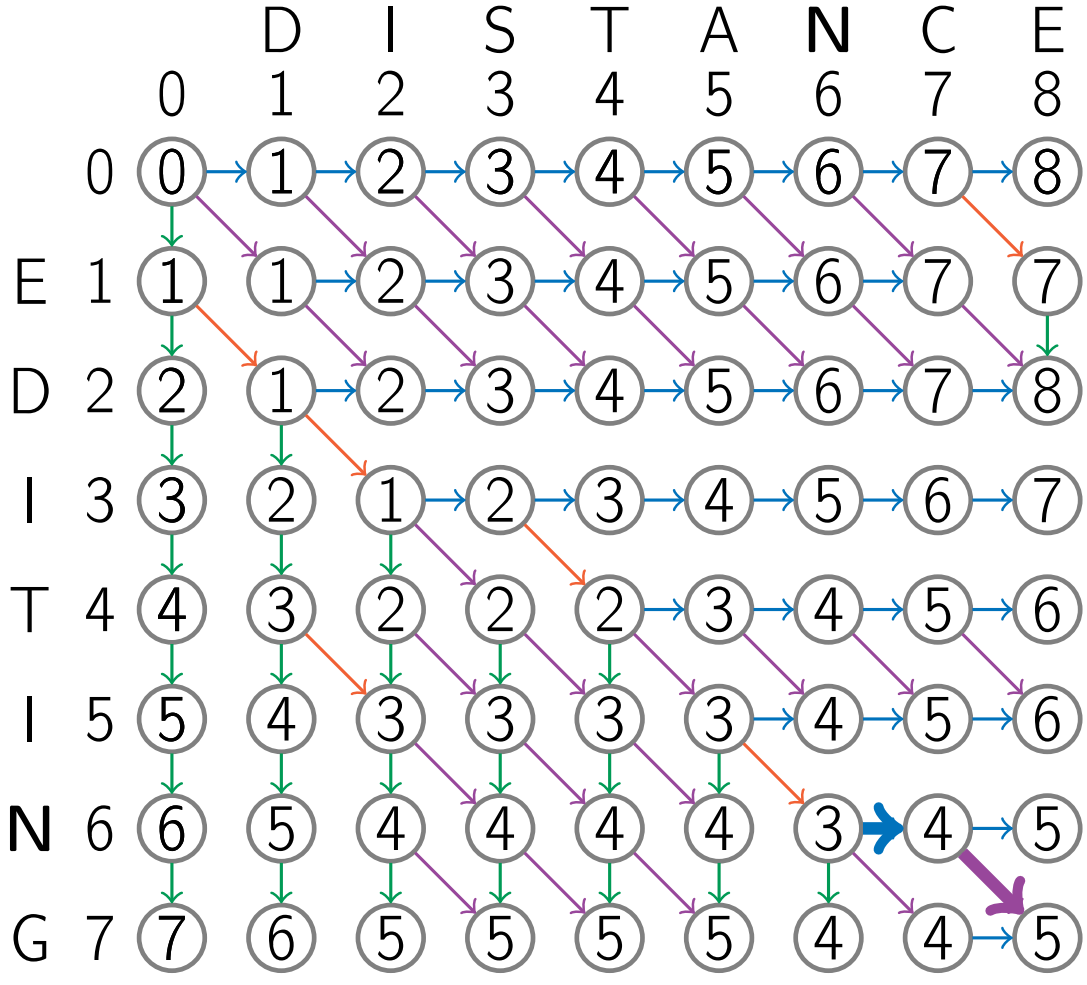
let's consider a mismatch

G
E

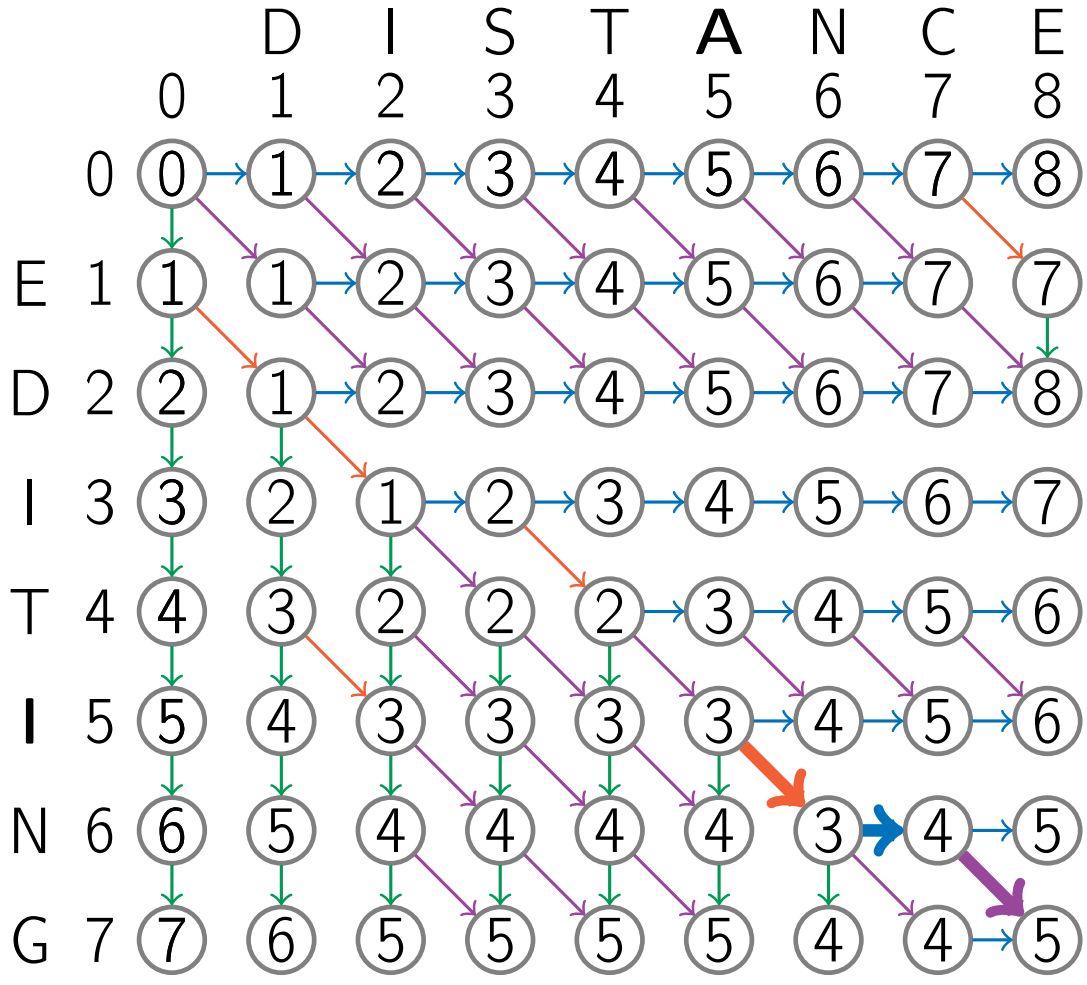


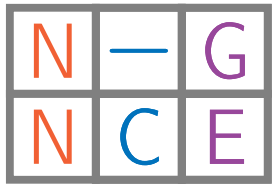
we continue in a similar fashion

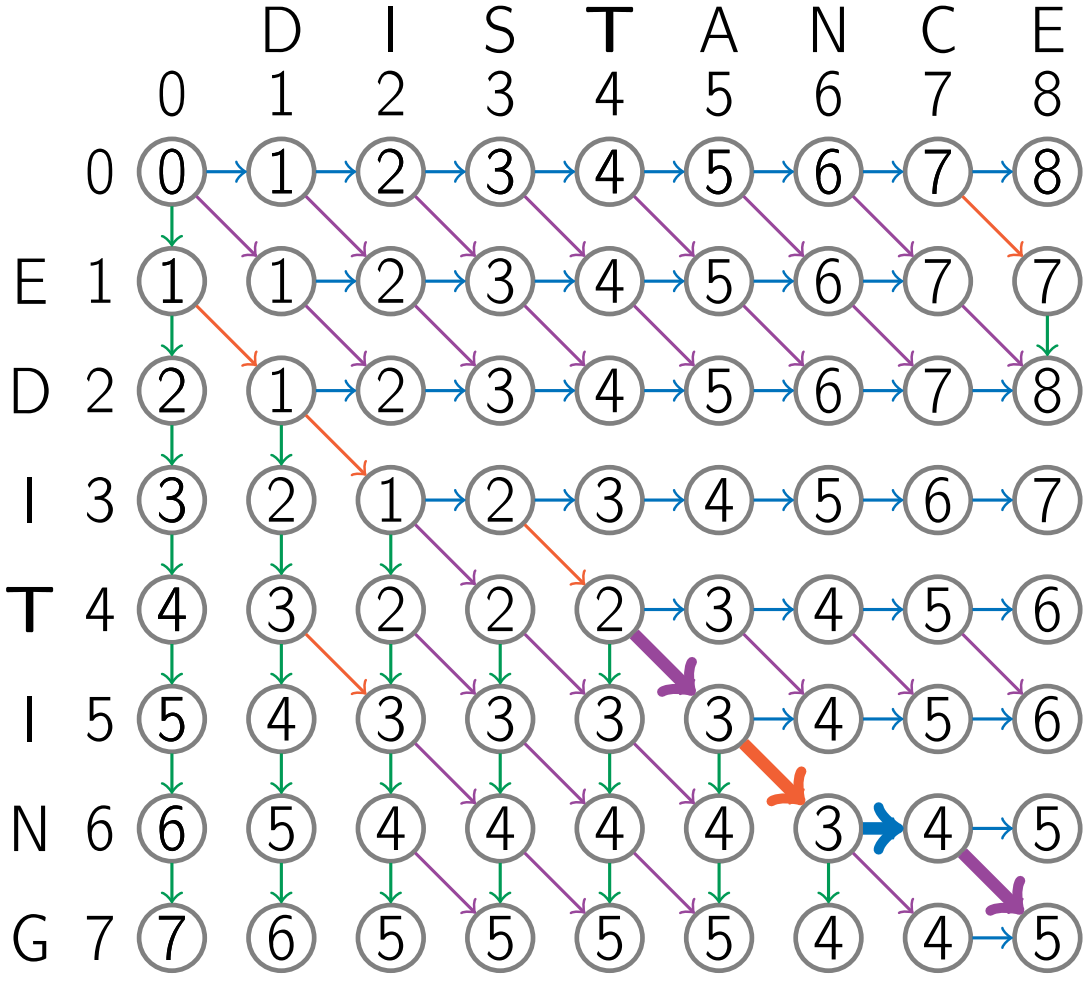
G
E



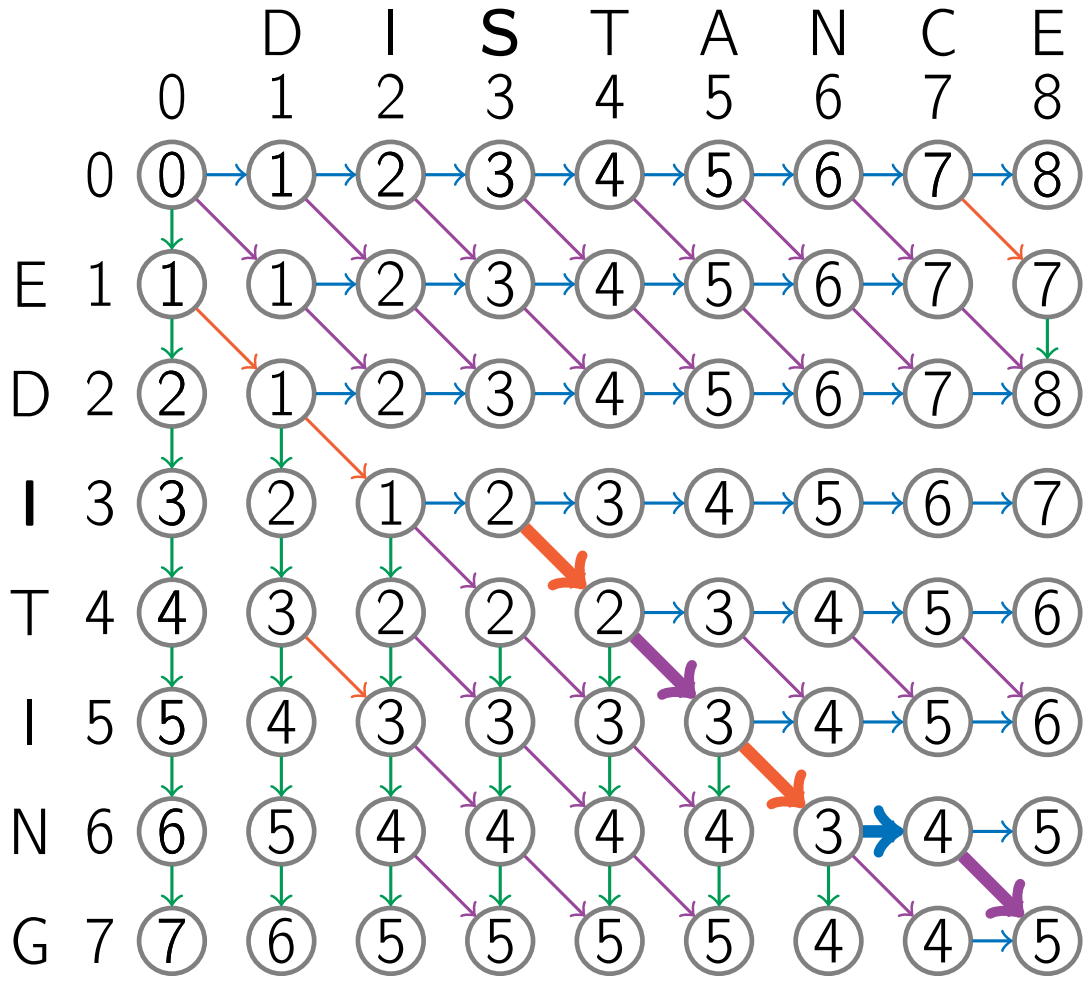
—	G
C	E


D I S T A N C E
0 1 2 3 4 5 6 7 8
0
E 1
D 2
I 3
T 4
I 5
N 6
G 7

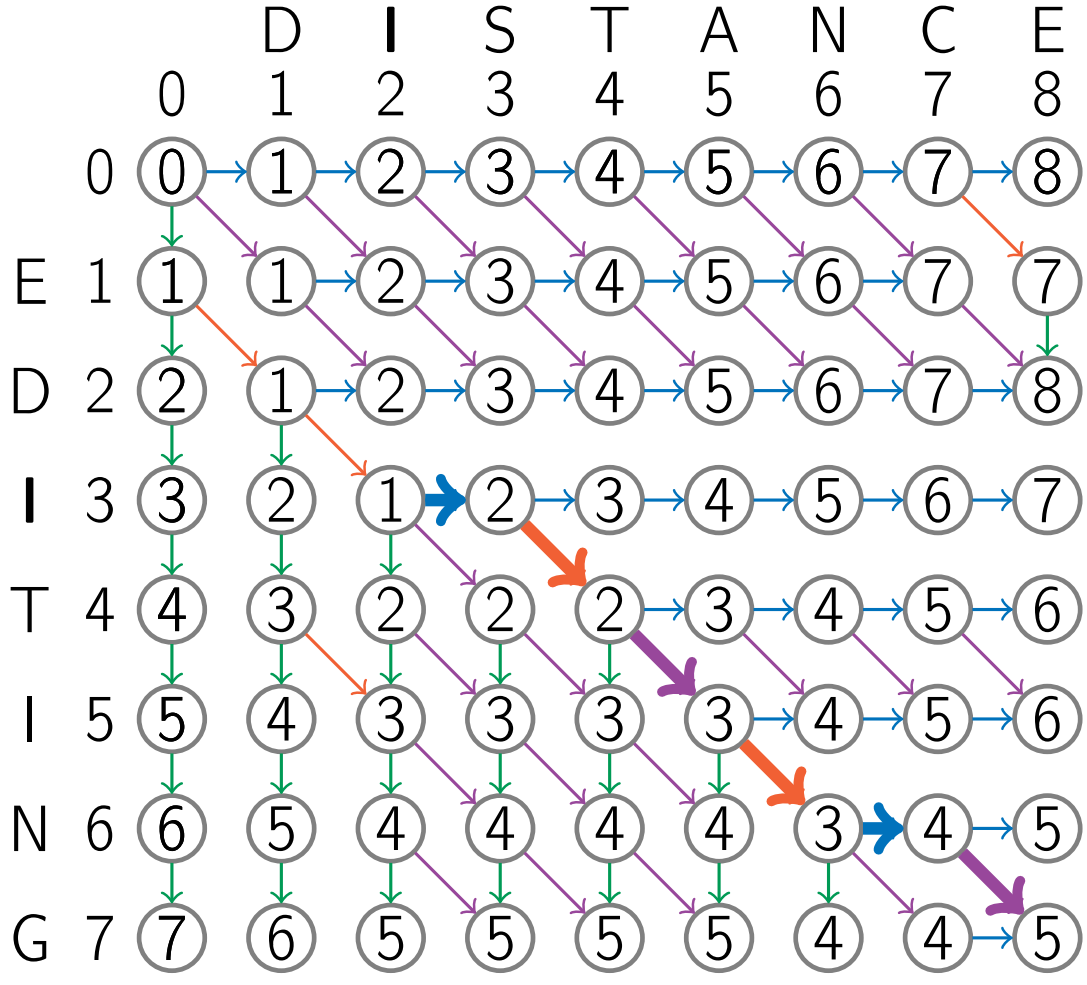

N — G
N C E


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 2 3 4 5 6 7 7
D 2 2 1 2 3 4 5 6 7 8
I 3 3 2 1 2 3 4 5 6 7
T 4 4 3 2 2 2 3 4 5 6
I 5 5 4 3 3 3 3 4 5 6
N 6 6 5 4 4 4 4 3 4 5
G 7 7 6 5 5 5 5 4 4 5

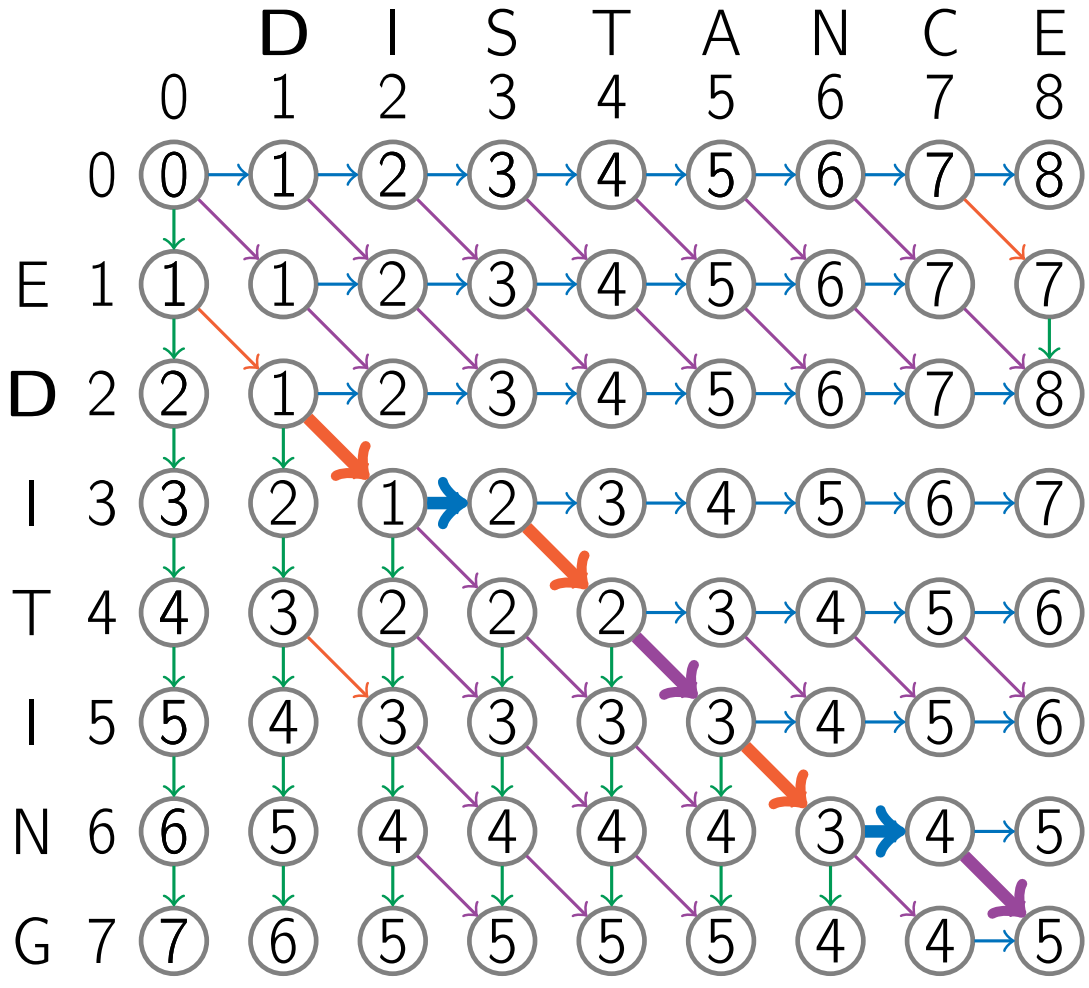
I	N	—	G
A	N	C	E


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 2 3 4 5 6 7 7
D 2 2 1 2 3 4 5 6 7 8
I 3 3 2 1 2 3 4 5 6 7
T 4 4 3 2 2 2 3 4 5 6
I 5 5 4 3 3 3 3 4 5 6
N 6 6 5 4 4 4 4 3 4 5
G 7 7 6 5 5 5 5 4 4 5

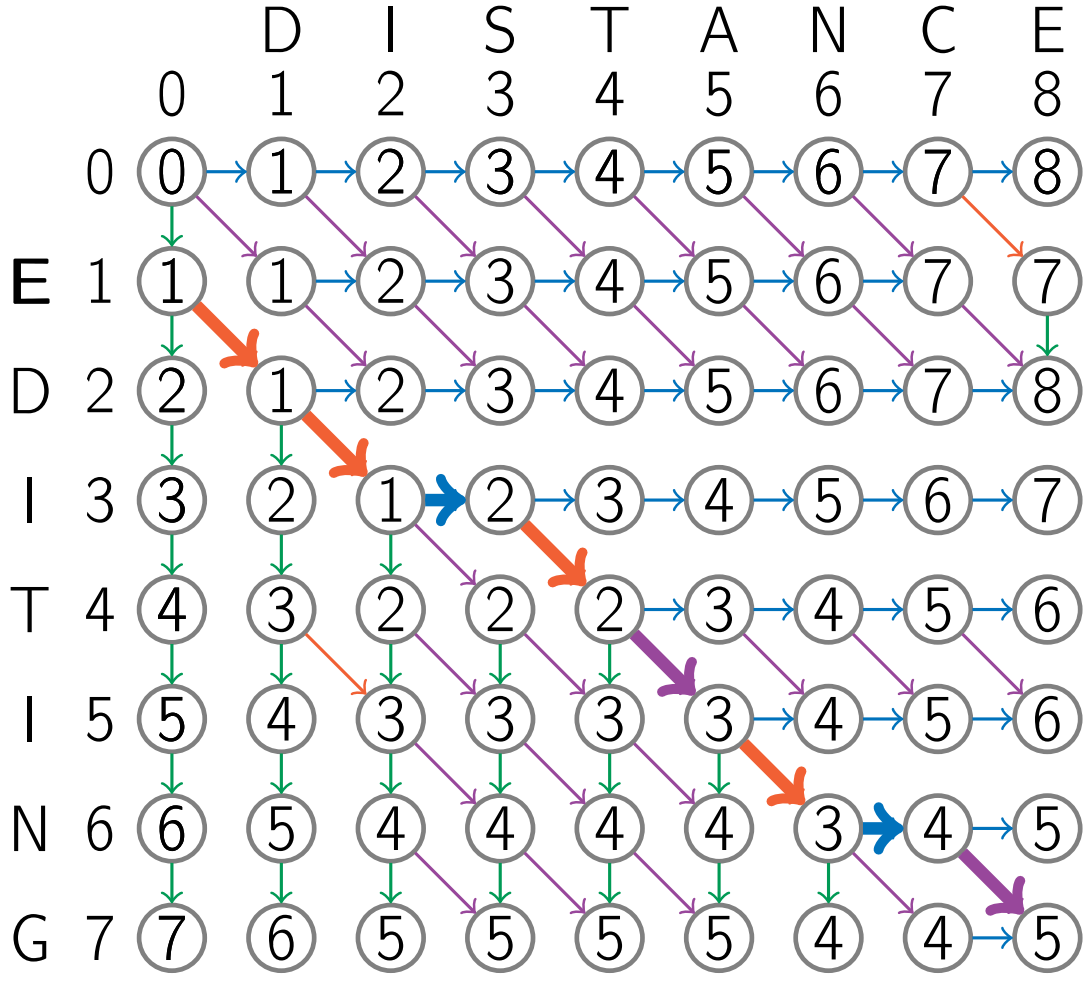
T	I	N	—	G
T	A	N	C	E


D I S T A N C E
0 1 2 3 4 5 6 7 8
0 0 1 2 3 4 5 6 7 8
E 1 1 1 2 3 4 5 6 7 7
D 2 2 1 2 3 4 5 6 7 8
I 3 3 2 1 2 3 4 5 6 7
T 4 4 3 2 2 2 3 4 5 6
I 5 5 4 3 3 3 3 4 5 6
N 6 6 5 4 4 4 4 3 4 5
G 7 7 6 5 5 5 5 4 4 5

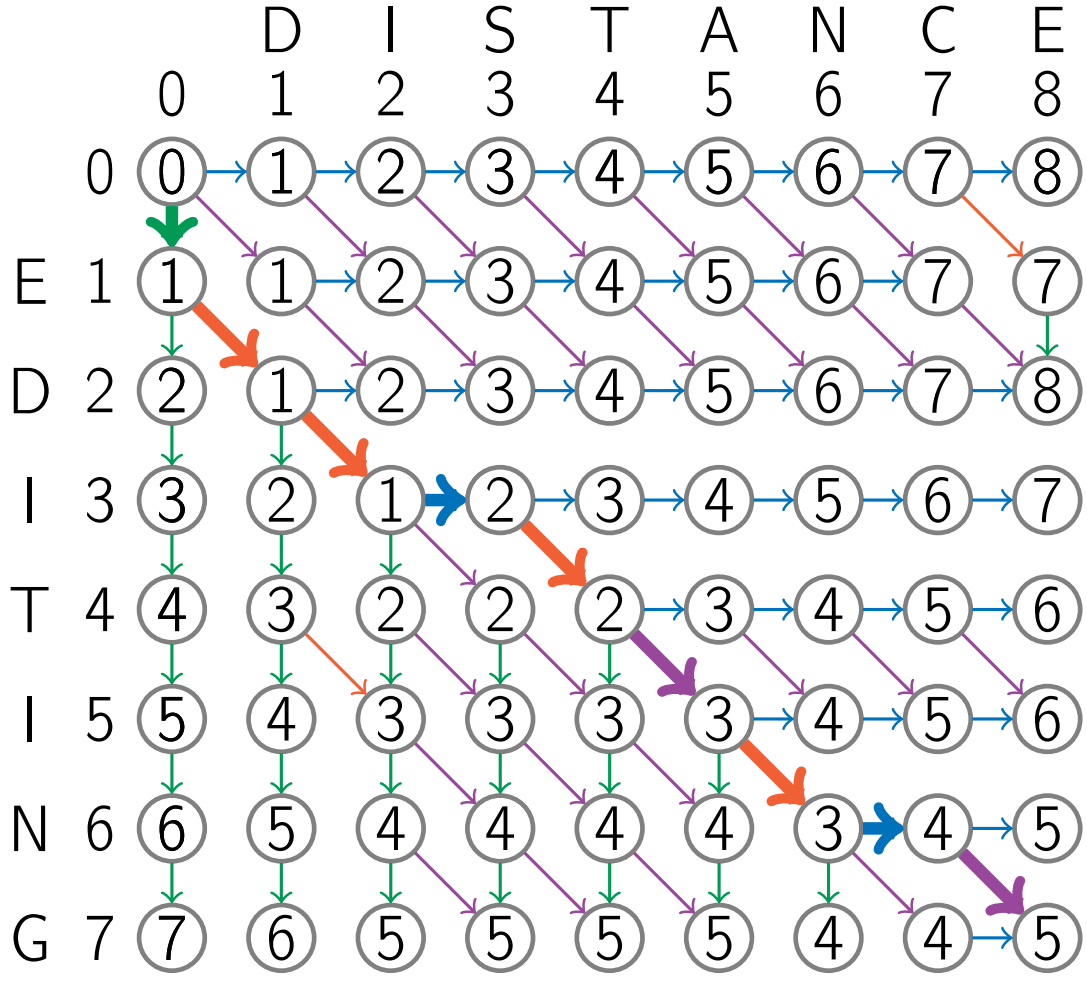
—	T	I	N	—	G
S	T	A	N	C	E



I	—	T	I	N	—	G
I	S	T	A	N	C	E



D	I	—	T	I	N	—	G
D	I	S	T	A	N	C	E



E	D	I	—	T	I	N	—	G
—	D	I	S	T	A	N	C	E

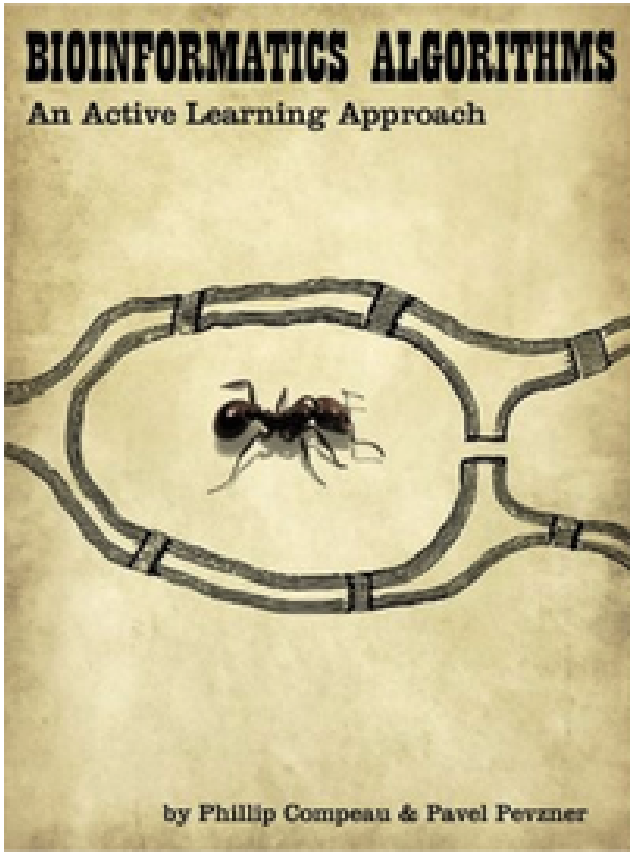
OutputAlignment(i, j)

```
if i = 0 and j = 0:
  return
if backtrack(i, j) = ↓:
  OutputAlignment(i − 1, j)
  print A[i] / −
else if backtrack(i, j) = →:
  OutputAlignment(i, j − 1)
  print − / B[j]
else:
  OutputAlignment(i − 1, j − 1)
  print A[i] / B[j]
```

OutputAlignment(i, j)

```
if i = 0 and j = 0:
    return
if i > 0 and D(i, j) = D(i − 1, j) + 1:
    OutputAlignment(i − 1, j)
    print A[i] / −
else if j > 0 and D(i, j) = D(i, j − 1) + 1:
    OutputAlignment(i, j − 1)
    print − / B[j]
else:
    OutputAlignment(i − 1, j − 1)
    print A[i] / B[j]
```

Comparing Genes, Proteins, and Genomes MOOC (a part of Bioinformatics Specialization on Coursera)



Bioinformatics Algorithms textbook at bioinformaticsalgorithms.org (2nd two-volume edition was published in 2015)