Dynamic Programming

Minimum Edit Distance and Longest Increasing Subsequence

Introduction

- Scenario: spell checker handling misspelled words

Introduction

- When a spell checker encounters a possible misspelled word, it suggests possible words that are close to the misspelled word by looking at other words in its dictionary

- How does the spell checker determine which words are close to the misspelled word?

Introduction

- How do you measure how "close" one word is from another?

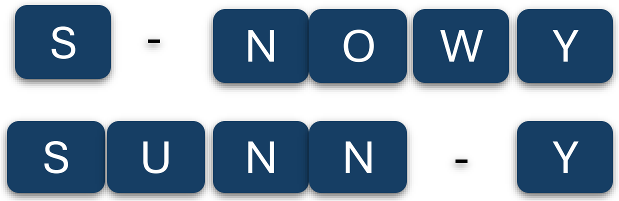
Alignments

- A natural measure of the distance between two strings is the extent to which they can be aligned or matched up

- An alignment is a way of writing the strings one above the other

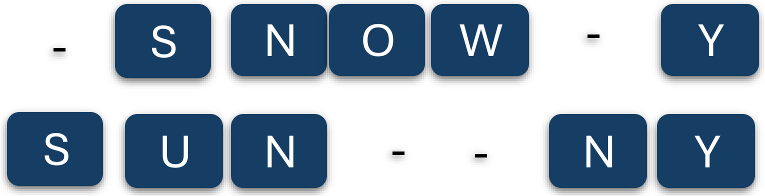
Alignments

- Possible alignments of SNOWY and SUNNY



S	-	N	O	W	Y
S	U	N	N	-	Y

Cost: 3



-	S	N	O	W	-	Y
S	U	N	-	-	N	Y

Cost: 5

Edit distance

- Named as such because it can also be thought of as the minimum number of edits needed to transform first string into the second

- Also known as the sequence alignment problem

Edit distance

- The (minimum) edit distance between two strings is the cost of their best possible alignment

- Best possible alignment = an alignment that minimizes the cost

- aka Levenshtein distance, after Vladimir Levenshtein, who considered this distance in 1965

Edit distance

- **Input:** Two sequences *X[1..m]* and *Y[1..n]* (usually strings, but could also be numbers)

- **Output:** The edit distance and the best possible alignment of the two sequences

How can we solve the problem?

Use **Brute-force** approach.

Brute Force Algorithm

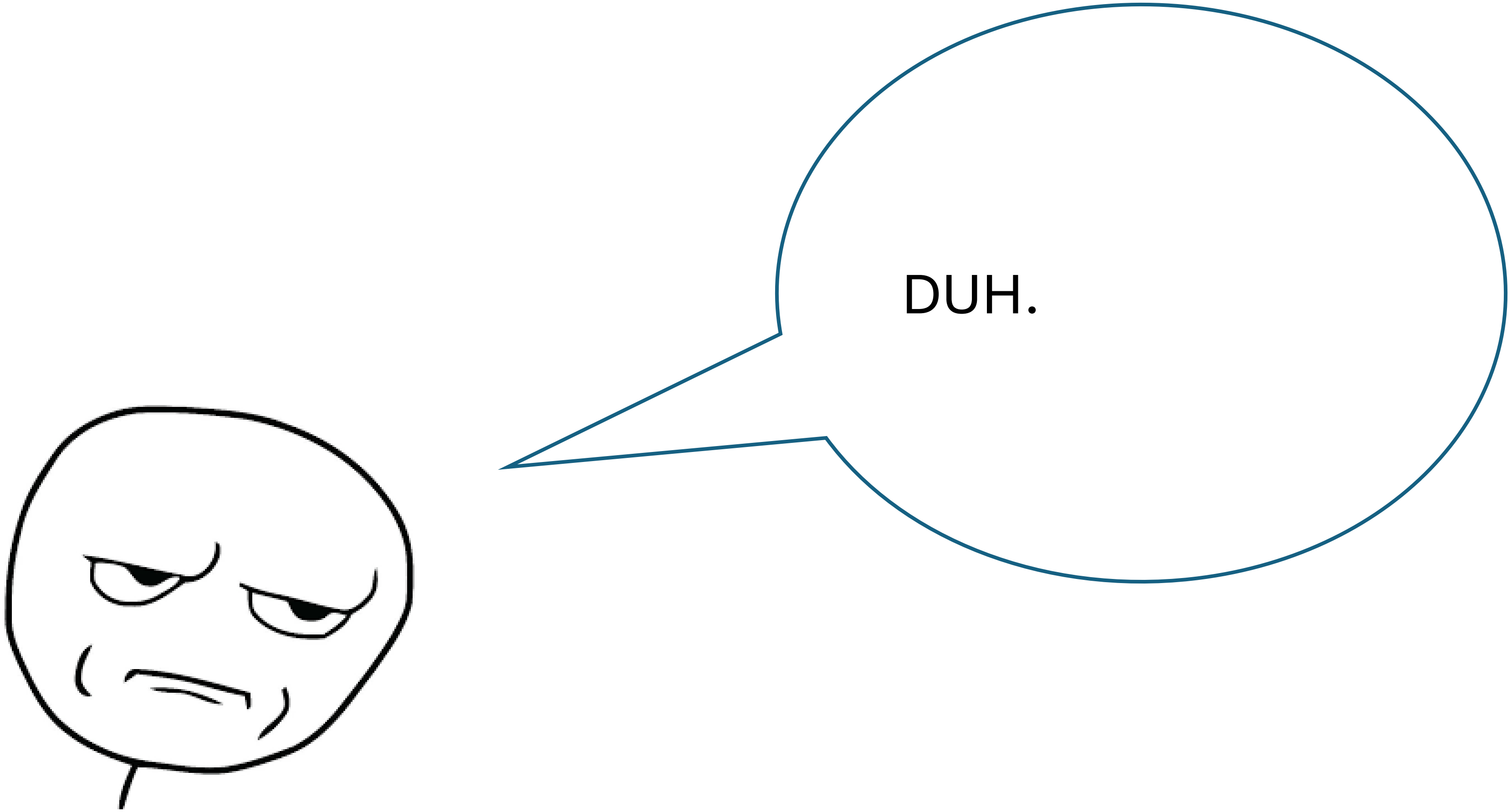
- Generate all possible alignments for the two sequences and compute their respective costs

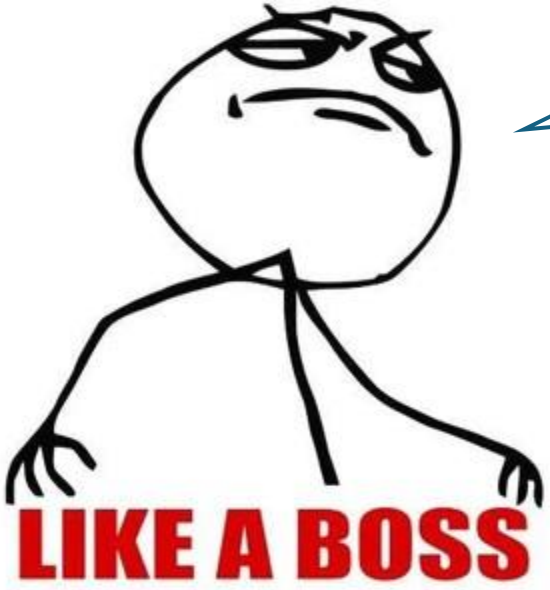
- Return the alignment with the least cost

Inefficient

- Since there are so many possible alignments between two strings, it would be inefficient to search through all of them for the best one

Can we do better?


DUH.



USE **DP APPROACH**.

DP Solution

- First and most crucial question: What are the subproblems?

- We need to divide the problem into smaller subproblems such that it will have an optimal substructure

Subproblems

- We can look at some prefix of the first string *X[1..i]* and some prefix of the second string *Y[1..j]*

- We'll call this subproblem **E(i,j):** the edit distance of prefix *X[1..i]* and prefix *Y[1..j]*

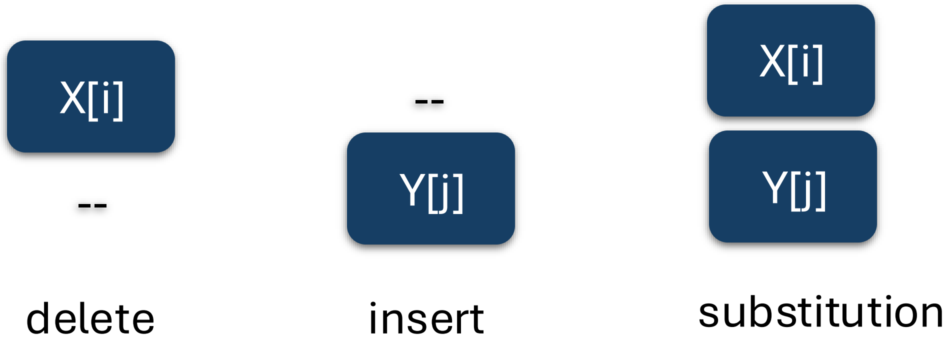
Defining E(i, j)

- We need to express E(i, j) in terms of smaller subproblems

- What do we know about the best alignment of *X[1..i]* and *Y[1..j]*?

Three cases

- We know that the rightmost column of the best alignment between *X[1..i]* and *Y[1..j]* can only be one of the three things:



3 operations

- Deletion
- Insertion
- Replacement
- Copy

Example

kitten → knitting

- Operation: insertion
- Knitten

Example

kitten → knitting

- Operation: Substitution
- Knittin

Example

kitten → knitting

- Operation: Insertion
- Knitting

Computing the solution

- The answers to all the subproblems form a m x n table
- What should be the order for solving these subproblems?

Two possible orderings:

1. one row at a time moving from left to right;
2. one column at a time moving from top to bottom

Computing the solution

- What remains to be solved is the base case: the smallest subproblems

- In this case, our base cases are E(0,*) and E(*,0)

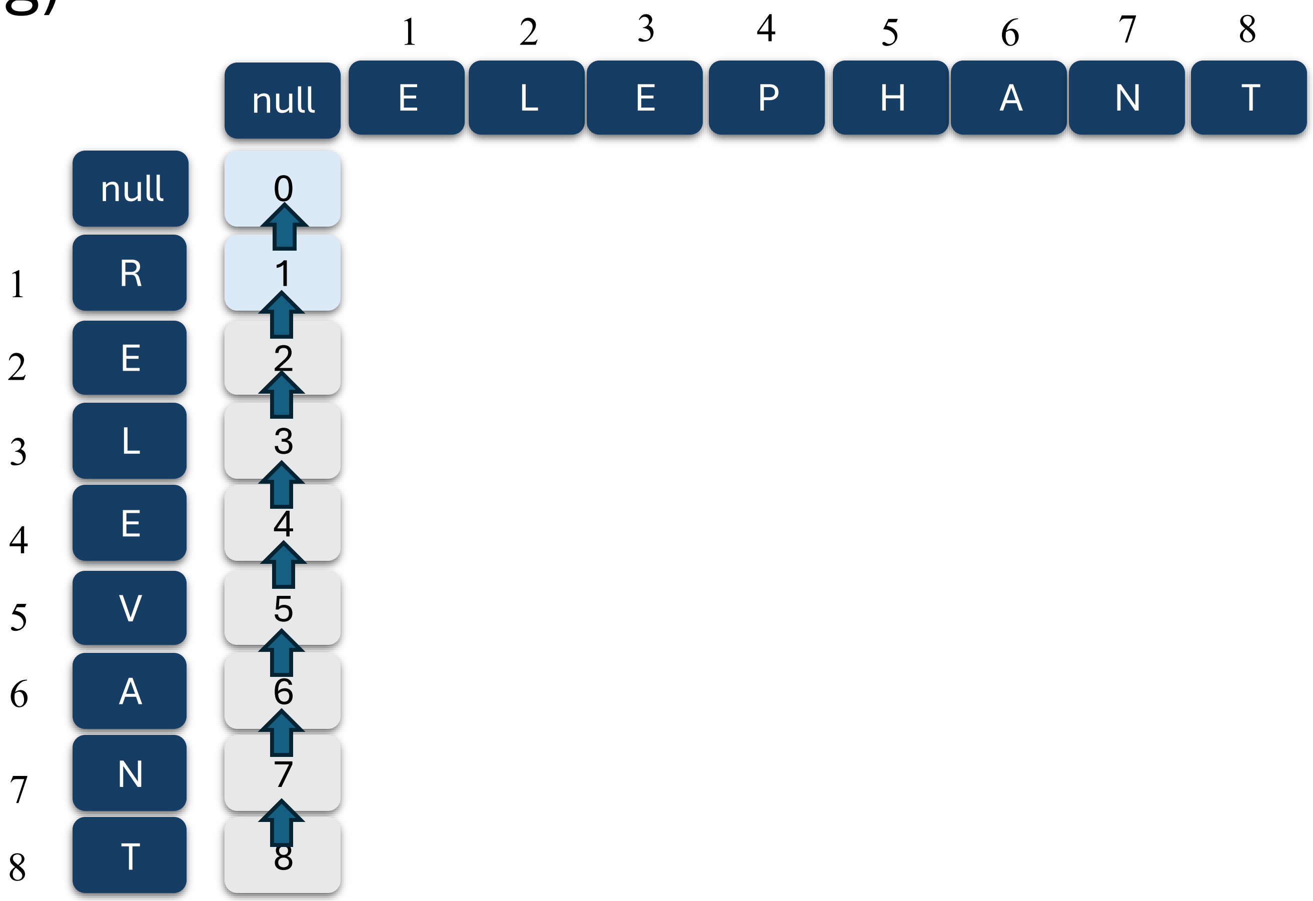
Computing the solution

- E(0,j) is the edit distance between the 0-length prefix of X (empty string) and the first j letters of Y

			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8
1	R									
2	E									
3	L									
4	E									
5	V									
6	A									
7	N									
8	T									

Computing the solution

- E(i,0) is the edit distance between the first i letters of X and the 0-length prefix of Y (empty string)

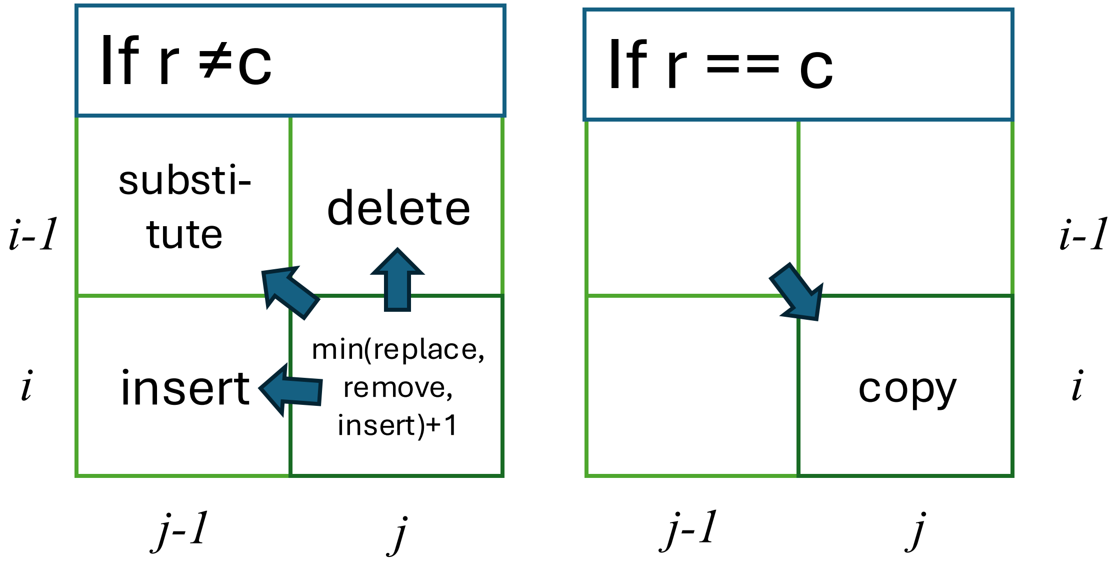


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0								
1	R	1								
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

Defining E(i, j)

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

where diff(i,j) = 0 if X[i] == Y[j], and 1 otherwise



If r ≠c

substi-tute

delete

insert

min(replace, remove, insert)+1

If r == c

copy

Backtrace

- To construct the alignment of the X and Y, we need to keep a backtrace

- Idea is similar to constructing LCS

Backtrace

- Every time we enter a cell, remember where we came from
- Trace back path from the bottom-right corner and follow directions to read off the alignment

- Trace a path starting from the bottom-right corner (M,N) and follow the directions all the way back to
- the top-left corner (0,0); the directions along the way will tell you the alignment:

direction = ↑ X	direction = ← --	direction = ↖ X
deletion --	insertion Y	aligned Y

Backtrace

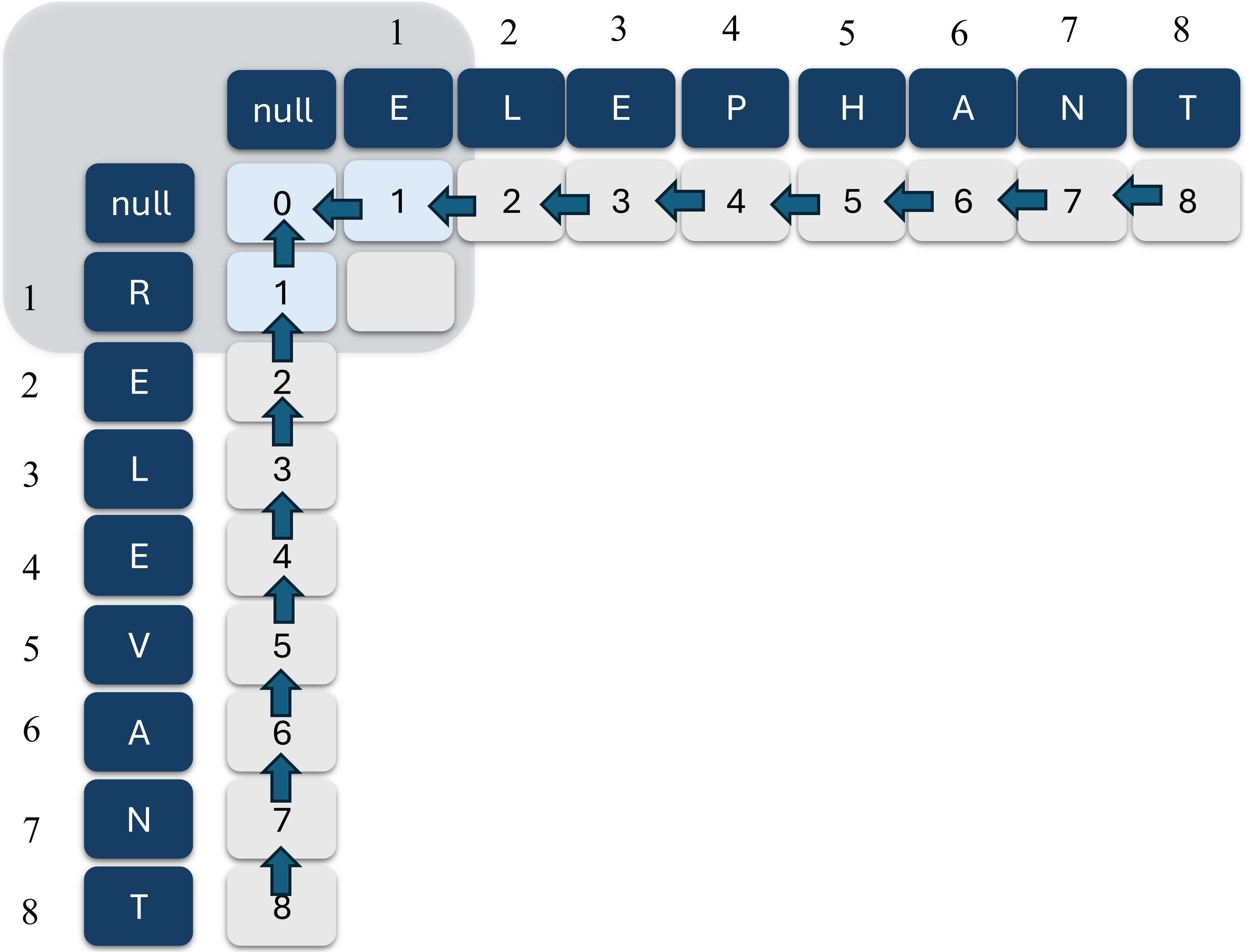
- Insertion: LEFT = E(i, j-1)
- Deletion: UP = E(i-1, j)
- Substitution: Diagonal = E(i-1, j-1)

Multiple Directions

- In keeping backtrace directions, you can store more than 1 direction if there are ties
- Any path from bottom-right to the top-left is a sequence alignment

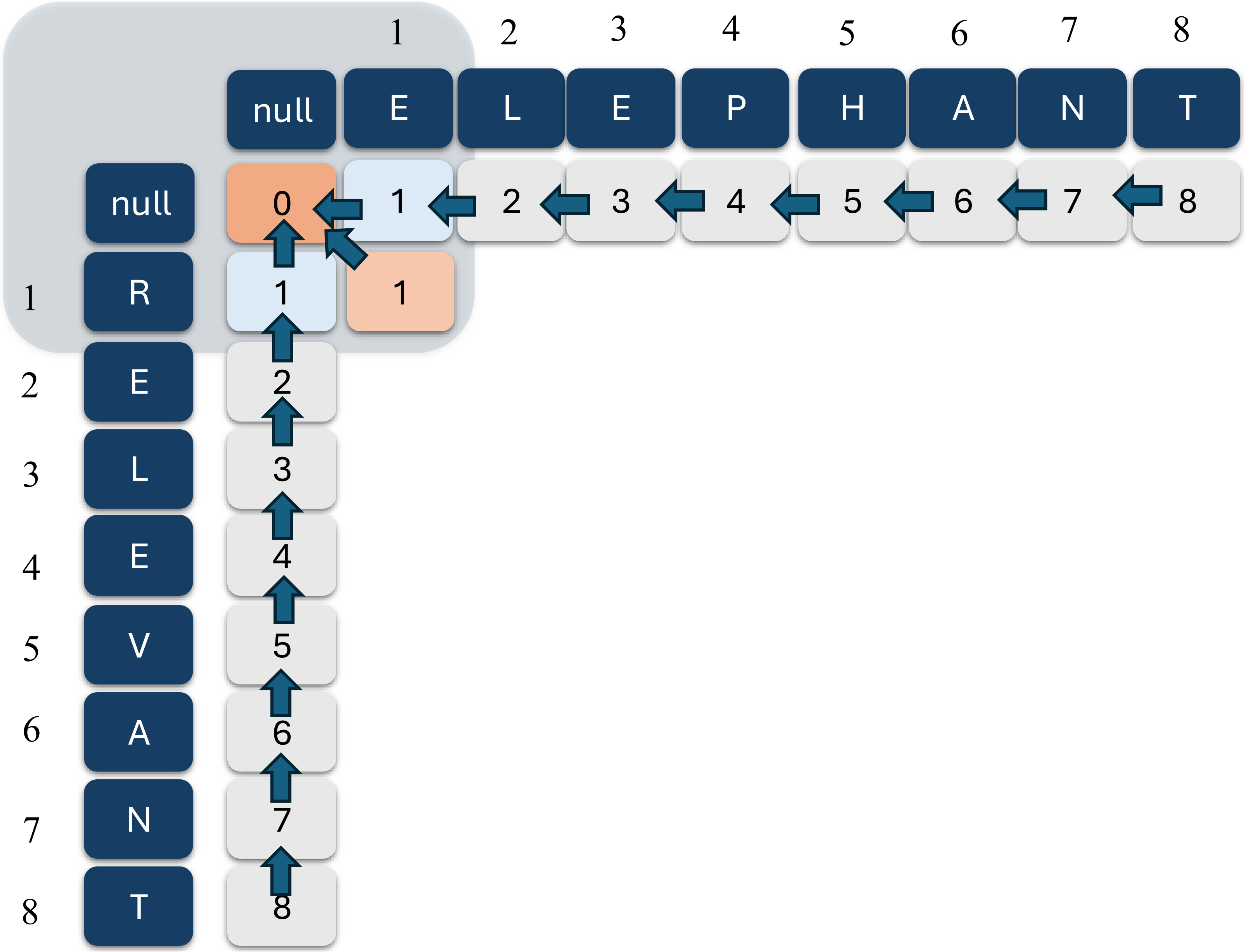
Computing the solution

- $E(0, j) = j$
- $E(I, 0) = i$

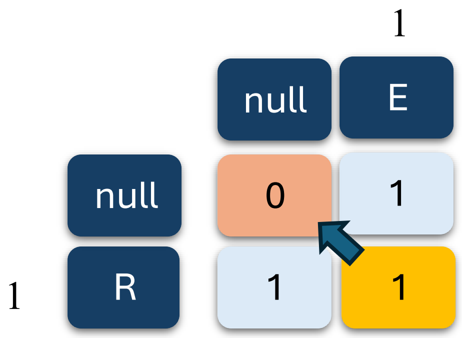


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1								
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



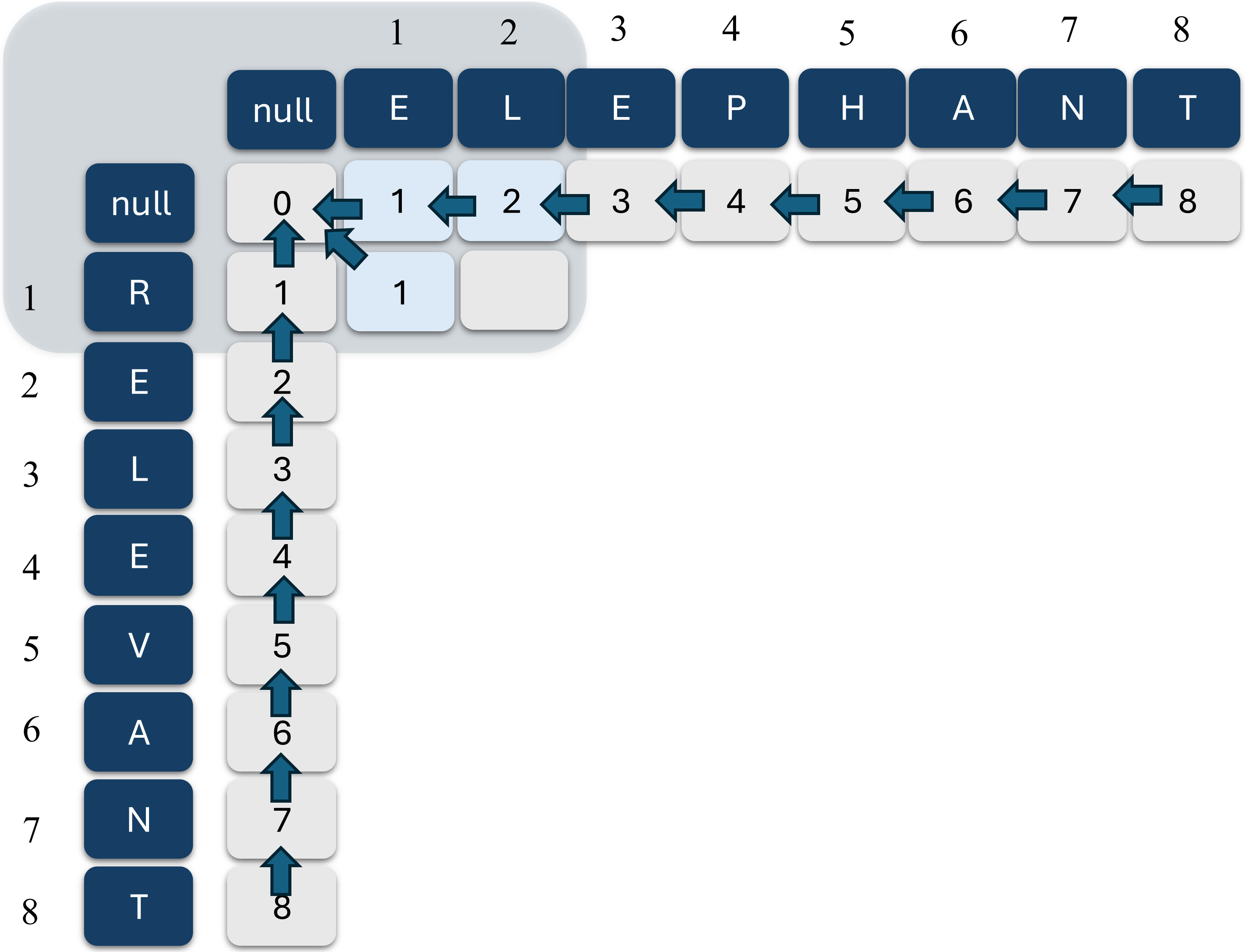
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1							
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								



			1
		null	E
	null	0	1
1	R	1	1

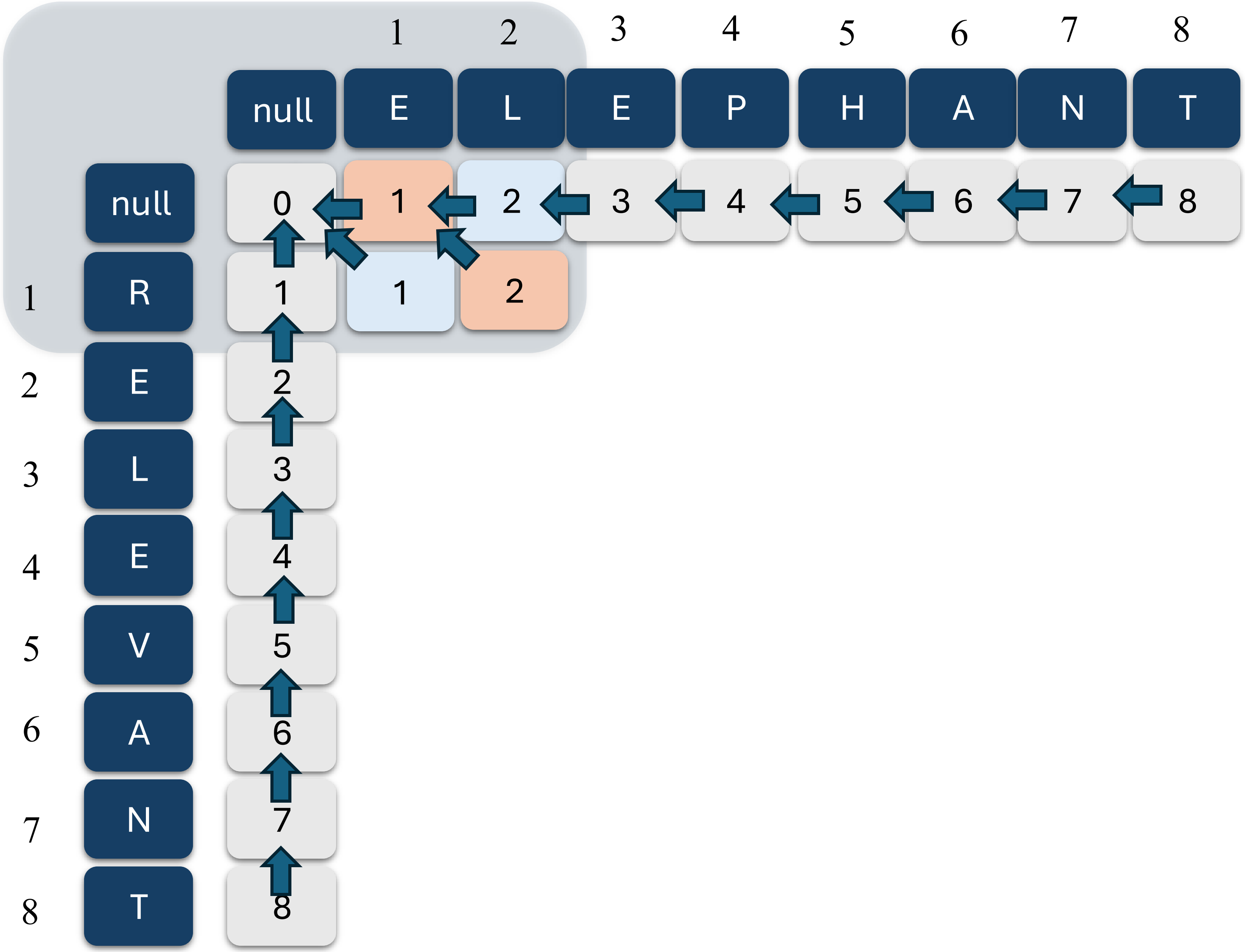
Replace R with E

$$E(i,j) = min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i,j) + E(i-1, j-1) \end{cases}$$

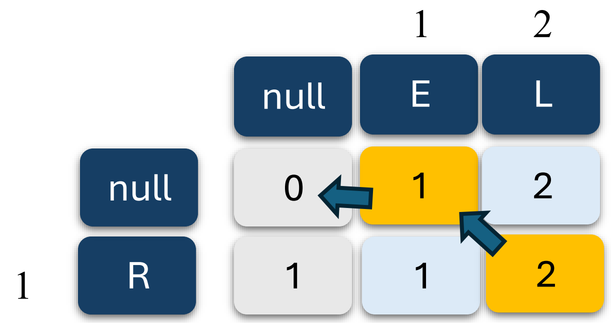


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1							
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



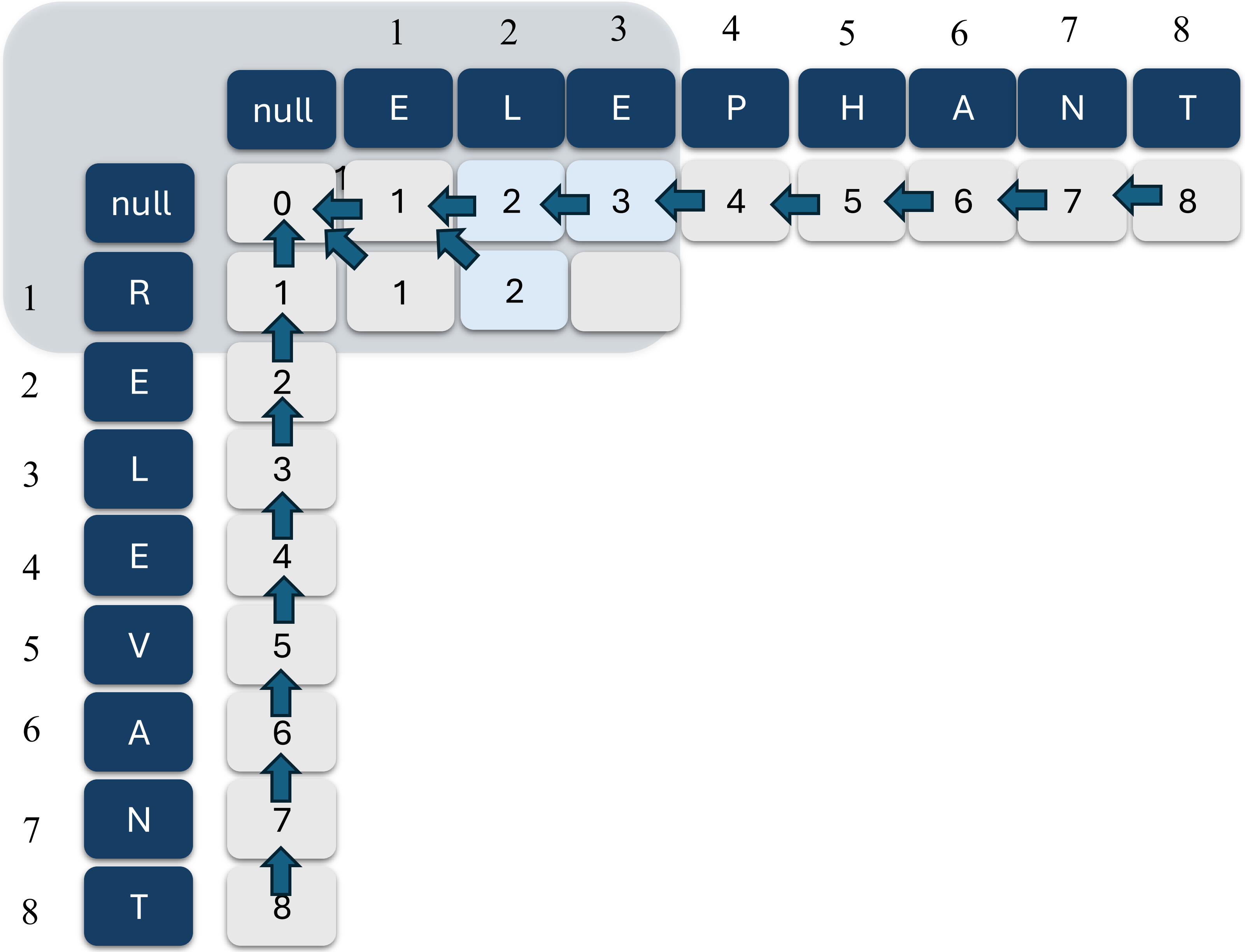
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2						
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								



			1	2
		null	E	L
	null	0	1	2
1	R	1	1	2

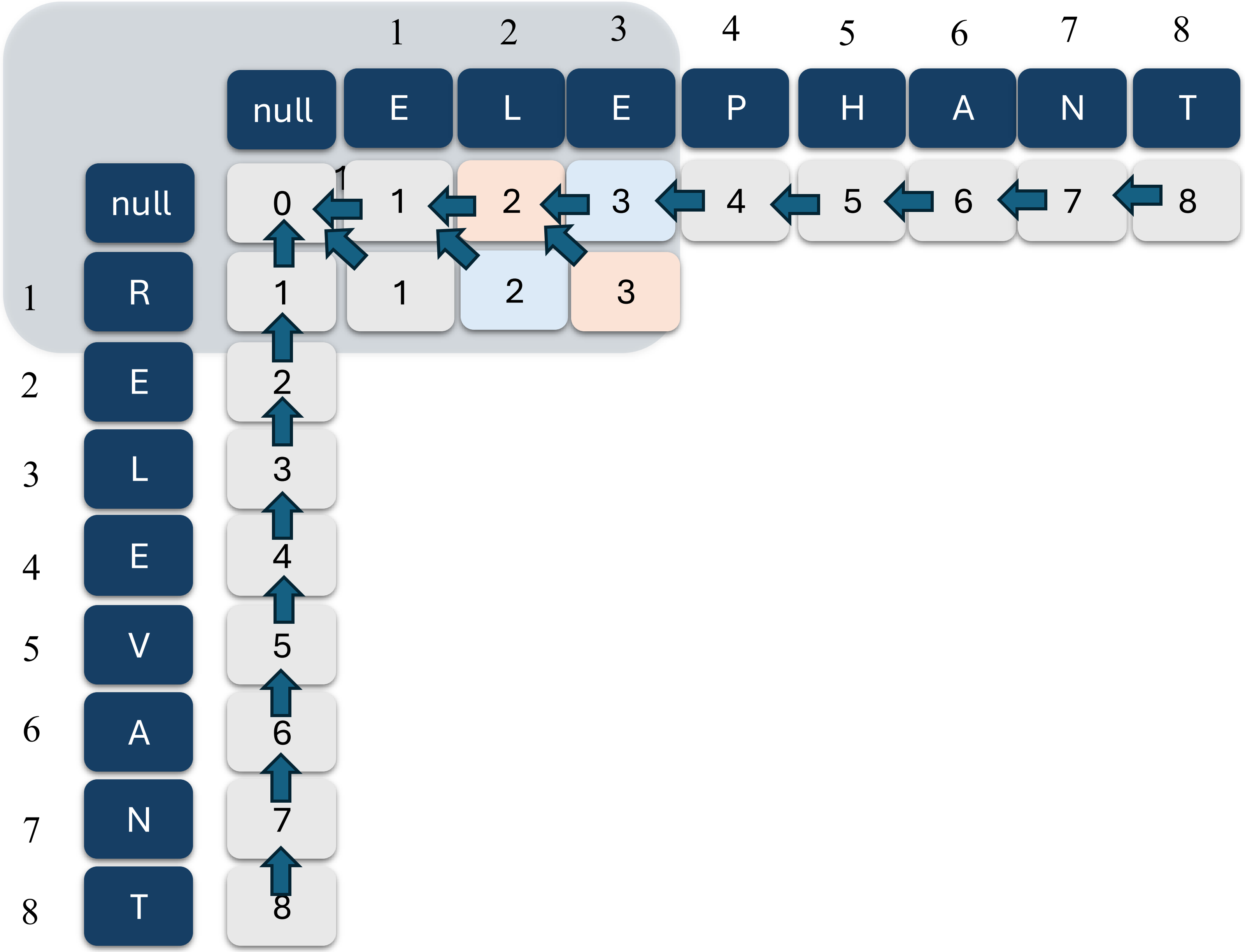
Insert E
Replace R with L

$$E(i,j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i,j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



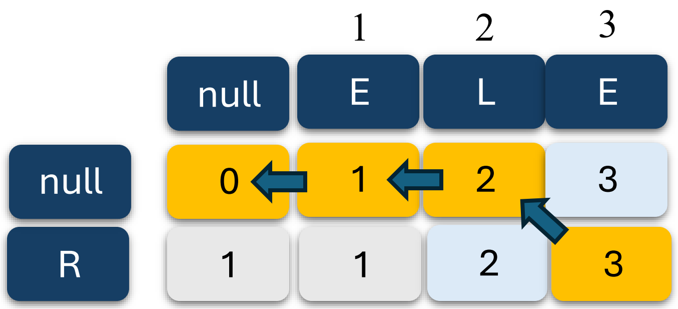
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2						
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3					
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

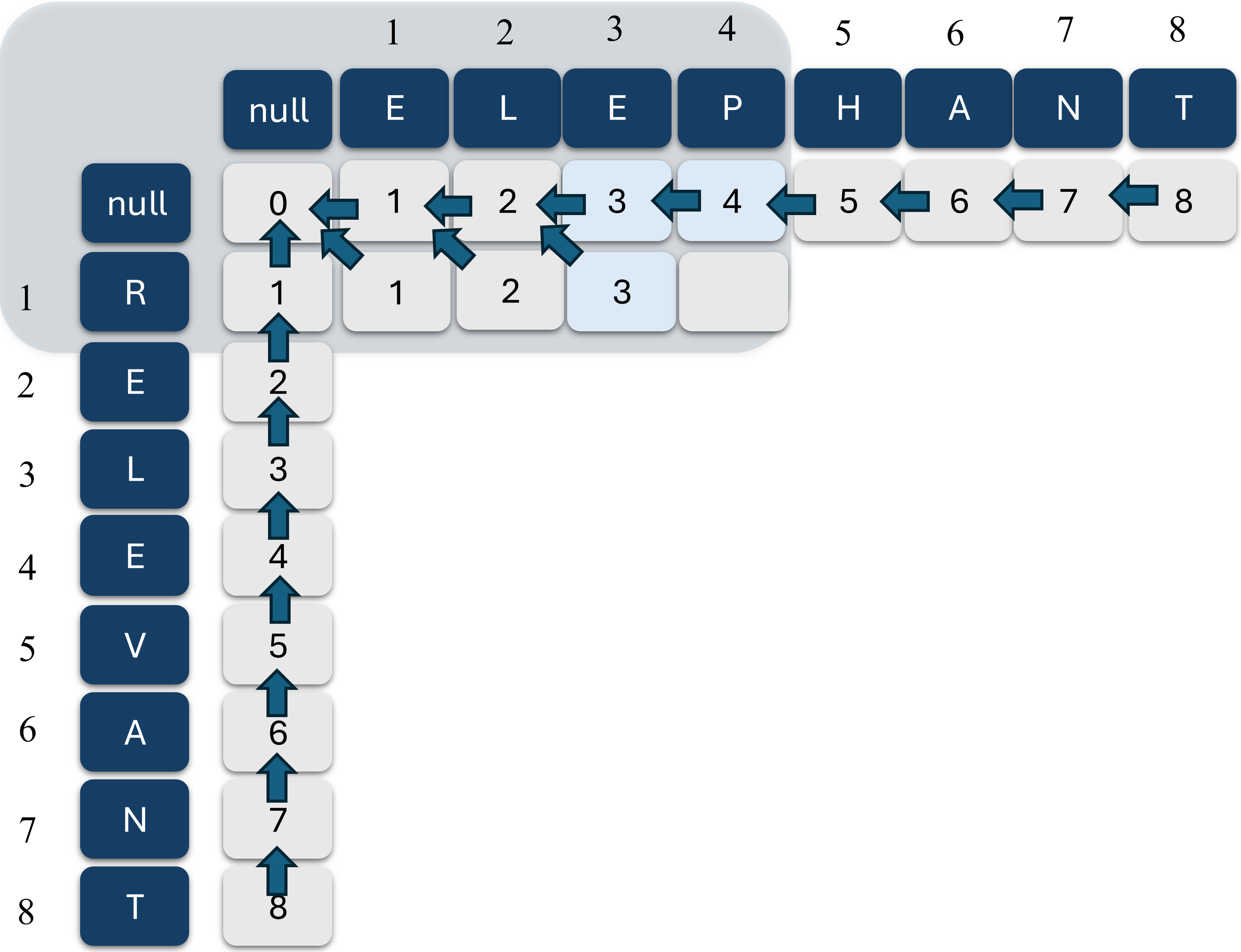
1



		1	2	3
	null	E	L	E
null	0	1	2	3
R	1	1	2	3

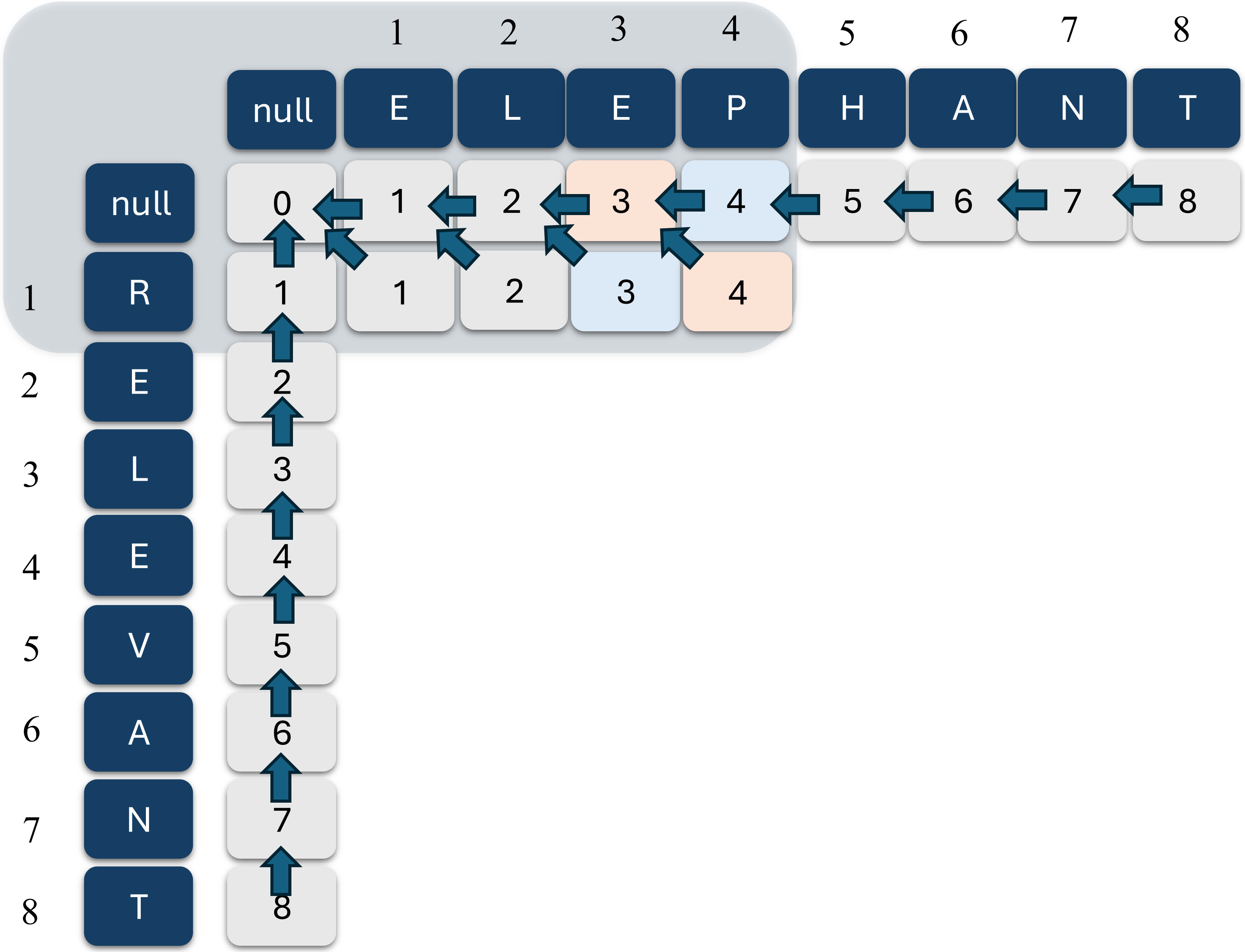
Insert E
Insert L
Replace R with E

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



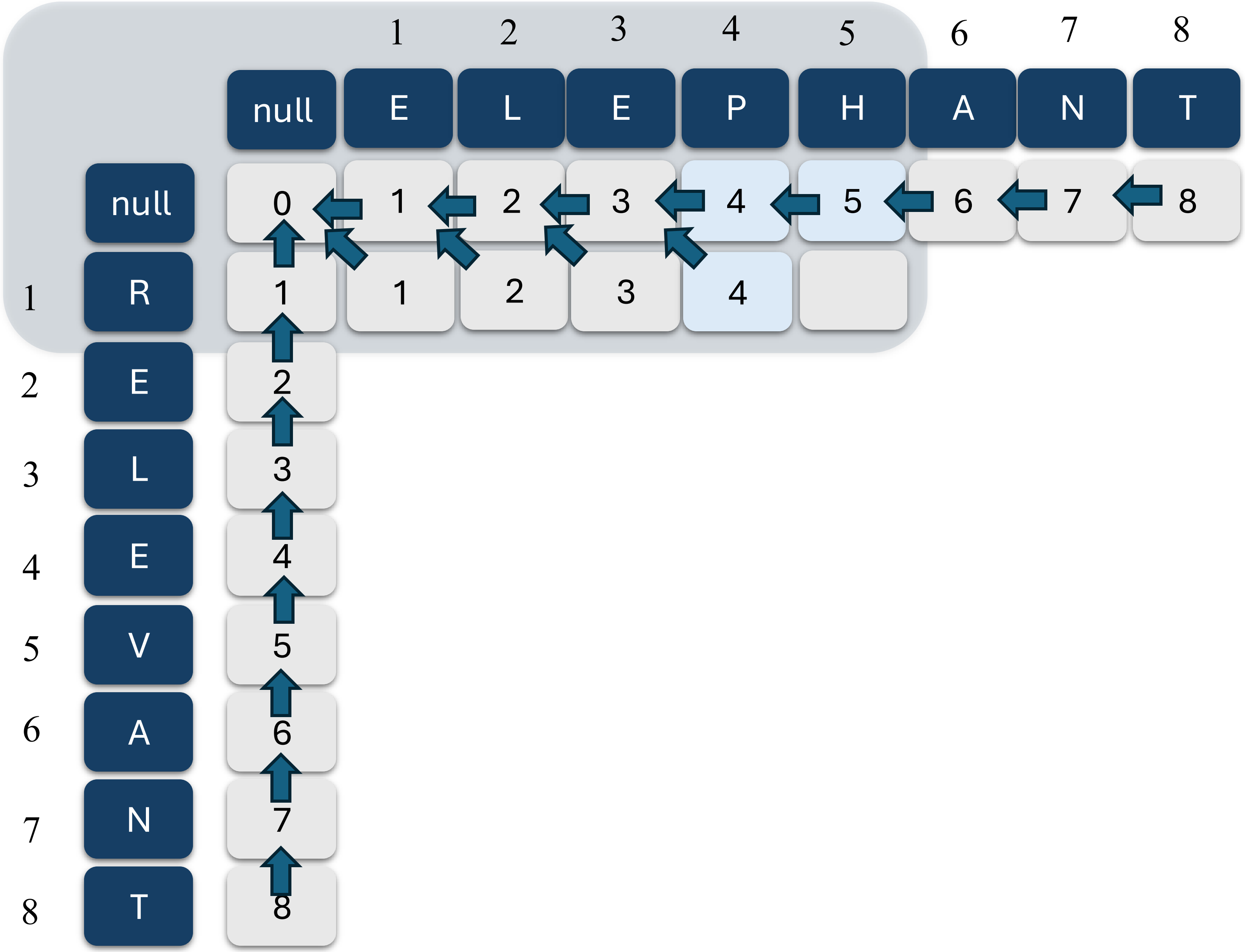
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3					
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = min\begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



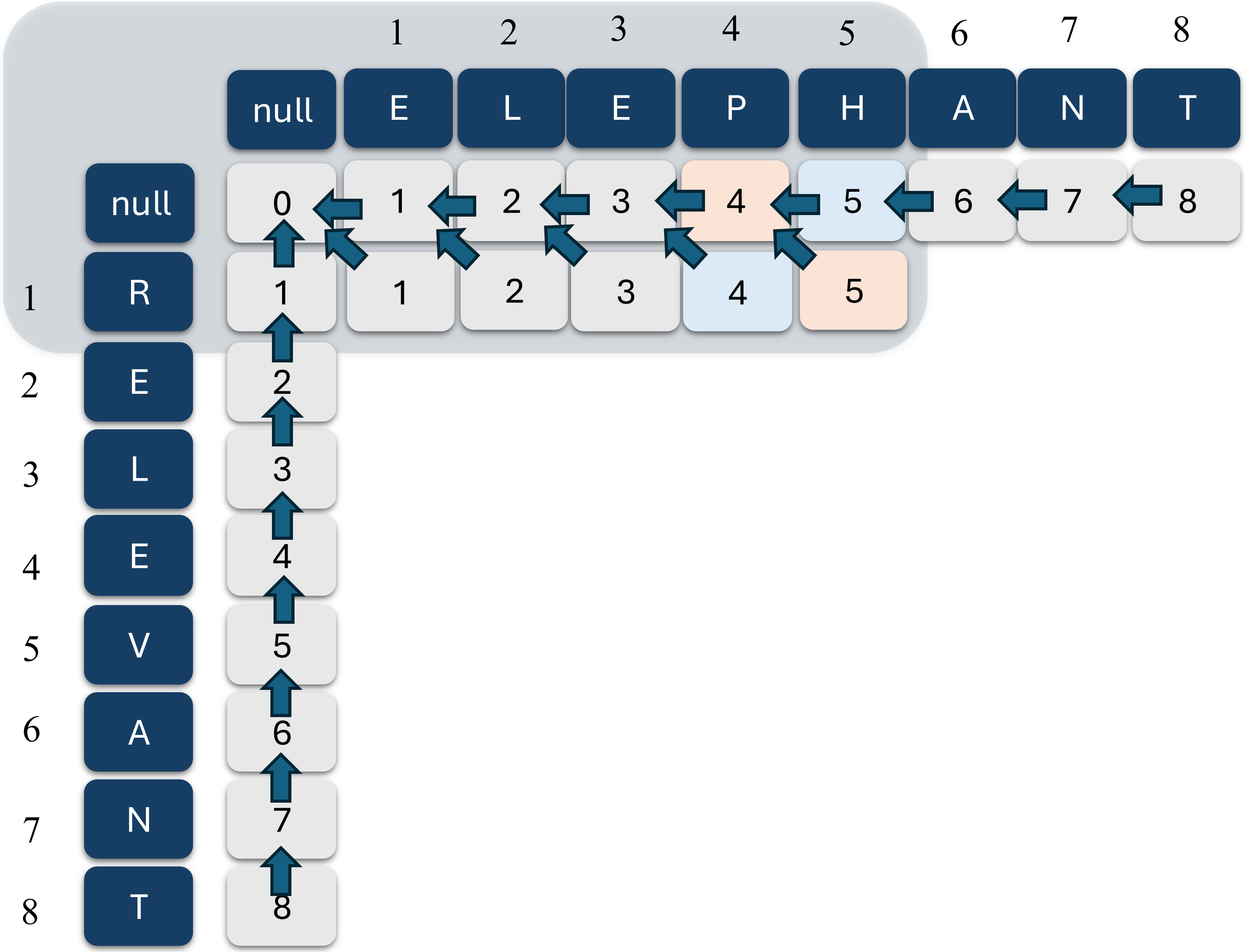
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4				
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

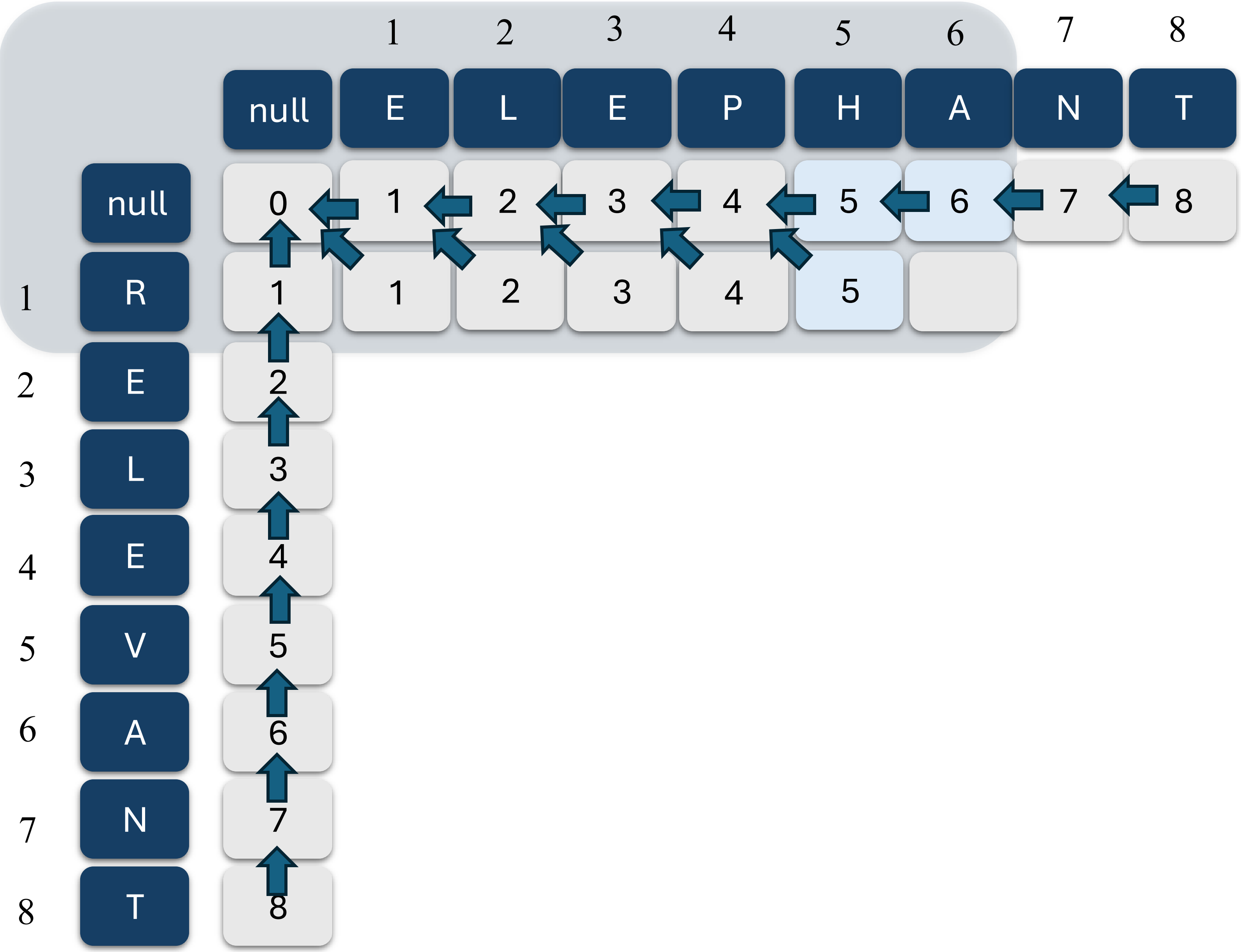
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4				
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

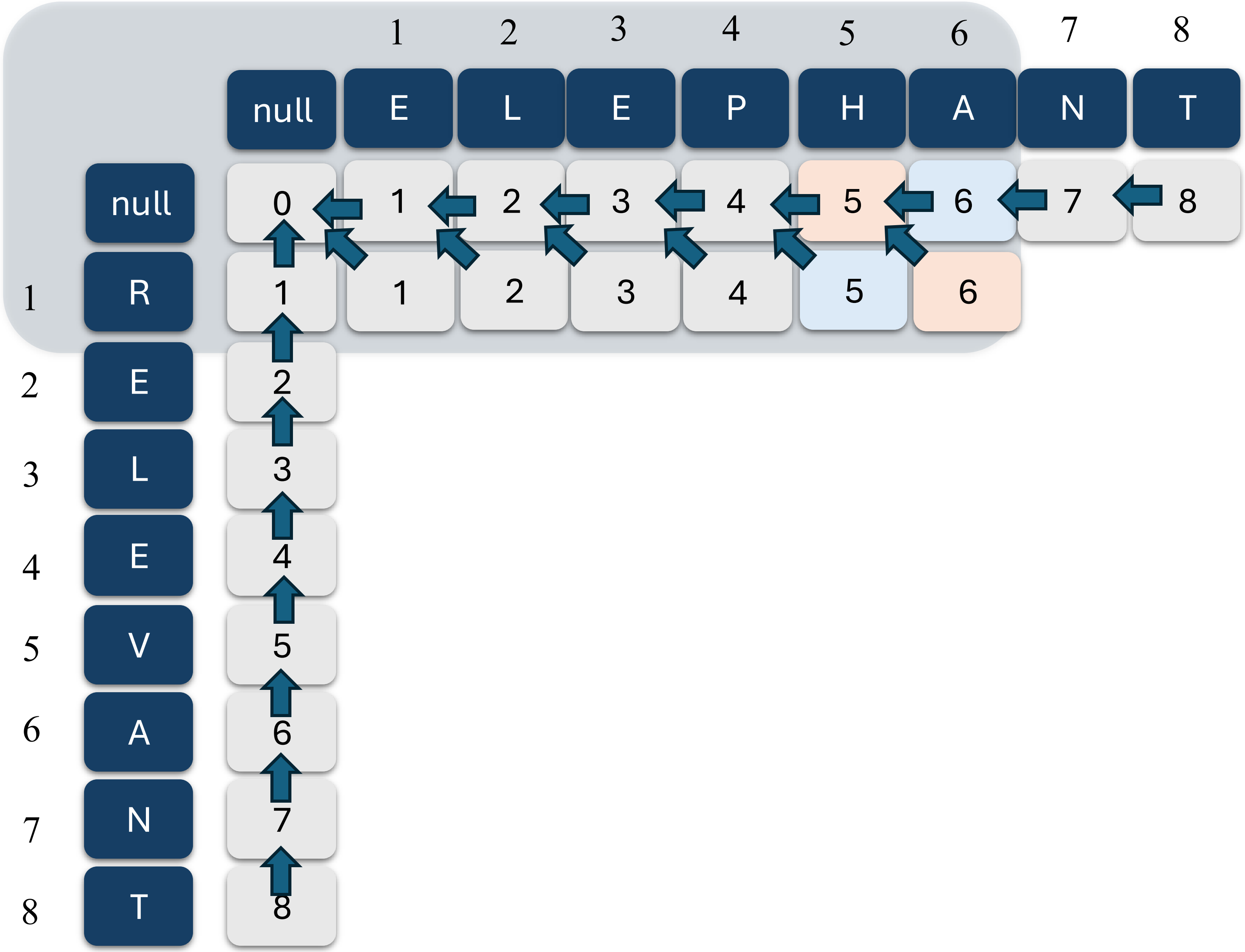
$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$





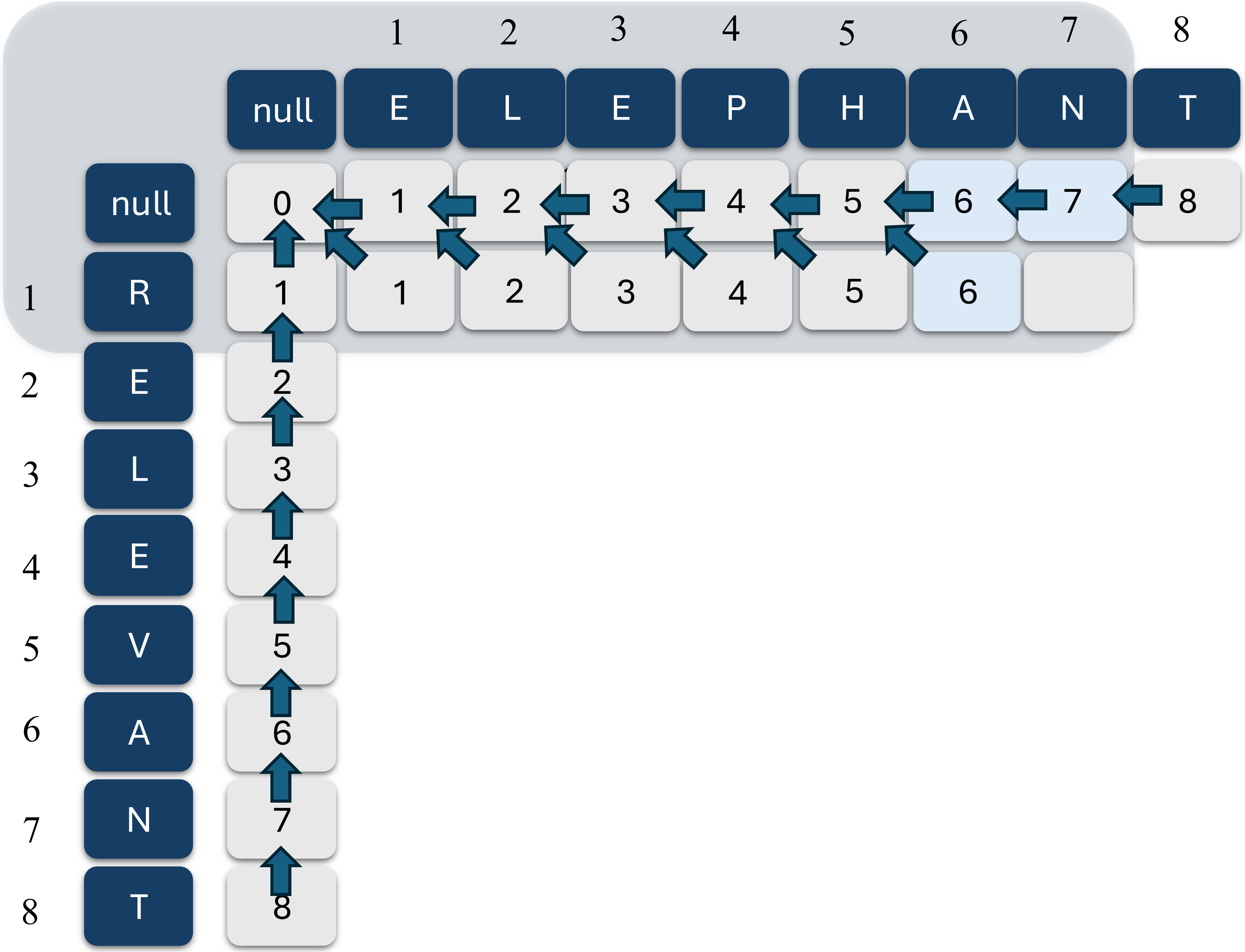
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5			
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



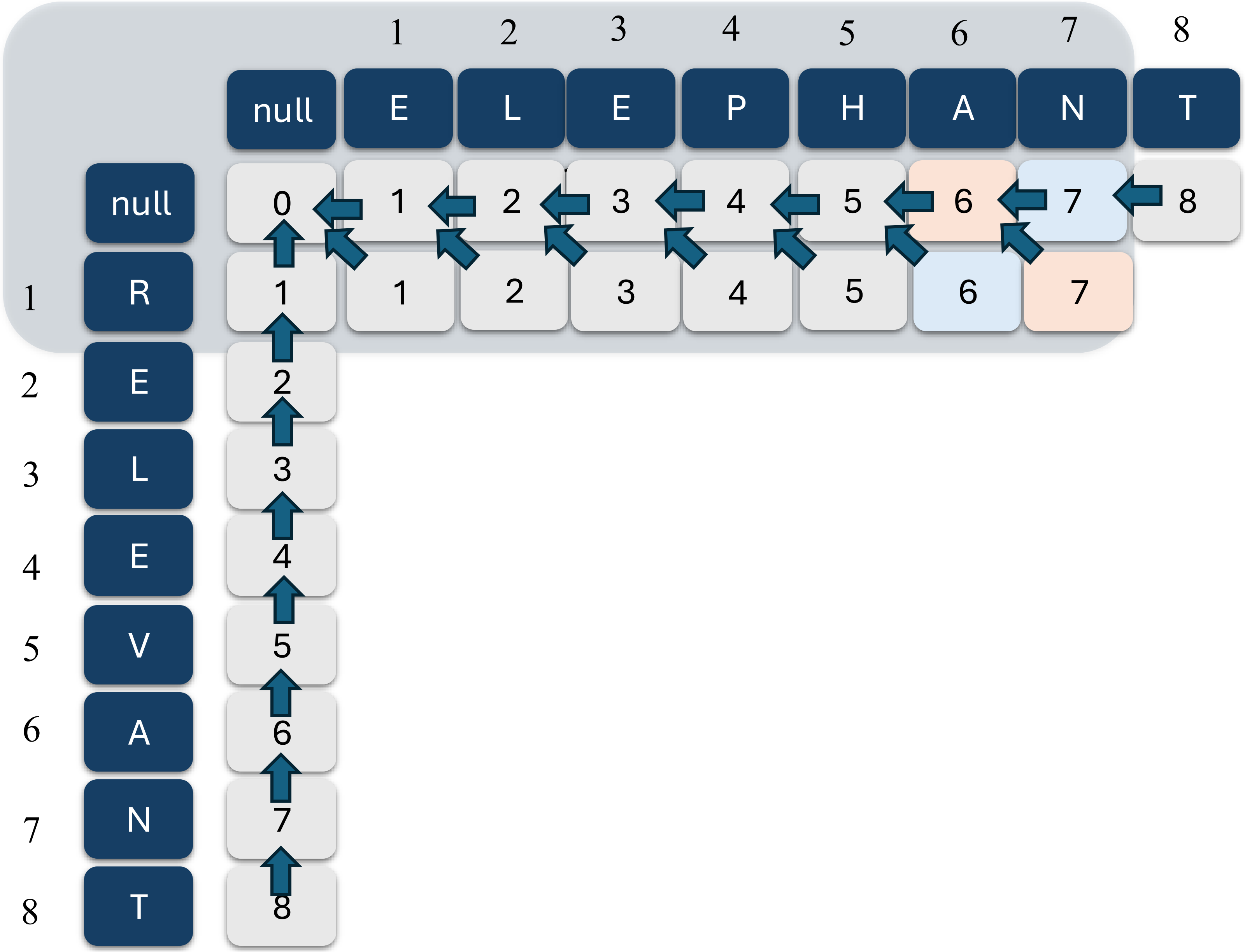
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6		
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6		
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

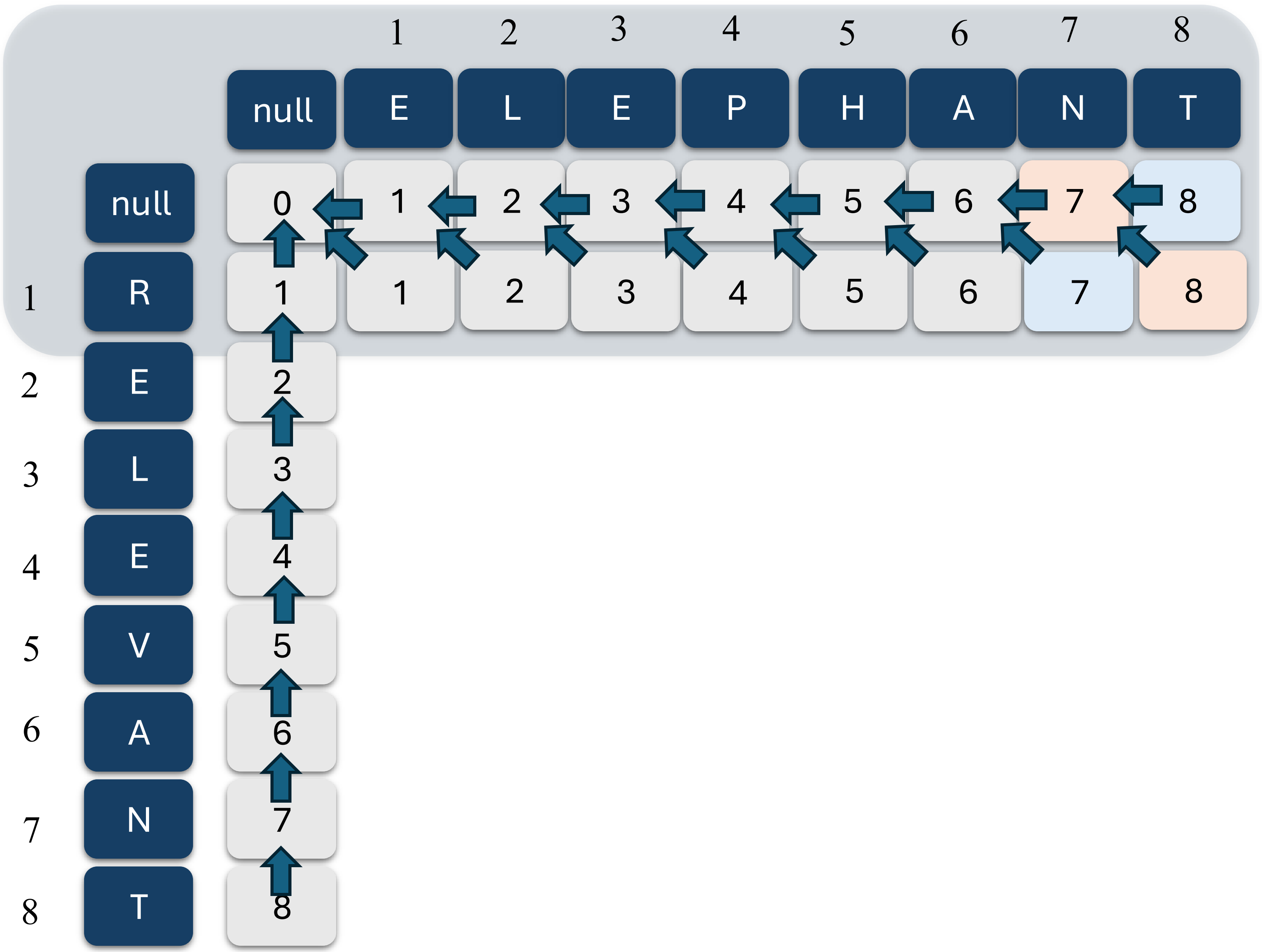


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

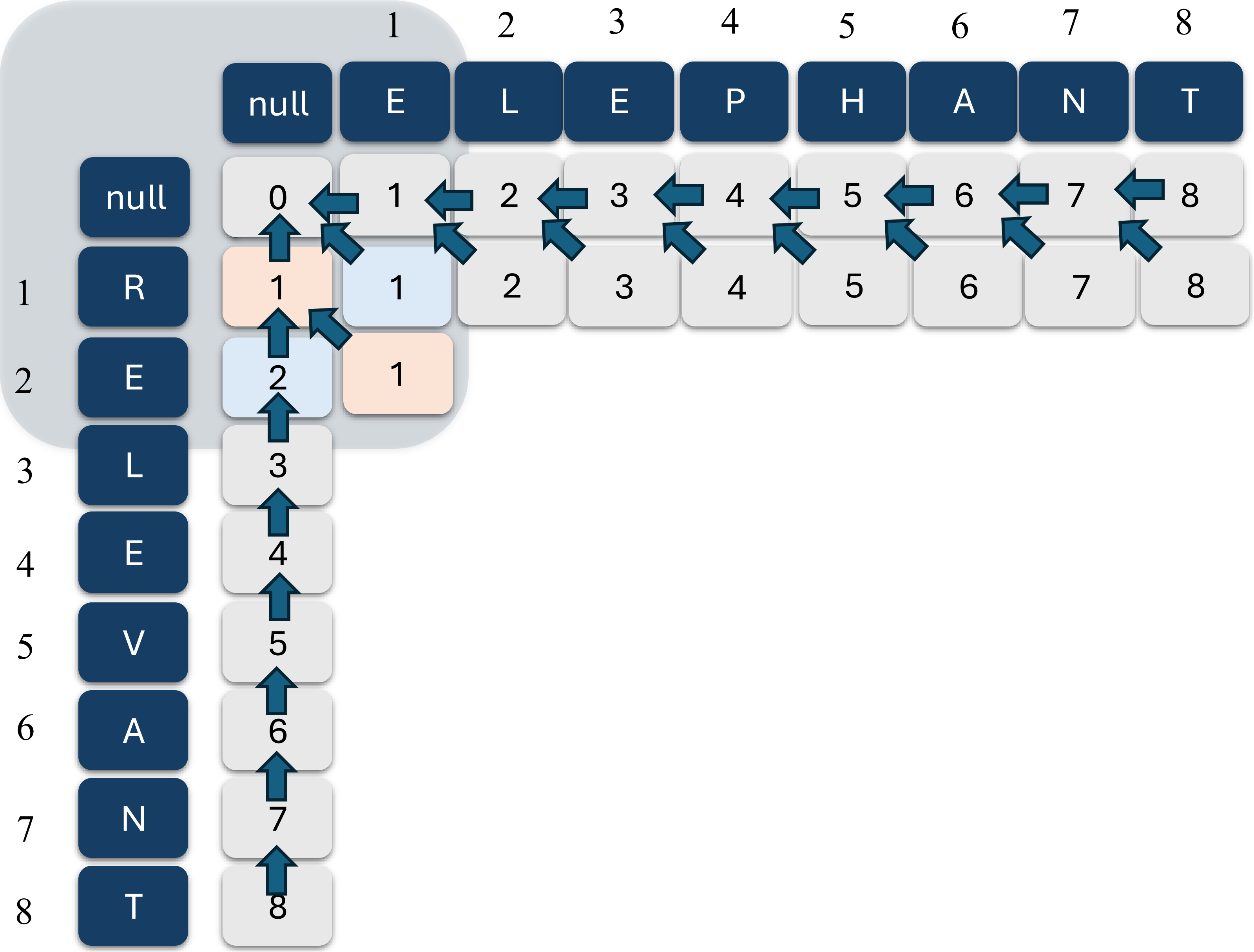
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ \mathit{diff}(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

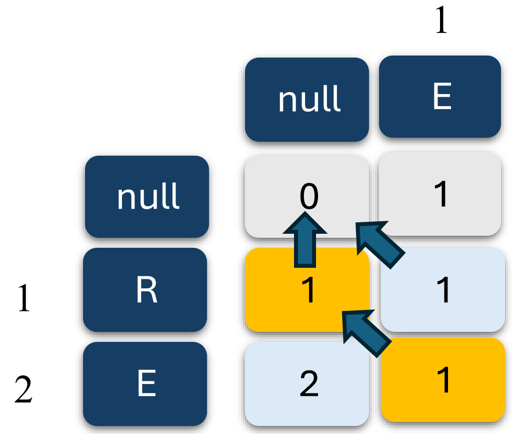


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2								
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



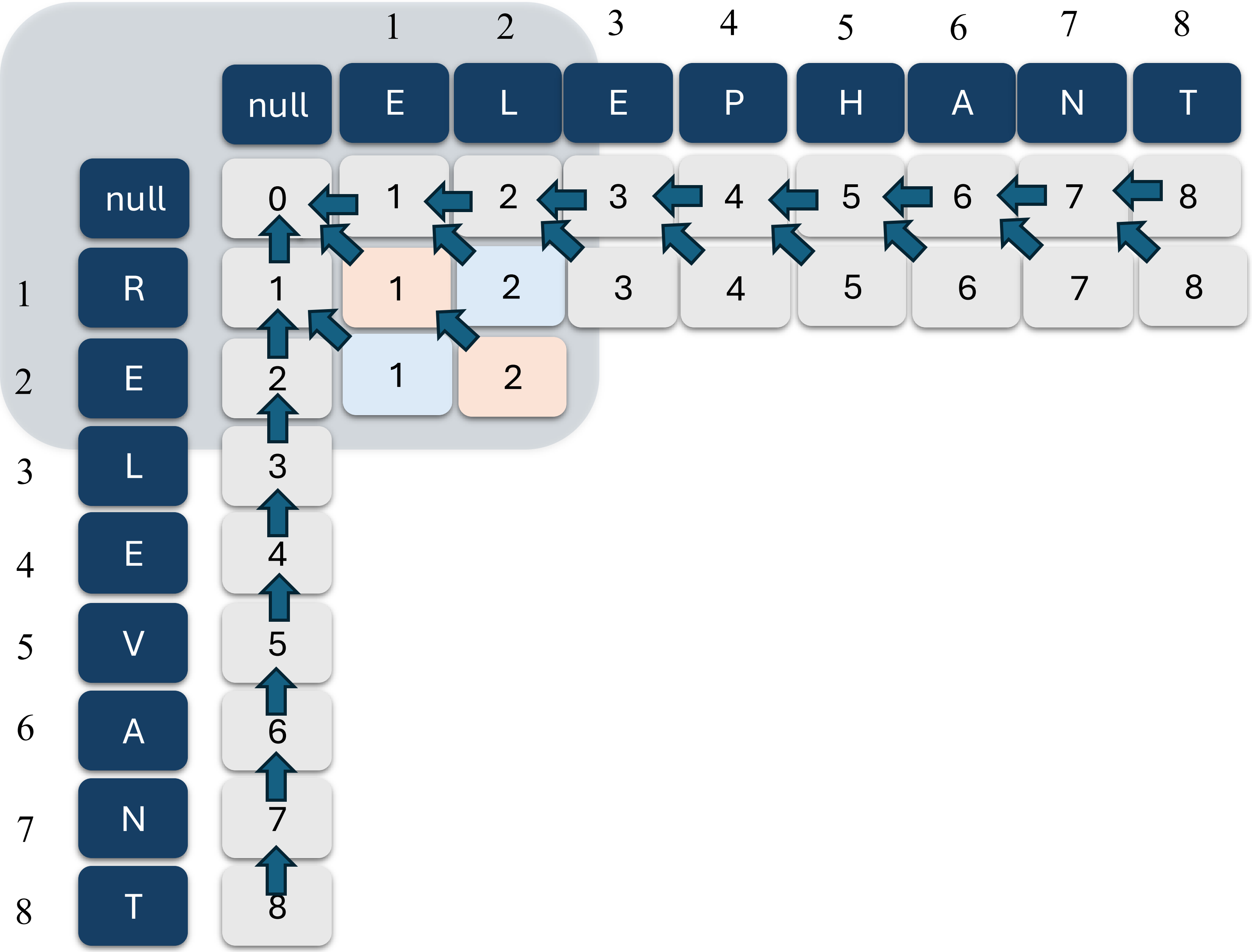
		0	1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1							
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								



			1
		null	E
	null	0	1
1	R	1	1
2	E	2	1

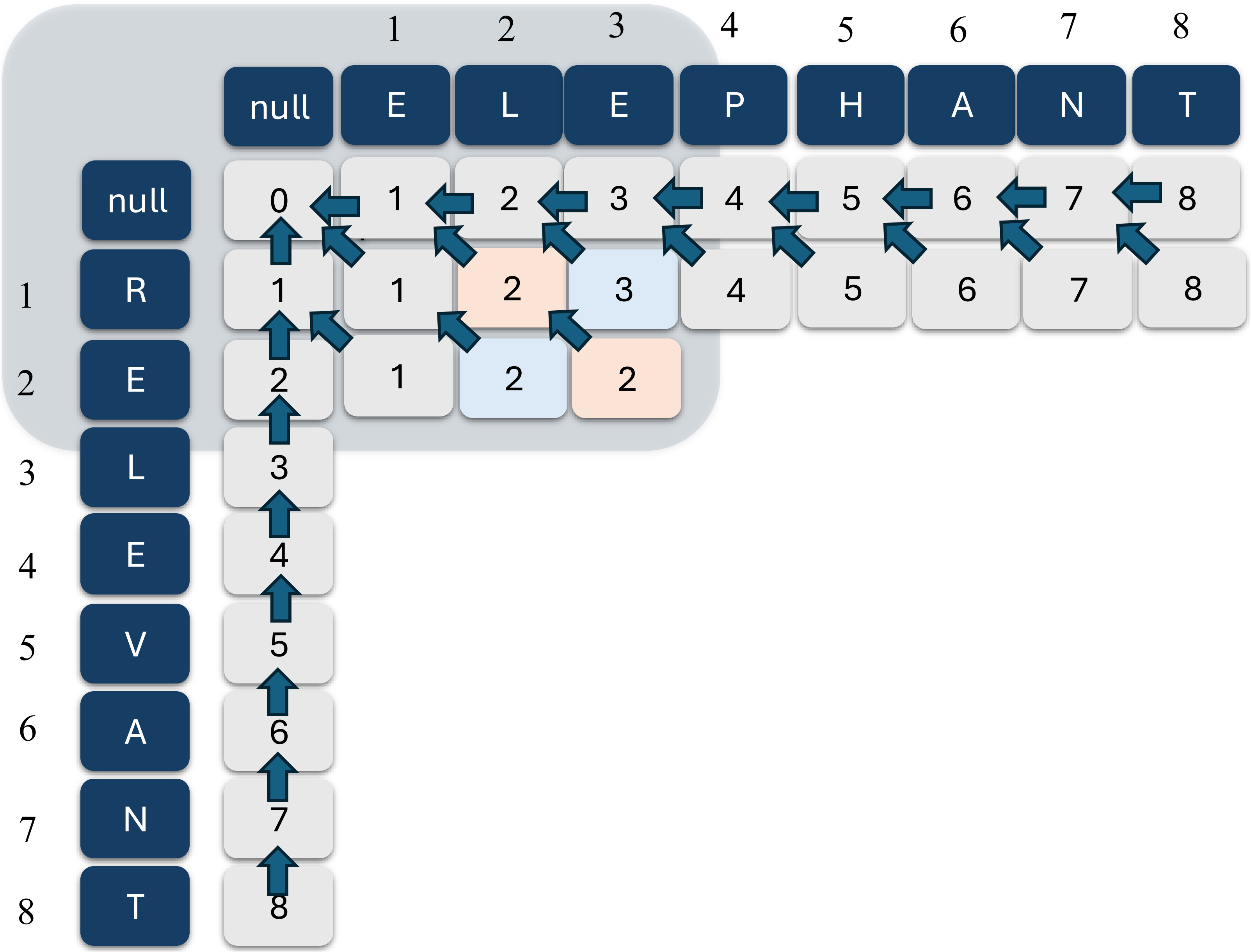
Delete R
Copy E

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



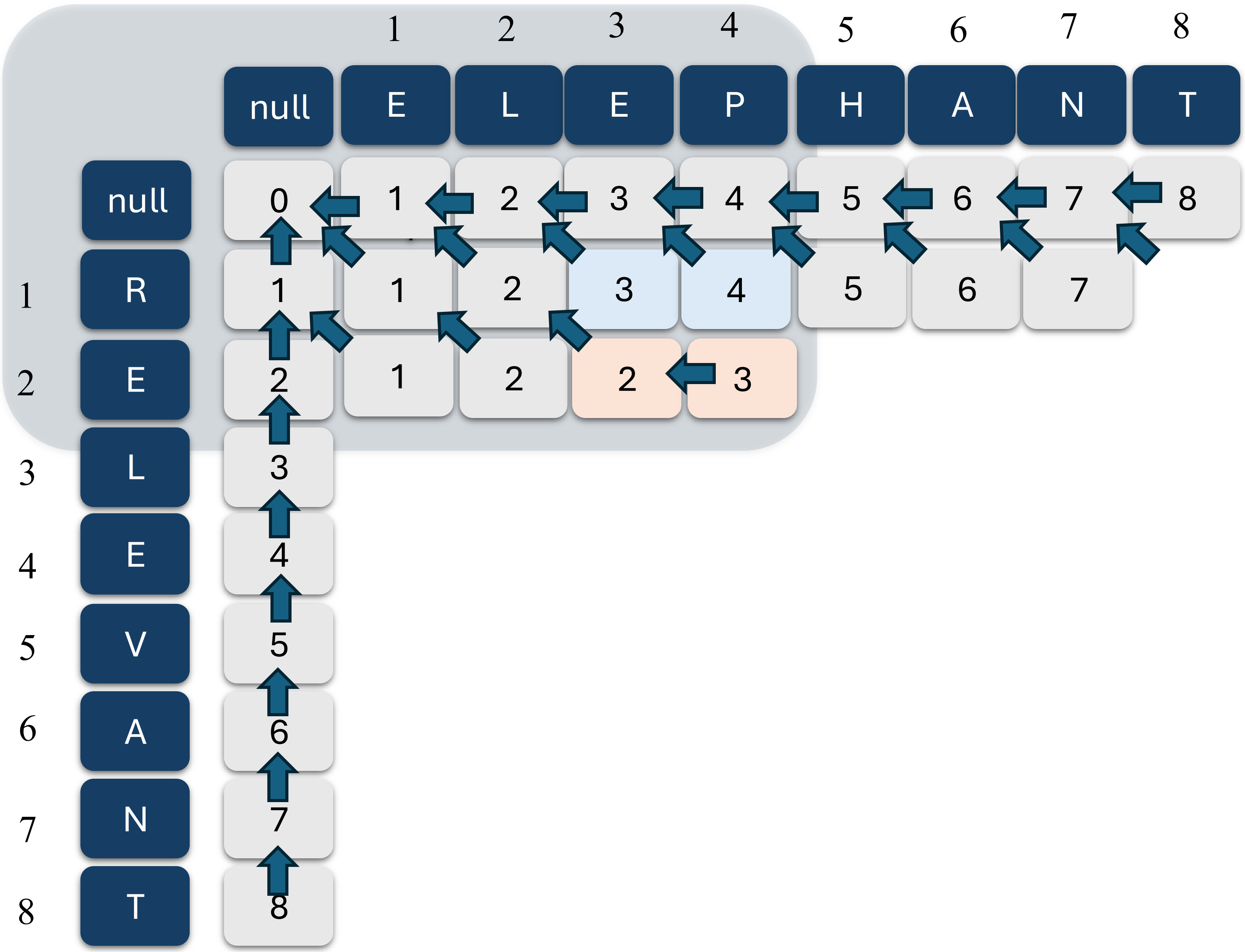
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2						
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min\begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

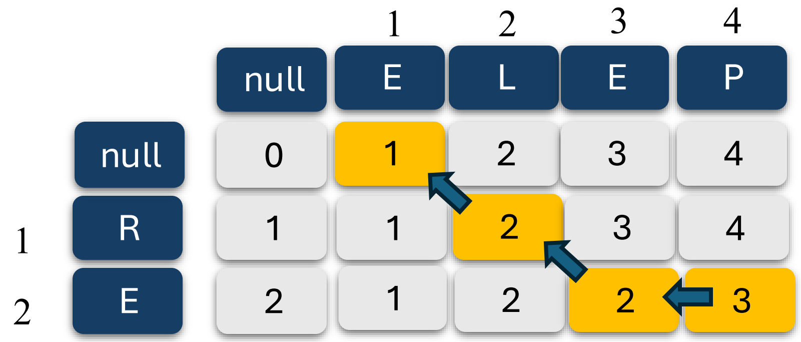


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2					
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



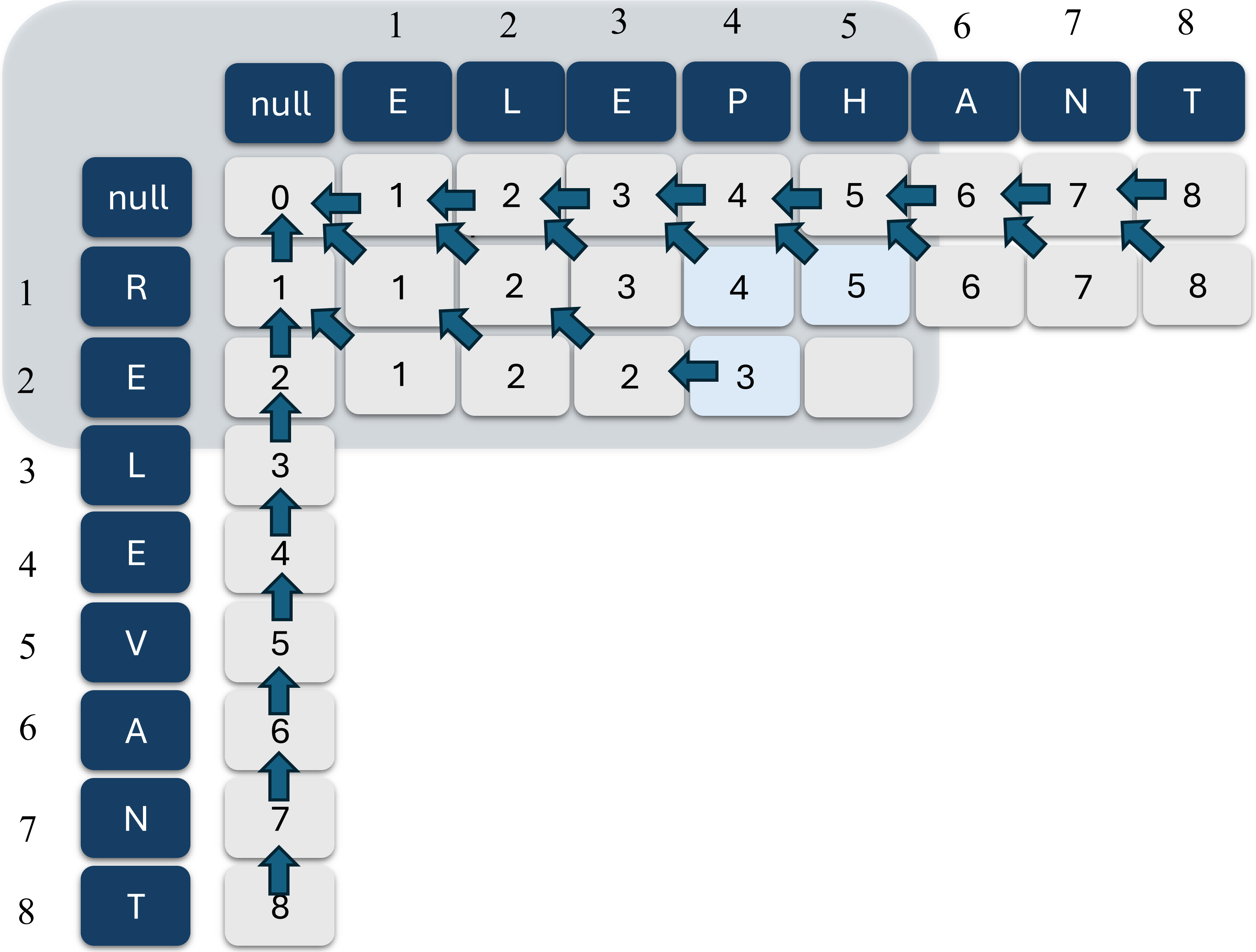
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	
2	E	2	1	2	2	3				
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								



			1	2	3	4
		null	E	L	E	P
	null	0	1	2	3	4
1	R	1	1	2	3	4
2	E	2	1	2	2	3

Insert E
Replace R with L
Copy E
Insert P

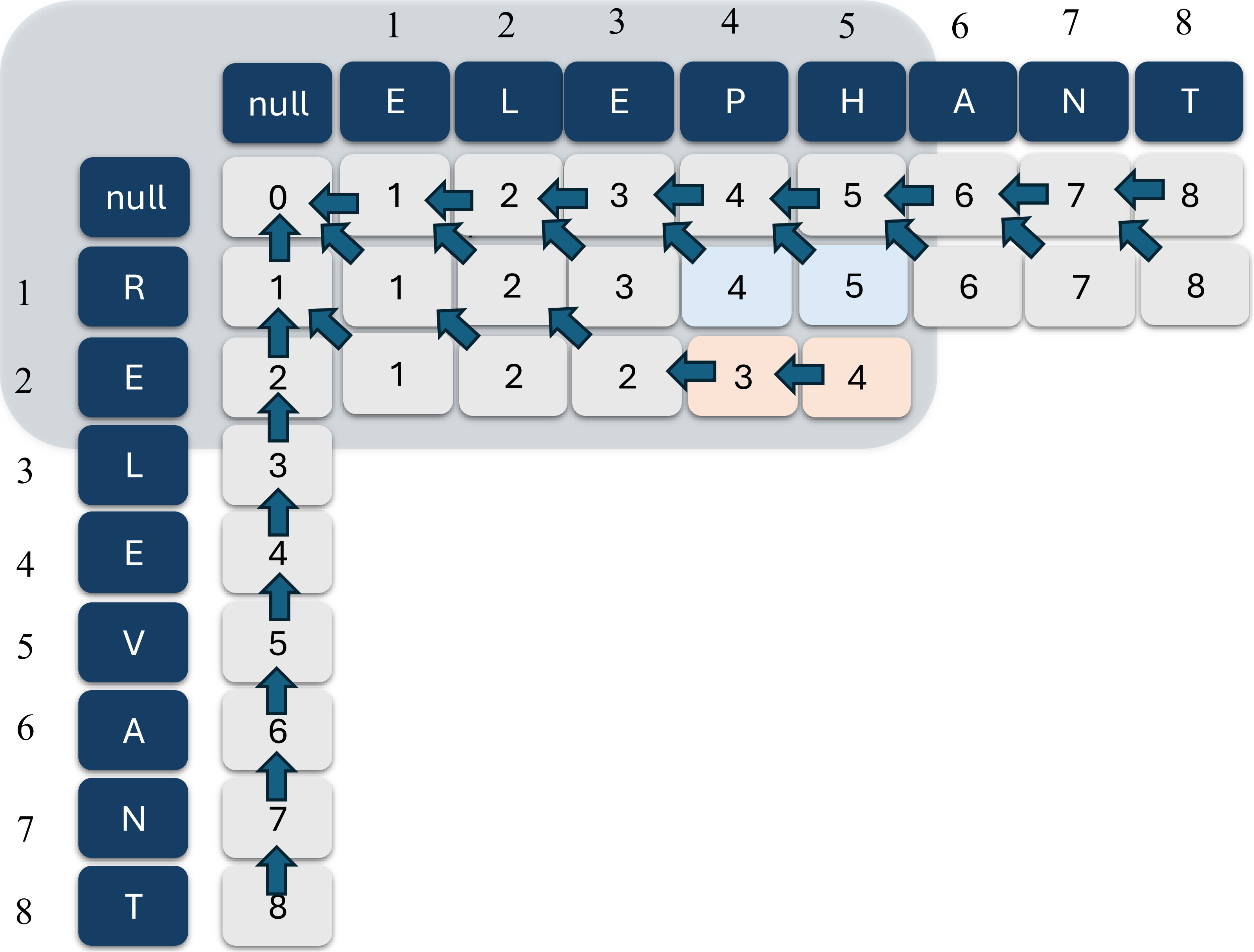
$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
null	0	1	2	3	4	5	6	7	8
1 R	1	1	2	3	4	5	6	7	8
2 E	2	1	2	2	3				
3 L	3								
4 E	4								
5 V	5								
6 A	6								
7 N	7								
8 T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

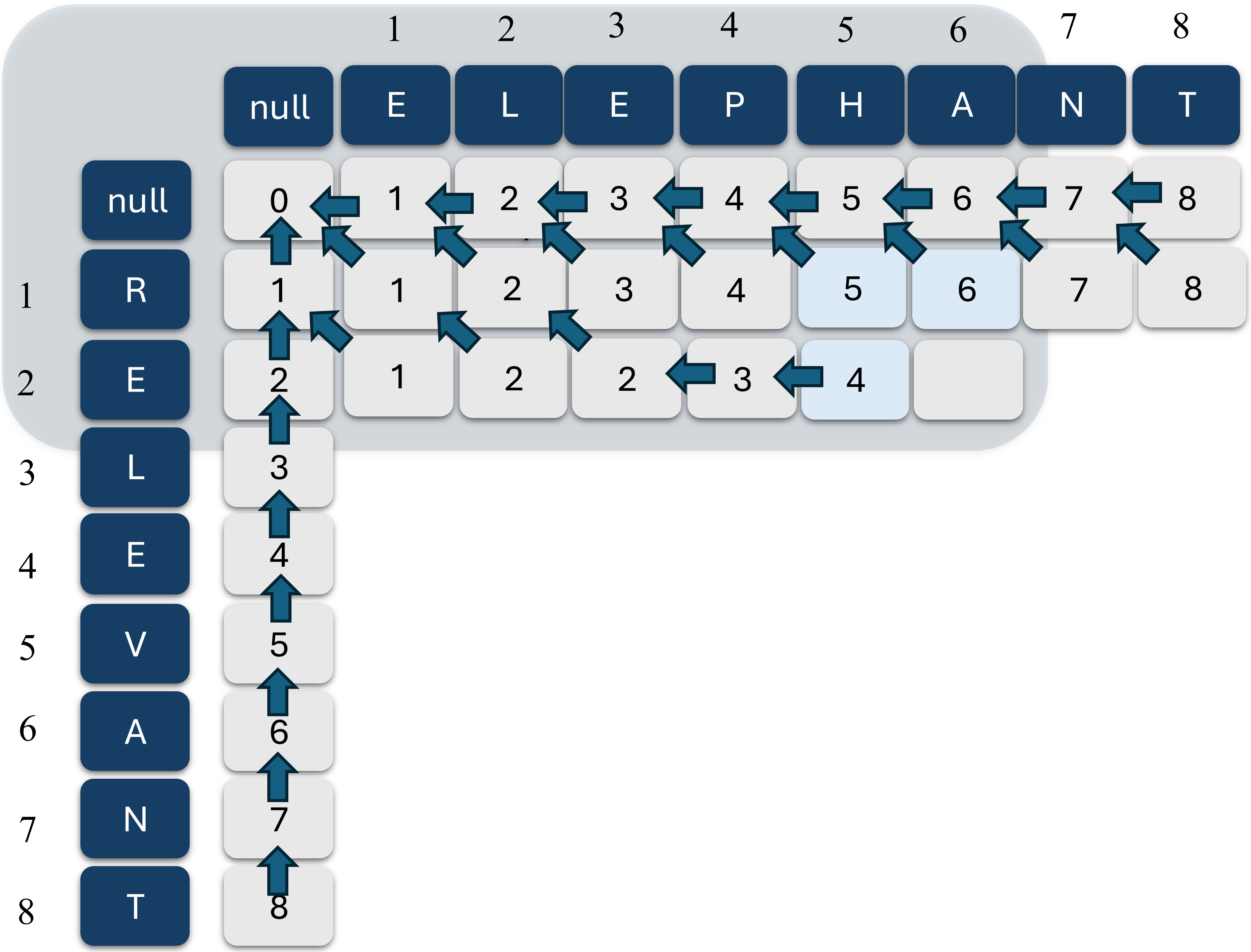
↖ ← ↑



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
null	0	1	2	3	4	5	6	7	8
1 R	1	1	2	3	4	5	6	7	8
2 E	2	1	2	2	3	4			
3 L	3								
4 E	4								
5 V	5								
6 A	6								
7 N	7								
8 T	8								

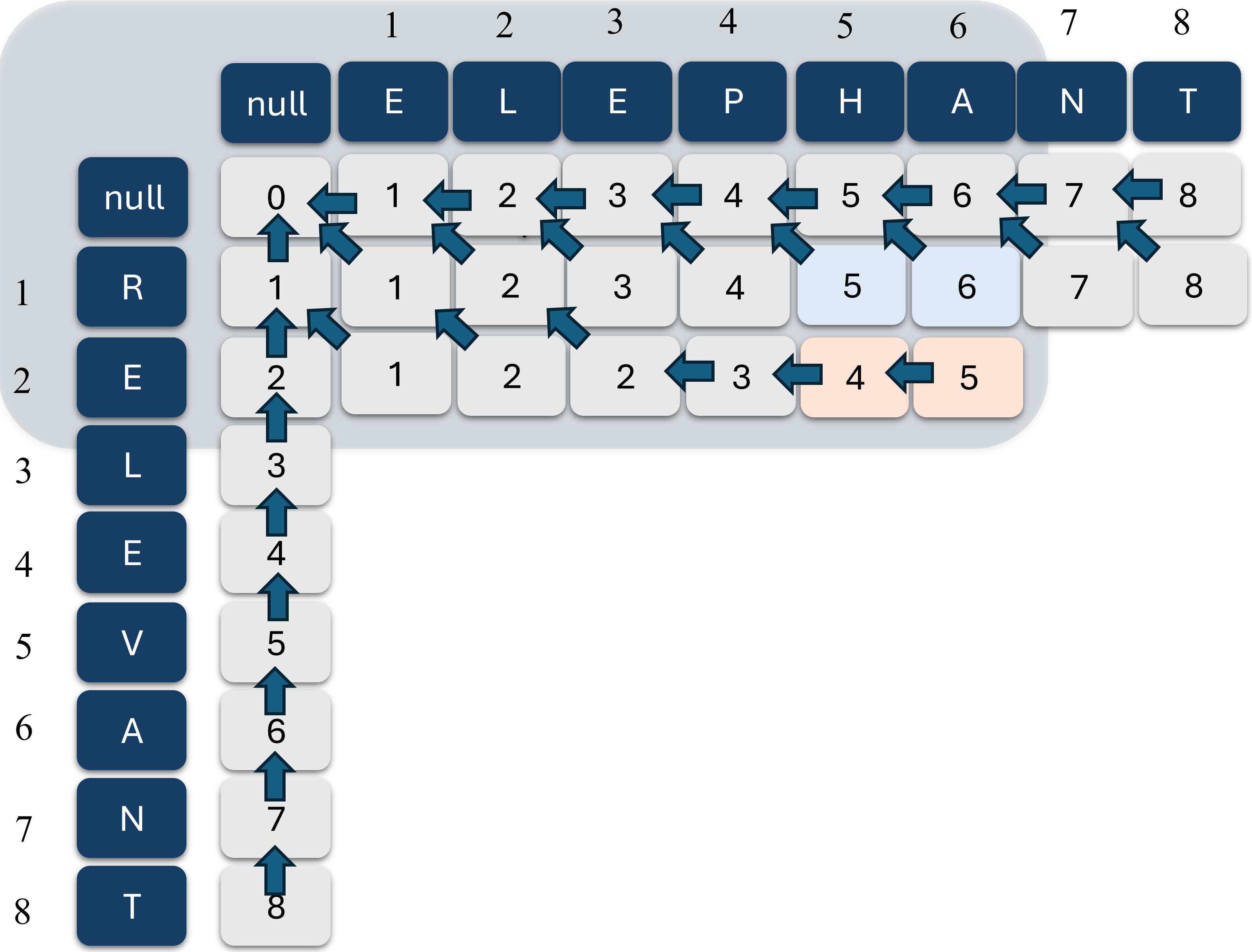
$$E(i, j) = min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4			
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

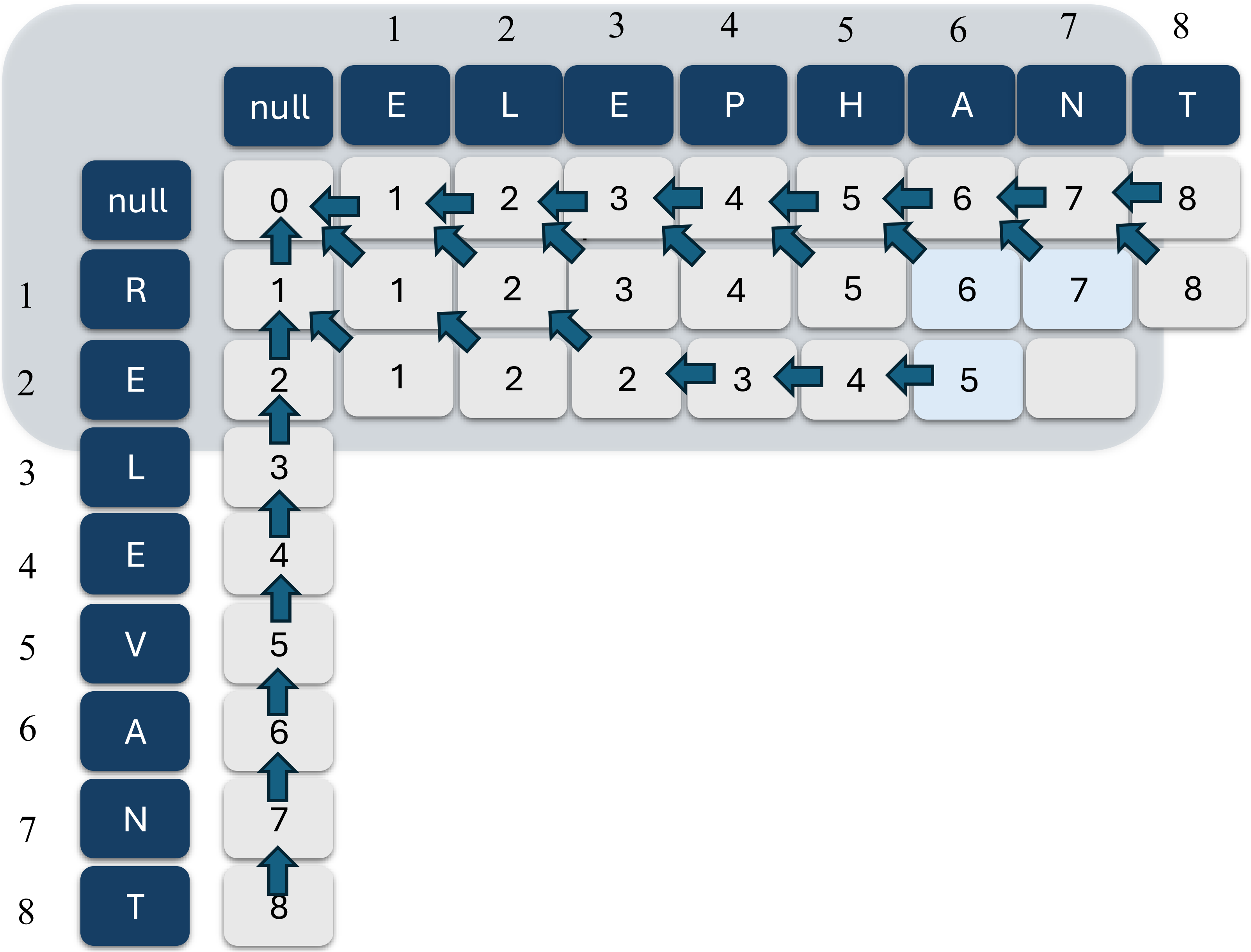
$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	←1	←2	←3	←4	←5	←6	←7	←8
1	R	↑1	↖1	↖2	↖3	↖4	↖5	↖6	↖7	↖8
2	E	↑2	↖1	↖2	↖2	←3	←4	←5		
3	L	↑3								
4	E	↑4								
5	V	↑5								
6	A	↑6								
7	N	↑7								
8	T	↑8								

↖ ← ↑

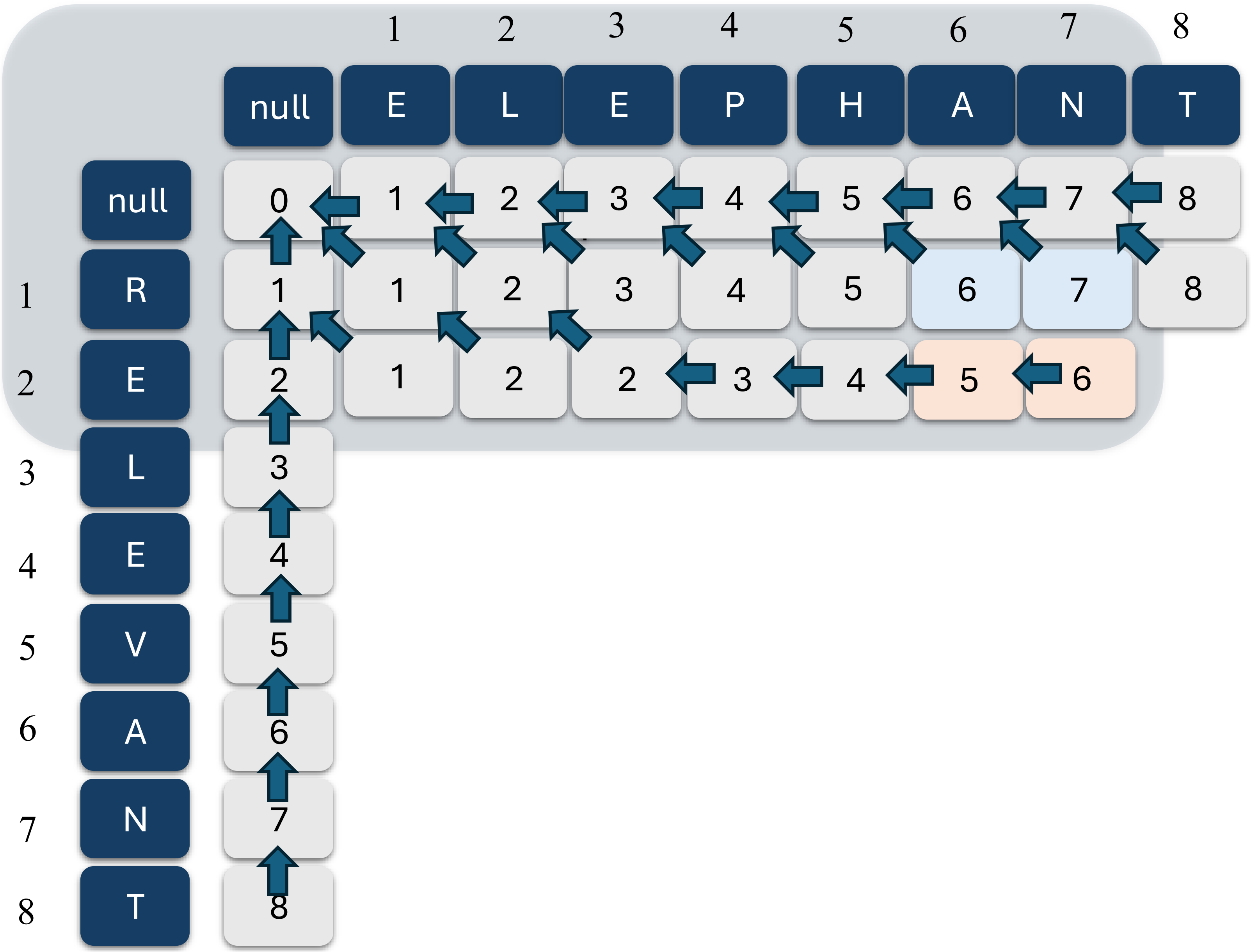
$$E(i,j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i,j) + E(i-1, j-1) \end{cases}$$



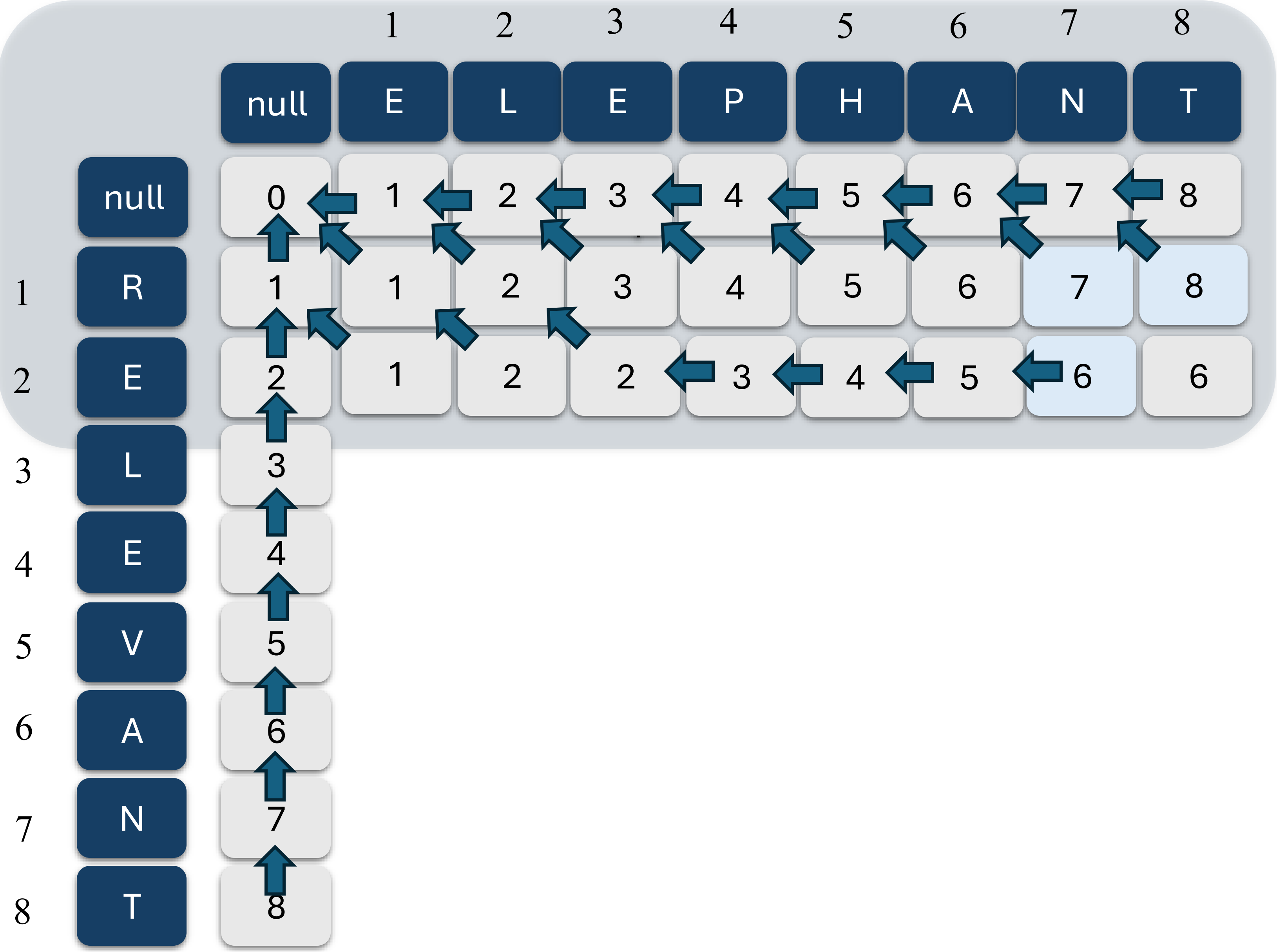
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5		
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

↖ ← ↑

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

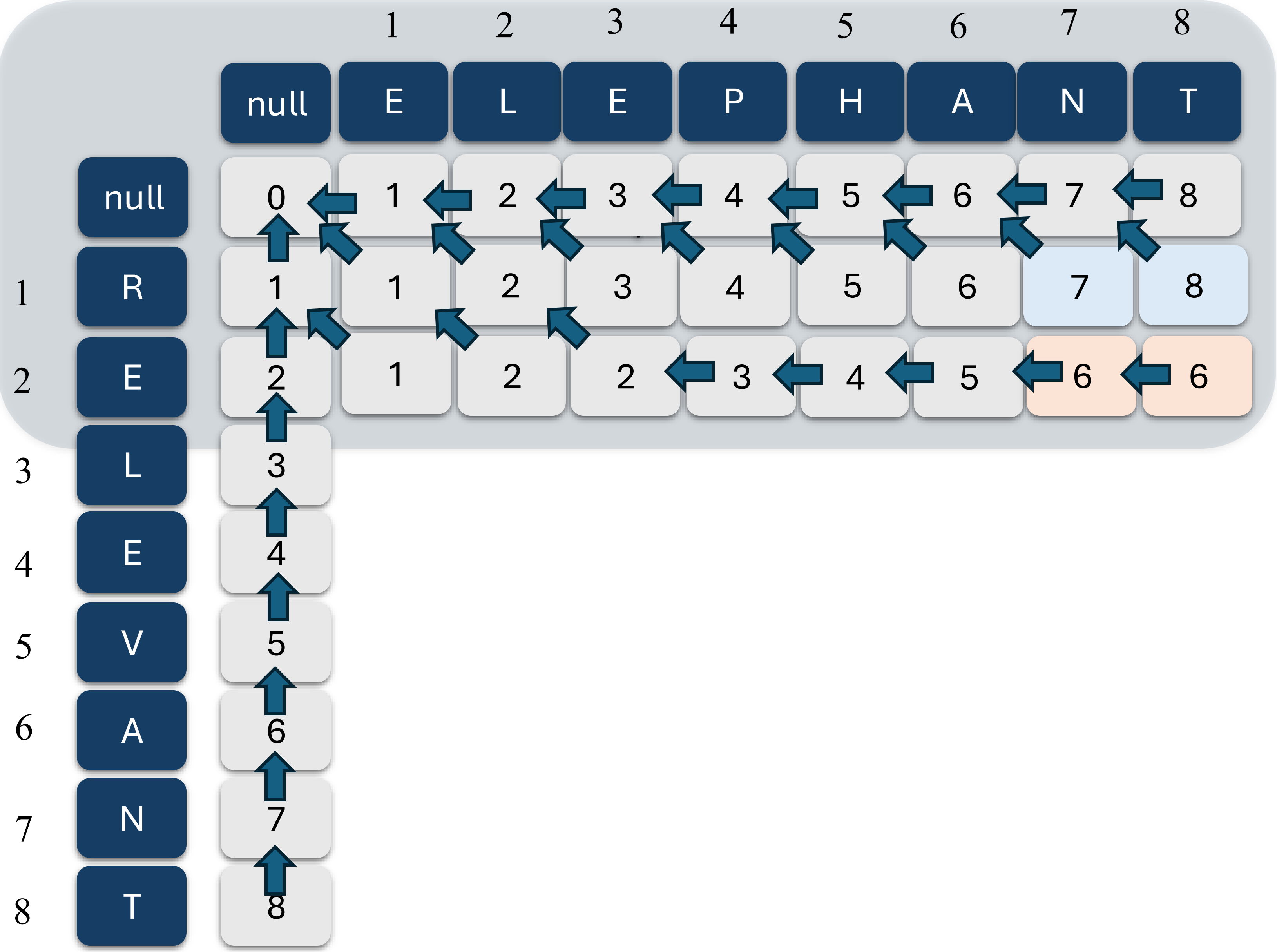


↖ ← ↑



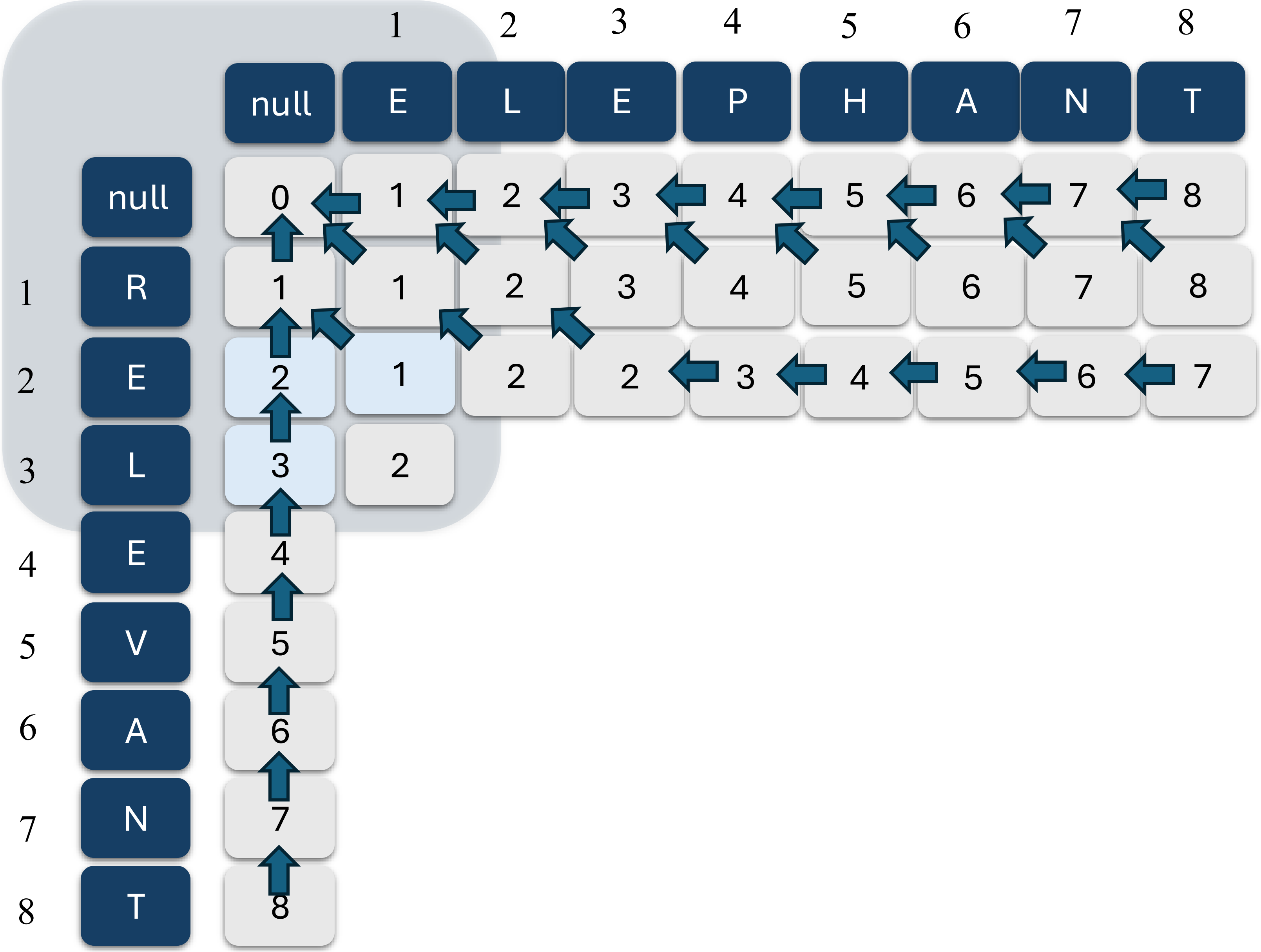
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	6
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



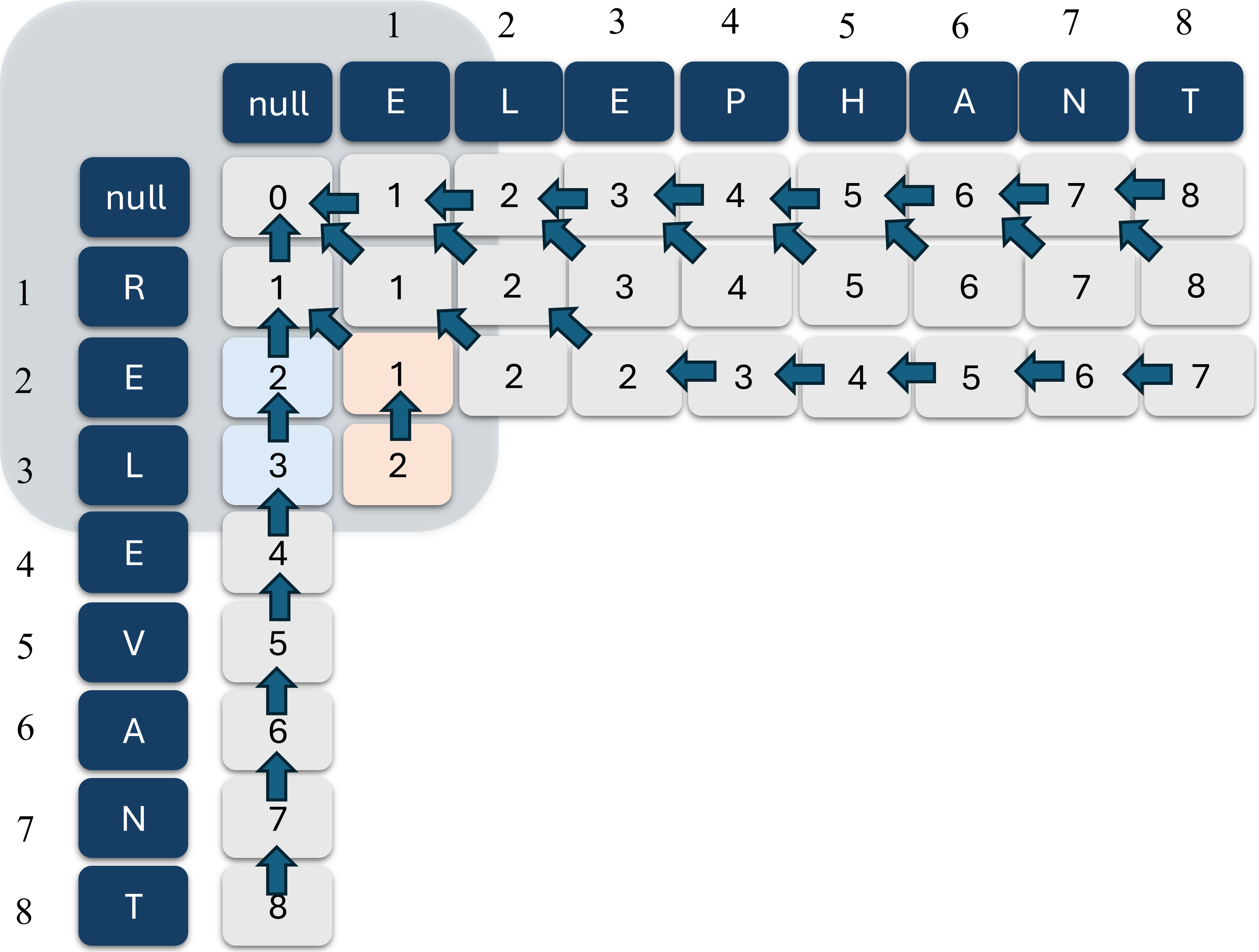
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	6
3	L	3								
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i,j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i,j) + E(i-1, j-1) \end{cases}$$

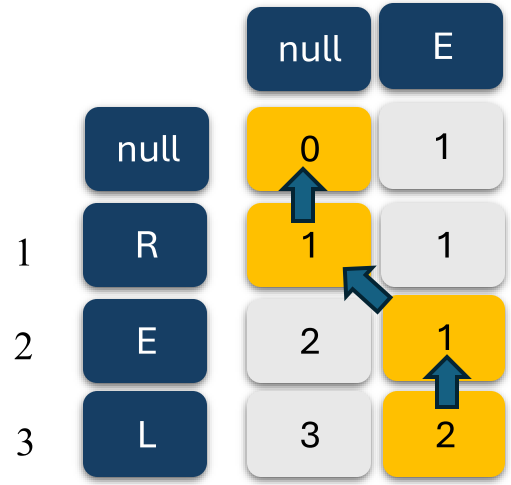


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2							
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



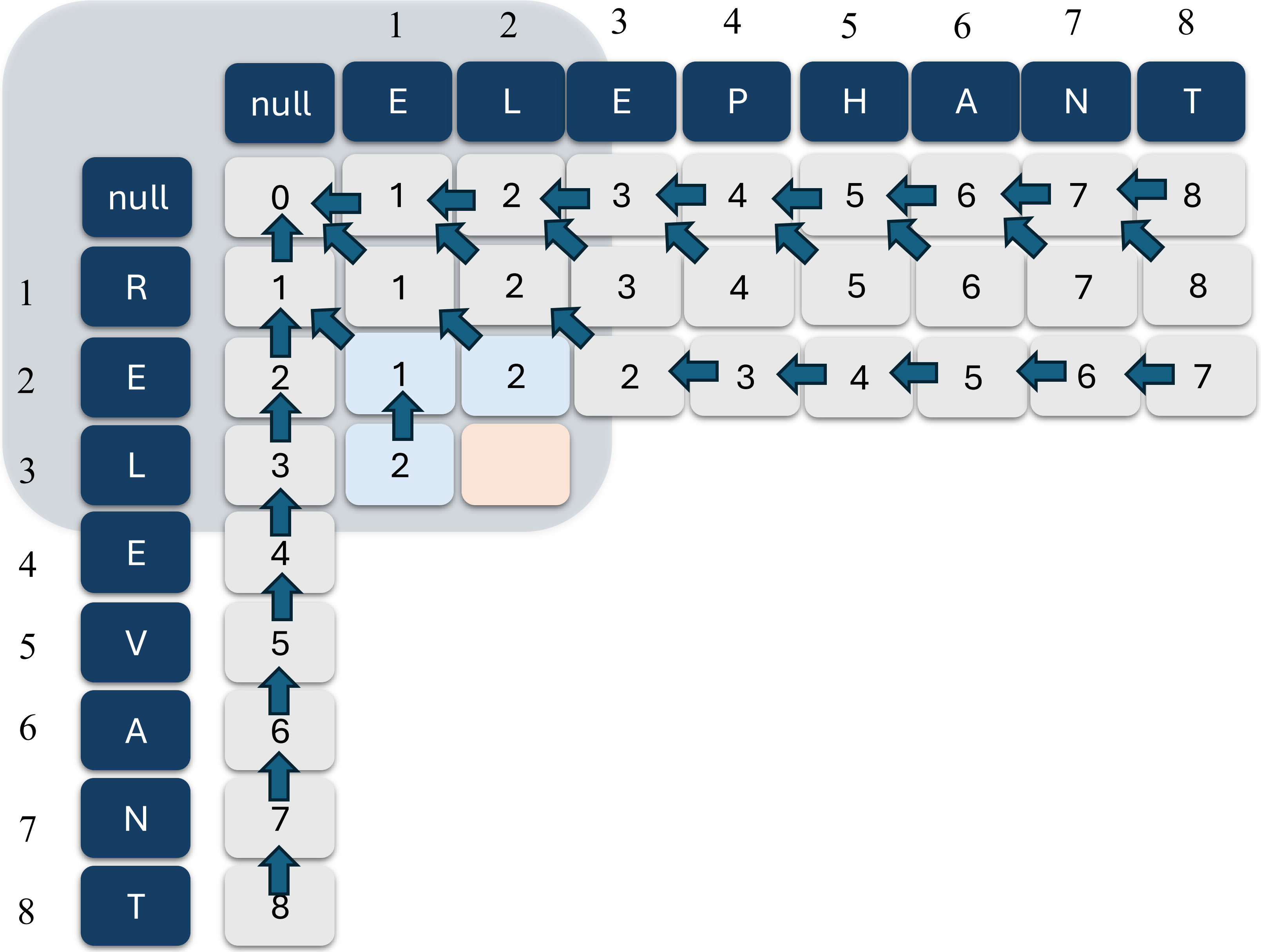
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2							
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								



		null	E
	null	0	1
1	R	1	1
2	E	2	1
3	L	3	2

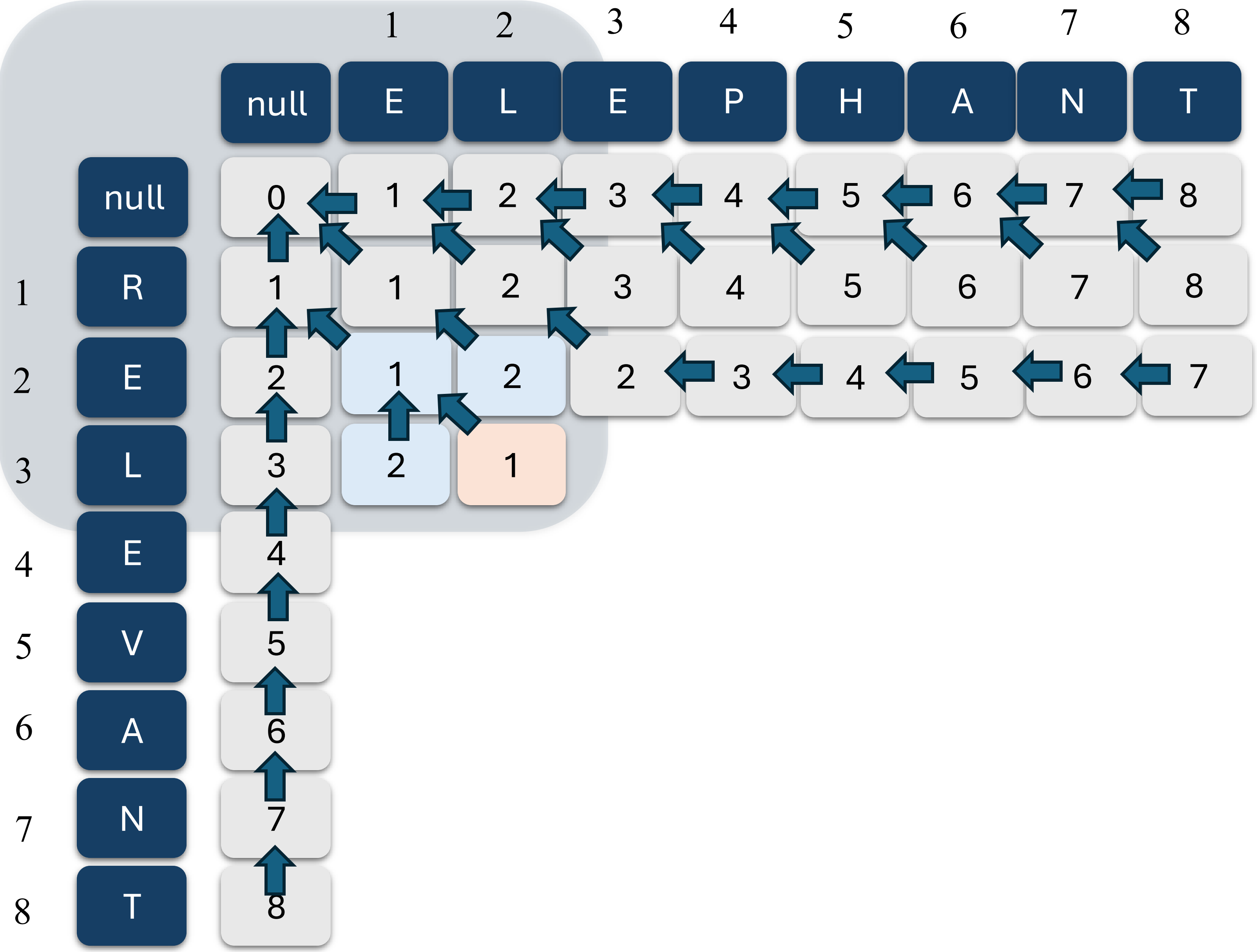
Delete R
Copy E
Delete L

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

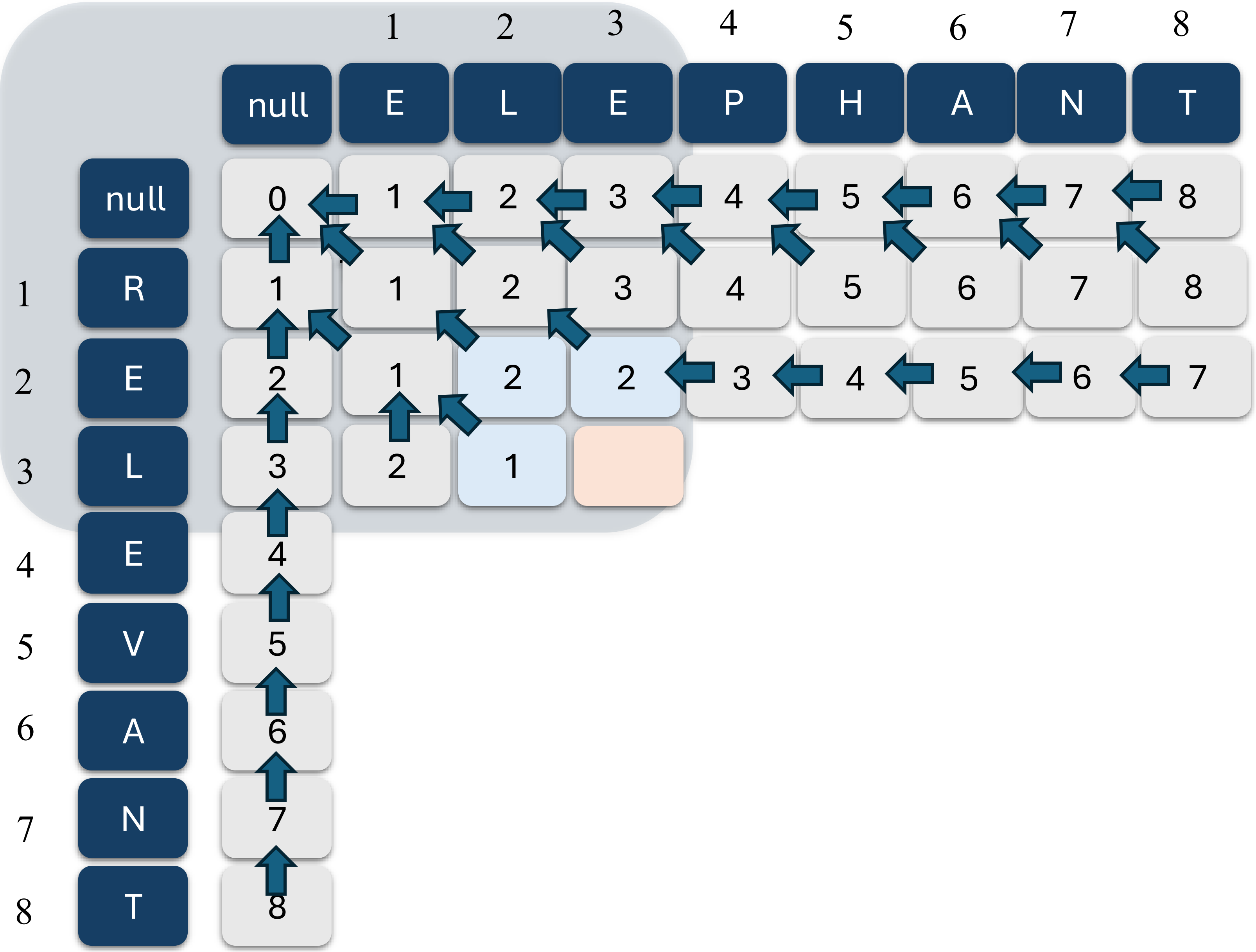


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2							
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



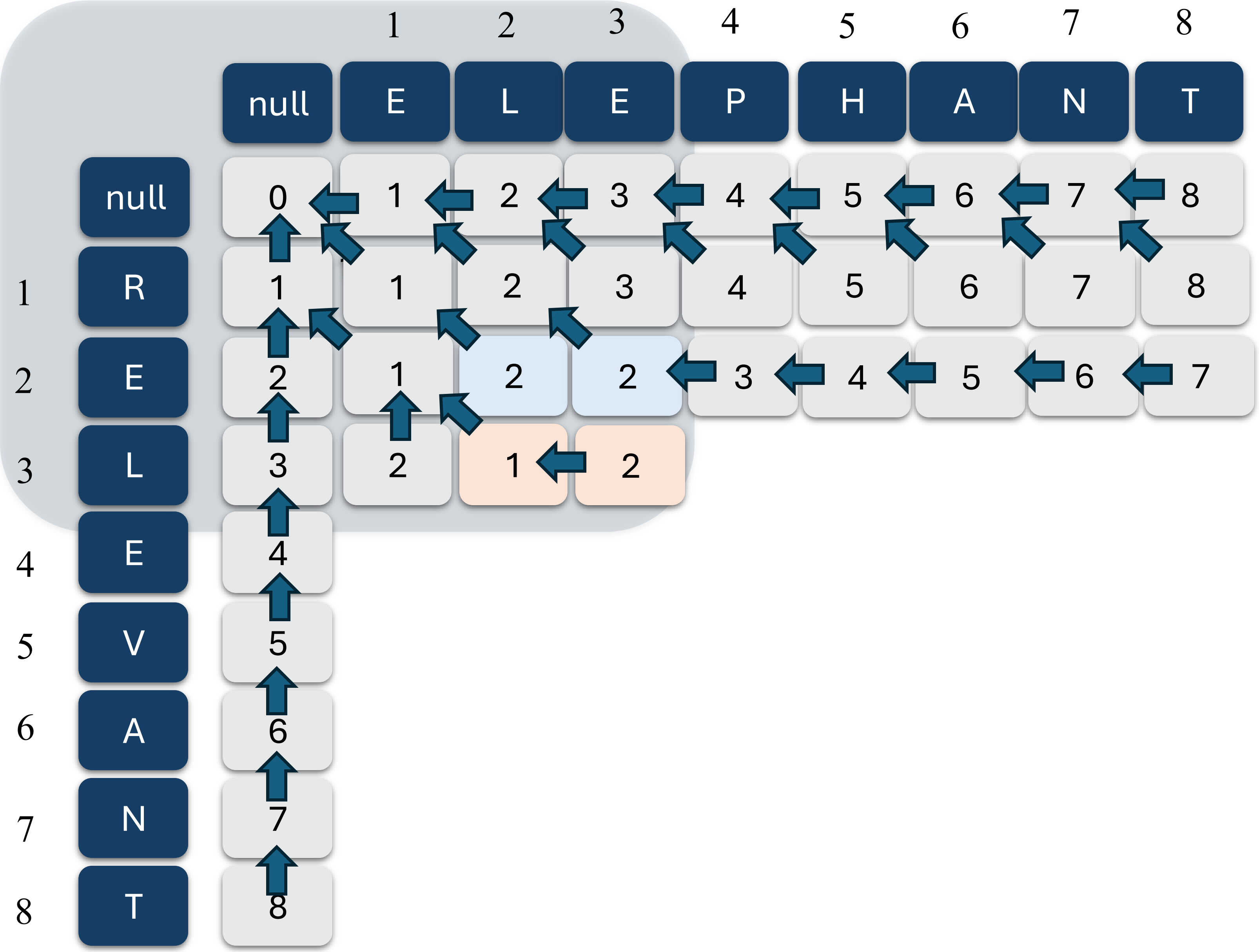
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1						
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

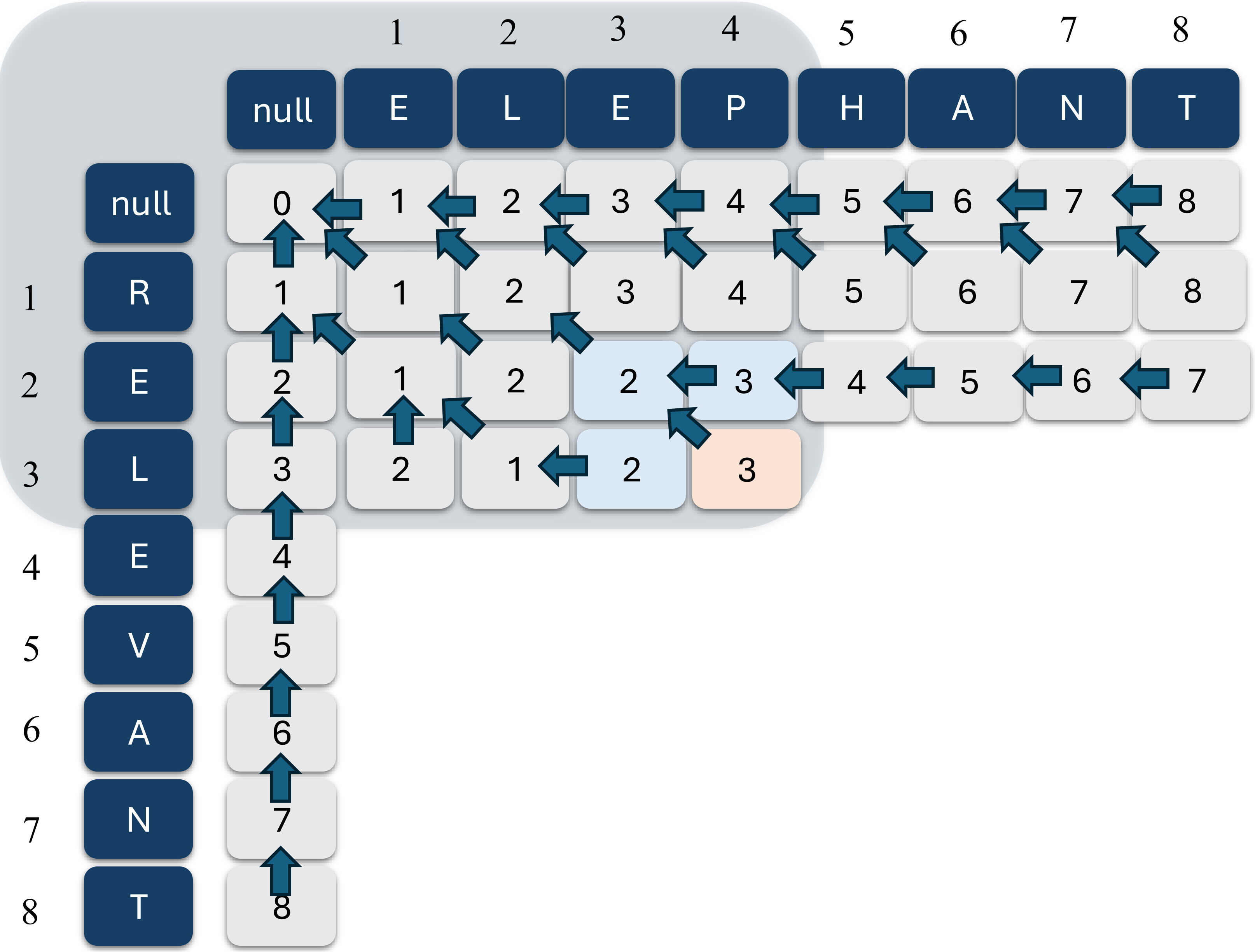
$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

↖ ← ↑



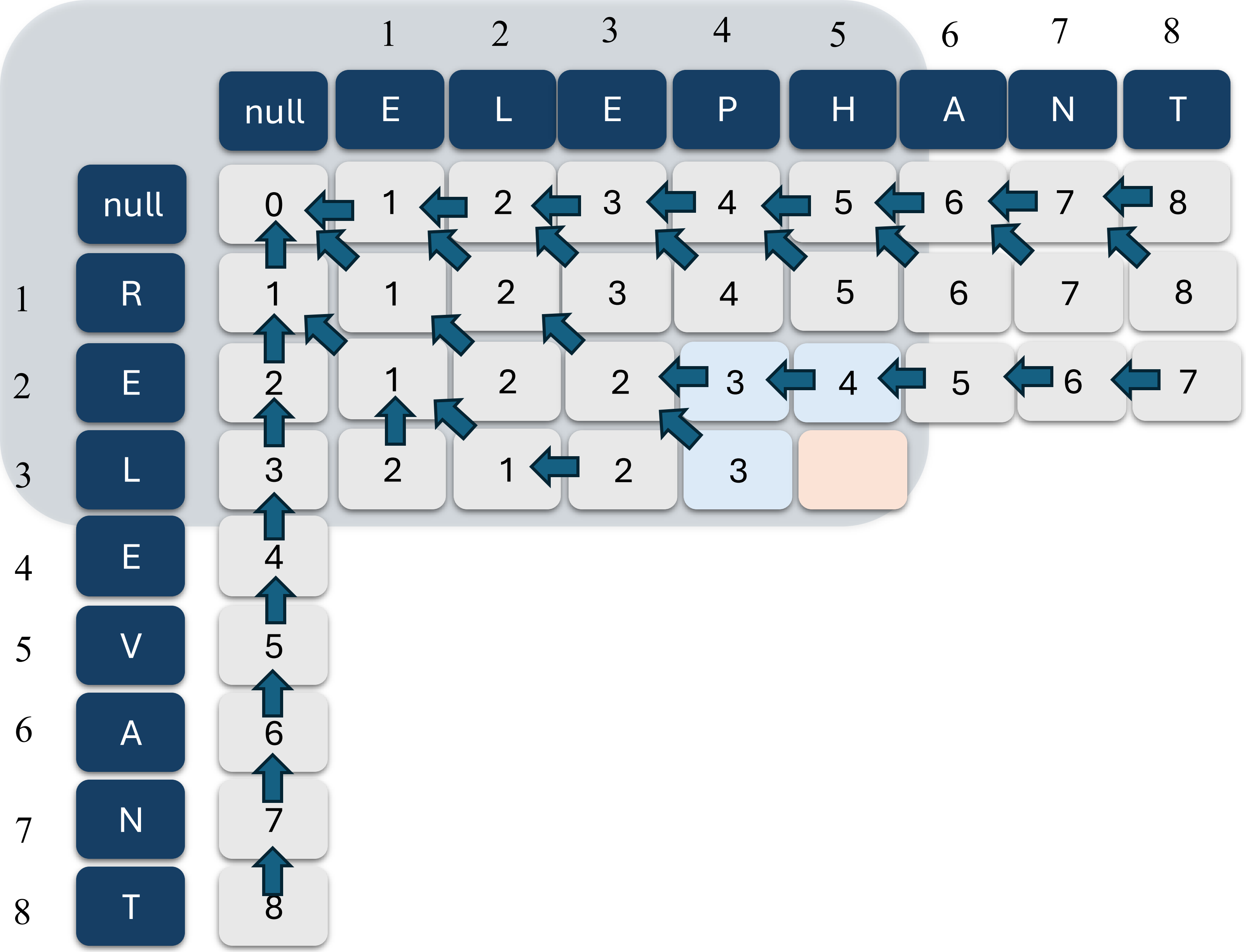
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2					
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3				
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

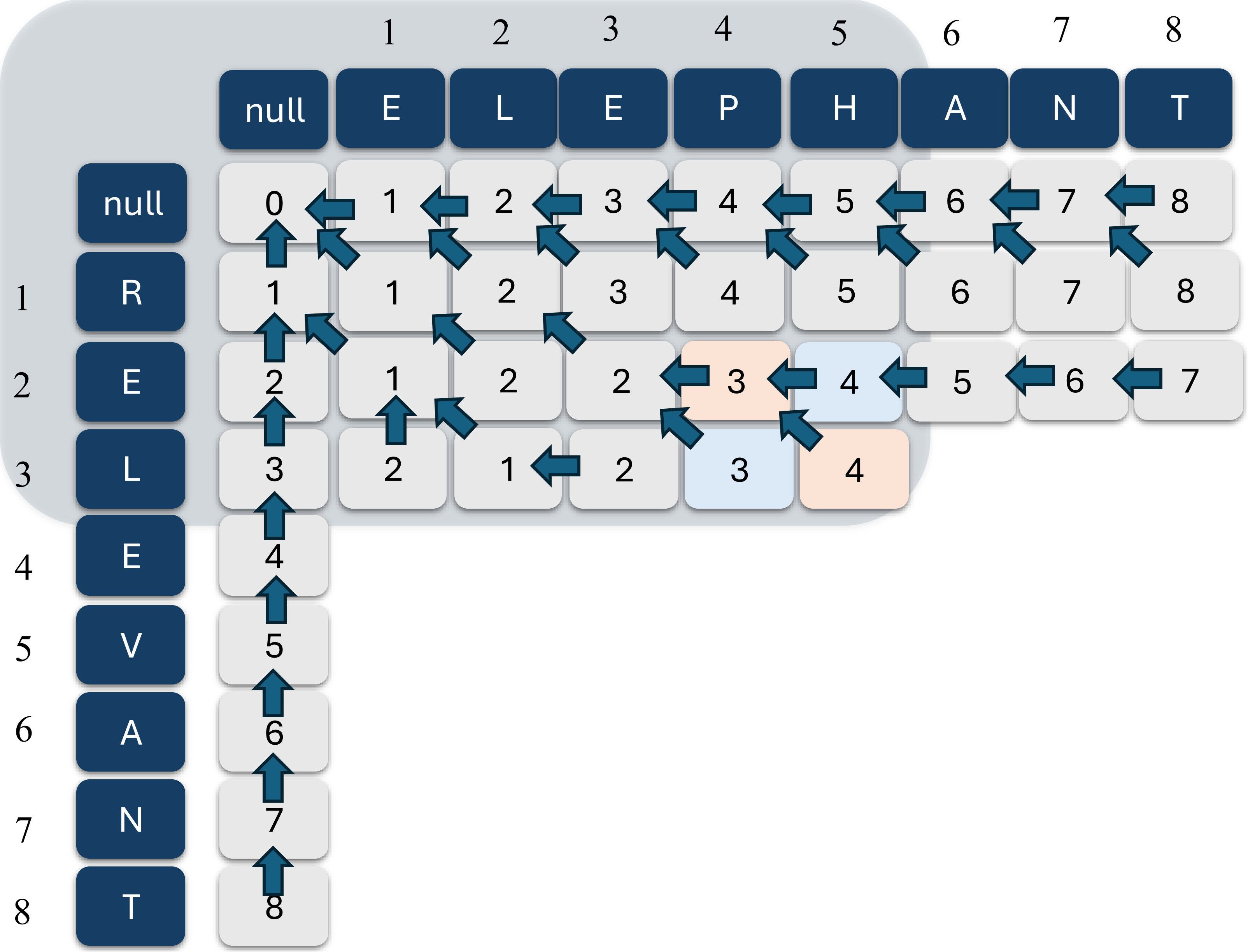
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3				
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

↖ ← ↑

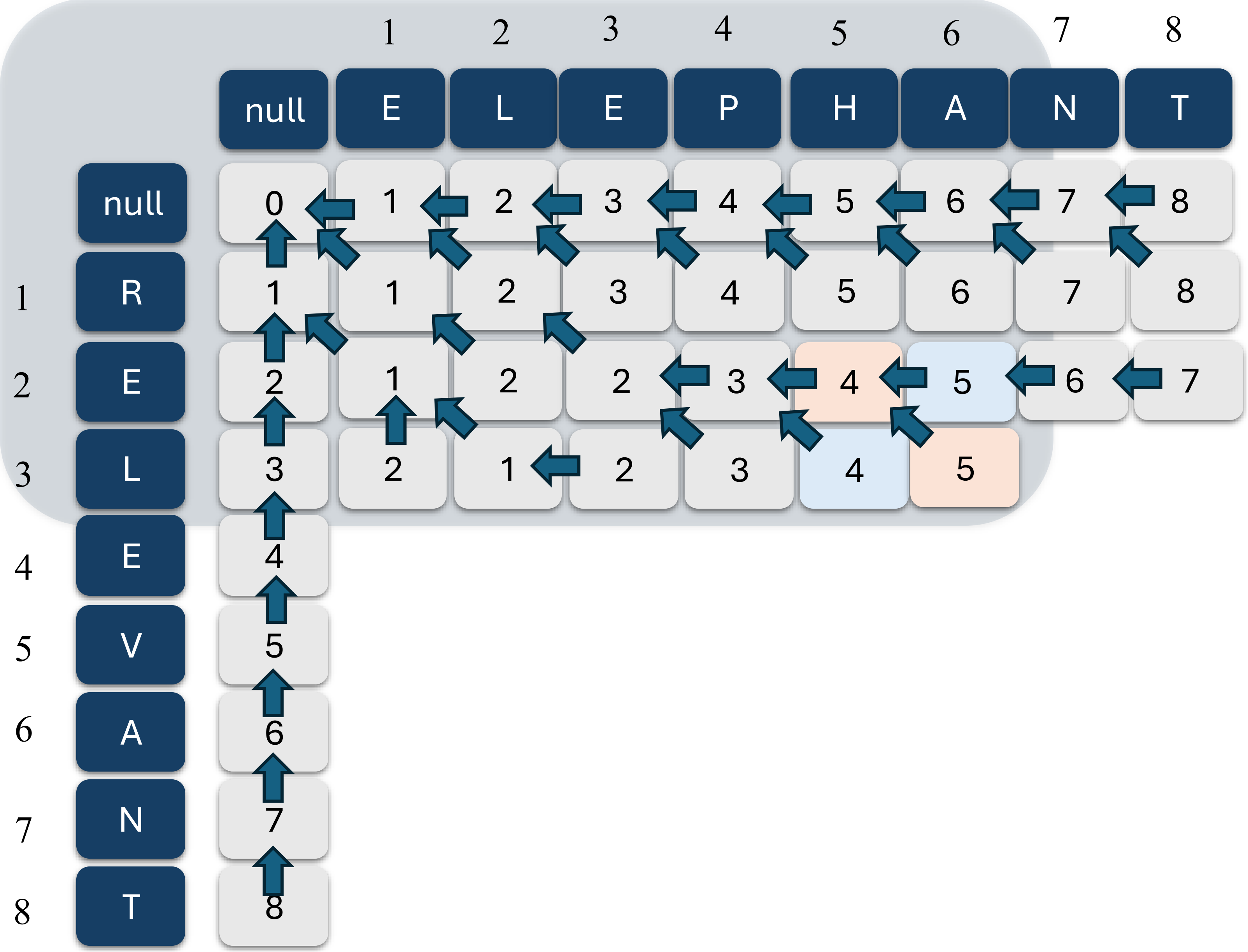
$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



		null	E	L	E	P	H	A	N	T
			1	2	3	4	5	6	7	8
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4			
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

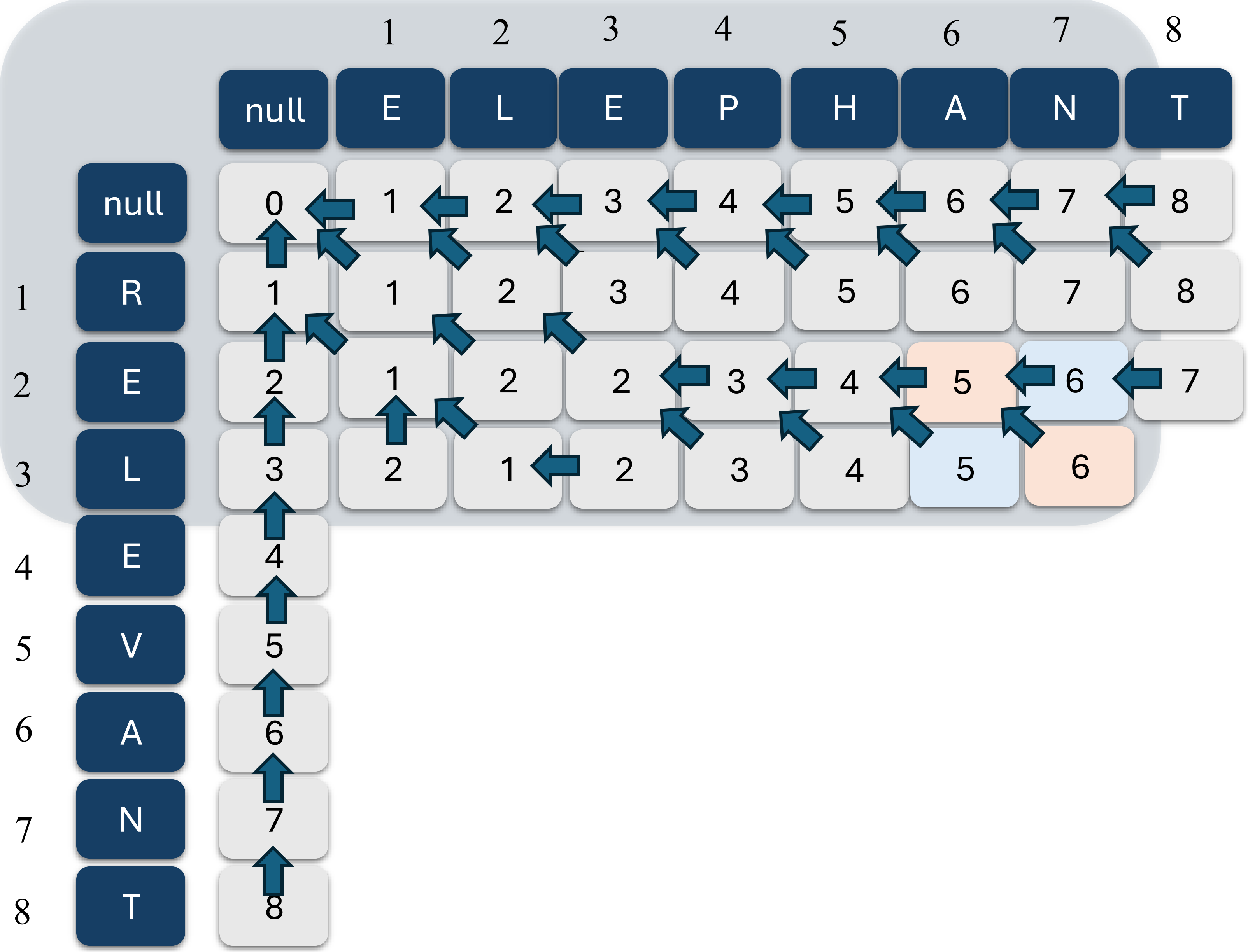
$$E(i,j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i,j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5		
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

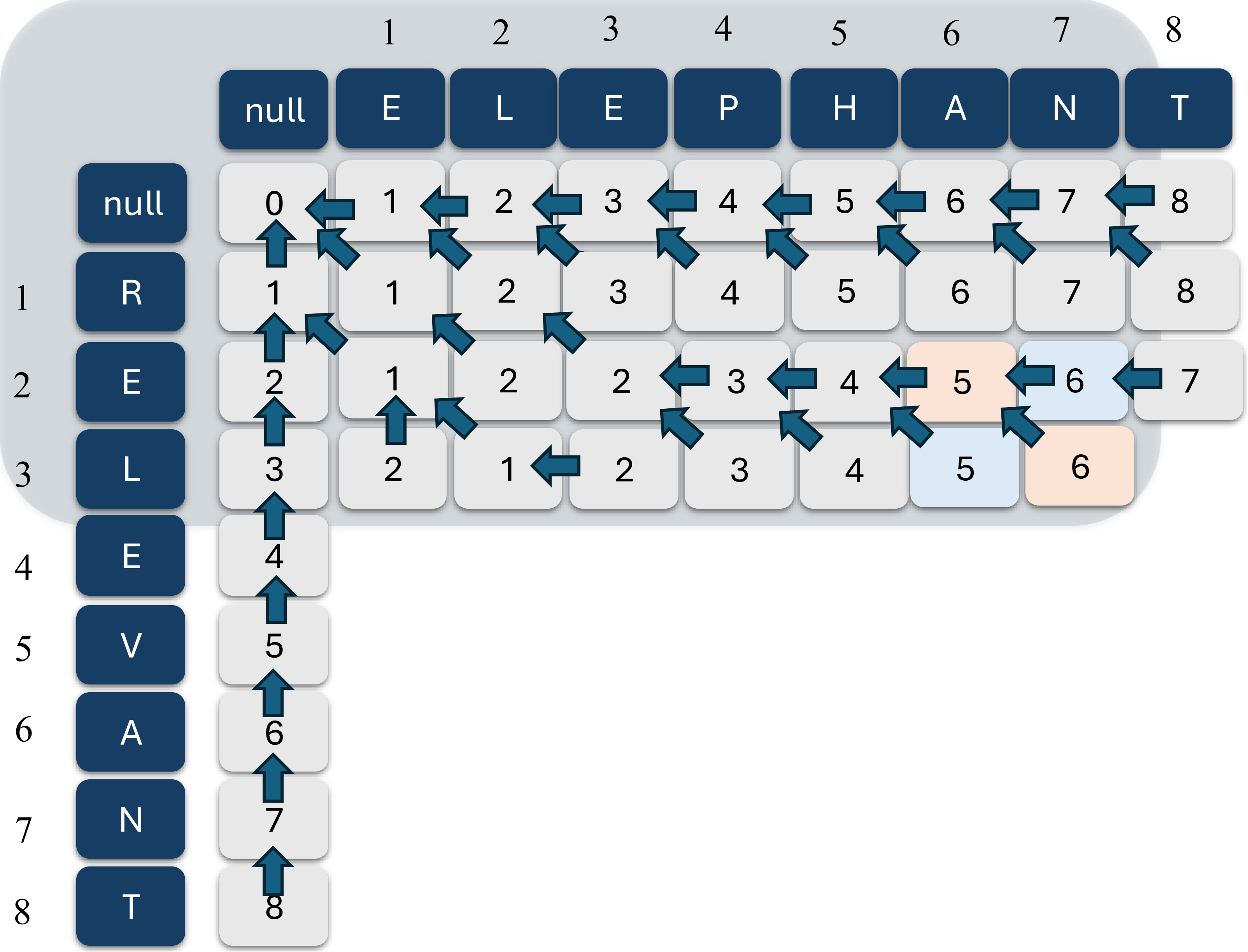
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

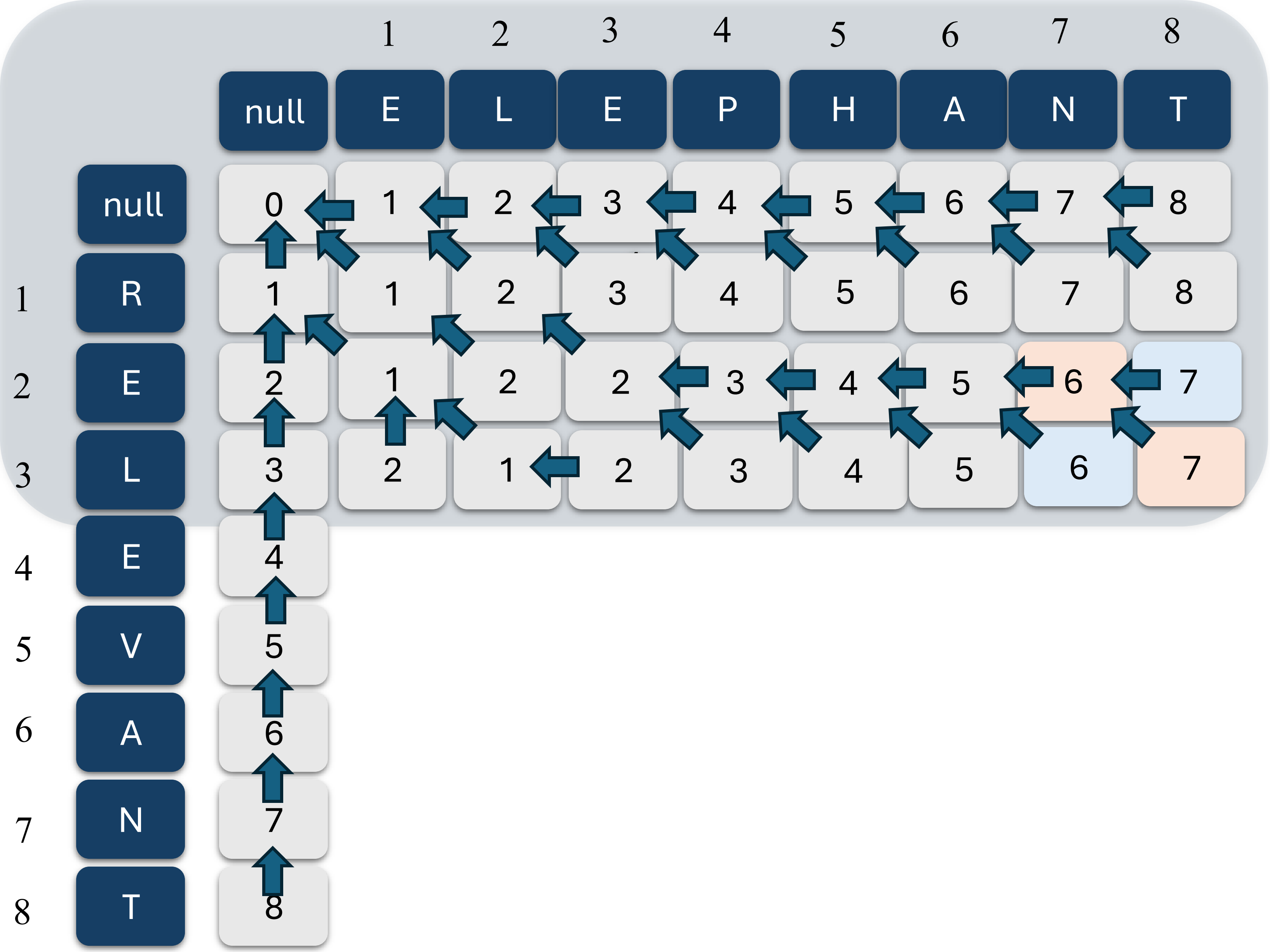
↖ ← ↑

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



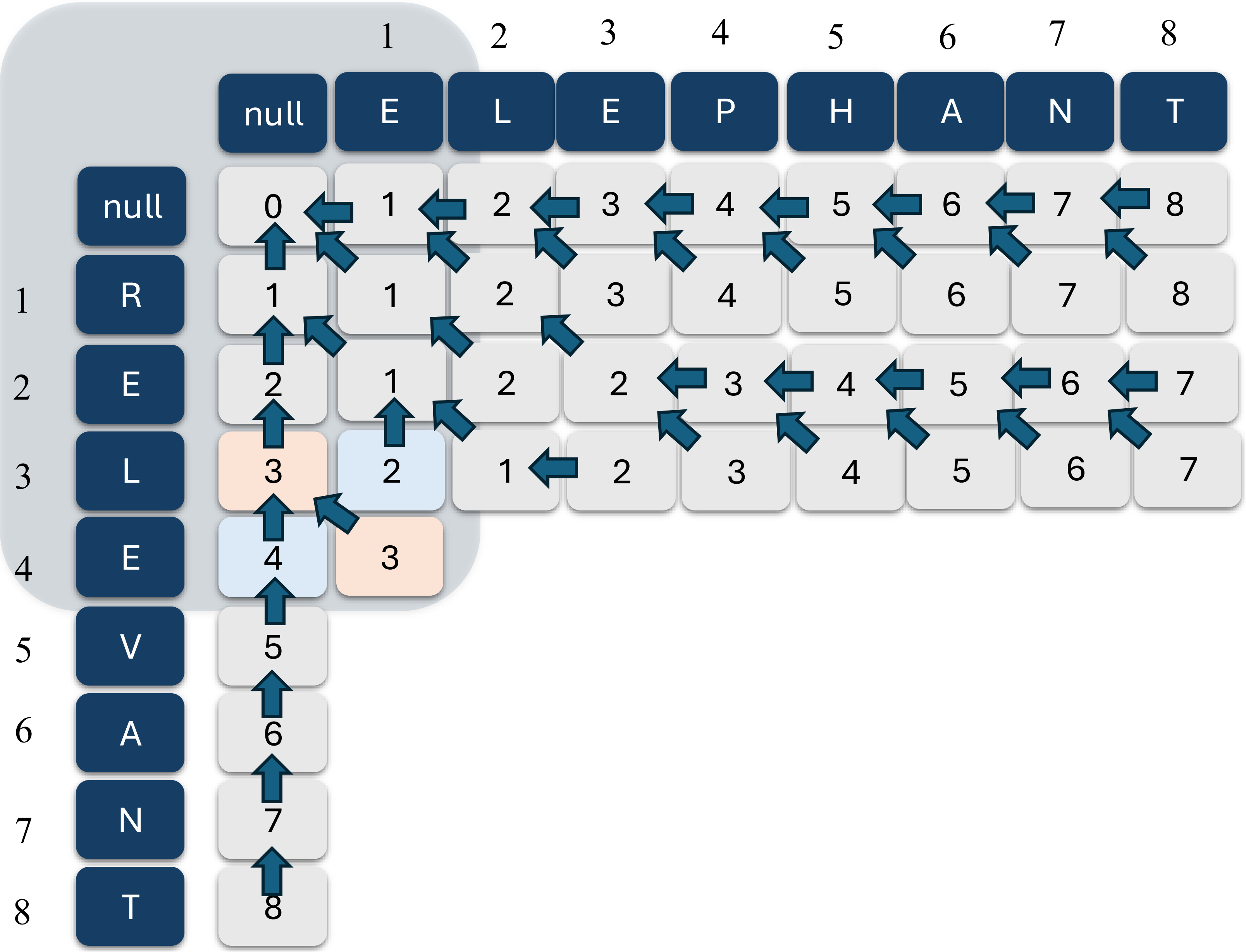
		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



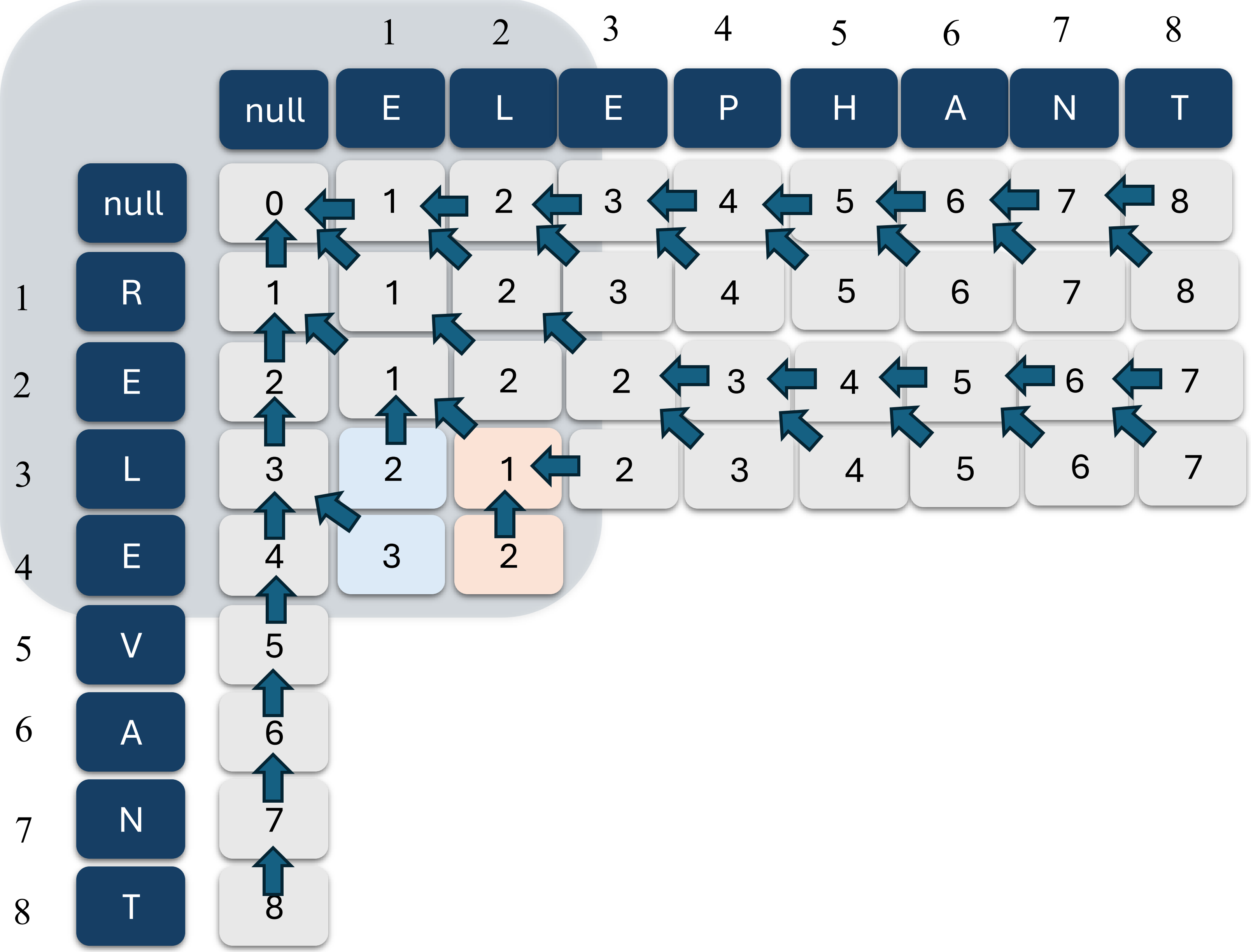
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4								
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3							
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

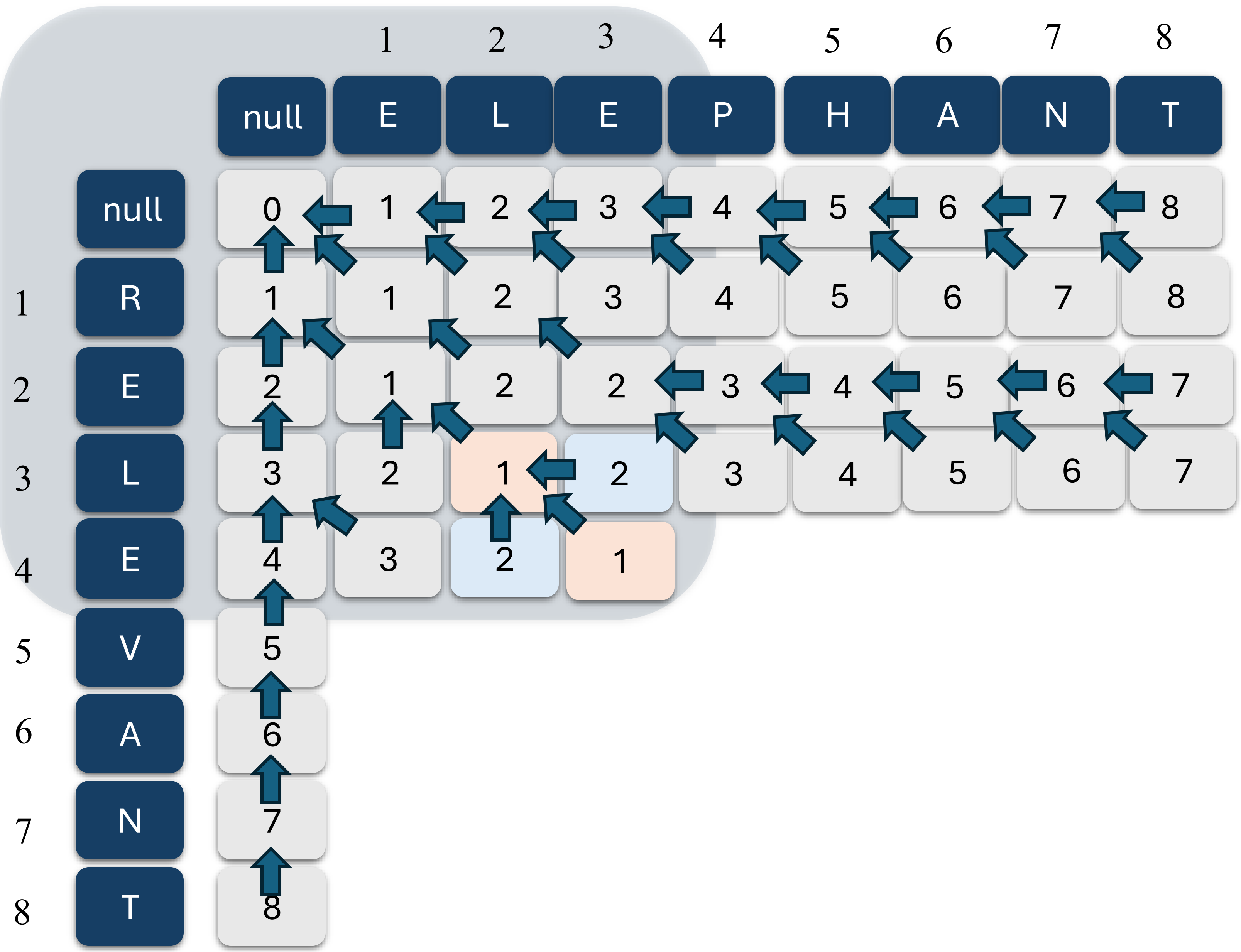


		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2						
5	V	5								
6	A	6								
7	N	7								
8	T	8								

↖ ⬅ ⬆

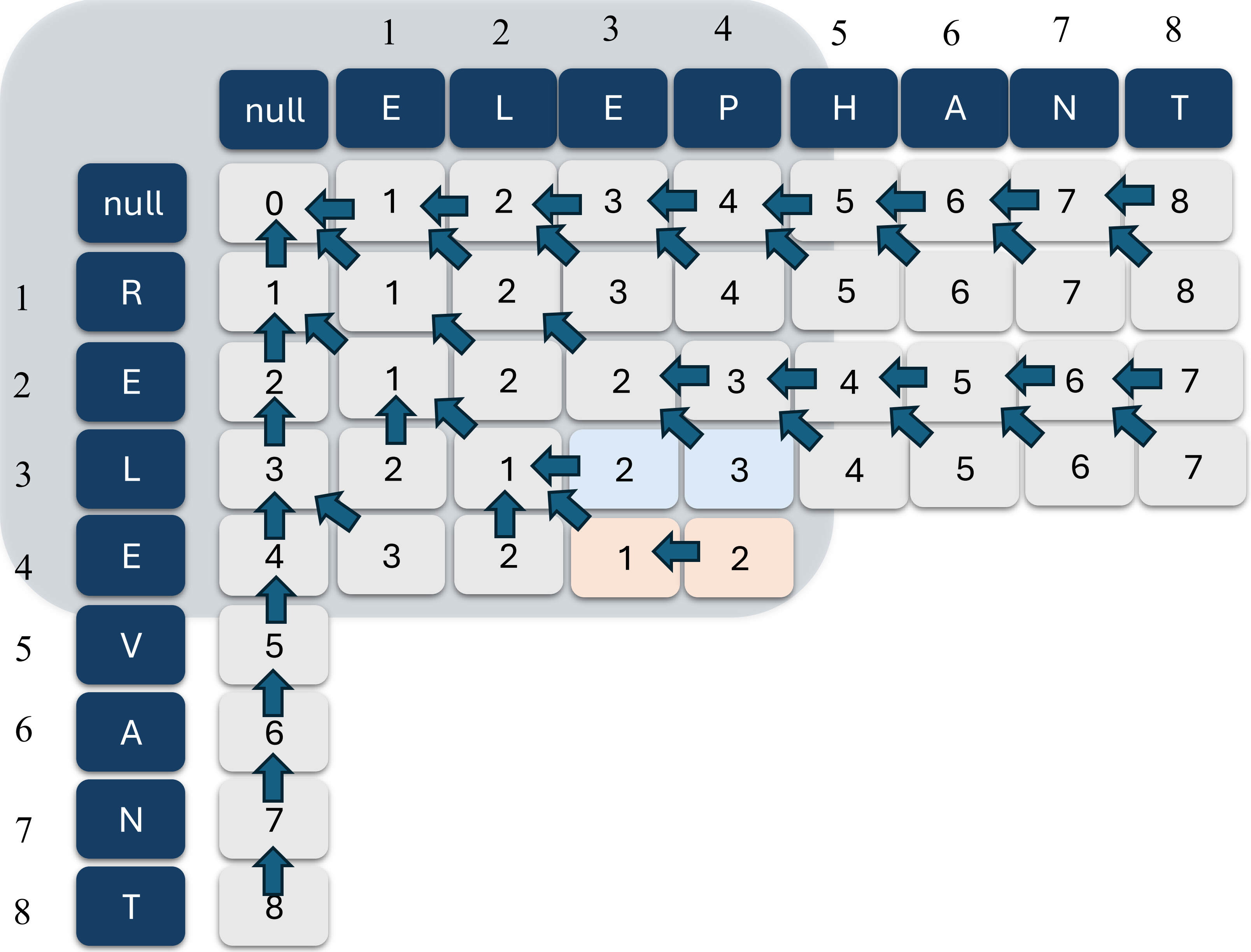
$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

↖ ← ↑

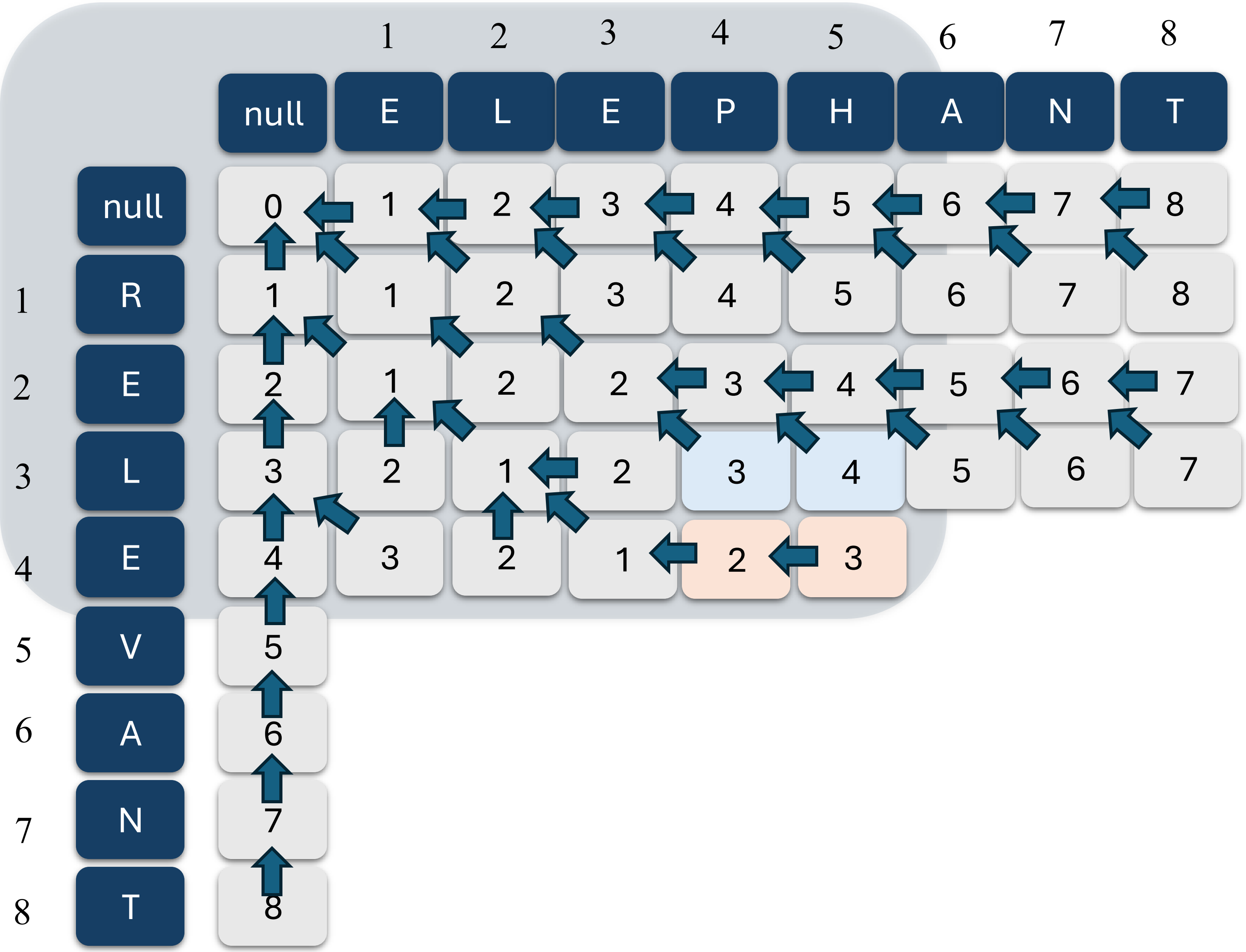


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1					
5	V	5								
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



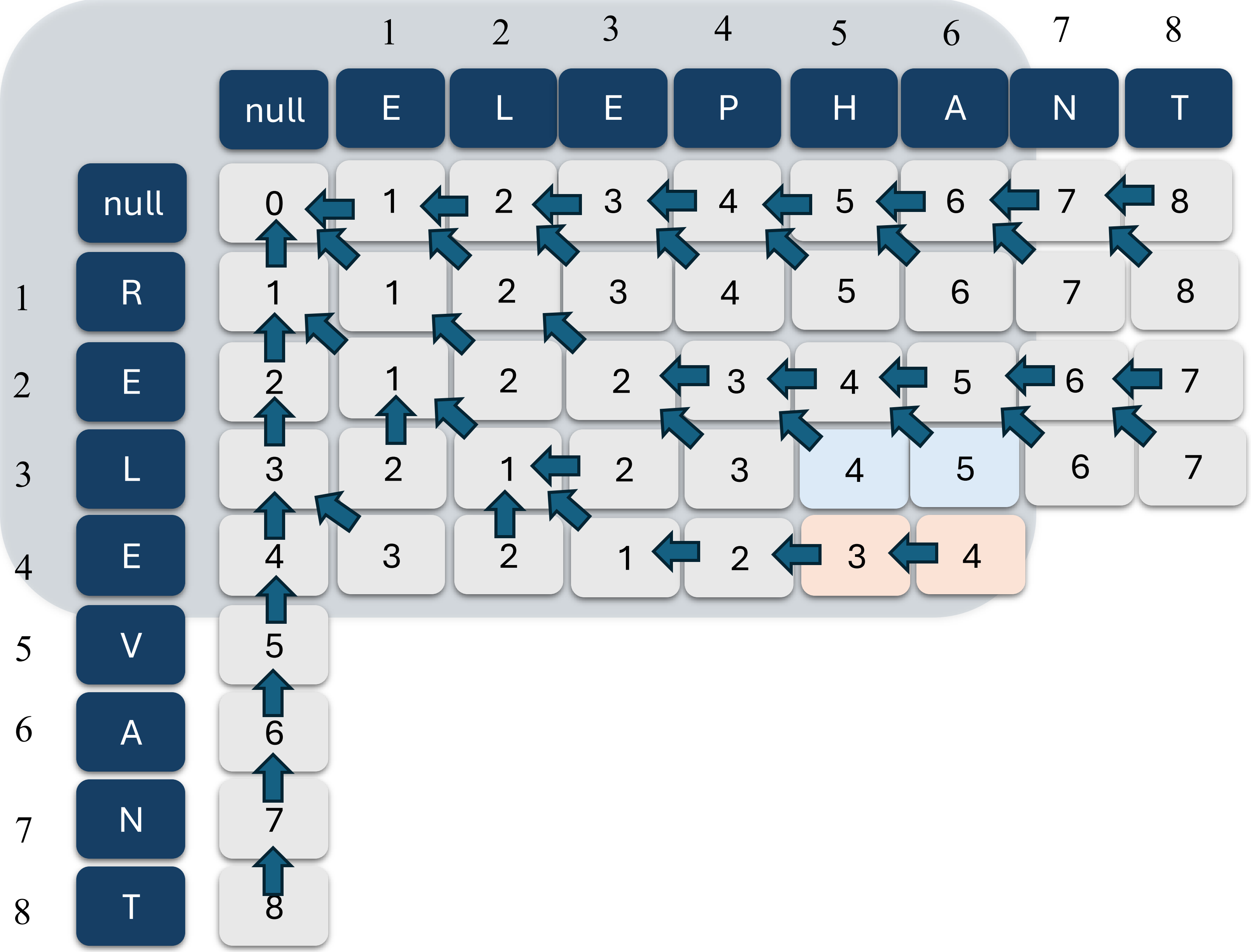
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3			
5	V	5								
6	A	6								
7	N	7								
8	T	8								

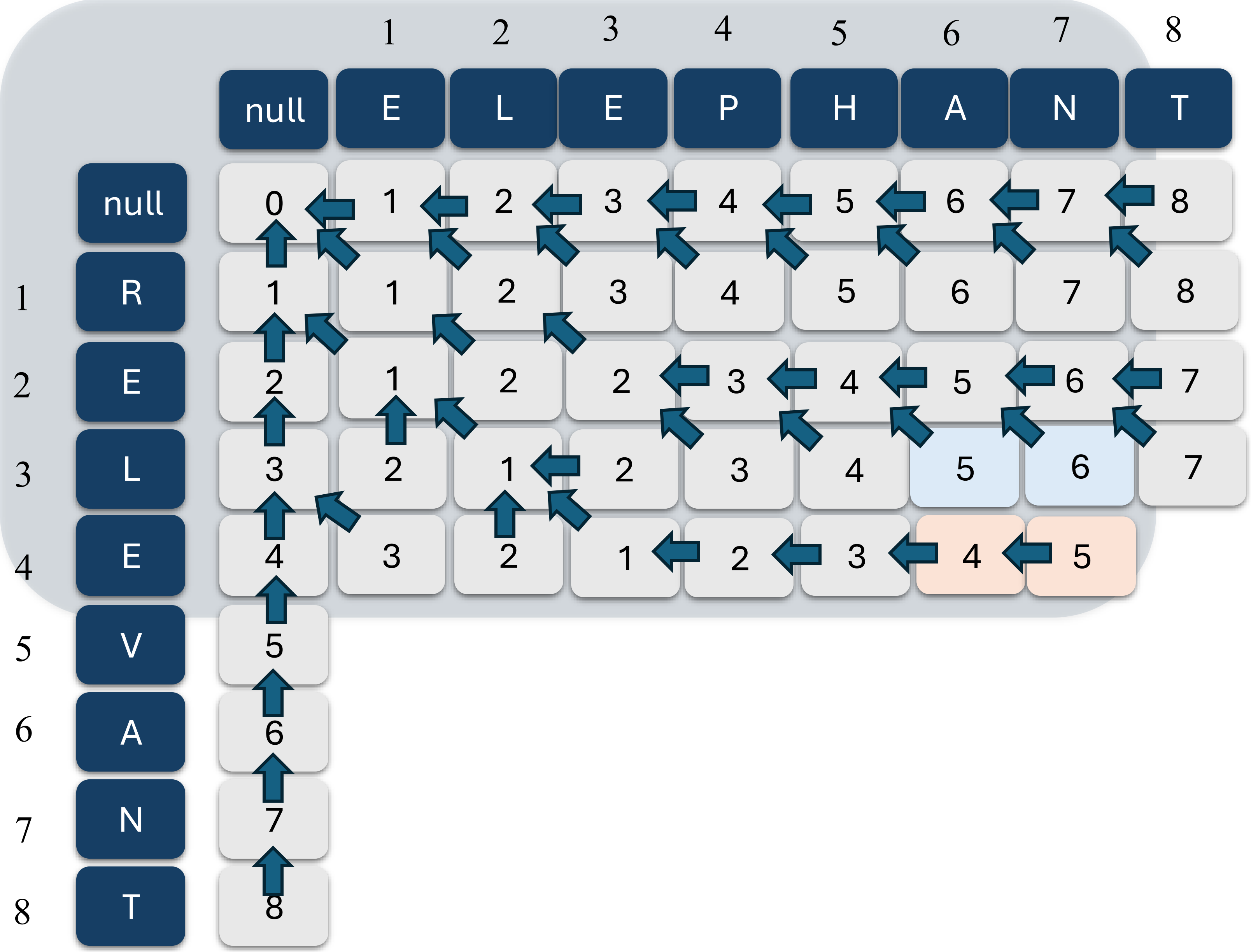
$$E(i, j) = min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4		
5	V	5								
6	A	6								
7	N	7								
8	T	8								

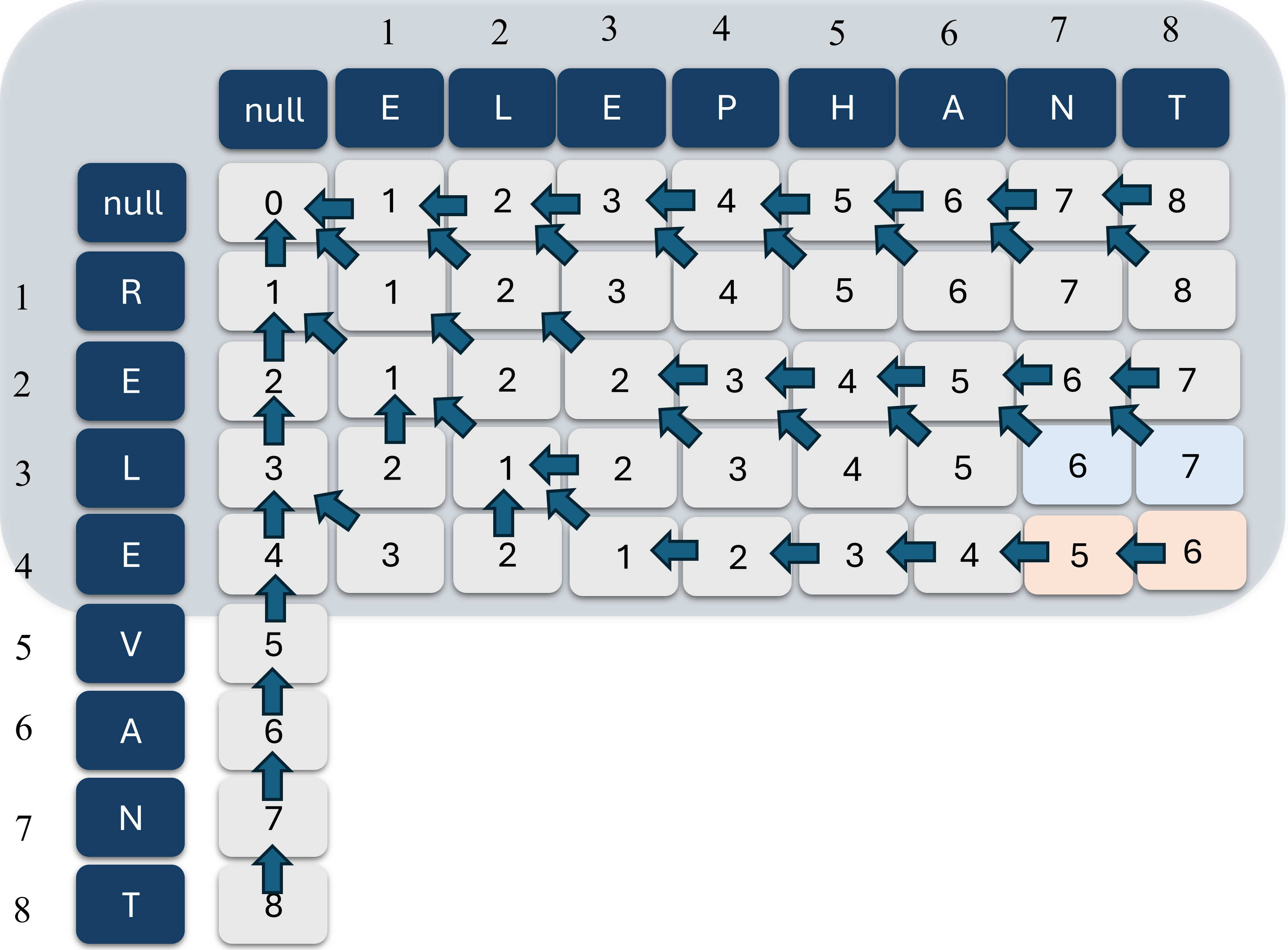
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	
5	V	5								
6	A	6								
7	N	7								
8	T	8								

↖ ← ↑

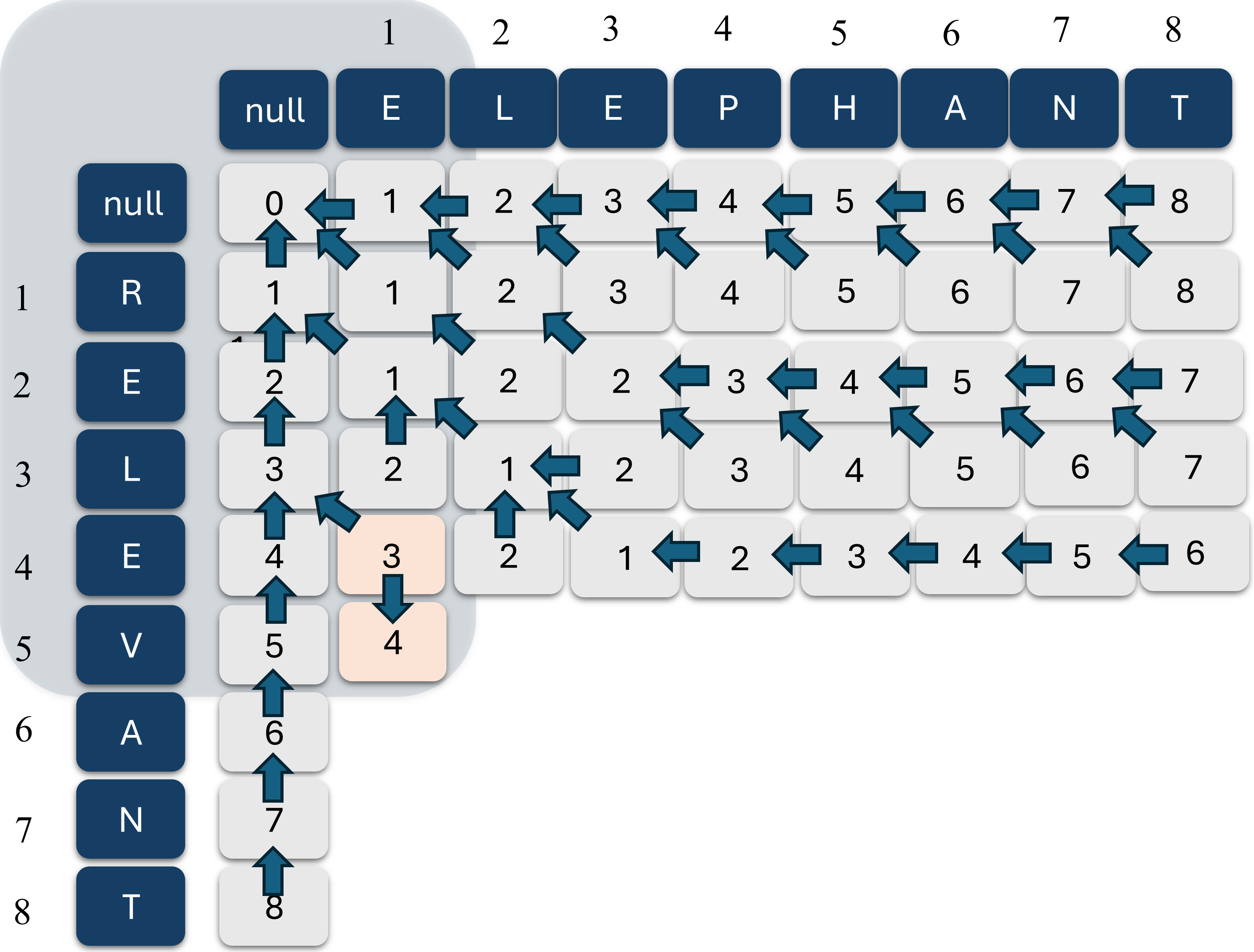
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5								
6	A	6								
7	N	7								
8	T	8								

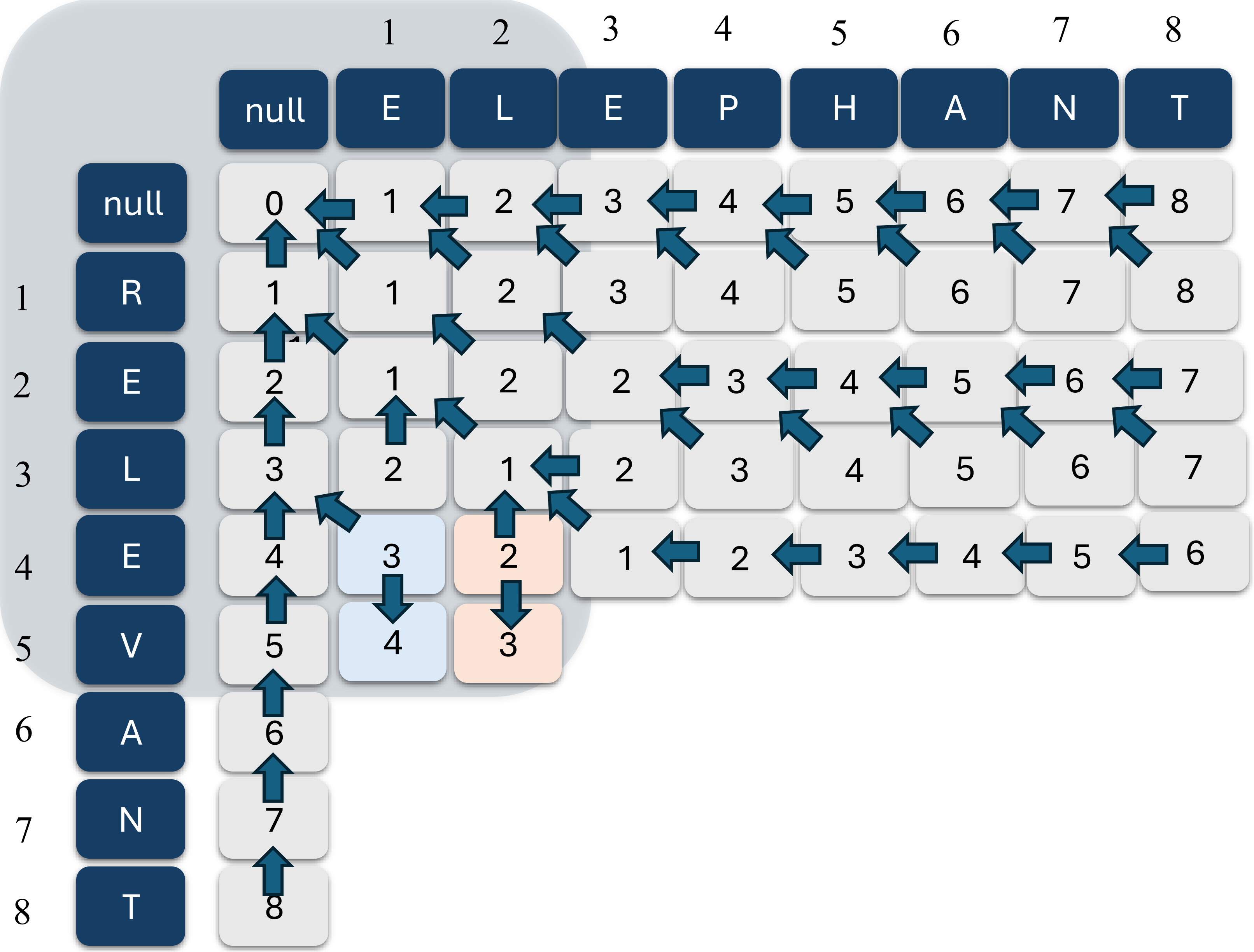
↖ ← ↑

$$E(i, j) = min\begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4							
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

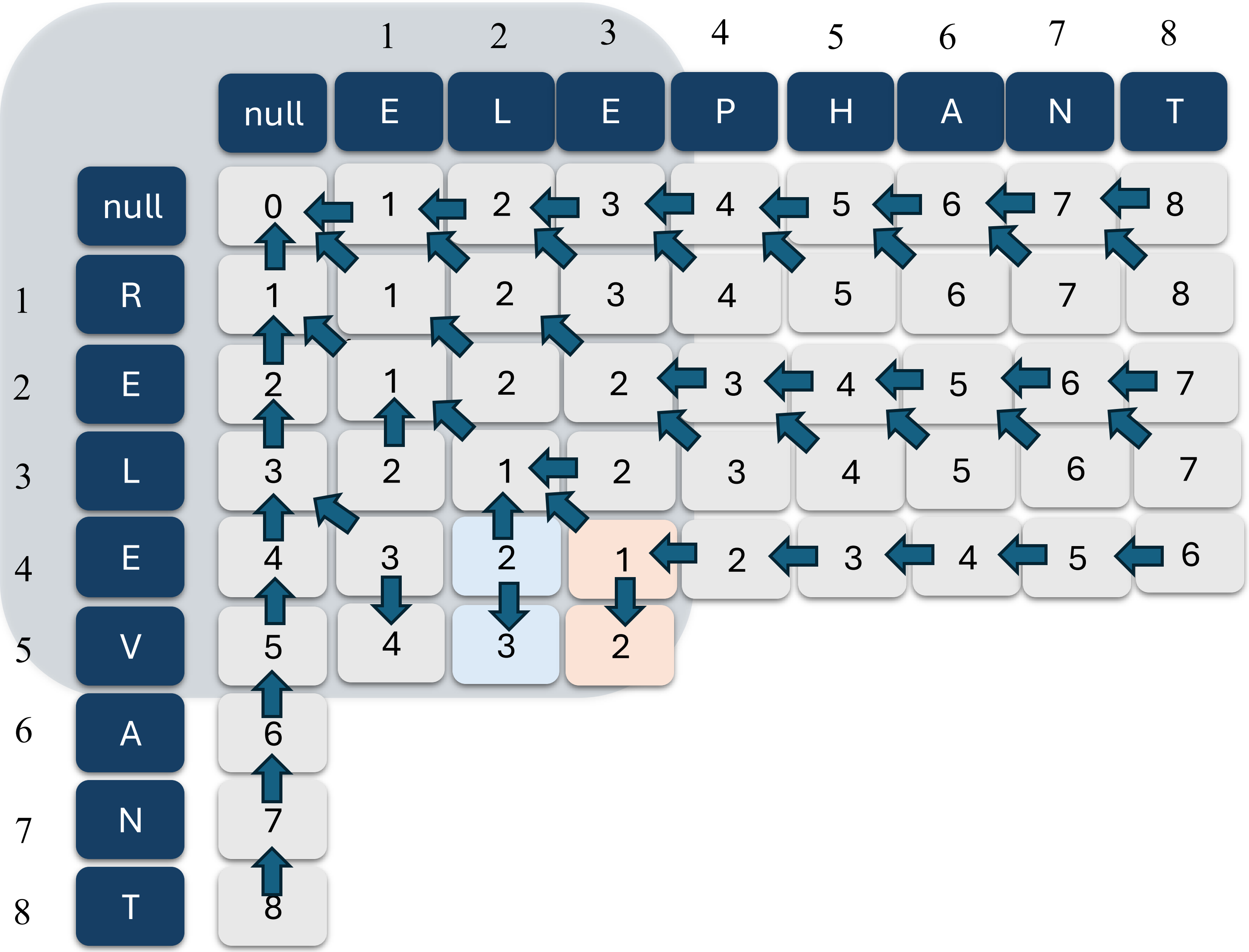


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3						
6	A	6								
7	N	7								
8	T	8								

↖ ← ↑

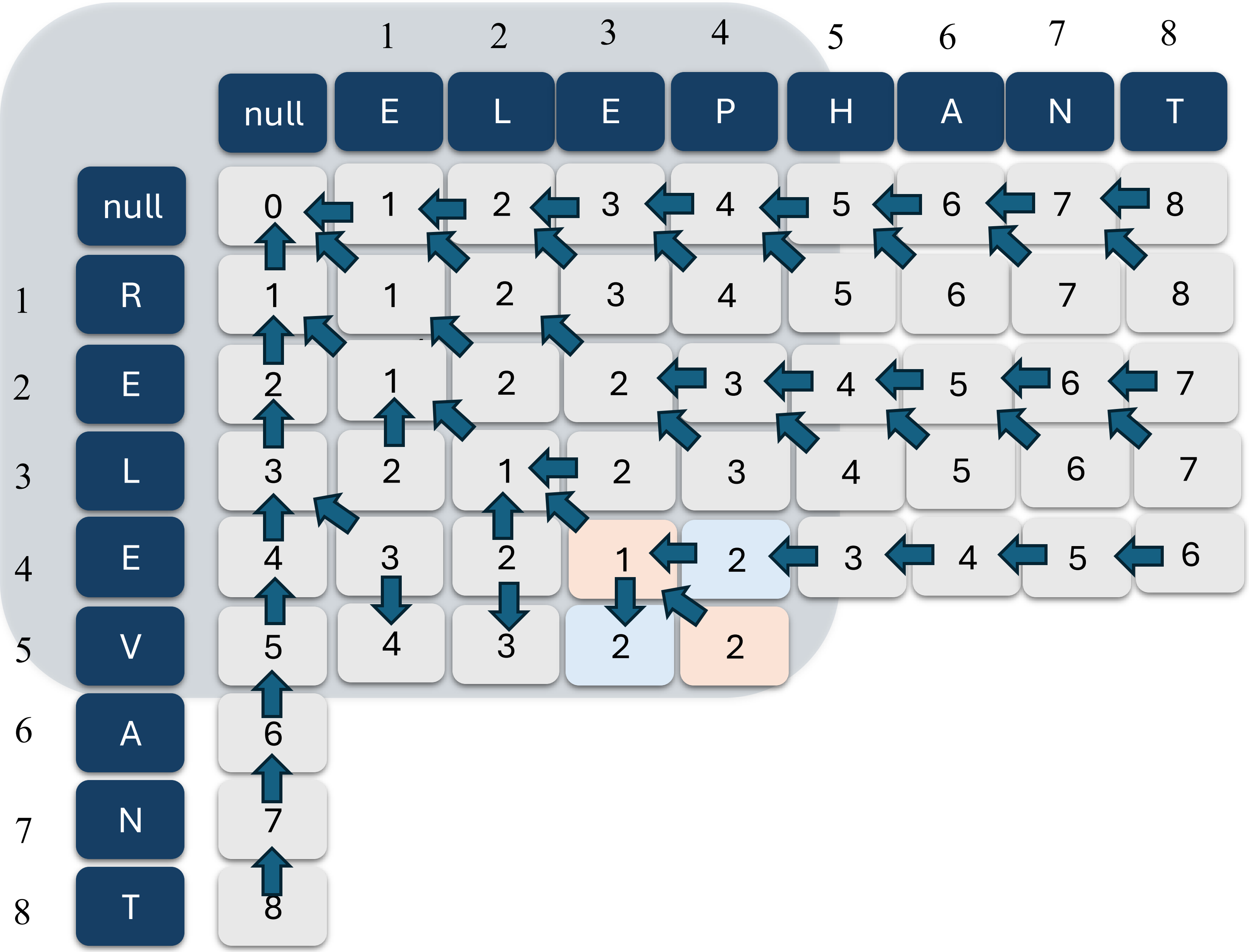
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2					
6	A	6								
7	N	7								
8	T	8								

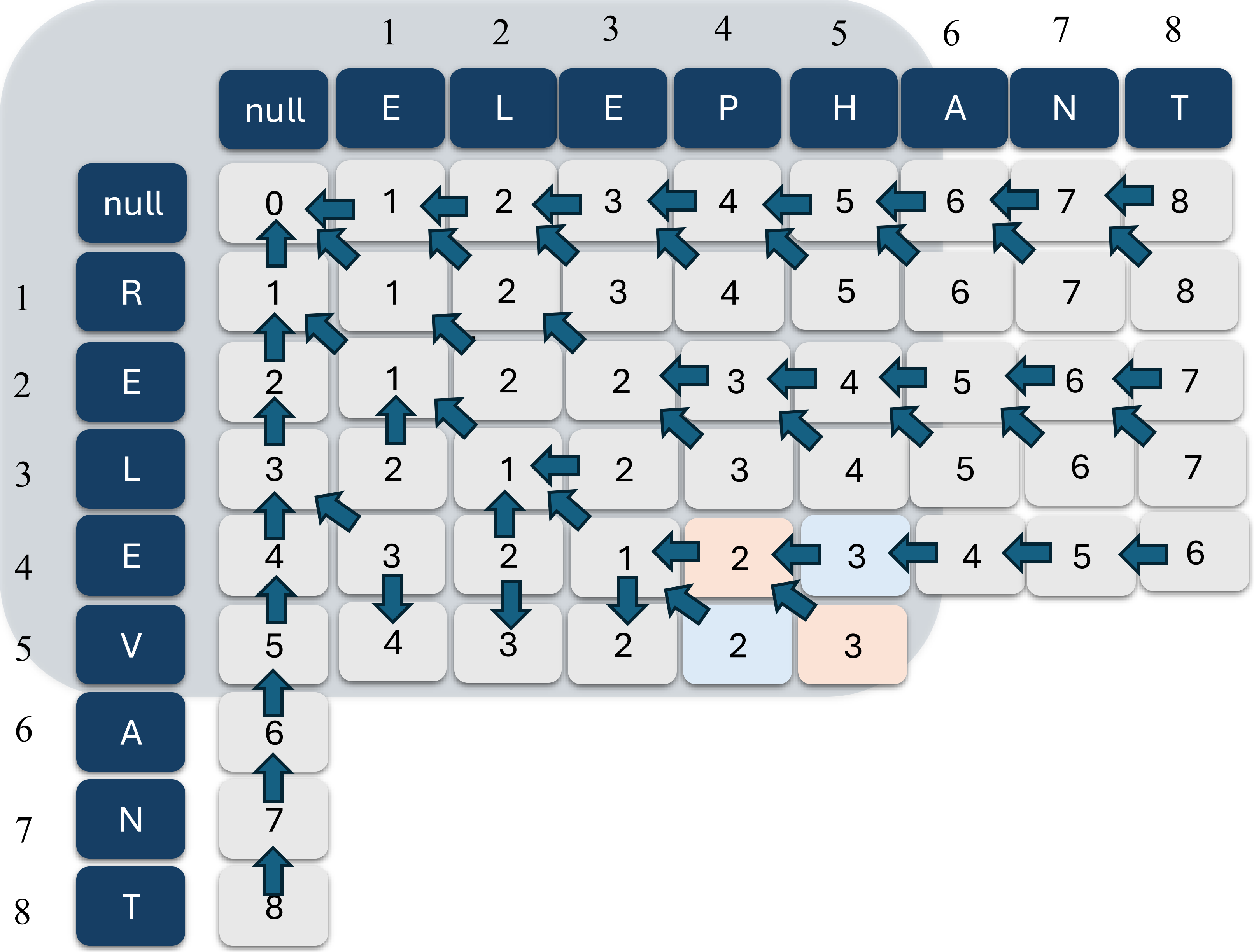
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2				
6	A	6								
7	N	7								
8	T	8								

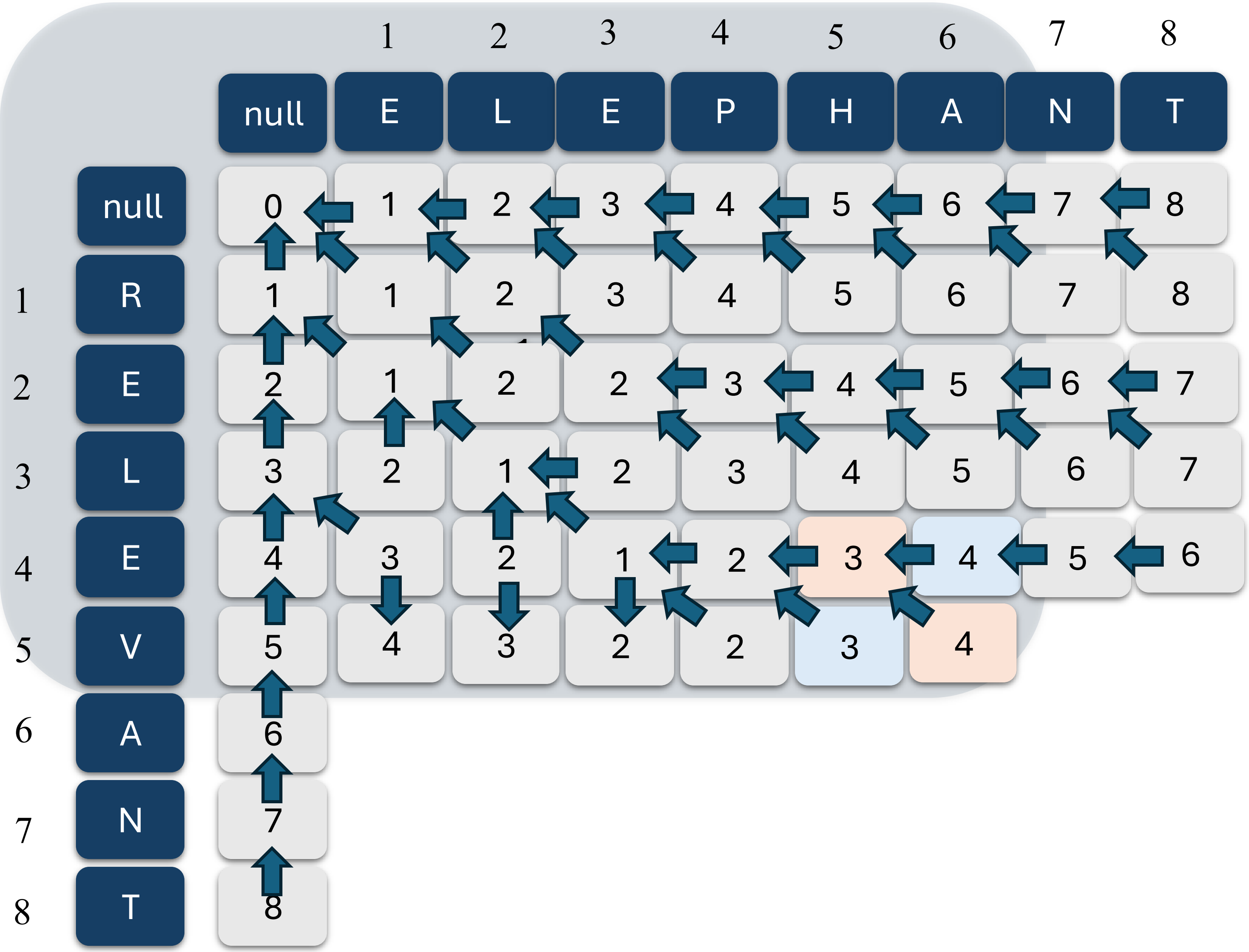
↖ ← ↑

$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3			
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



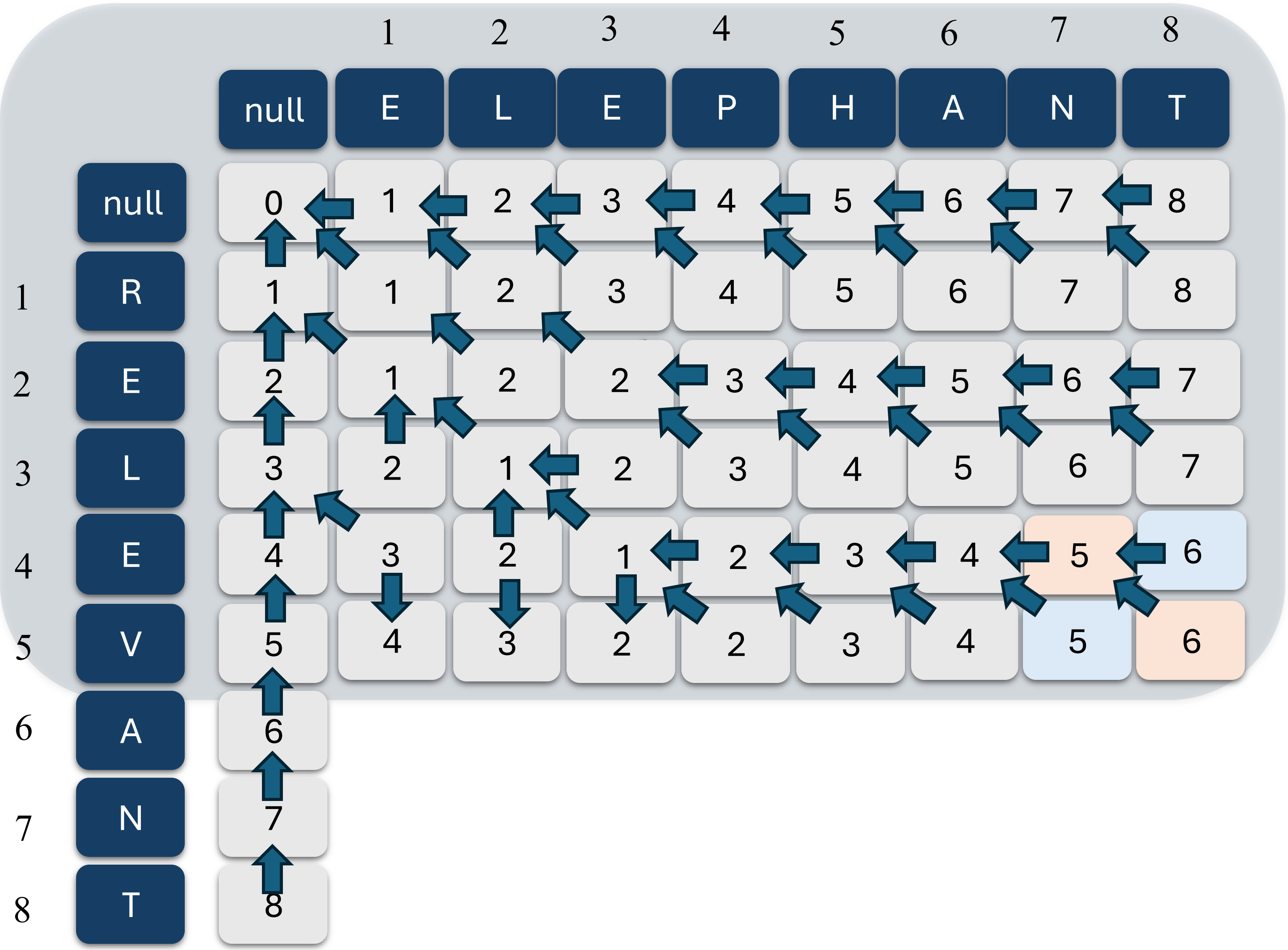
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4		
6	A	6								
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	
6	A	6								
7	N	7								
8	T	8								

↖ ← ↑

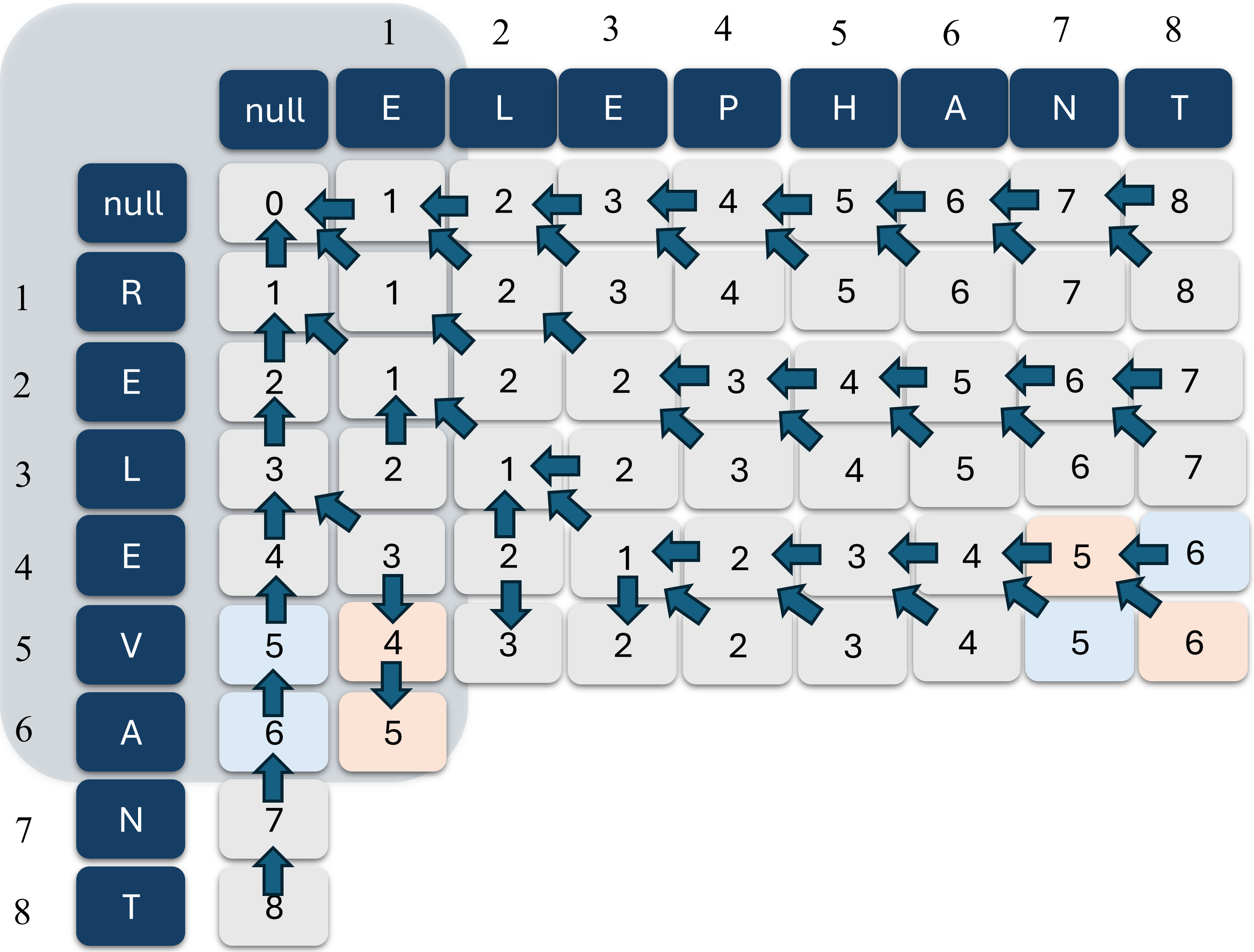
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	**R**	1	1	2	3	4	5	6	7	8
2	**E**	2	1	2	2	3	4	5	6	7
3	**L**	3	2	1	2	3	4	5	6	7
4	**E**	4	3	2	1	2	3	4	5	6
5	**V**	5	4	3	2	2	3	4	5	6
6	**A**	6								
7	**N**	7								
8	**T**	8								

↖ ← ↑

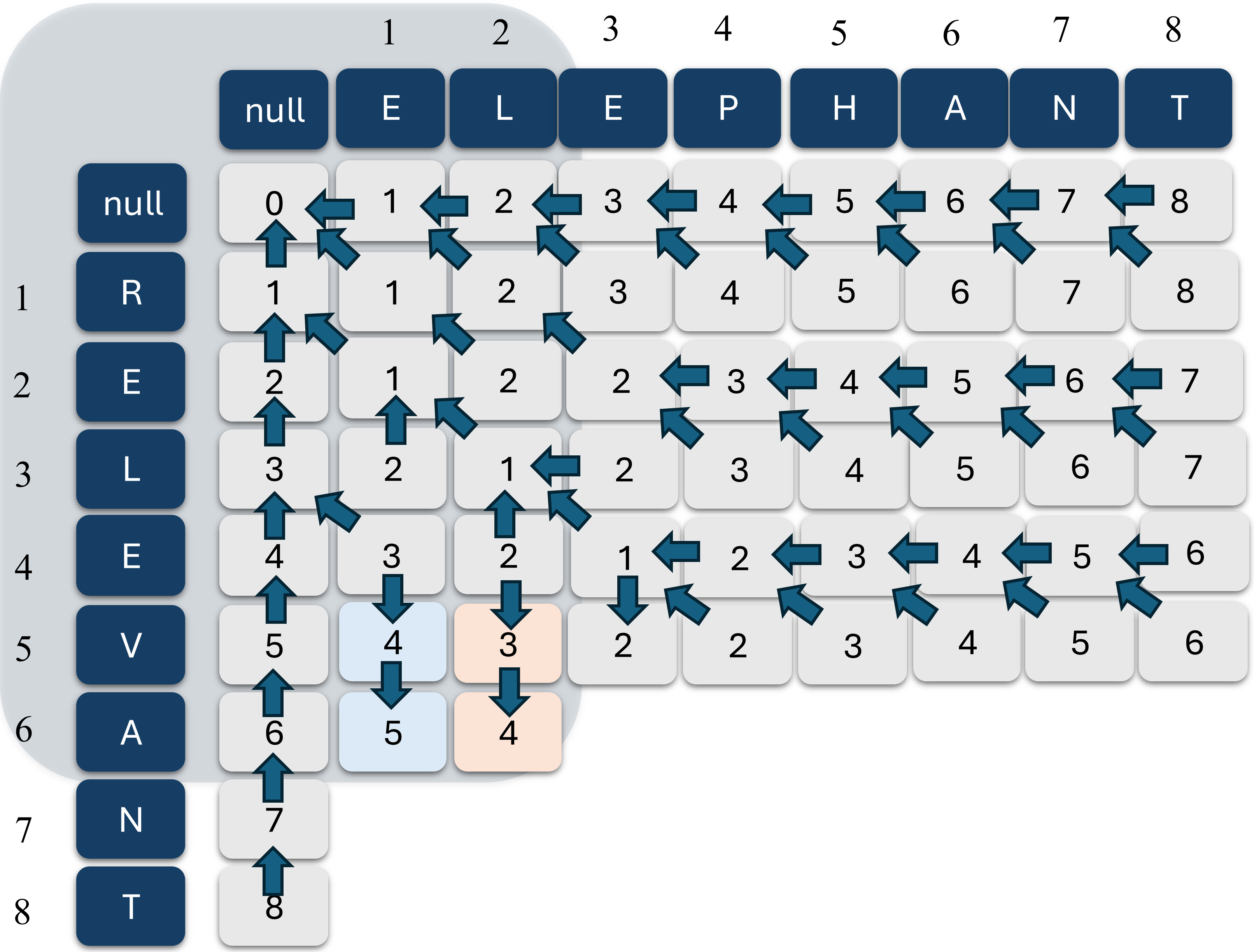
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
	null	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5							
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

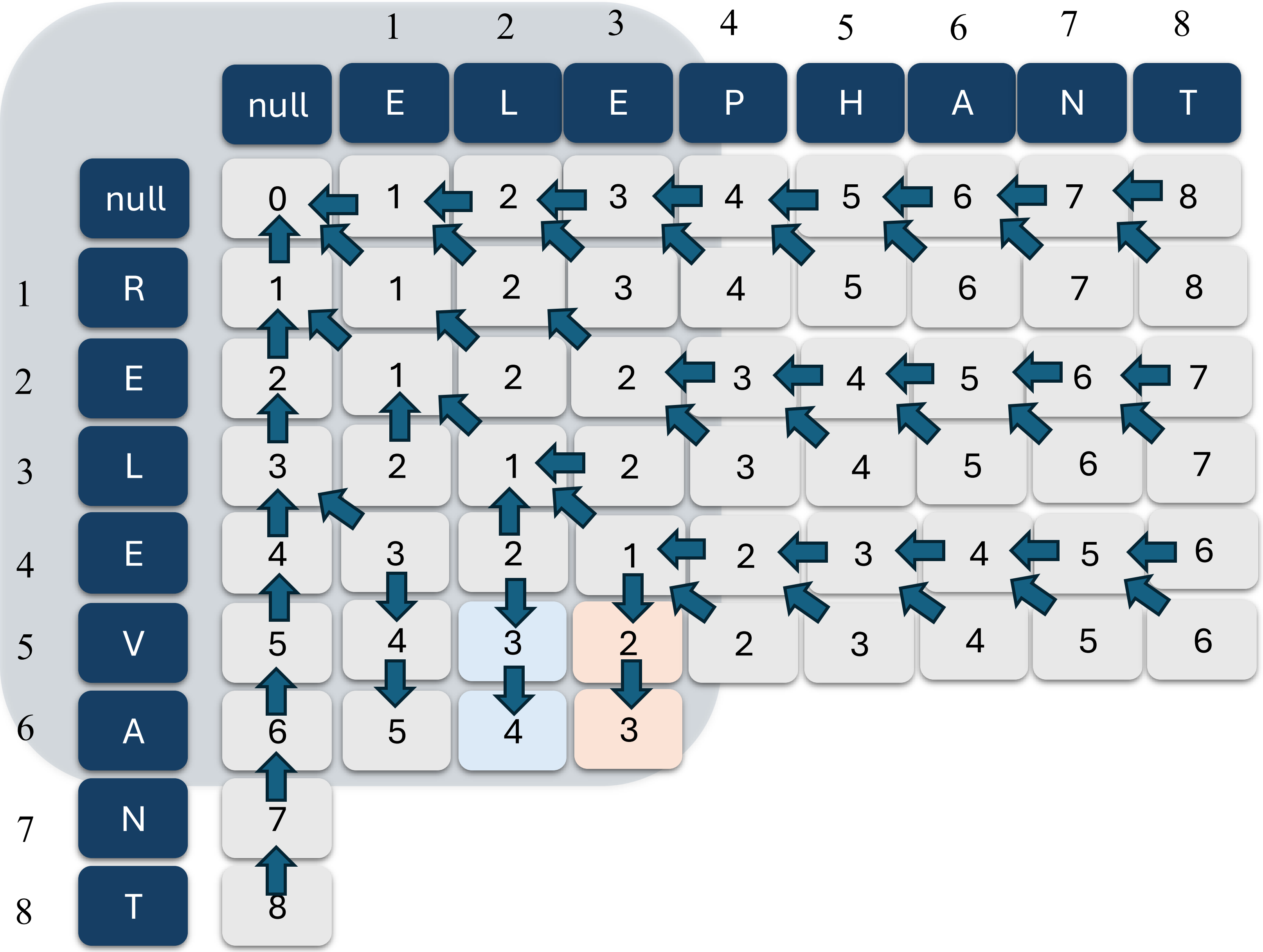
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4						
7	N	7								
8	T	8								

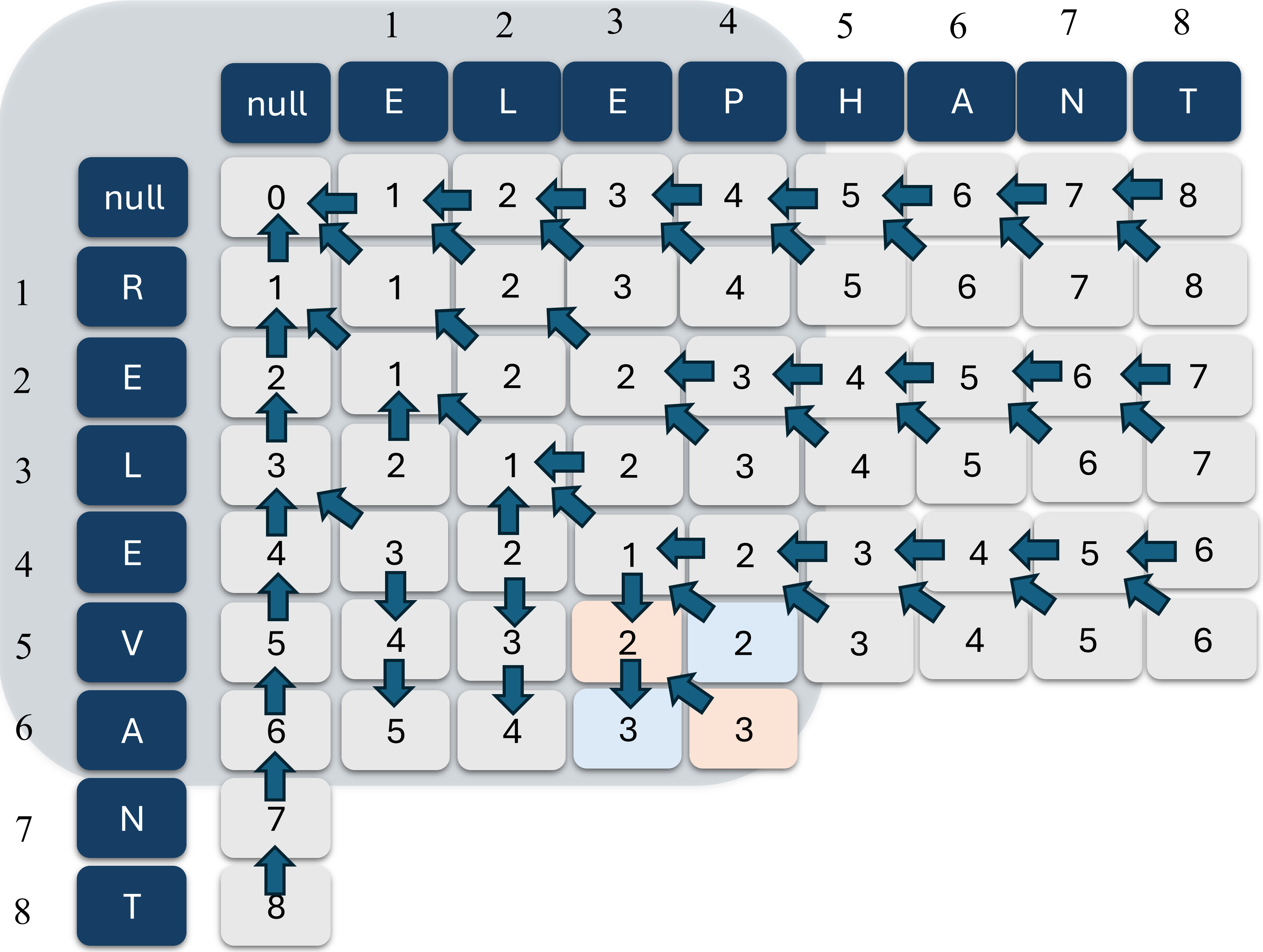
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



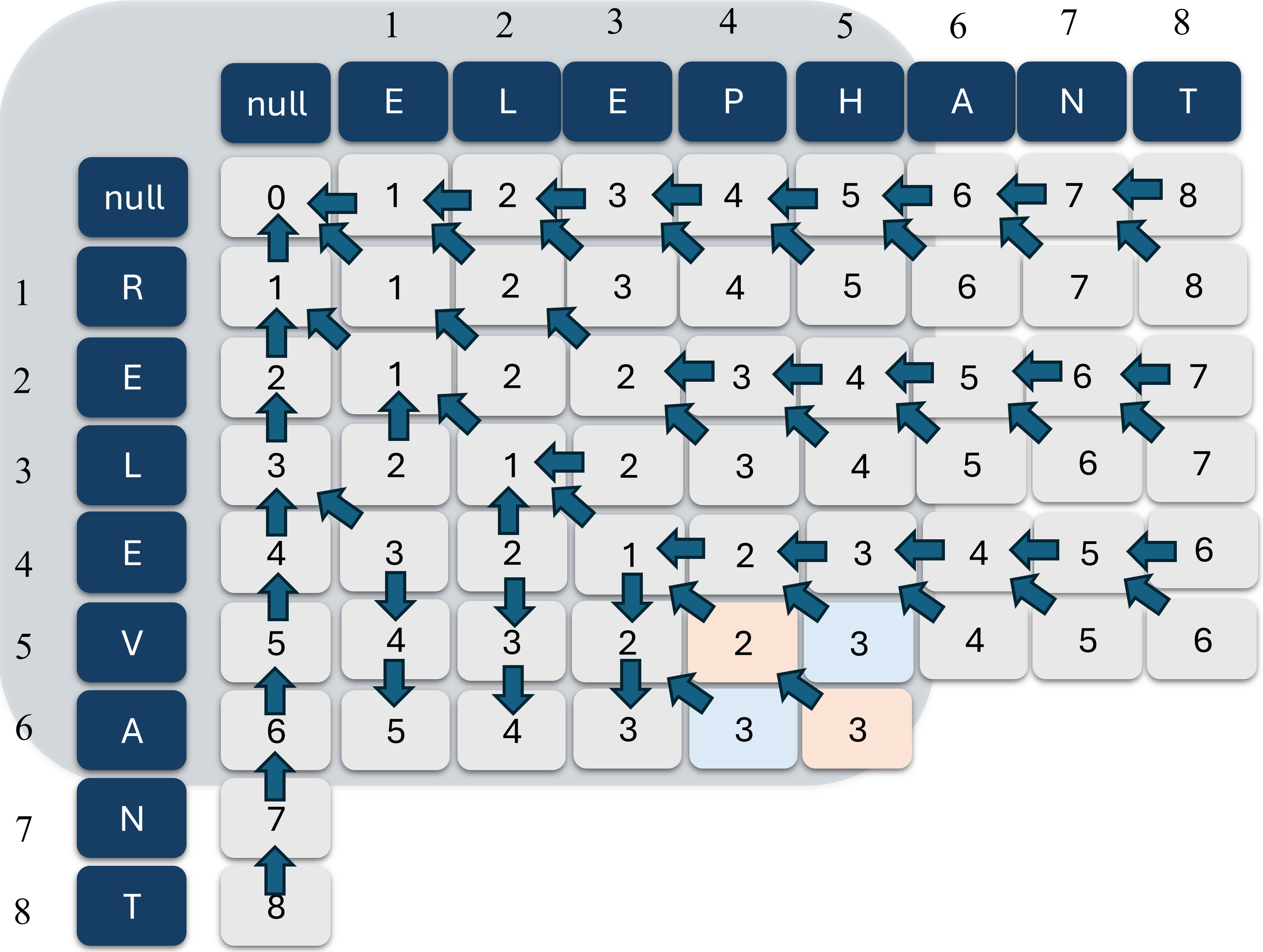
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3					
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



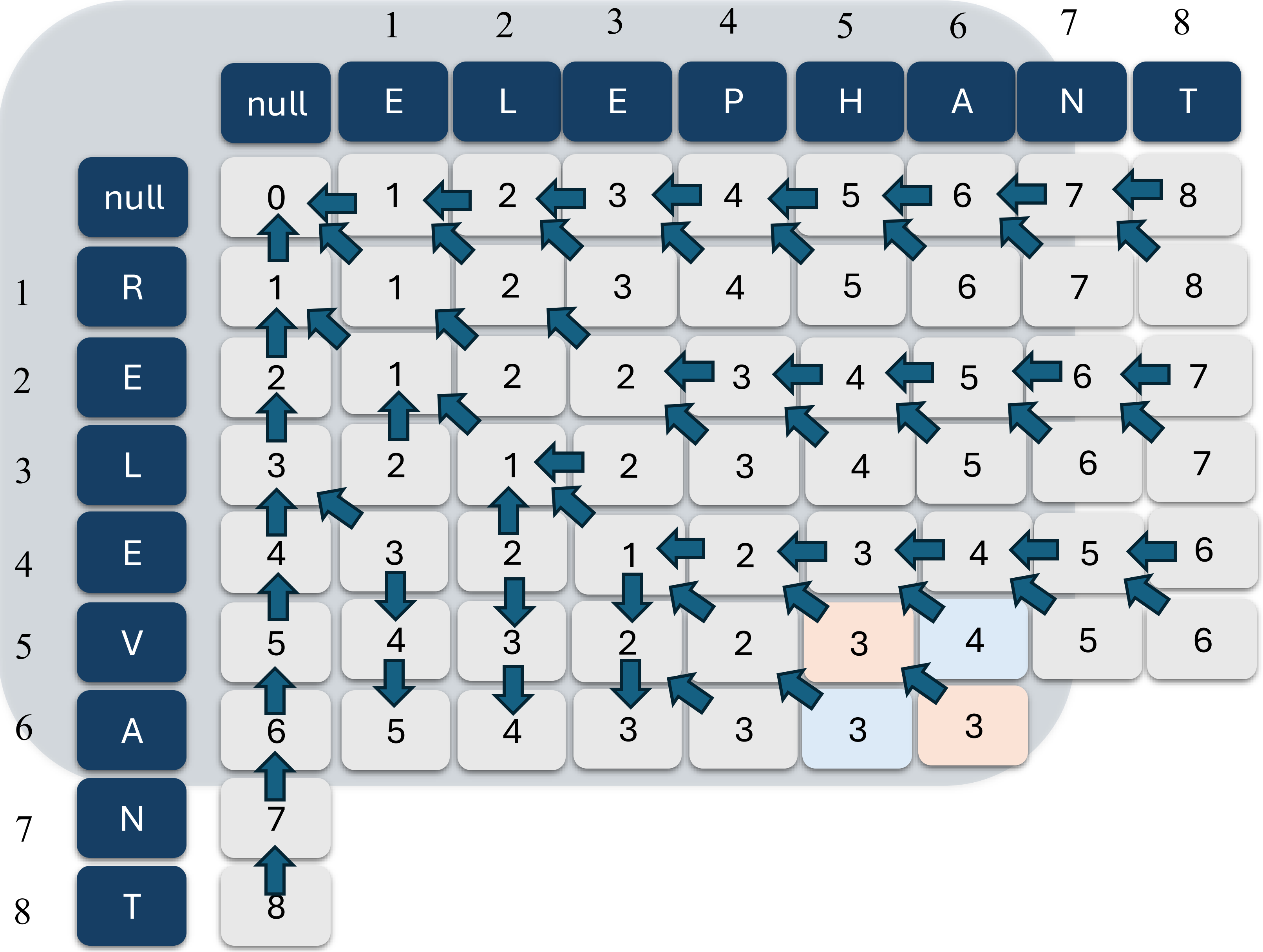
		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3				
7	N	7								
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3			
7	N	7								
8	T	8								

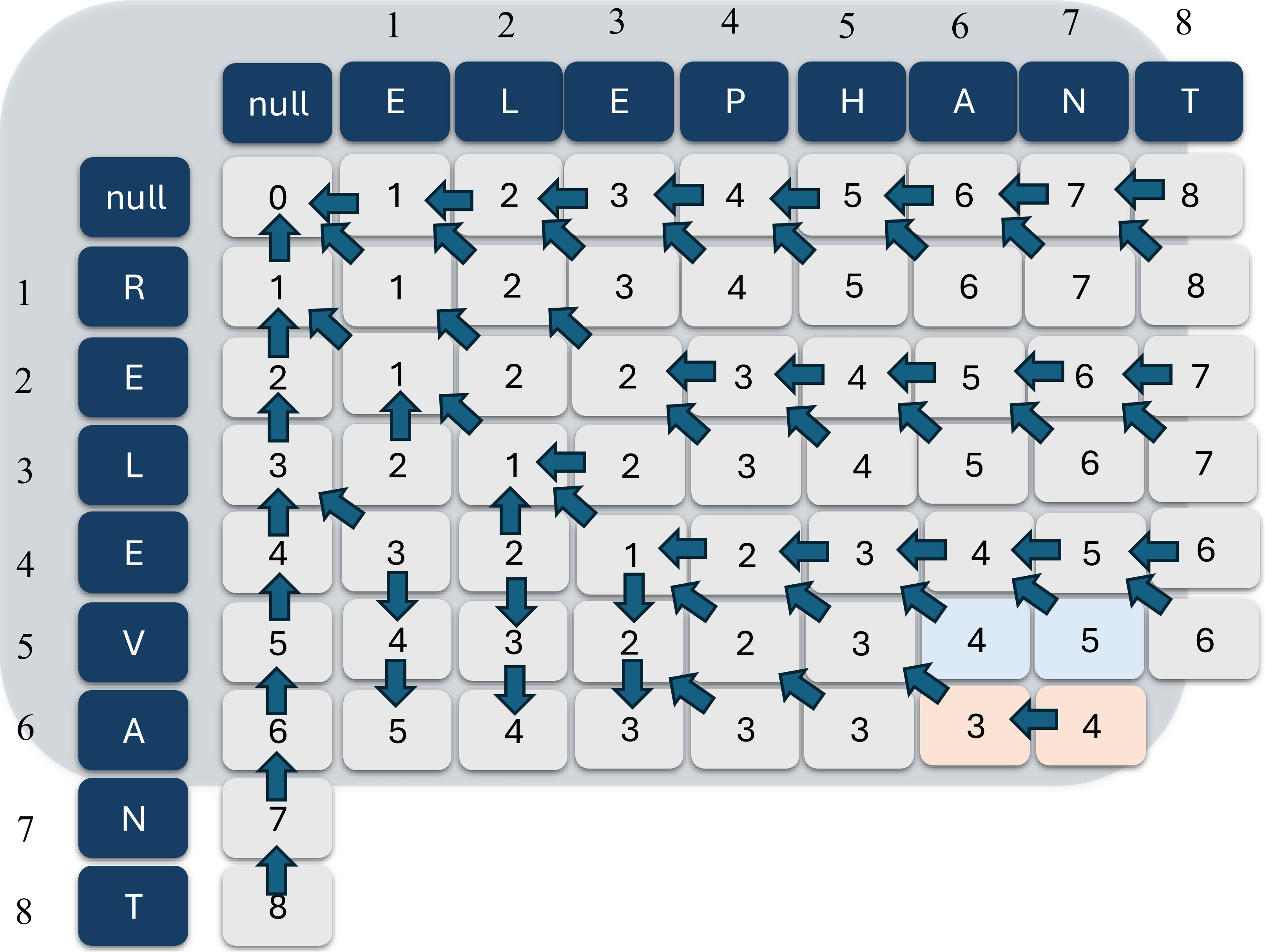
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3		
7	N	7								
8	T	8								

↖ ← ↑

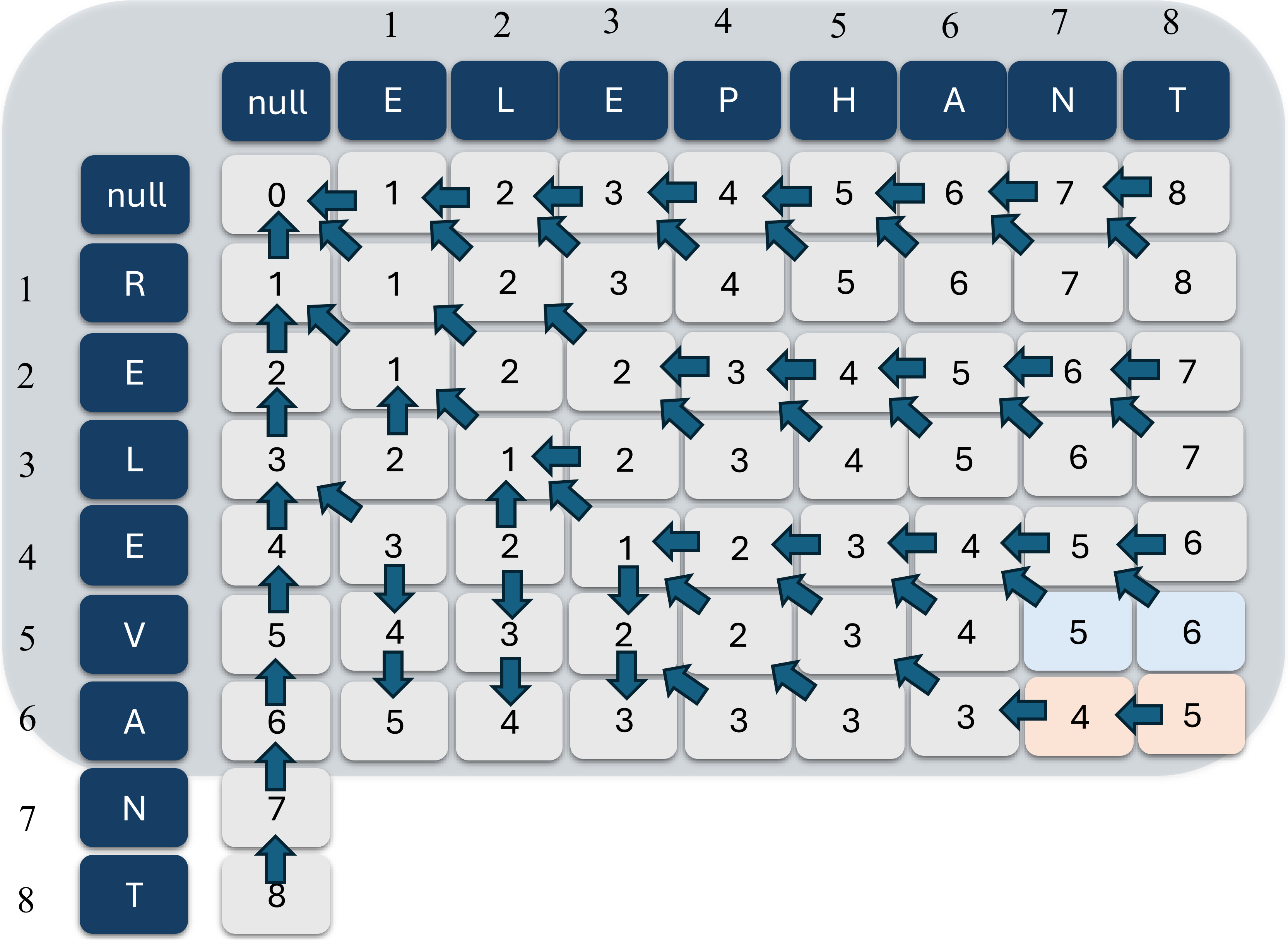
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



		0	1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	
7	N	7								
8	T	8								

↖ ← ↑

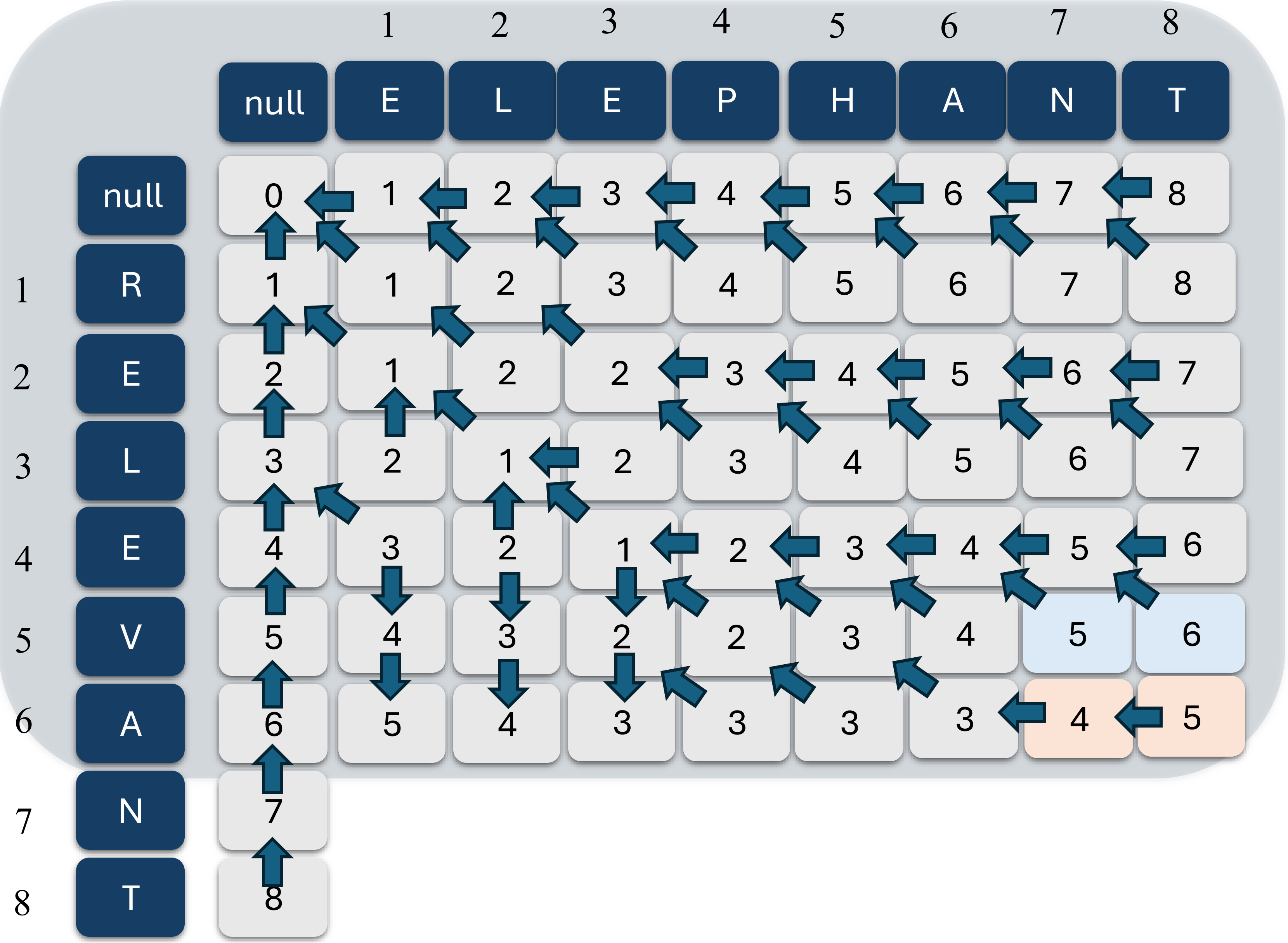
$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7								
8	T	8								

↖ ← ↑

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

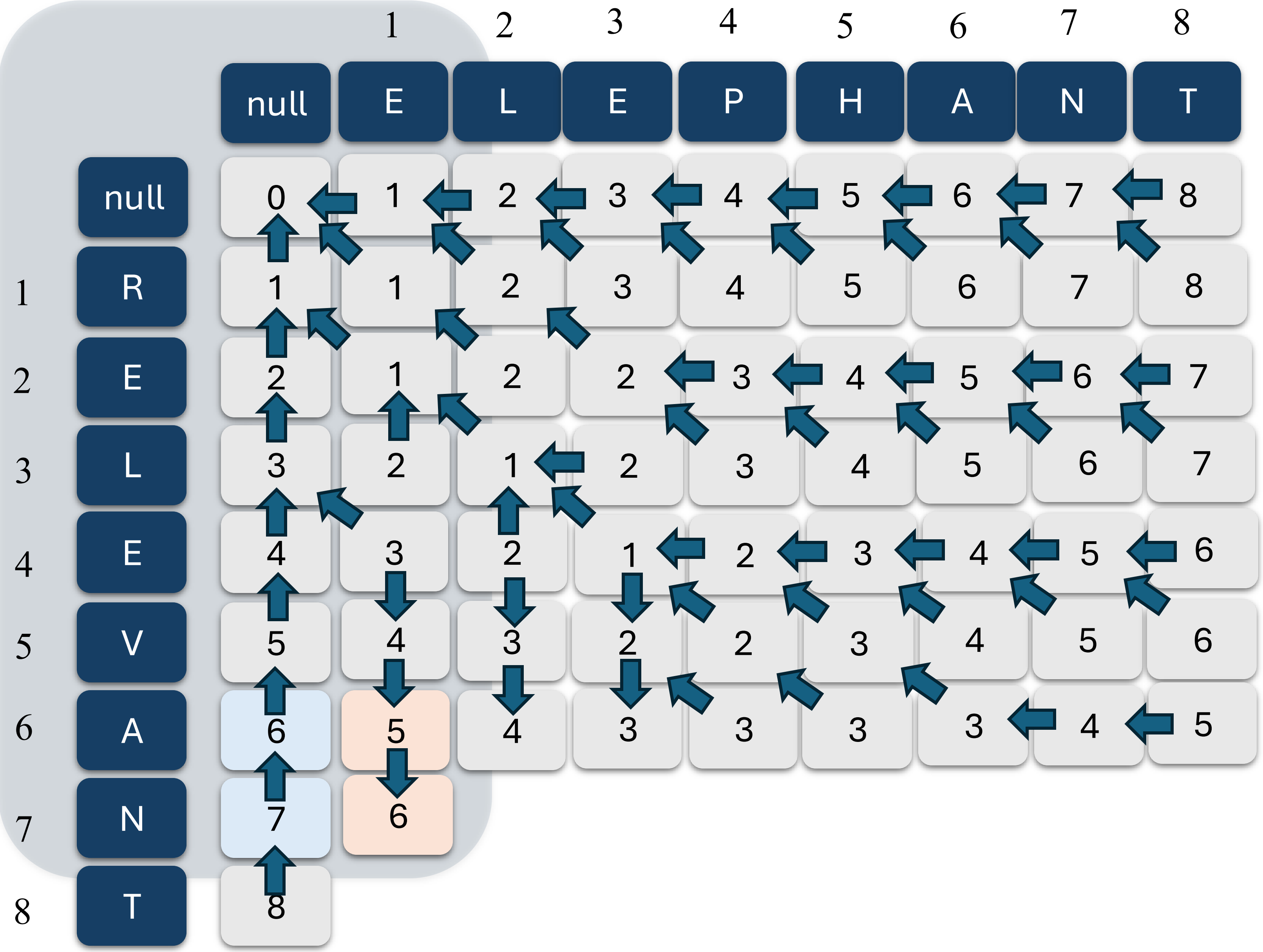


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7								
8	T	8								

↖ ← ↑

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

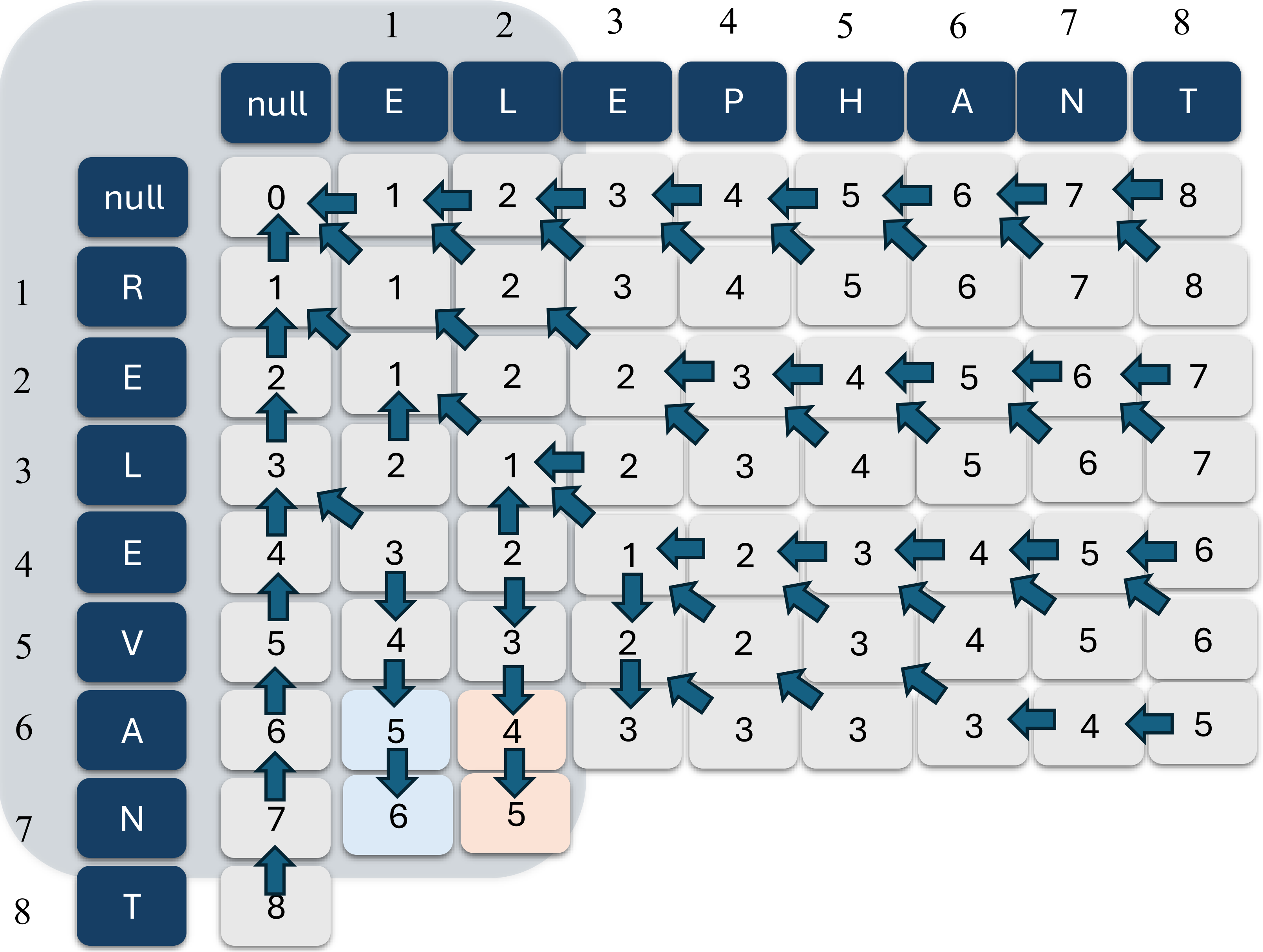
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6							
8	T	8								

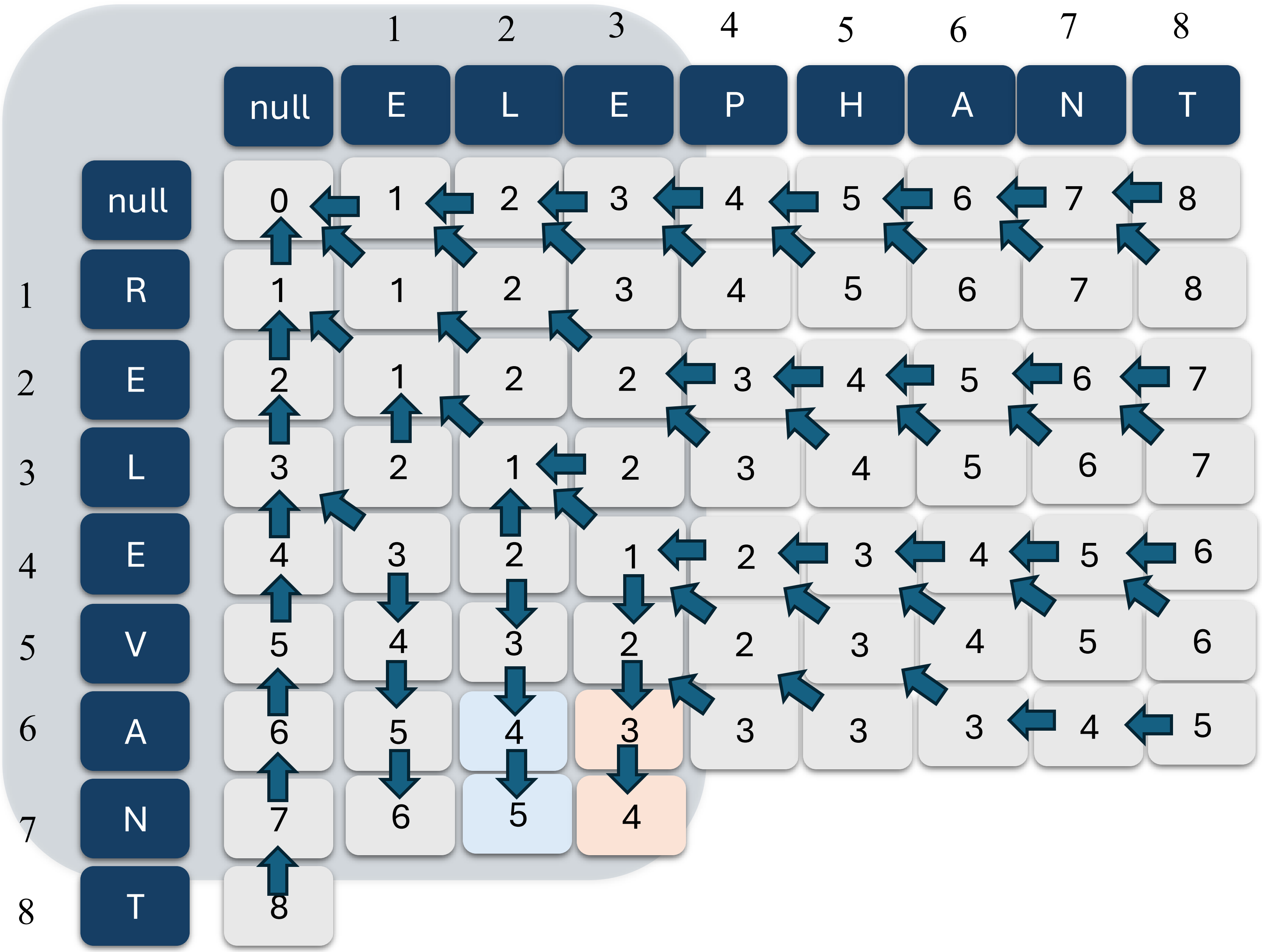
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5						
8	T	8								

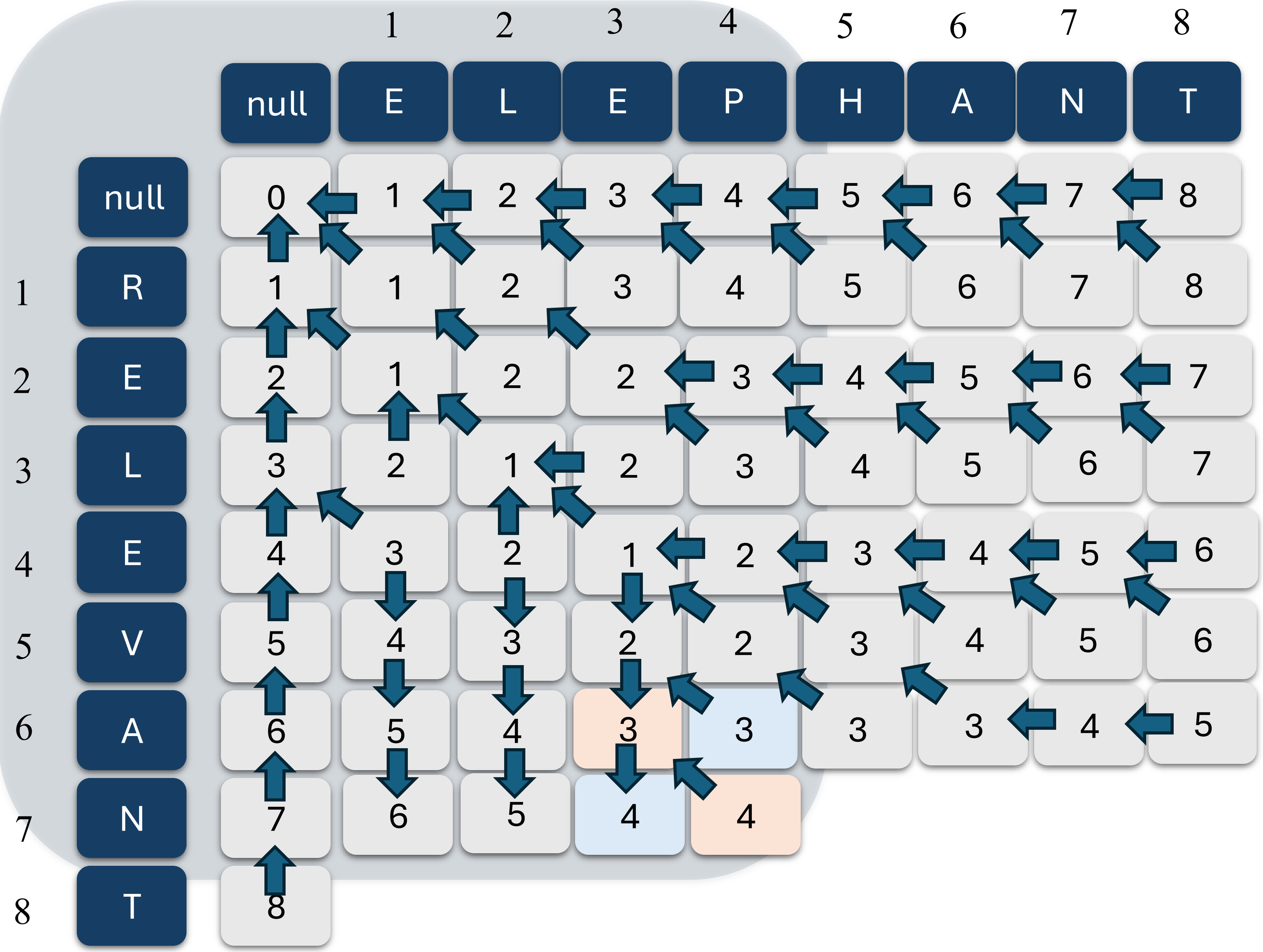
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null: 0	1	2	3	4	5	6	7	8
1	R: 1	1	2	3	4	5	6	7	8
2	E: 2	1	2	2	3	4	5	6	7
3	L: 3	2	1	2	3	4	5	6	7
4	E: 4	3	2	1	2	3	4	5	6
5	V: 5	4	3	2	2	3	4	5	6
6	A: 6	5	4	3	3	3	3	4	5
7	N: 7	6	5	4					
8	T: 8								

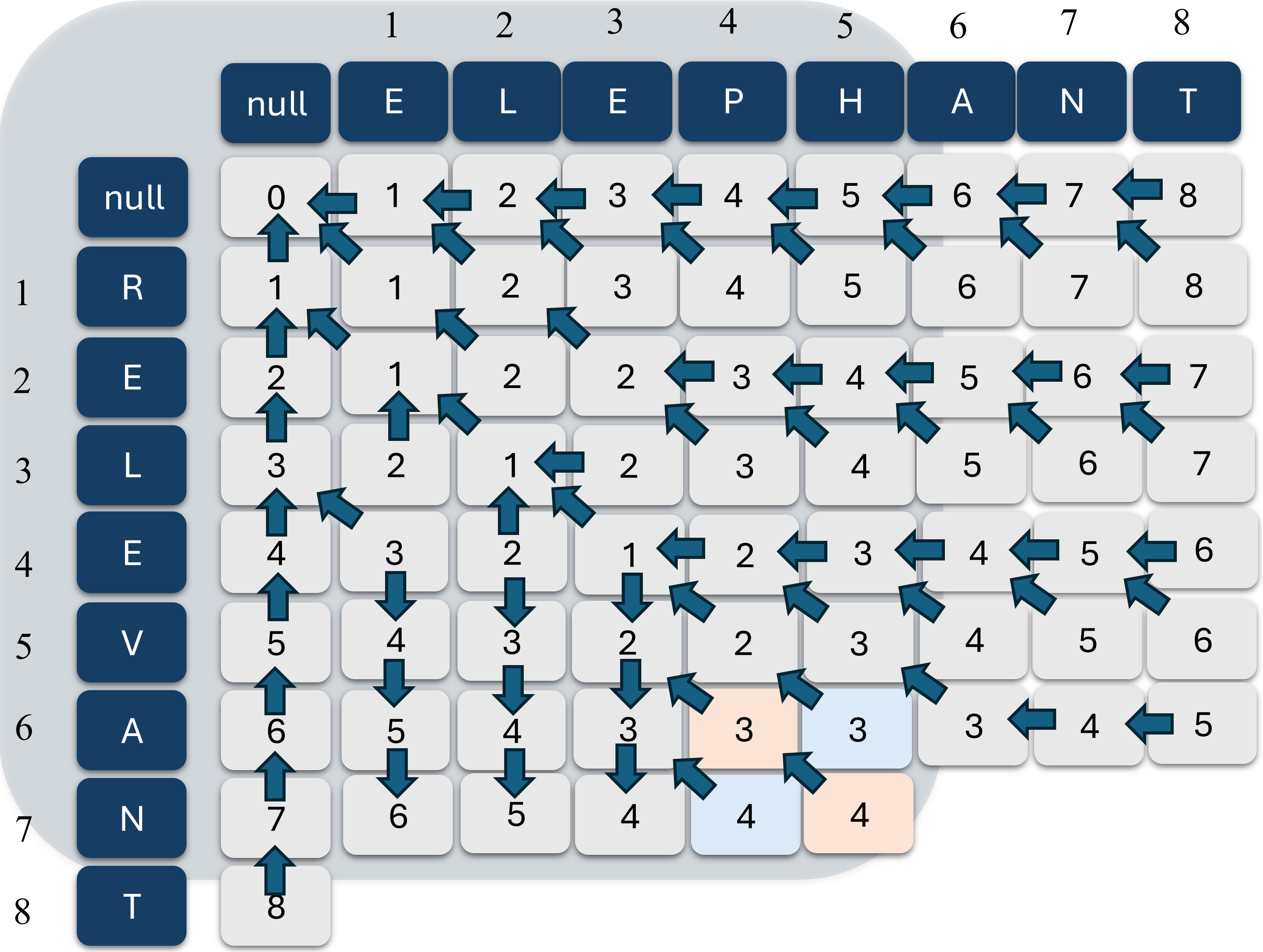
$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4				
8	T	8								

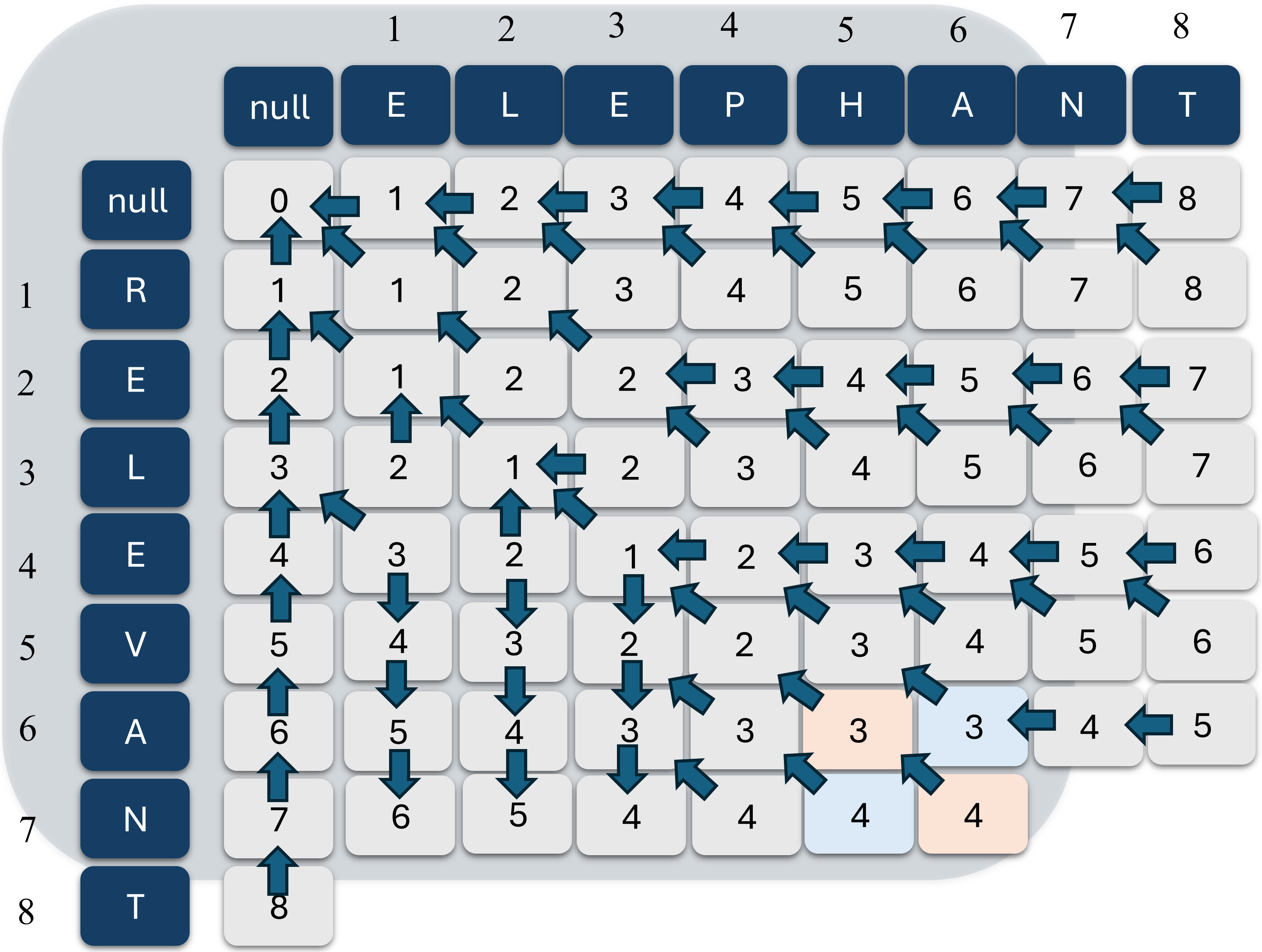
$$E(i, j) = min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4			
8	T	8								

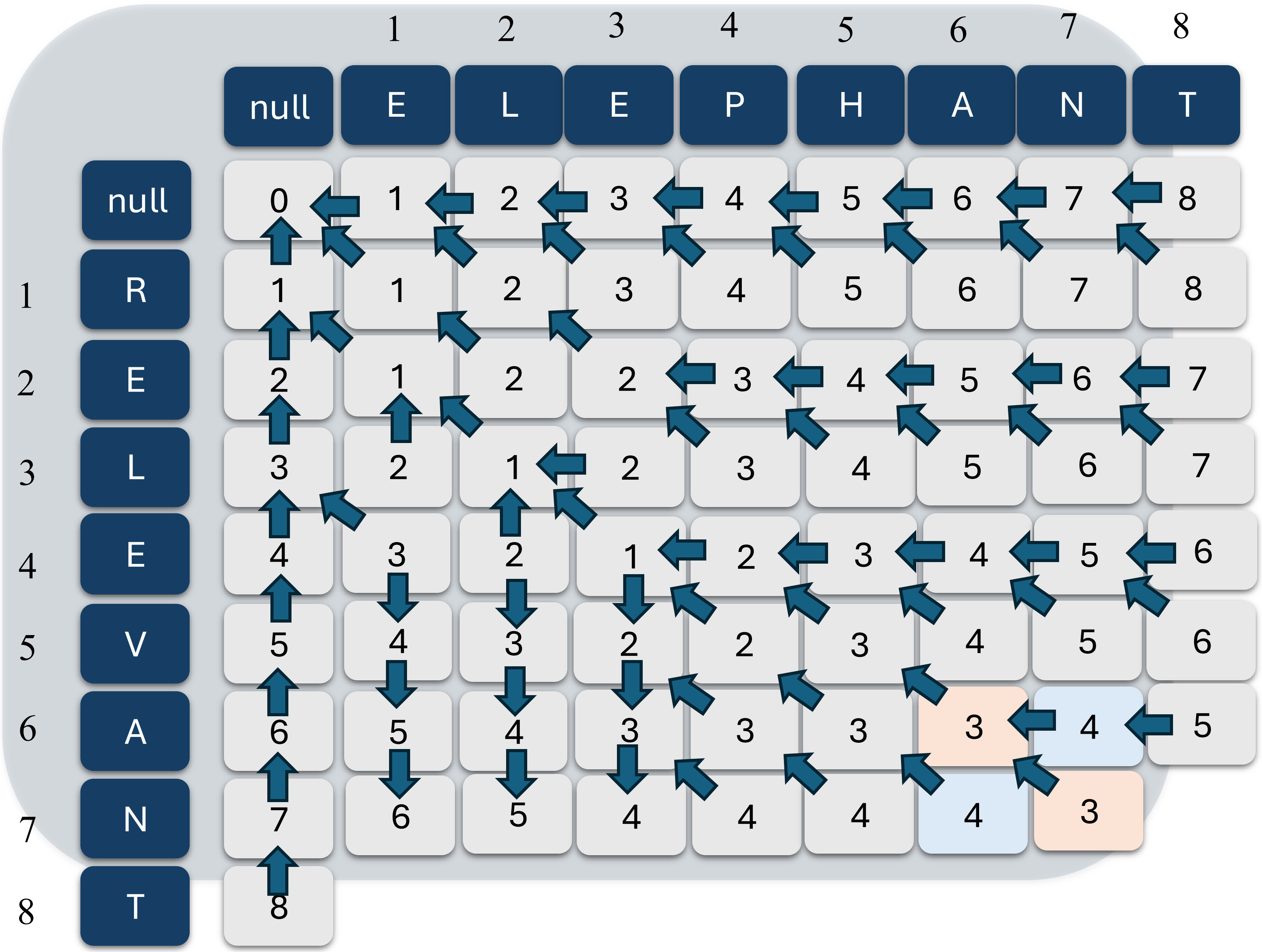
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4		
8	T	8								

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

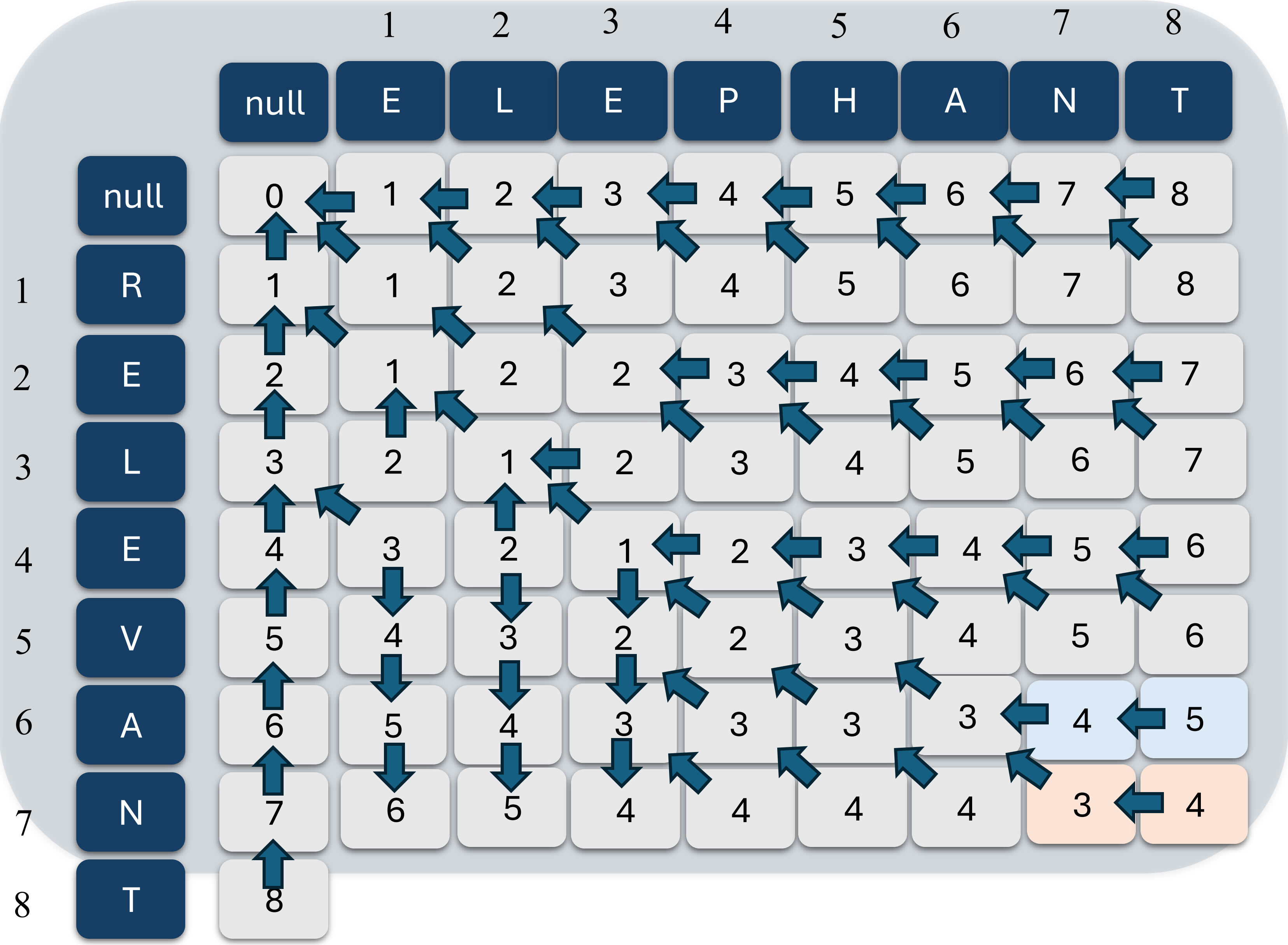


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	
8	T	8								

↖ ← ↑

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

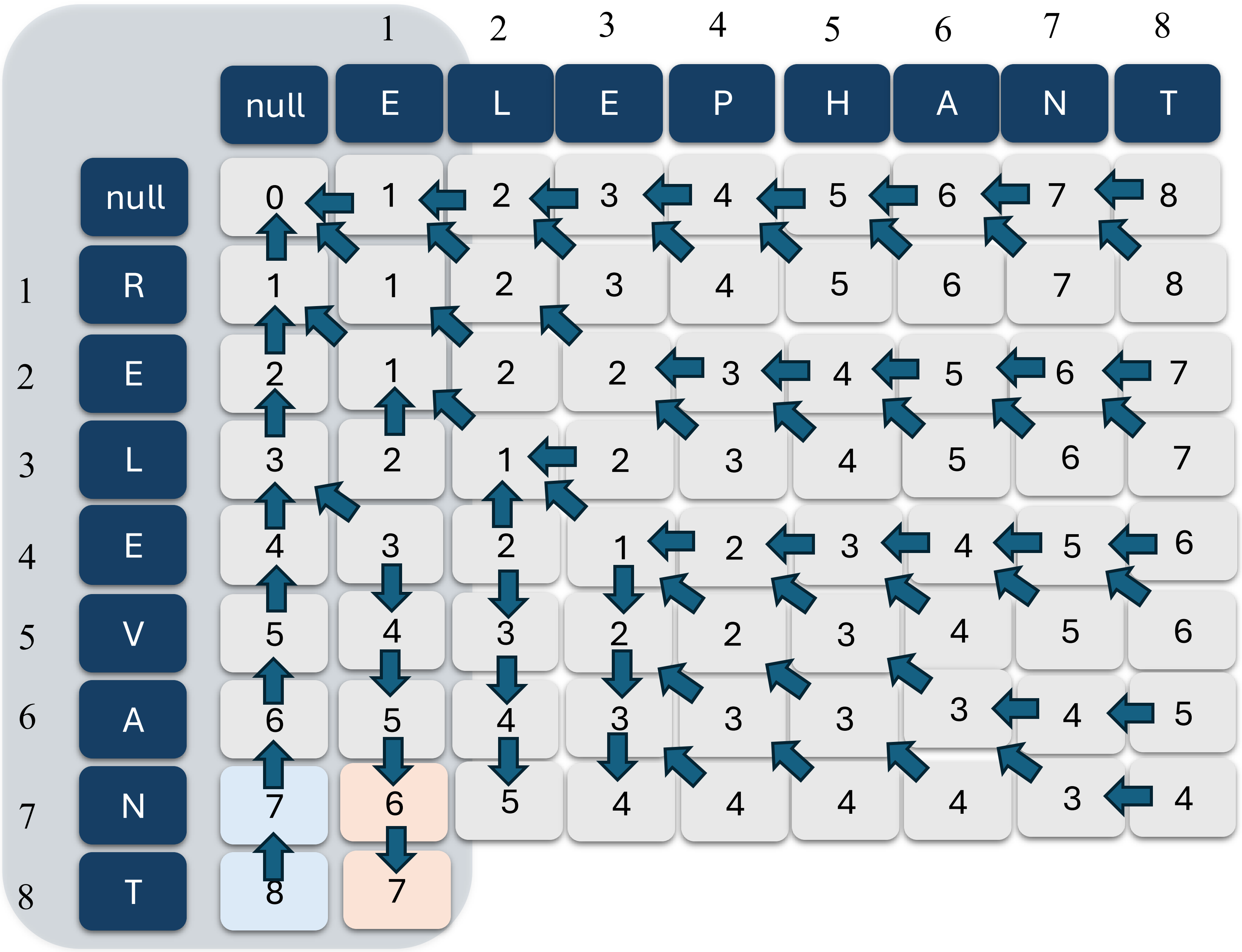
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8								

$$E(i, j) = min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

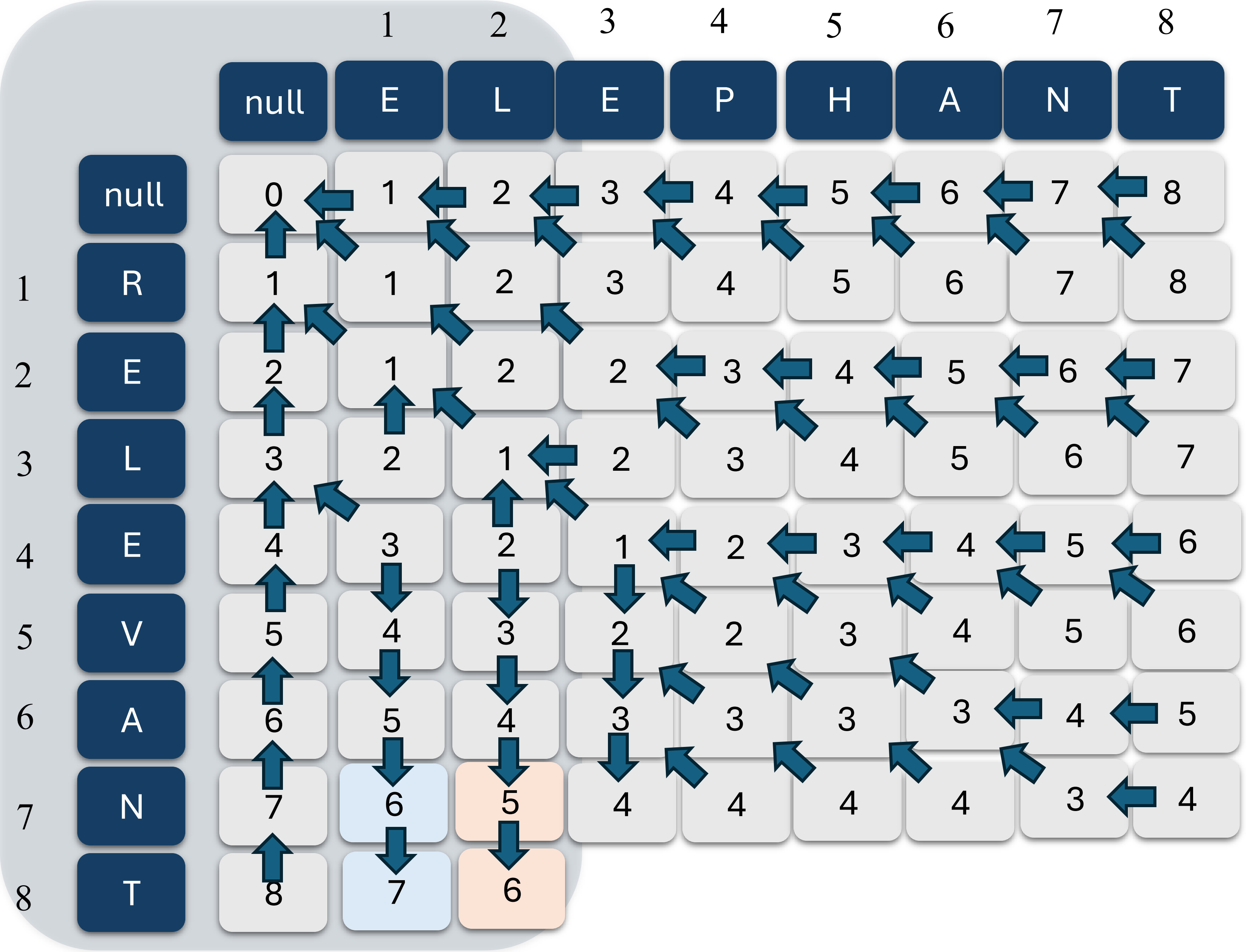
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7							

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

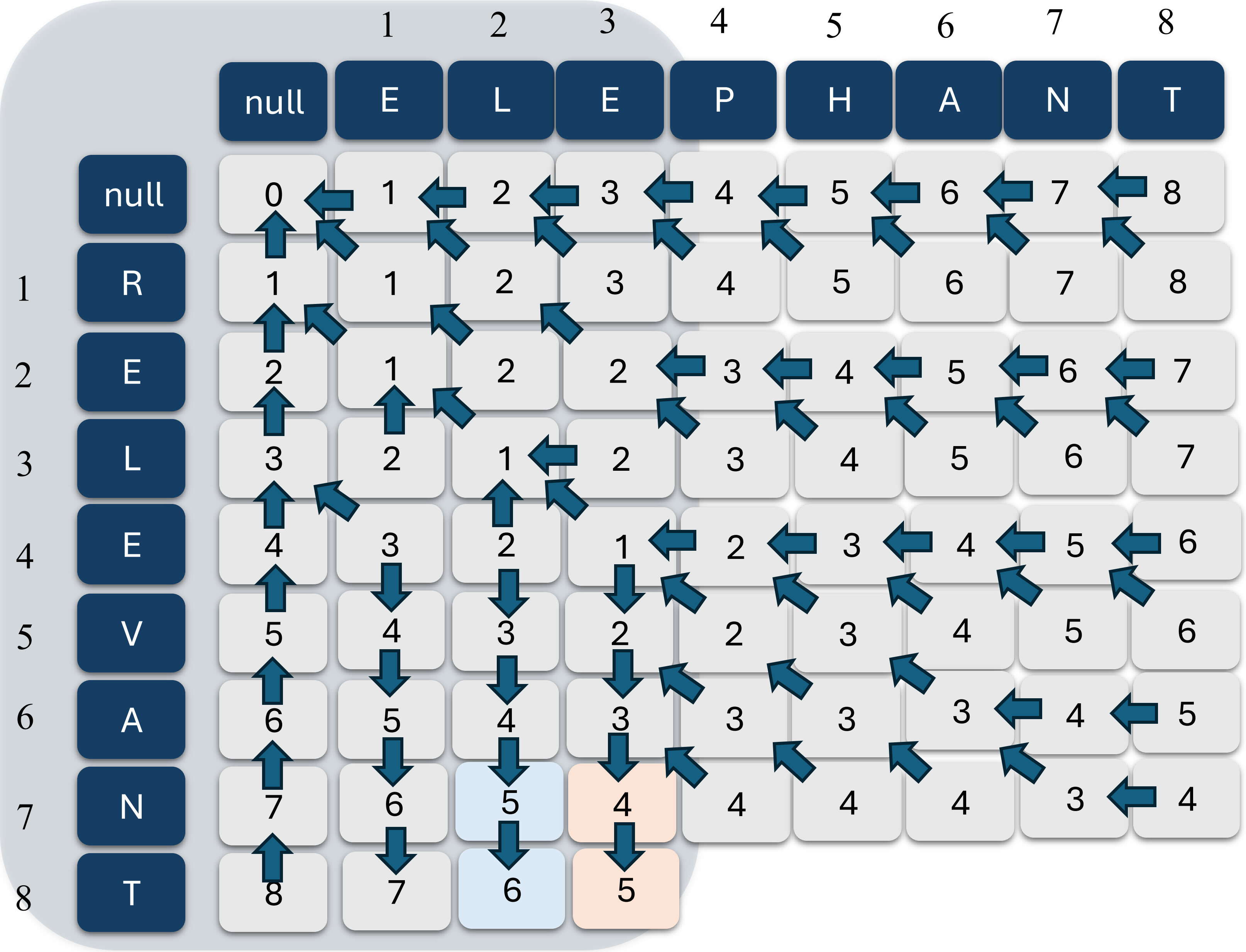
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6						

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

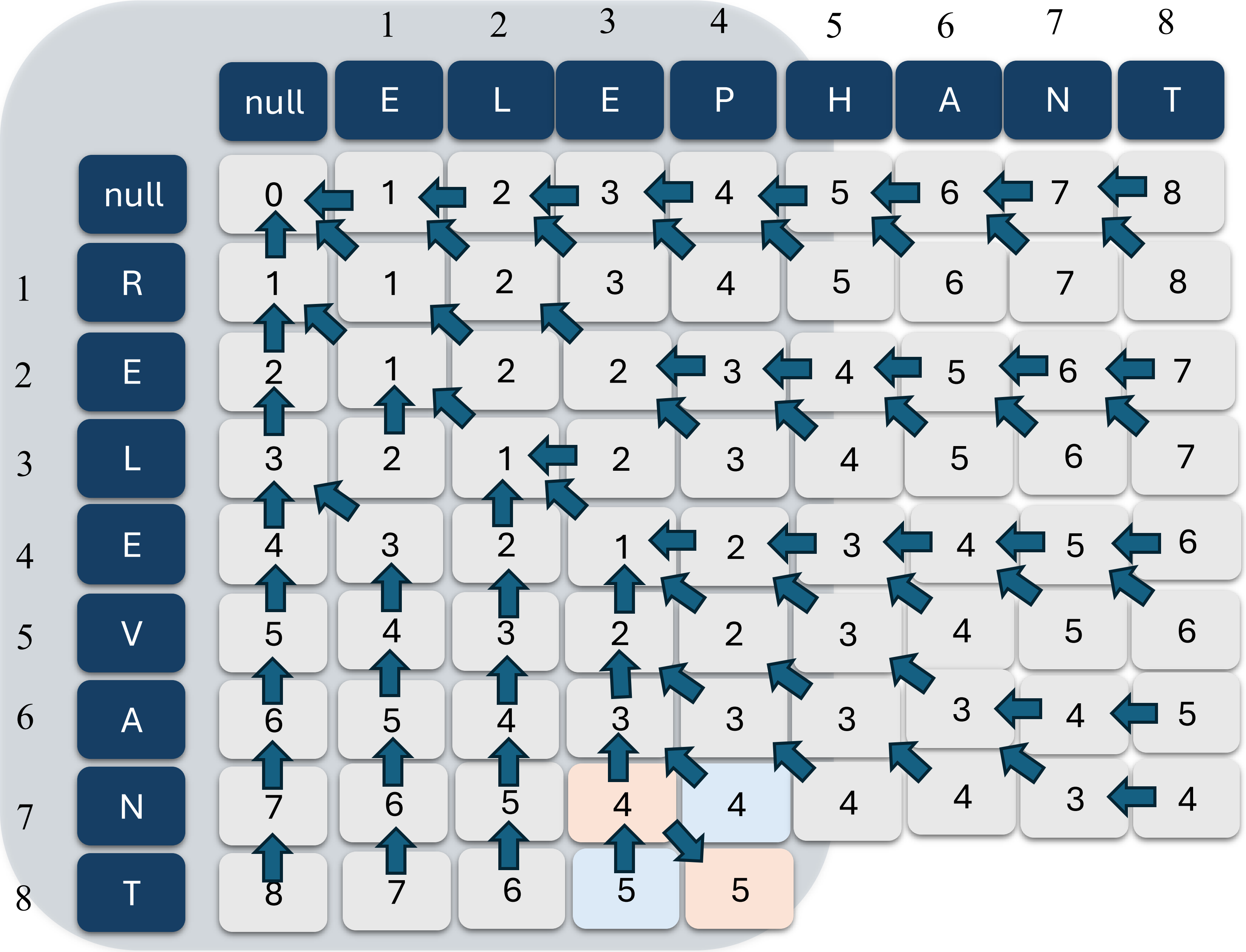
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5					

$$E(i, j) = \min \begin{cases} 1 + E(i\text{-}1, j) \\ 1 + E(i, j\text{-}1) \\ diff(i, j) + E(i\text{-}1, j\text{-}1) \end{cases}$$

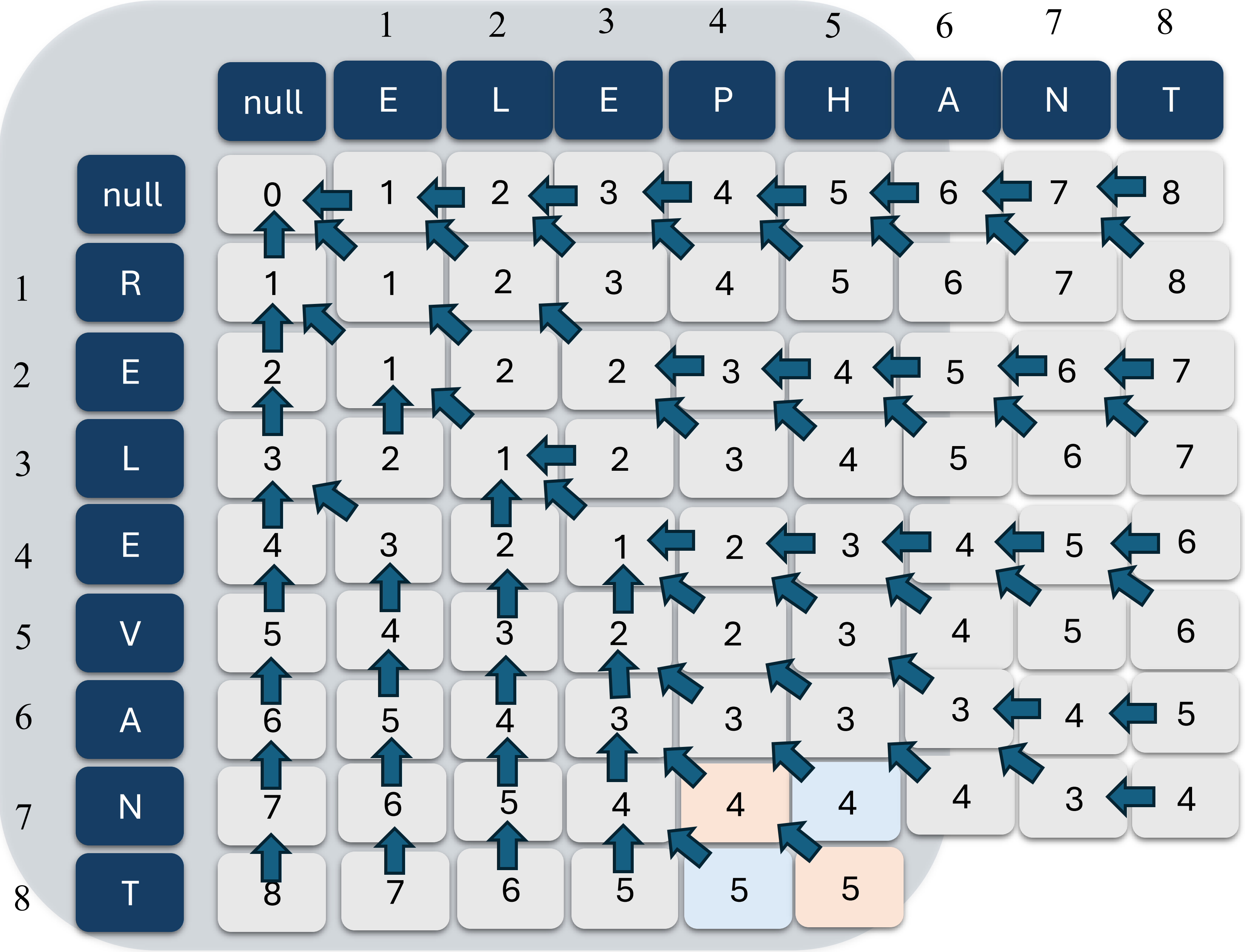
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5				

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

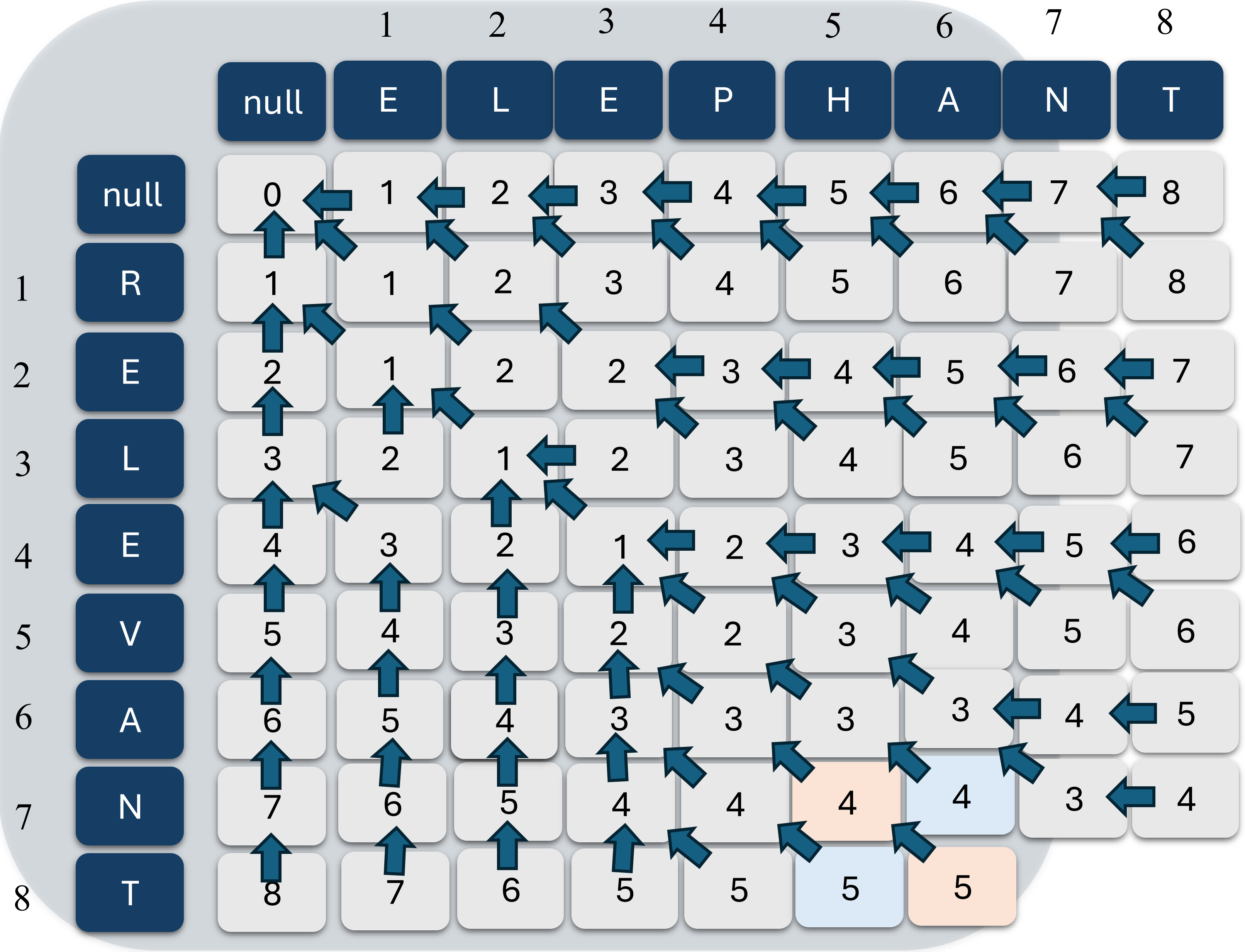
↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5			

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

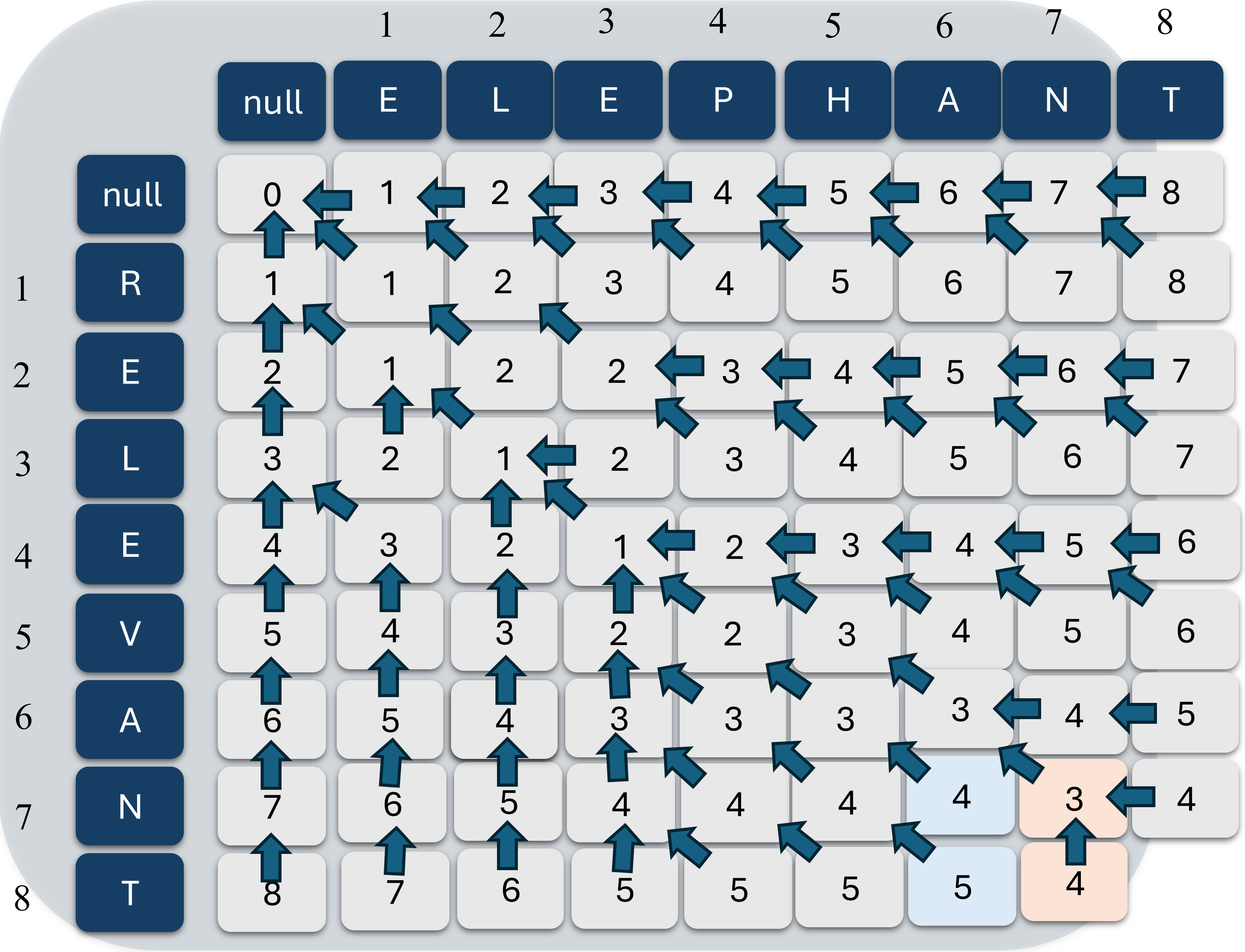
↖ ← ↑



		0	1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5		

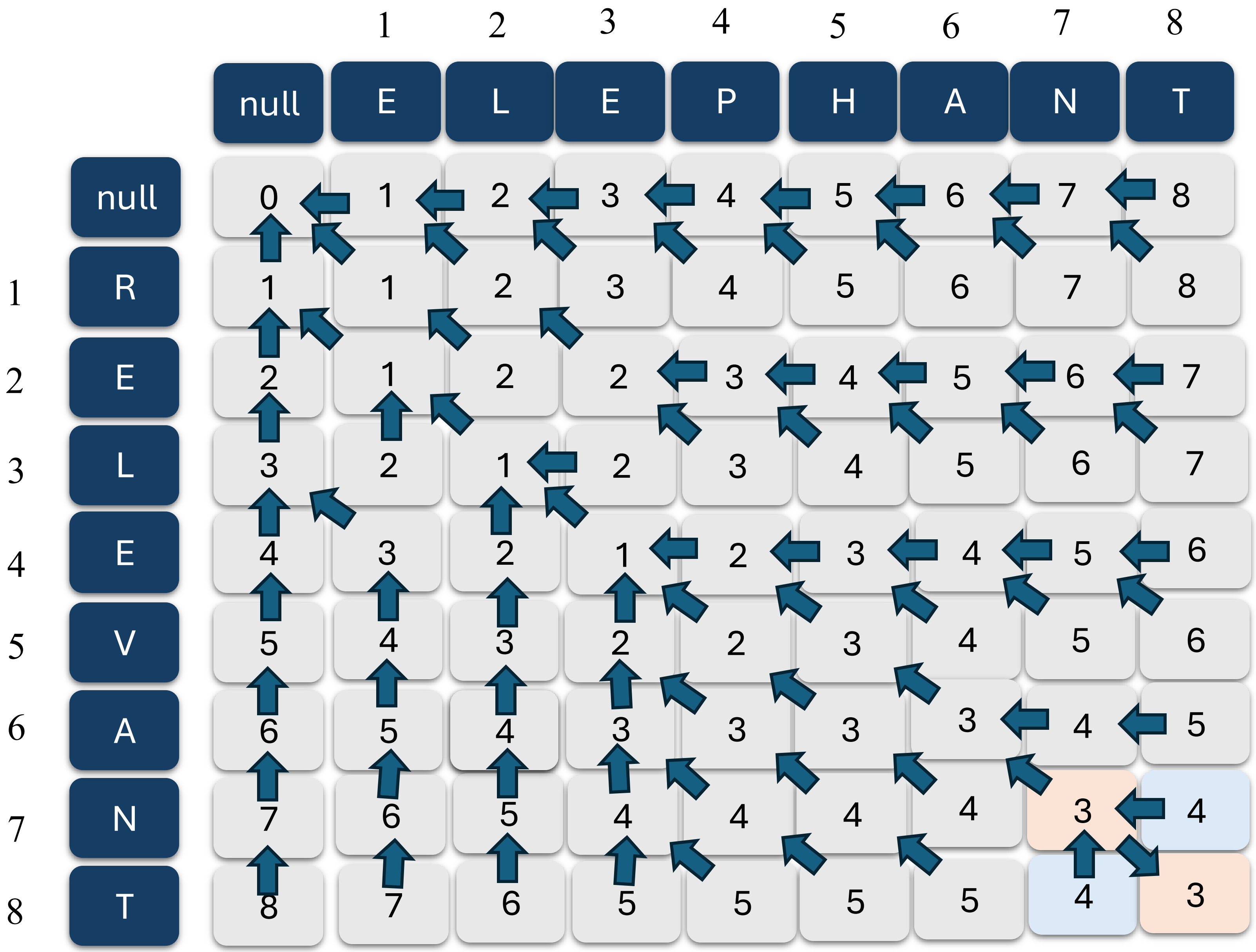
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

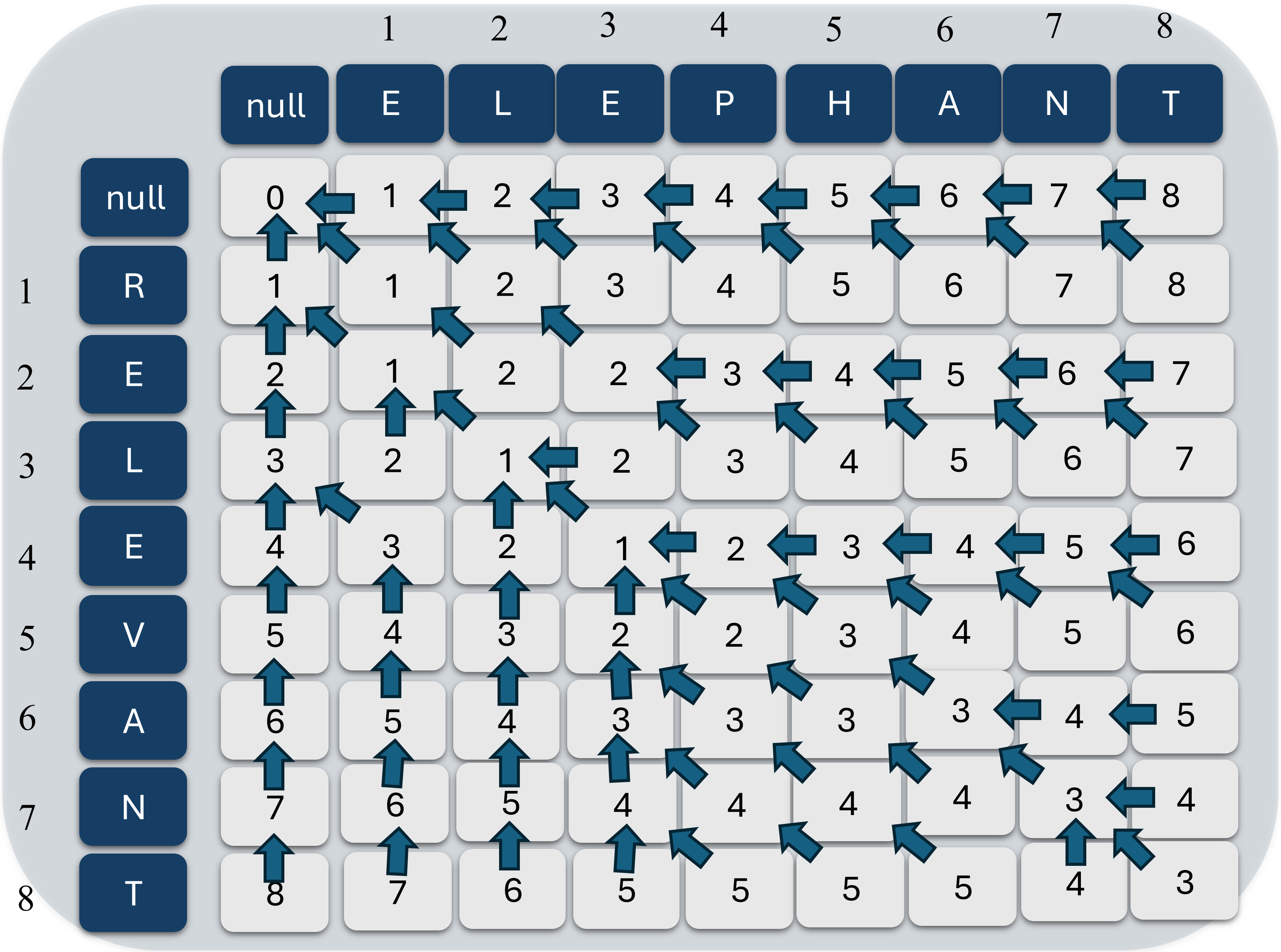


		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

↖ ← ↑

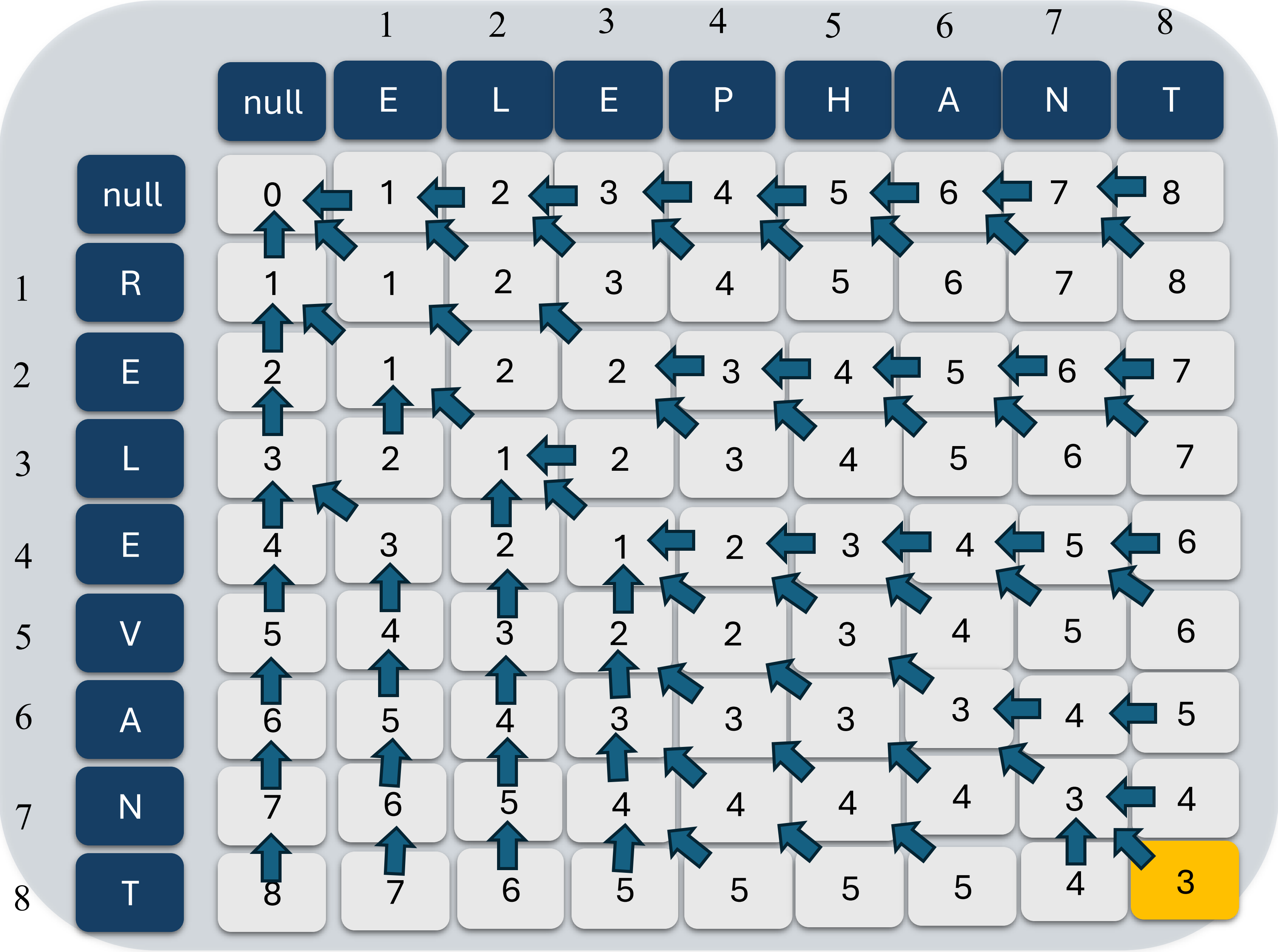
$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

↖ ← ↑

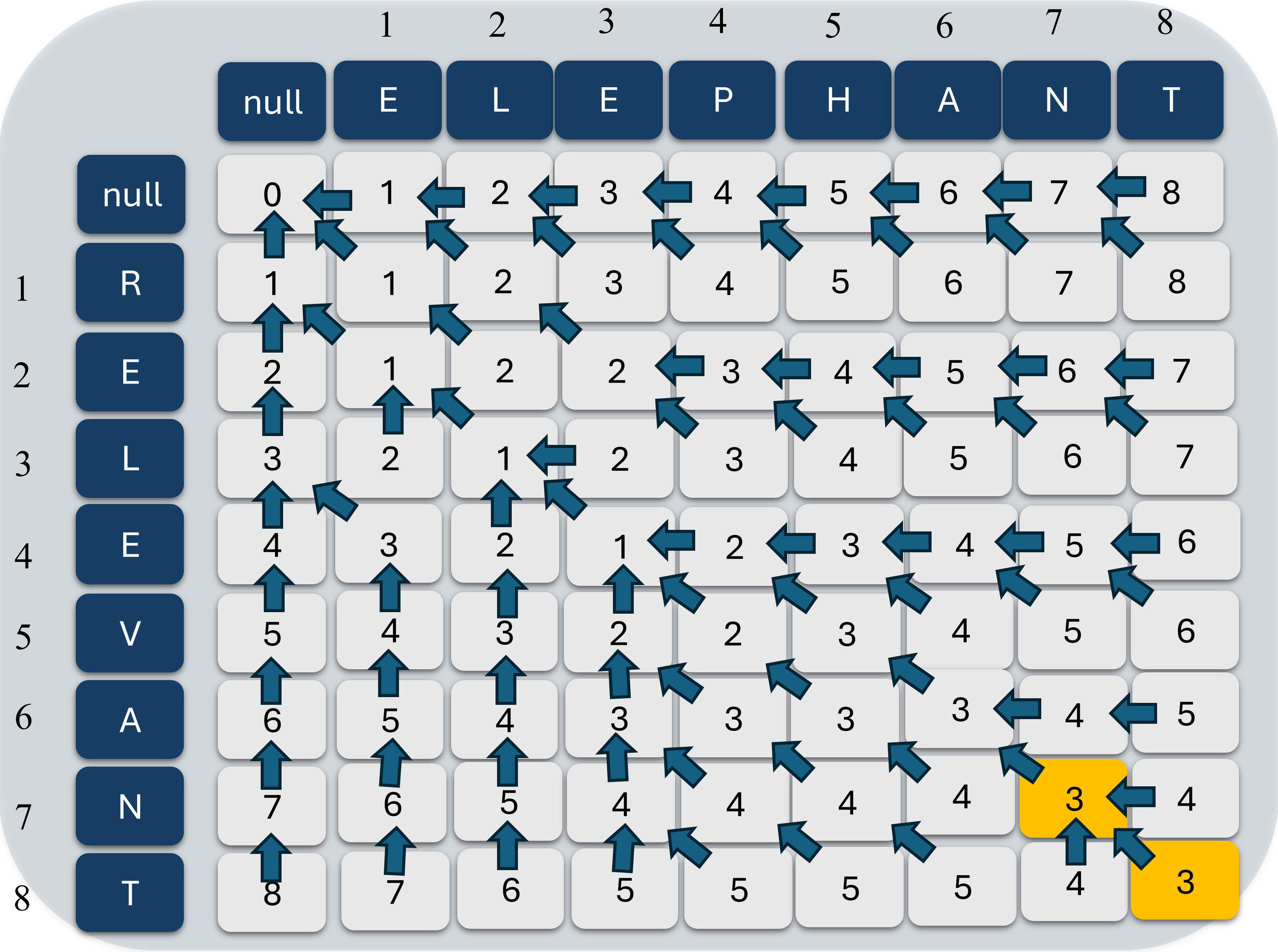


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

$$E(i, j) = \min \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ diff(i, j) + E(i-1, j-1) \end{cases}$$

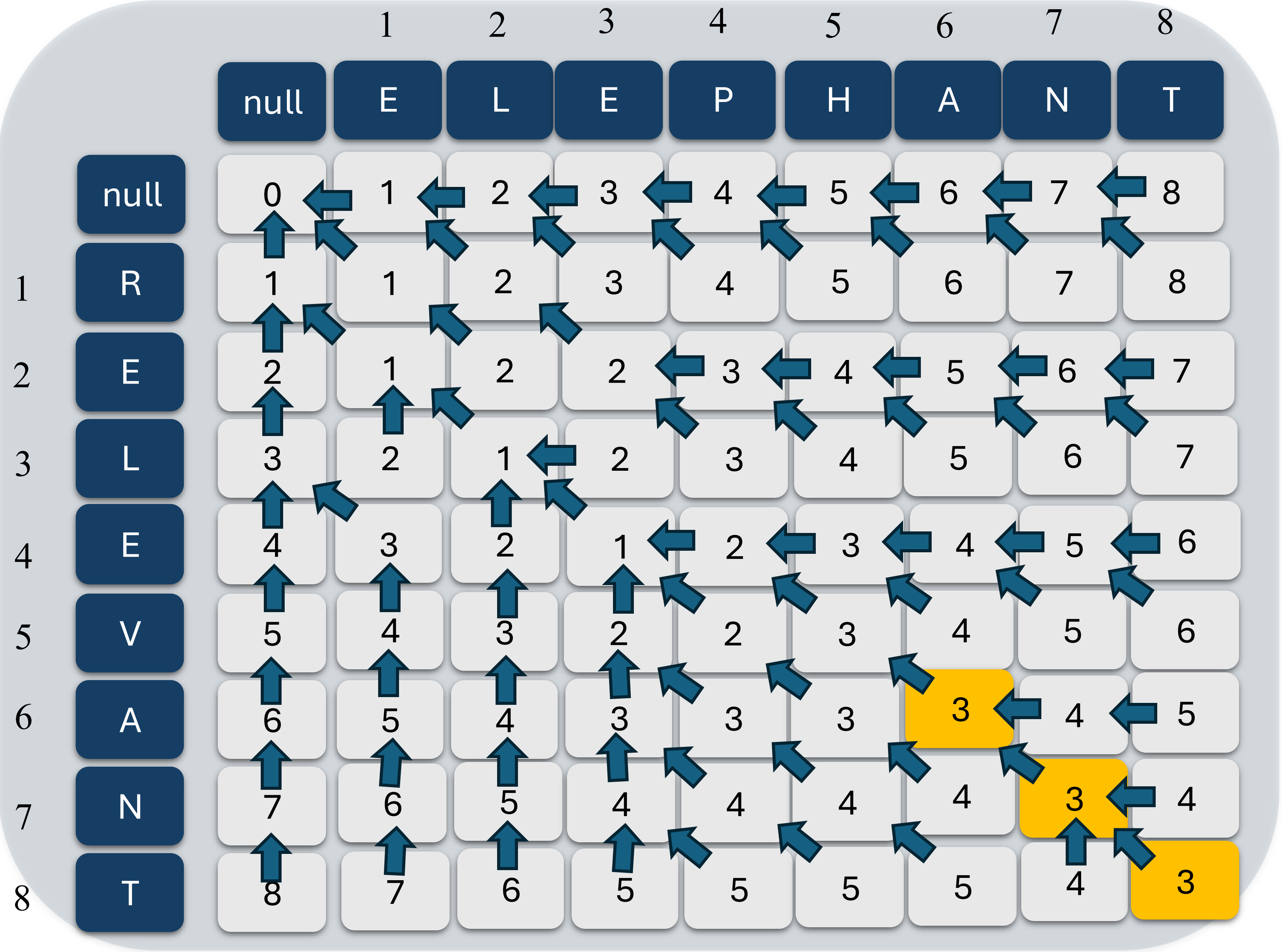


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

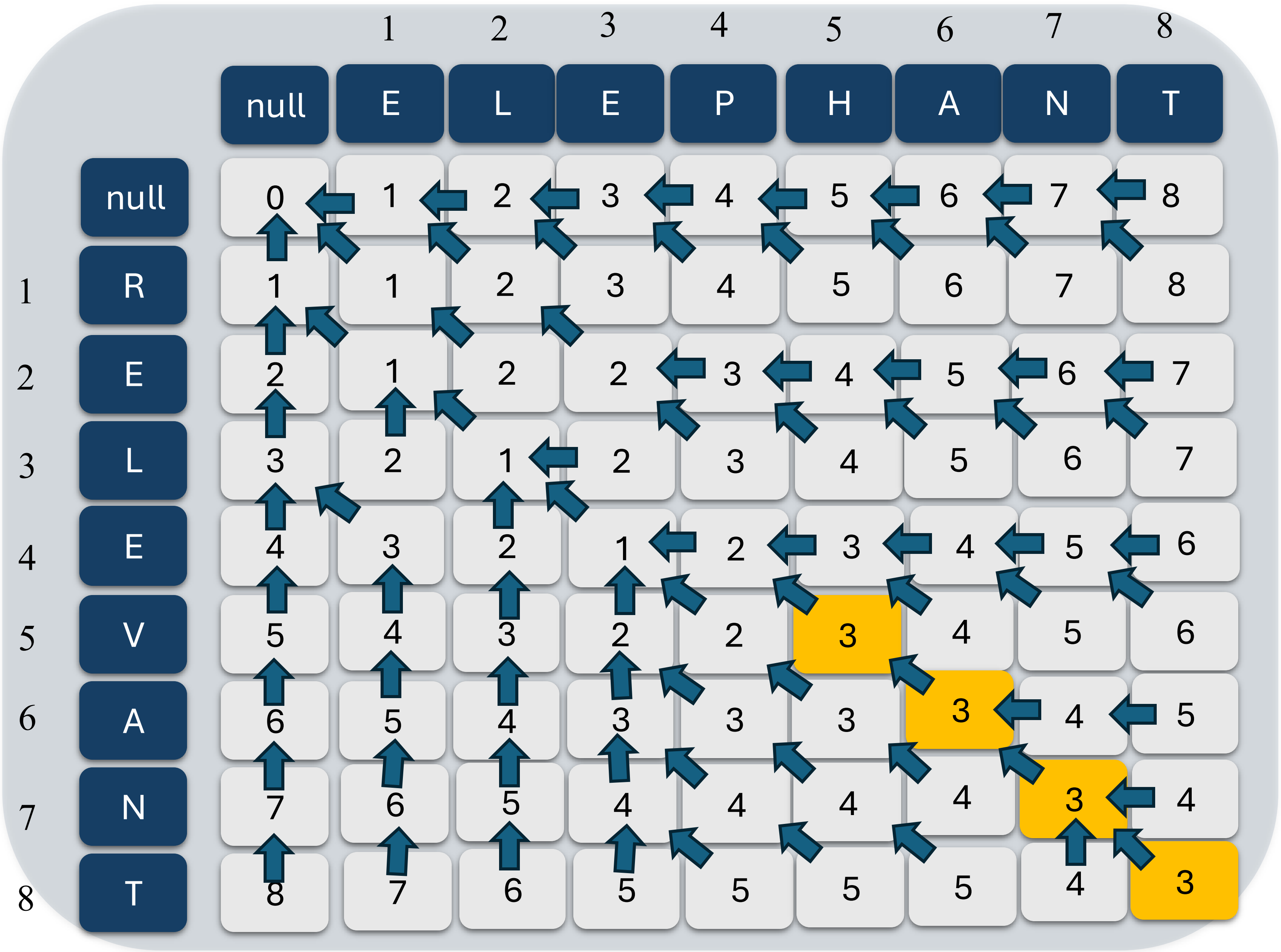


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

↖ ← ↑

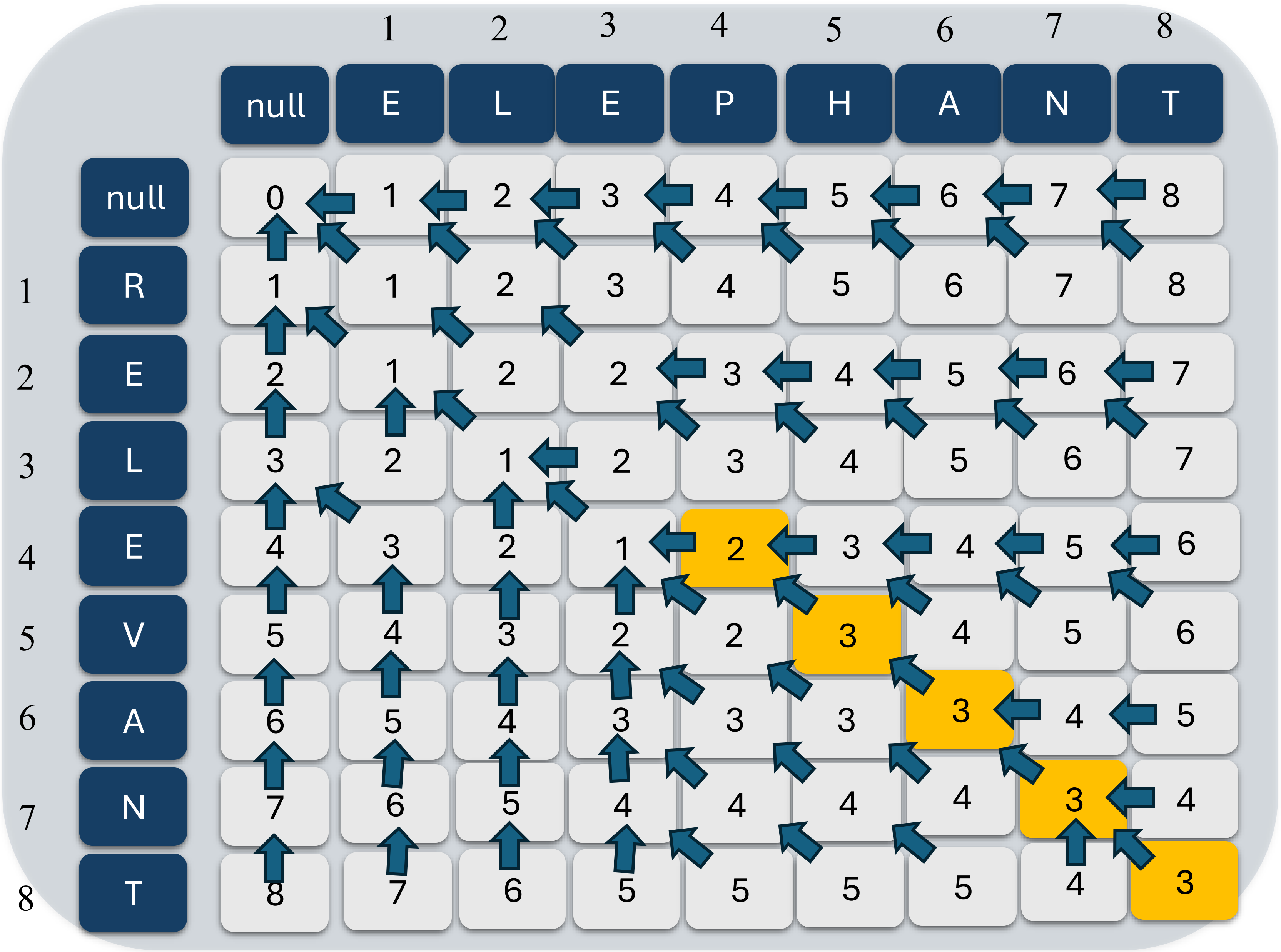


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

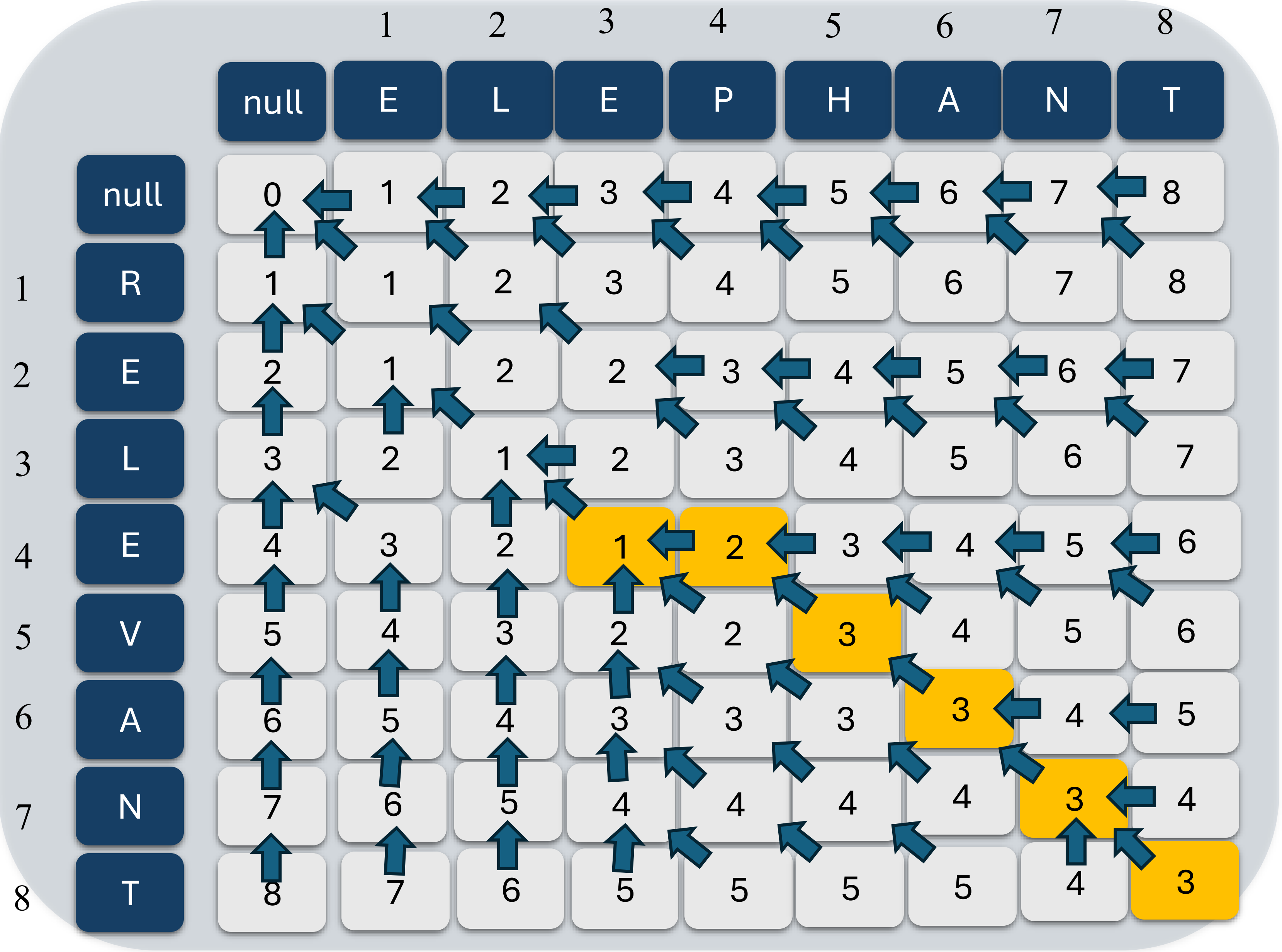


		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
1 R	1	1	2	3	4	5	6	7	8
2 E	2	1	2	2	3	4	5	6	7
3 L	3	2	1	2	3	4	5	6	7
4 E	4	3	2	1	2	3	4	5	6
5 V	5	4	3	2	2	3	4	5	6
6 A	6	5	4	3	3	3	3	4	5
7 N	7	6	5	4	4	4	4	3	4
8 T	8	7	6	5	5	5	5	4	3

↖ ← ↑

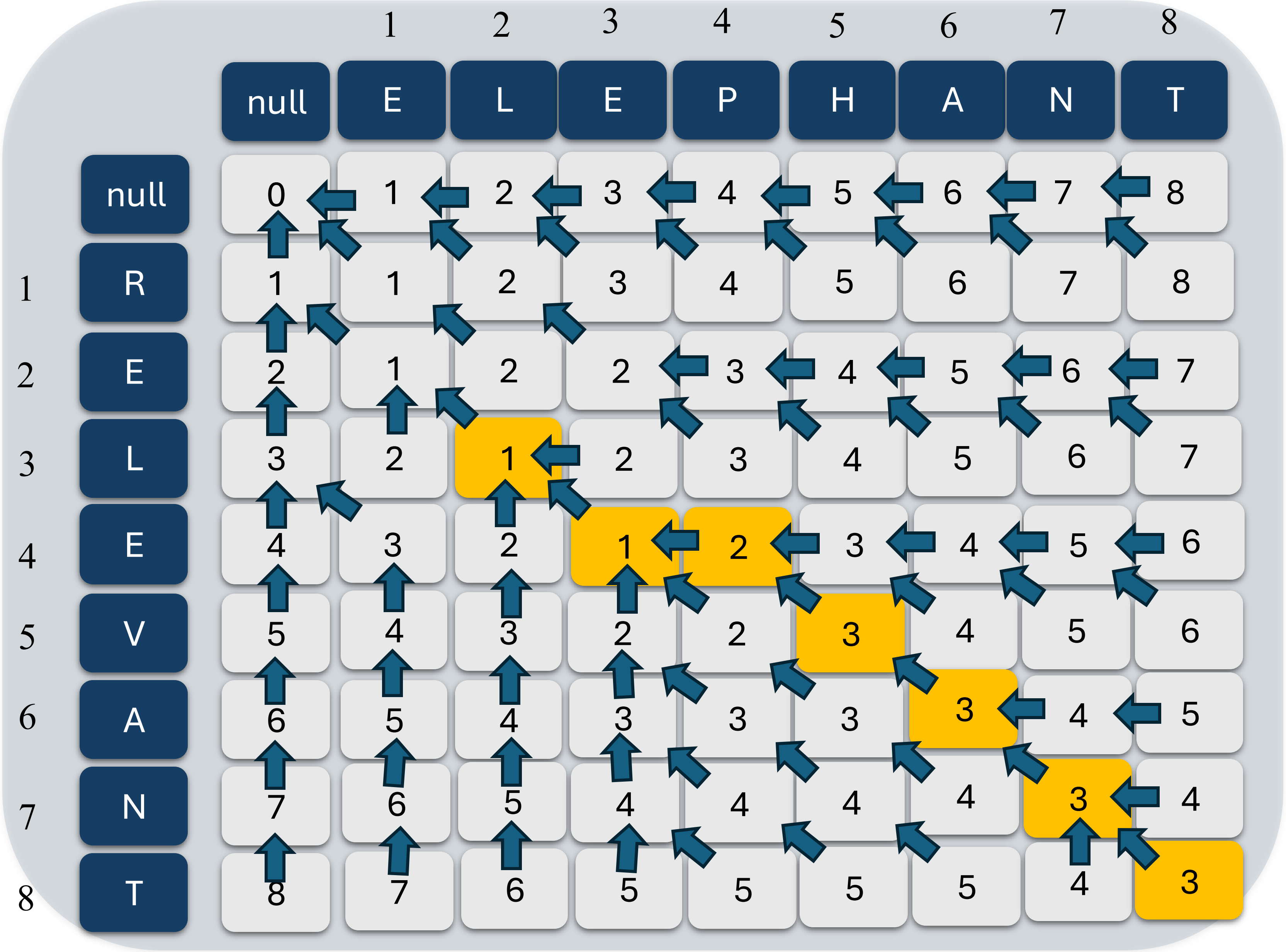


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

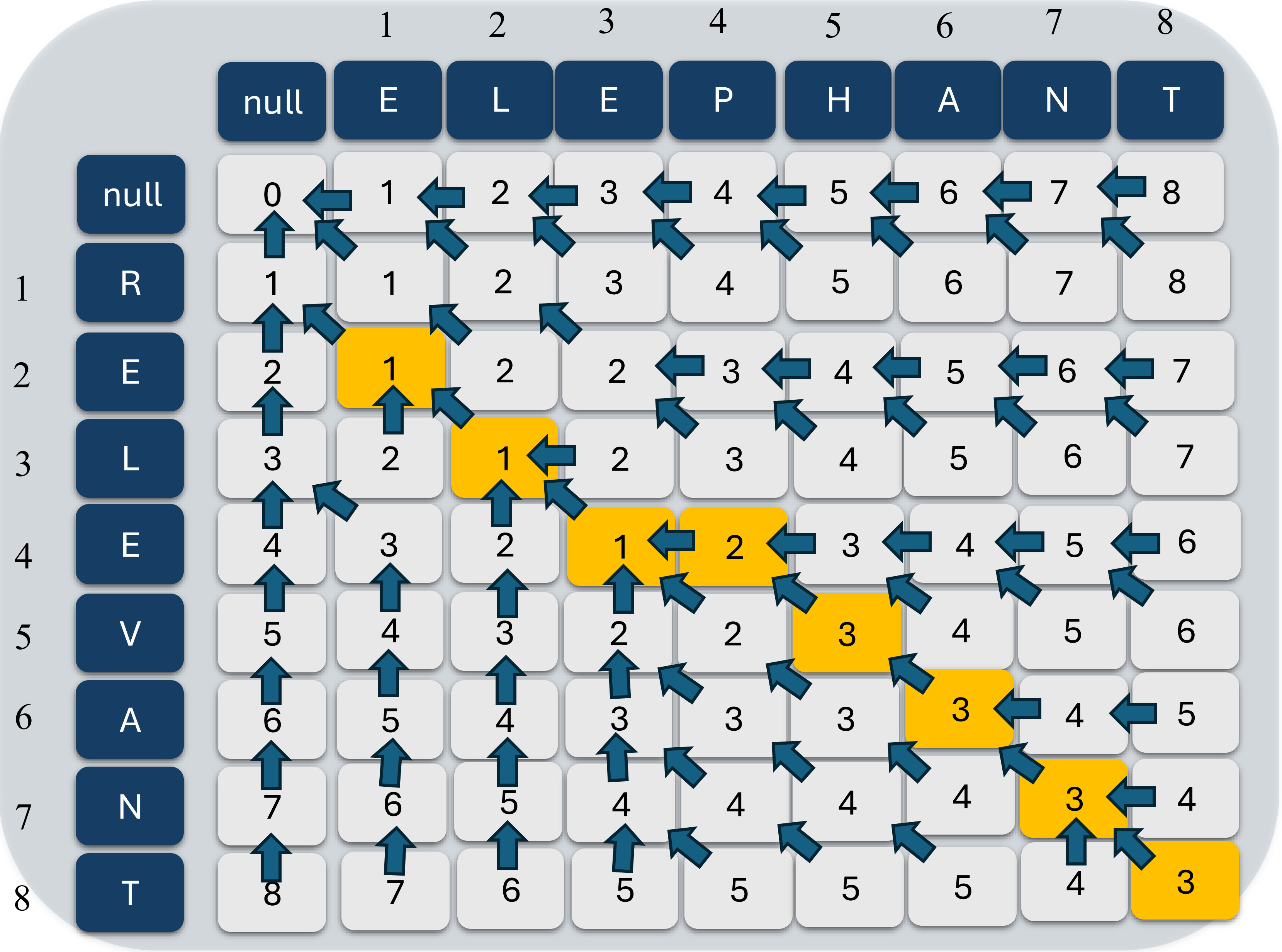
↖ ← ↑



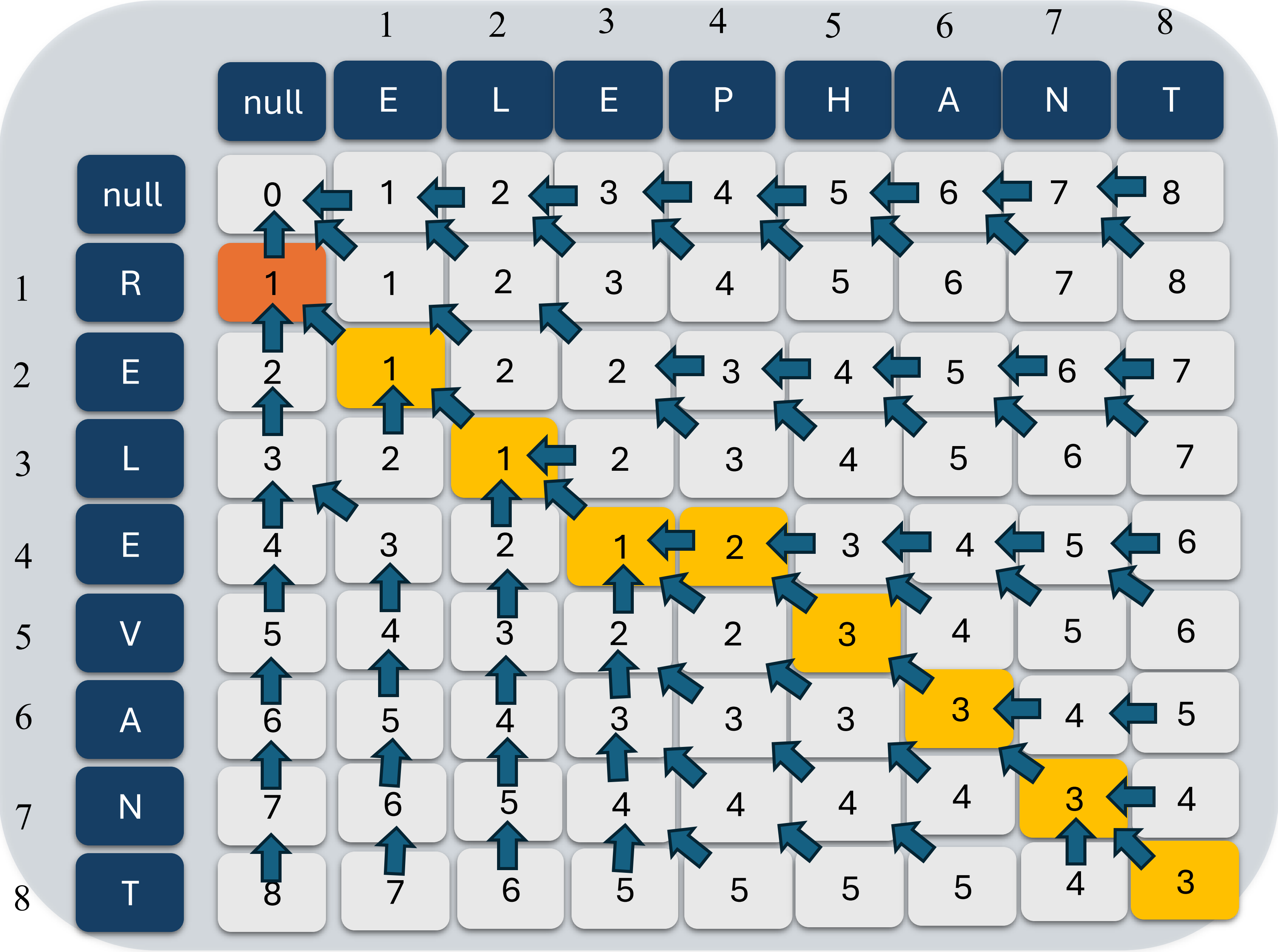
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

↖ ← ↑

↖ ← ↑

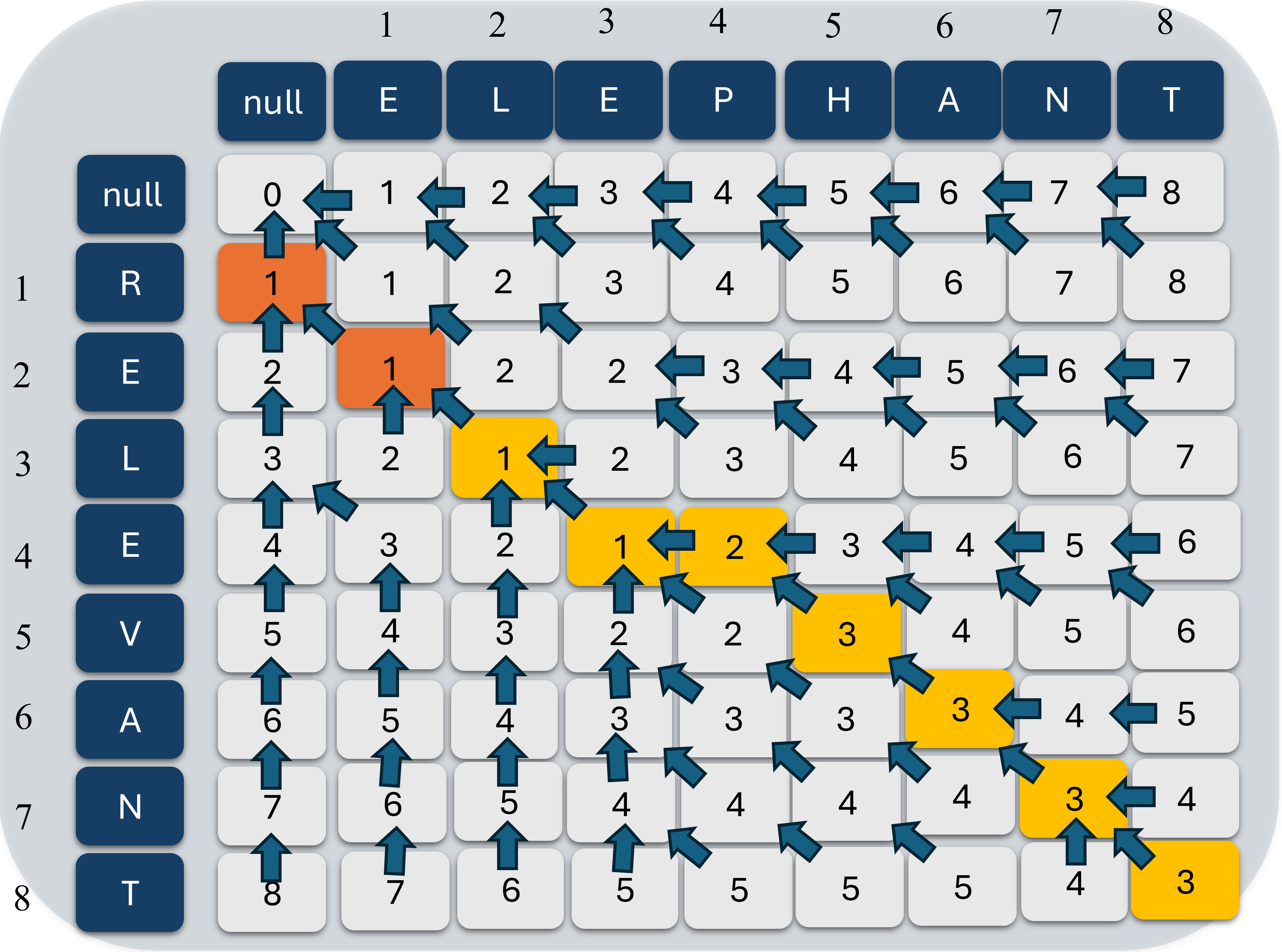


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3



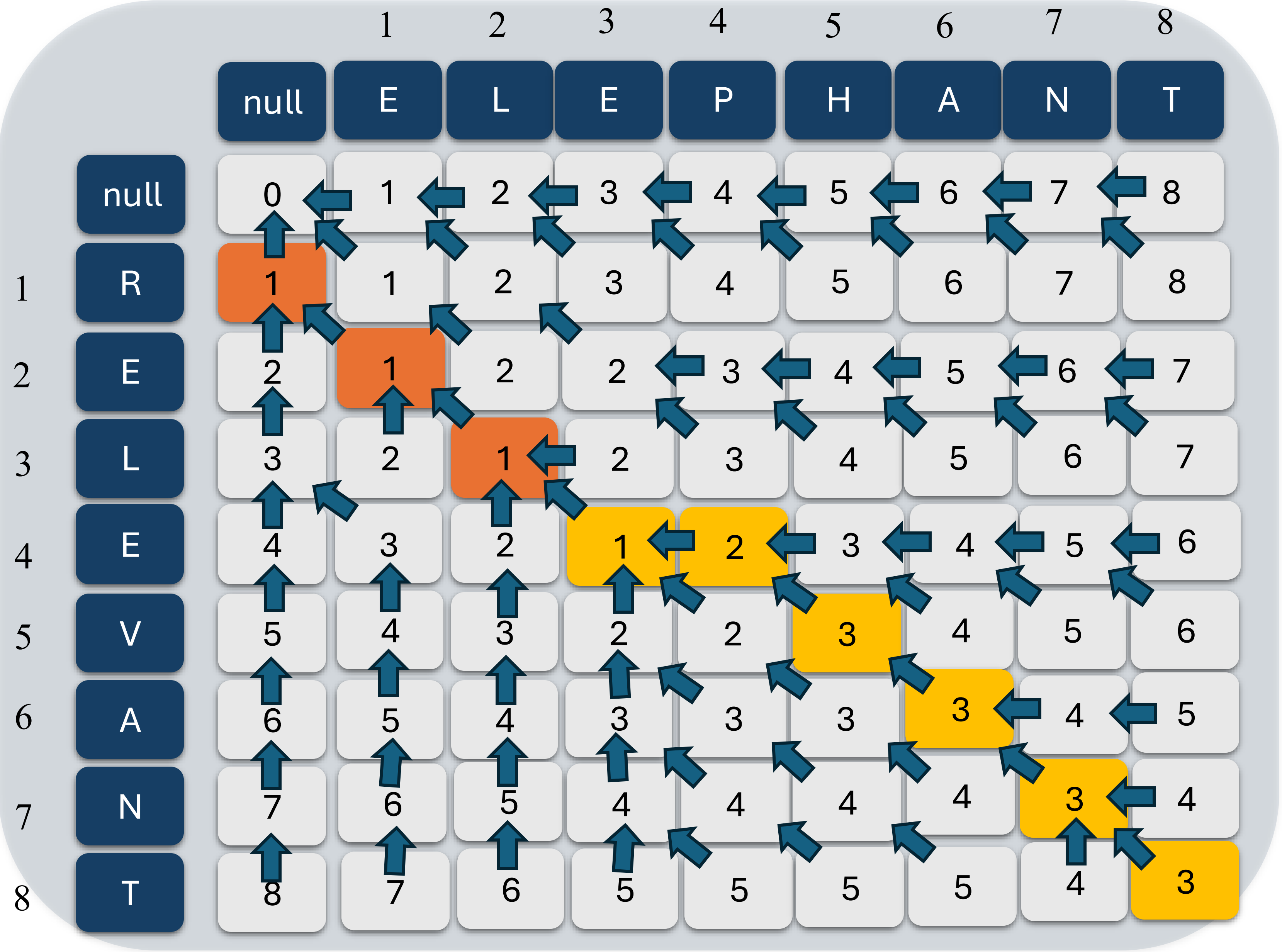
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

R	E	L	E	V	A	N	T
-							



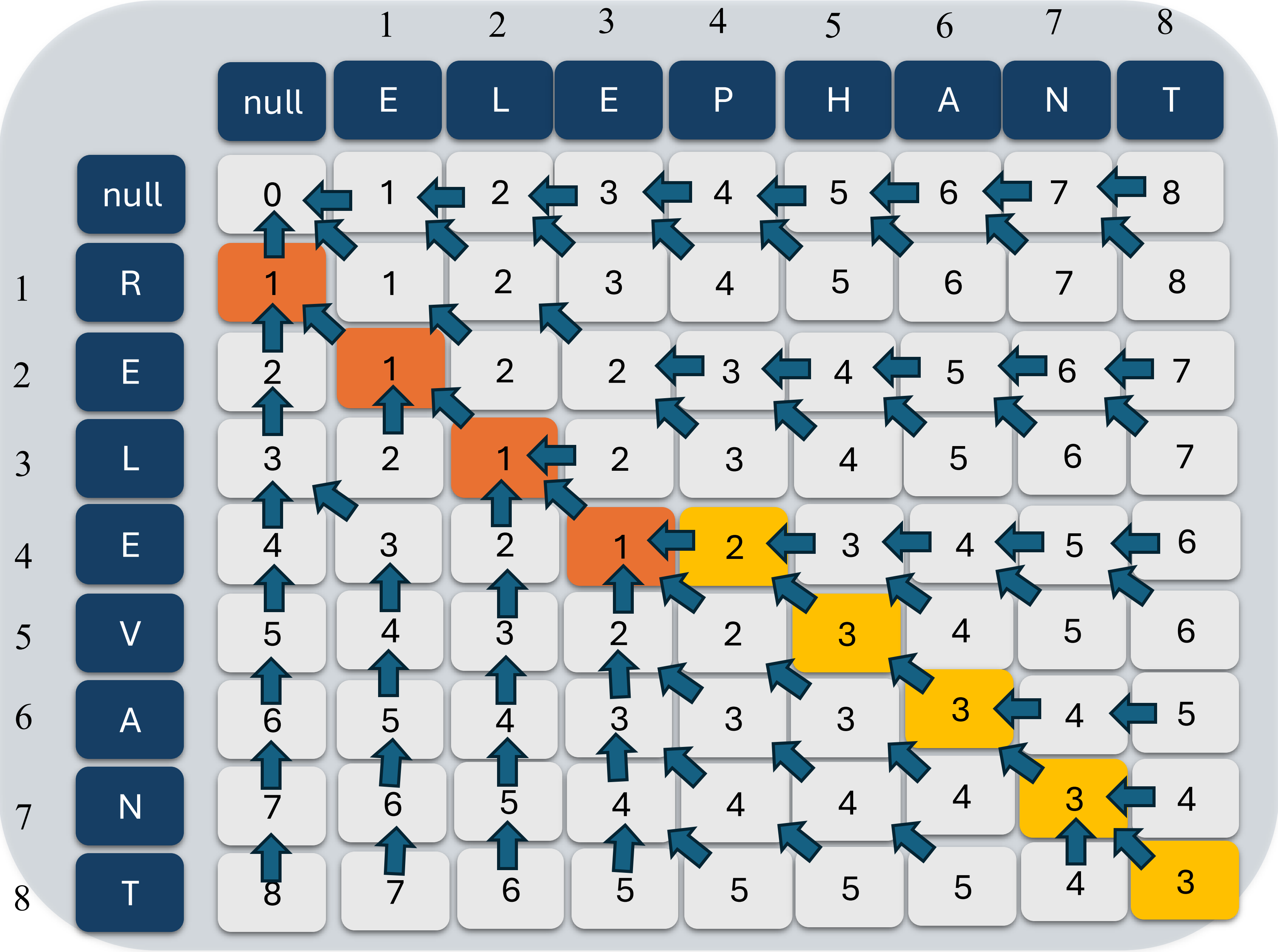
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

R	E	L	E	V	A	N	T
-	E						



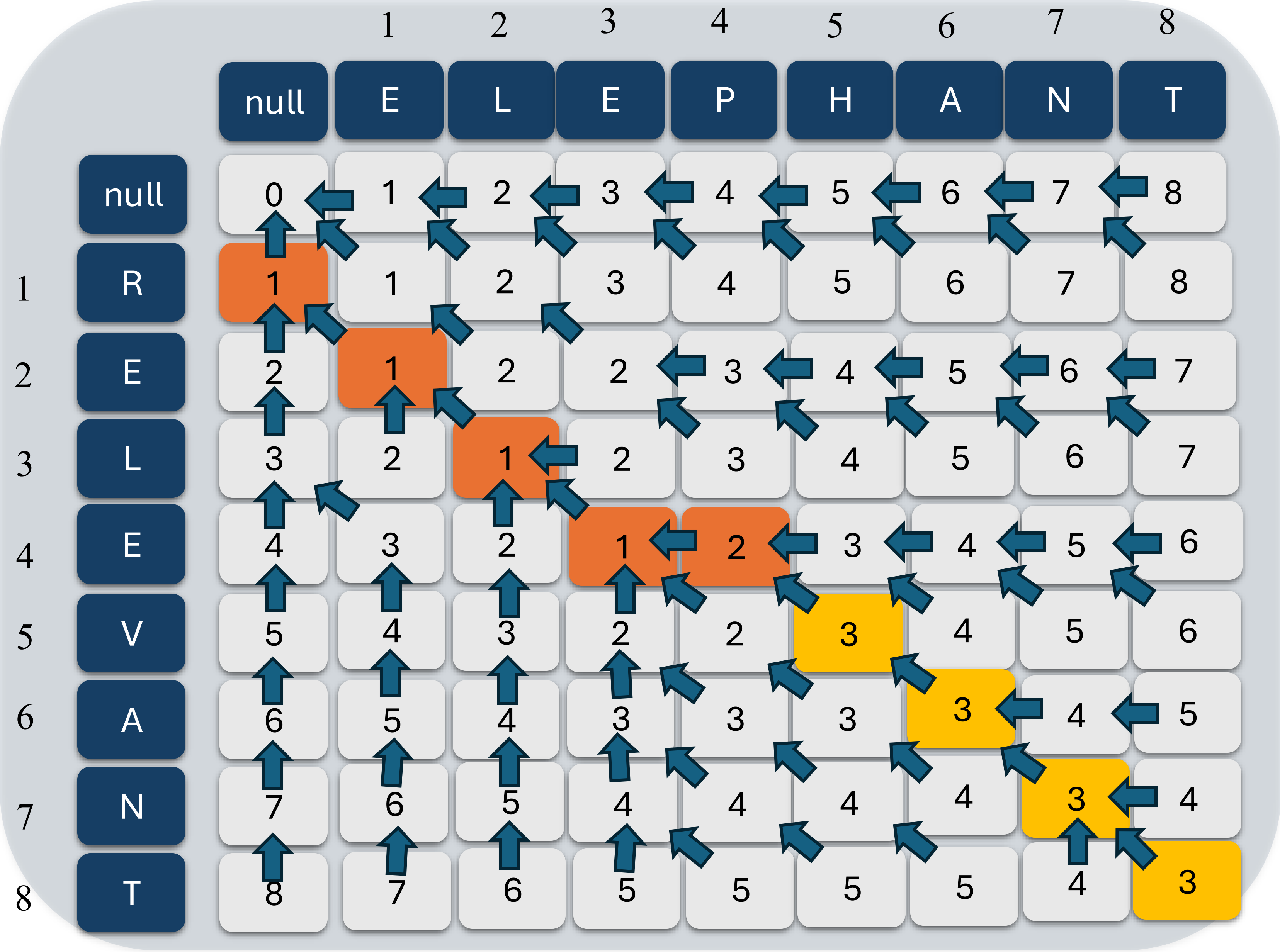
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

R	E	L	E	V	A	N	T
-	E	L					



			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

R E L E V A N T
- E L E

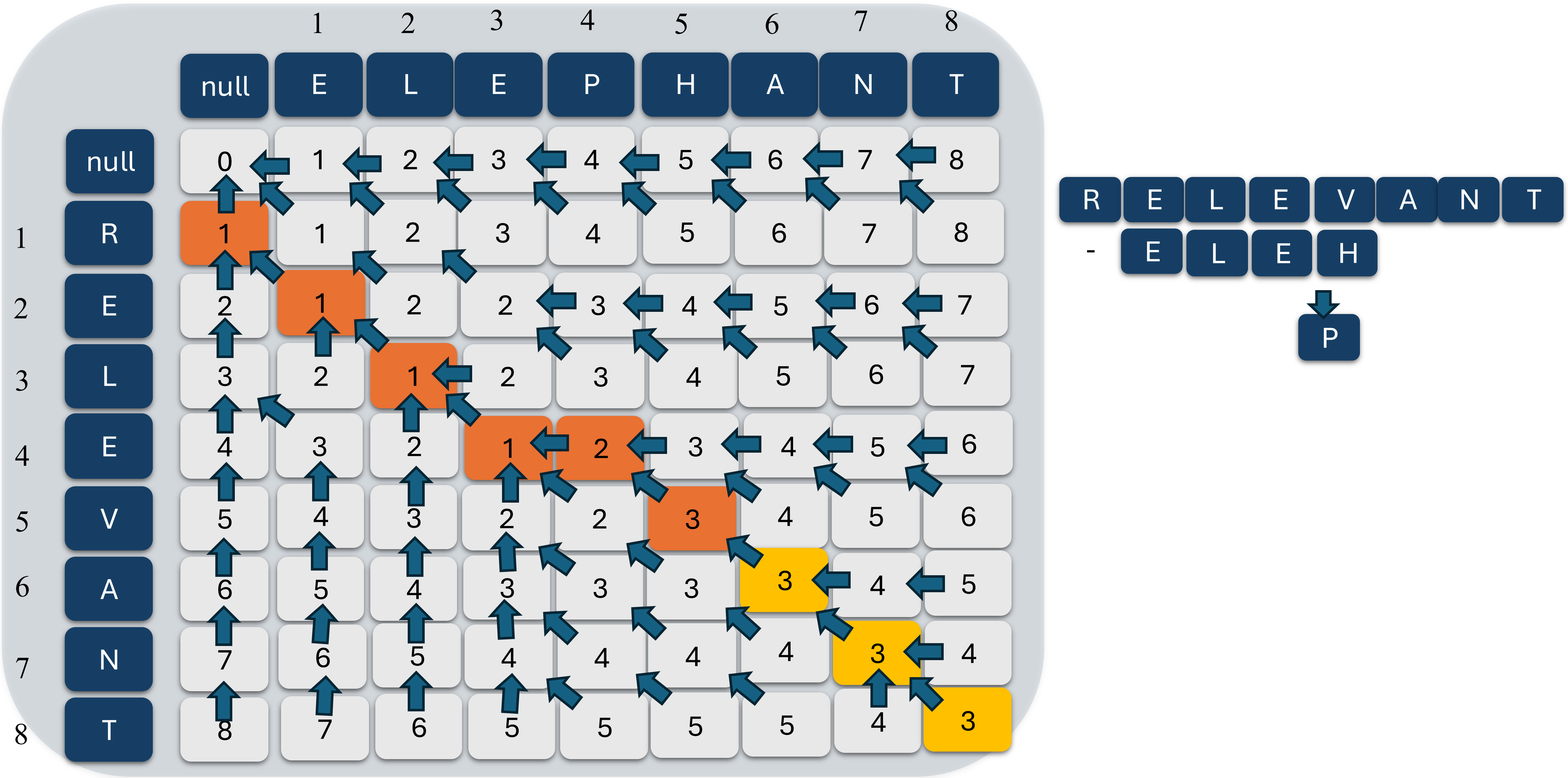


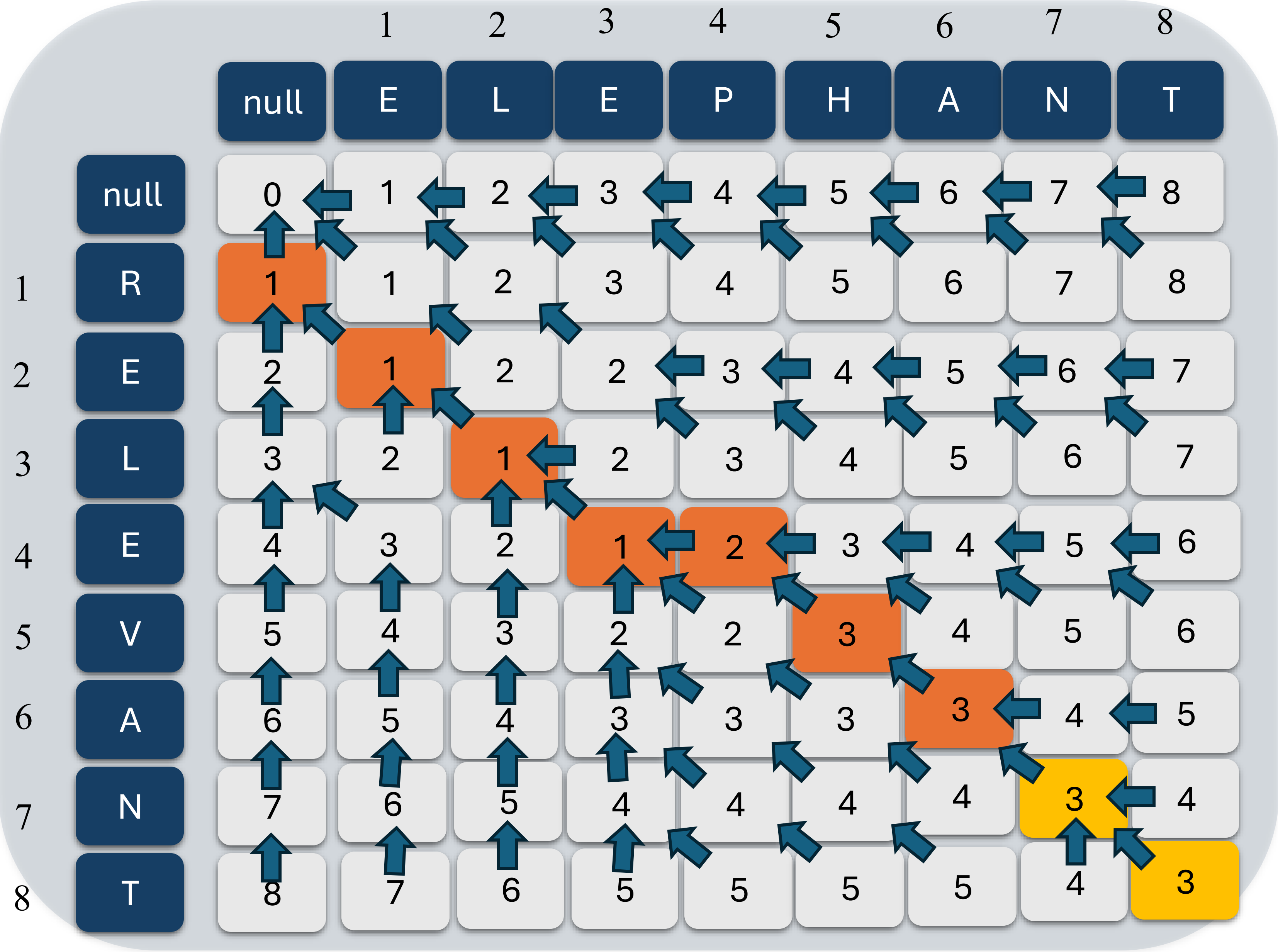
			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

R E L E V A N T

- E L E

P



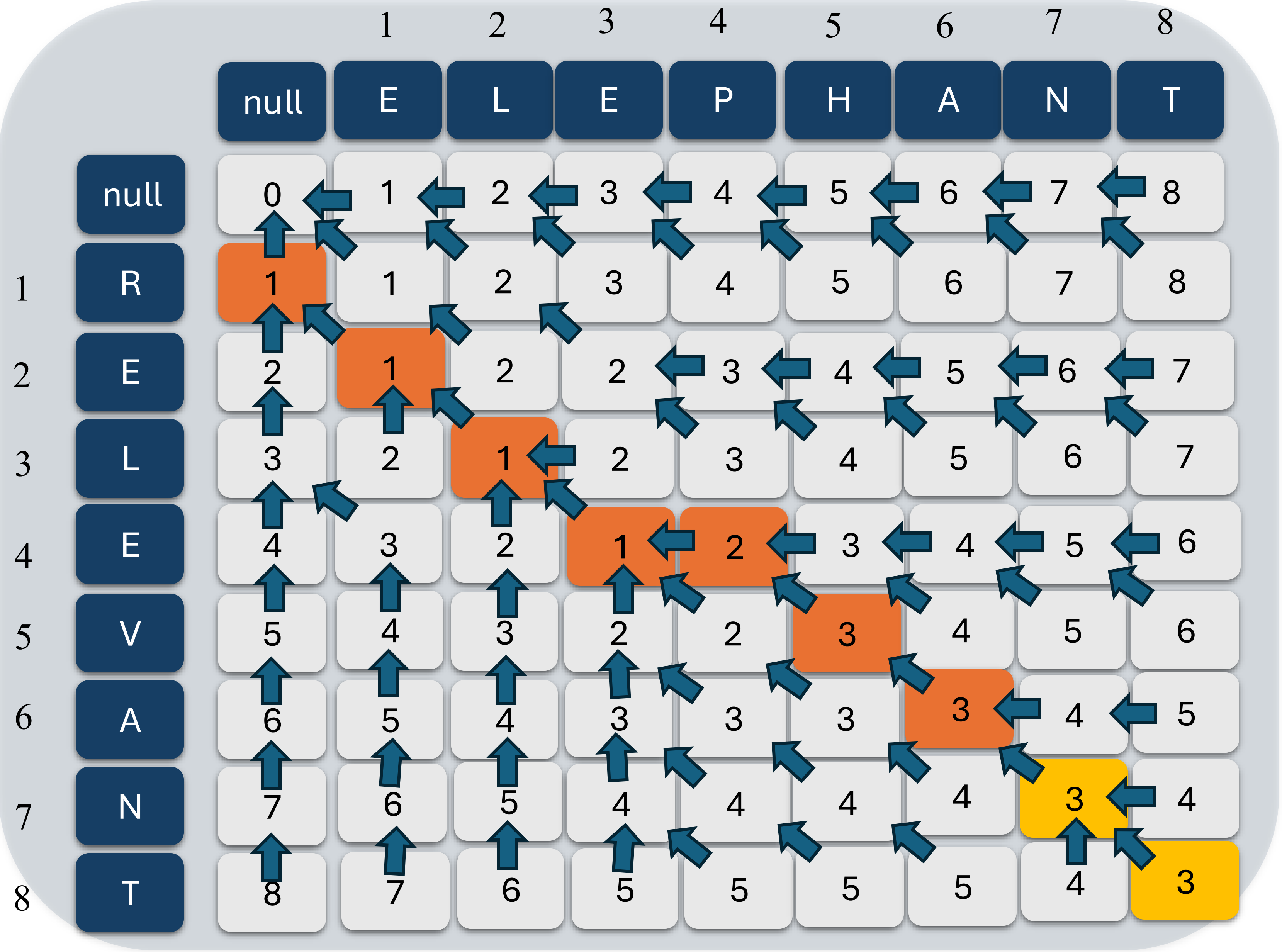


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

R E L E V A N T

- E L E H A

P

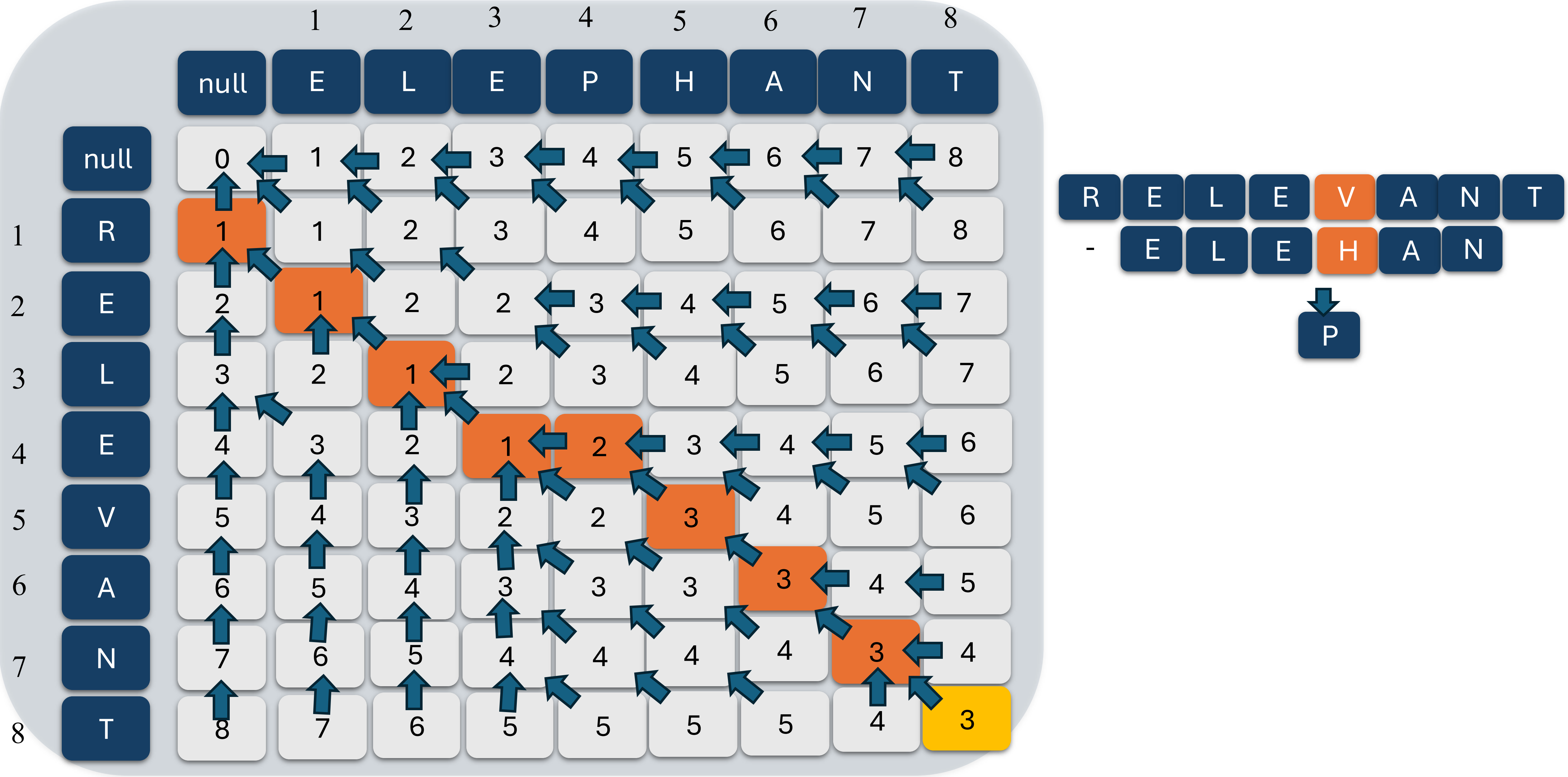


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

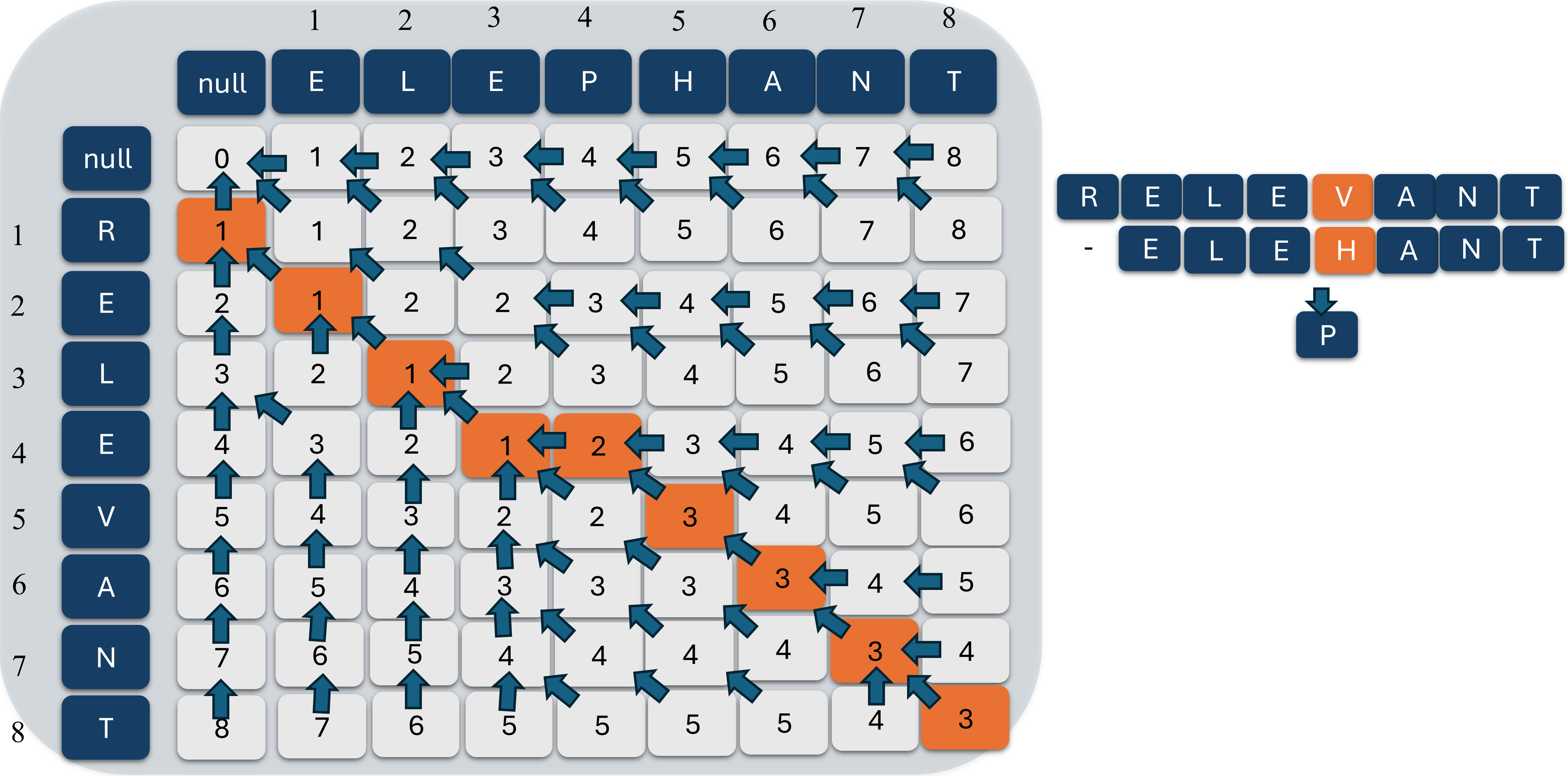
R E L E V A N T

- E L E H A

P



		1	2	3	4	5	6	7	8
	null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

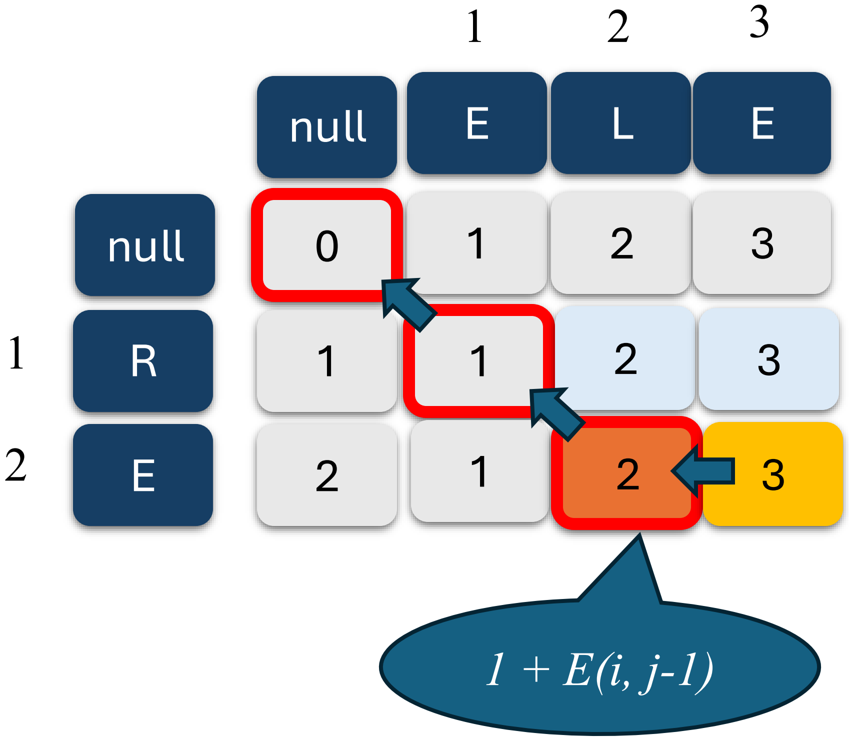


			1	2	3	4	5	6	7	8
		null	E	L	E	P	H	A	N	T
	null	0	1	2	3	4	5	6	7	8
1	R	1	1	2	3	4	5	6	7	8
2	E	2	1	2	2	3	4	5	6	7
3	L	3	2	1	2	3	4	5	6	7
4	E	4	3	2	1	2	3	4	5	6
5	V	5	4	3	2	2	3	4	5	6
6	A	6	5	4	3	3	3	3	4	5
7	N	7	6	5	4	4	4	4	3	4
8	T	8	7	6	5	5	5	5	4	3

Algorithm

```
minimum_edit_distance(sequence X, sequence Y):
    M = length(X)
    N = length(Y)
    #initialize table E of size M x N (M rows, N columns)
    for i = 0 to M:
        E(i, 0)= i # Base Case
    for j = 0 to N:
        E(0, j) = j # Base Case
    for i = 1 to M:
        for j = 1 to N:
                diff(i, j) = (X[i] == Y[j]) ? 0 : 1 # Diff
                E(i, j) = min{E(i-1, j)+1, E(i, j-1)+1, E(i-1, j-1)+diff(i, j)}
    return E(i, j)
```

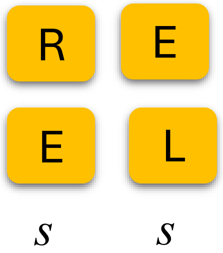
Some test cases



		1	2	3
	null	E	L	E
null	0	1	2	3
1 R	1	1	2	3
2 E	2	1	2	3

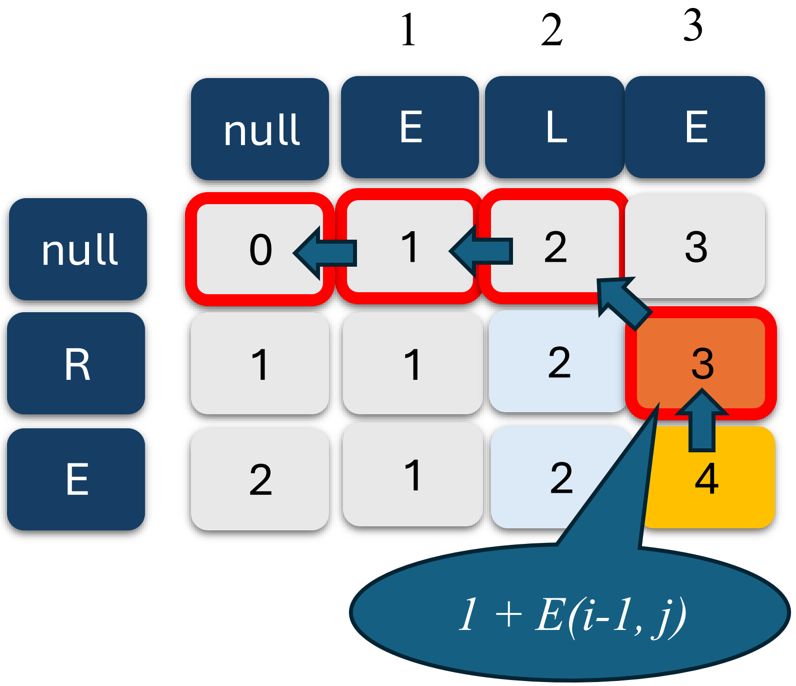
$1 + E(i, j\text{-}1)$

Cost of aligning RE and EL:



Cost of inserting E

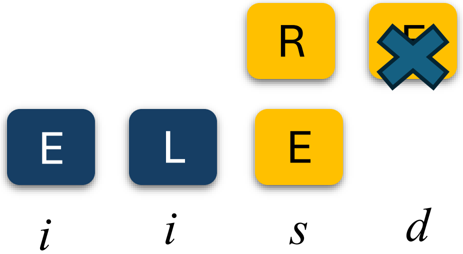
- *Replace R with E*
- *Replace E with L*
- *Insert E (+1)*



		1	2	3
	null	E	L	E
null	0	1	2	3
R	1	1	2	3
E	2	1	2	4

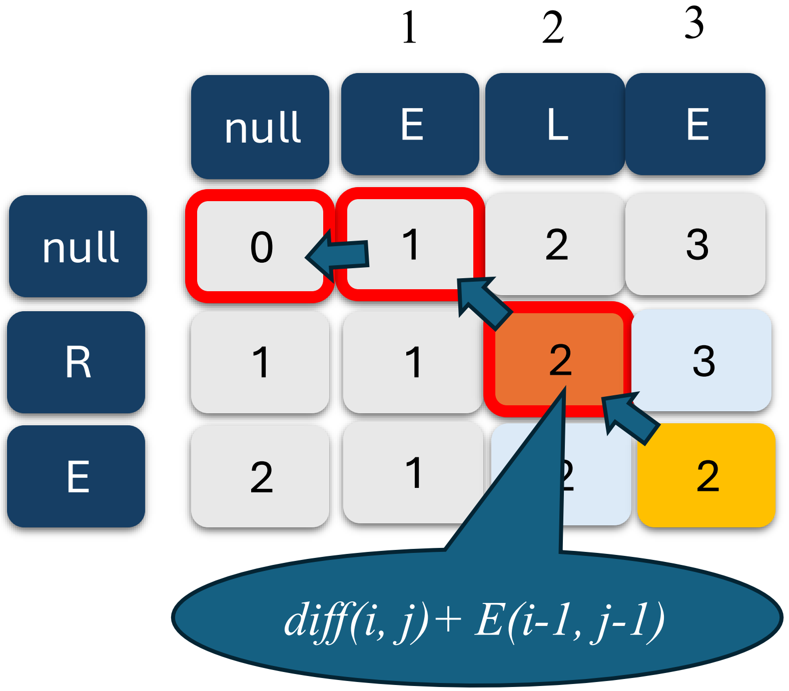
$1 + E(i\text{-}1, j)$

Cost of aligning R and ELE



Cost of Deleting E

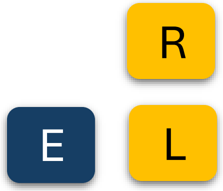
- *Insert E*
- *Insert L*
- *Replace R with E*
- *Delete E (+1)*



		1	2	3
	null	E	L	E
null	0	1	2	3
R	1	1	2	3
E	2	1	2	2

$diff(i, j) + E(i\text{-}1, j\text{-}1)$

Cost of aligning R and EL



Cost of Copying E

- *Insert E*
- *Replace R with L*
- *Copy E (+0)*

Longest Increasing Subsequence

Increasing Subsequence

A subsequence in which the numbers are getting strictly larger.

Example: A = [2, 5, 3, 7, 8, 4]

Increasing Subsequences:

- [2, 3, 4], [2, 3, 7], [2, 3, 8], [2, 3, 7, 8], [2, 5, 7, 8]

Longest Increasing Subsequence

- An increasing subsequence of greatest length

LIS

- Input: Sequence of numbers A = {a_1, a_2, … , a_n}
- Output: Longest increasing subsequence of A

How can we solve the problem?

I know brute force isn’t the way but I’ll say it anyway.

BRUTE-FORCE.

Brute Force Algorithm

- Find all increasing subsequences of A
- Return the increasing subsequence with greatest length

Brute Force Algorithm

```
brute_force_LIS(sequence A):
    subsequences = all subsequences of A
    max_length = 0
    for subseq in subsequences:
        if is_increasing(subseq):
            if length(subseq) > max_length:
                LIS = subseq
                max_length = length(subseq)
    return max_length, LIS
```

Analysis

- Finding all increasing subsequences = O(2^n)
- Involves checking each subsequence and checking if that subsequence is increasing

What’s a better way of solving for LIS?

OMG. Tinatanong pa ba yan?

Use **DP Approach**!

Solving Longest Increasing Subsequence using Longest Common Subsequence

- *Let B = sorted version of sequence A*
  - *takes O(N log N) time using merge sort*
- *LIS(A) = LCS(A,B)*
  - *takes $O(N^2)$ time ⇒ O(MN)*
  - *but M = N(B is just sorted version of A) so $O(N^2)$*

Solving Longest Increasing Subsequence using Longest Common Subsequence

- *Let B = sorted version of sequence A*
  - *takes* $O(N \log N)$ *time using merge sort*
- *LIS(A) = LCS(A,B)*
  - *takes* $O(N^2)$ *time* ⇒ $O(MN)$
  - *but M = N(B is just sorted version of A) so* $O(N^2)$

A = [5, 2, 8, 6, 3, 6, 9, 7]   B = [2, 3, 5, 6, 6, 7, 8, 9]

LCS (A, B) = [2, 3, 6, 9]

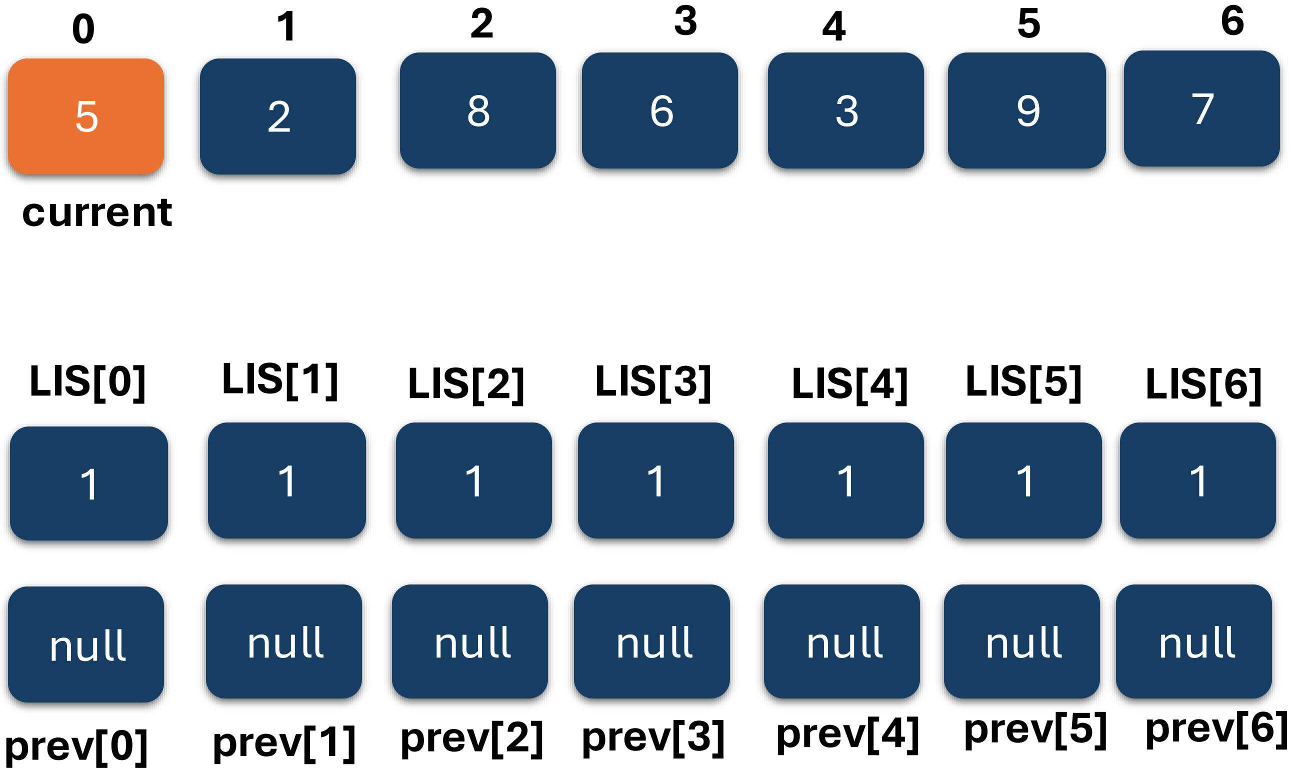
DP Idea

Example: [1,7,2,3,6],	LIS: 1236
Consider: [1,7,2,3],	LIS: 123
Now: [1,7,2],	LIS: 12 and 17

- What do you notice?

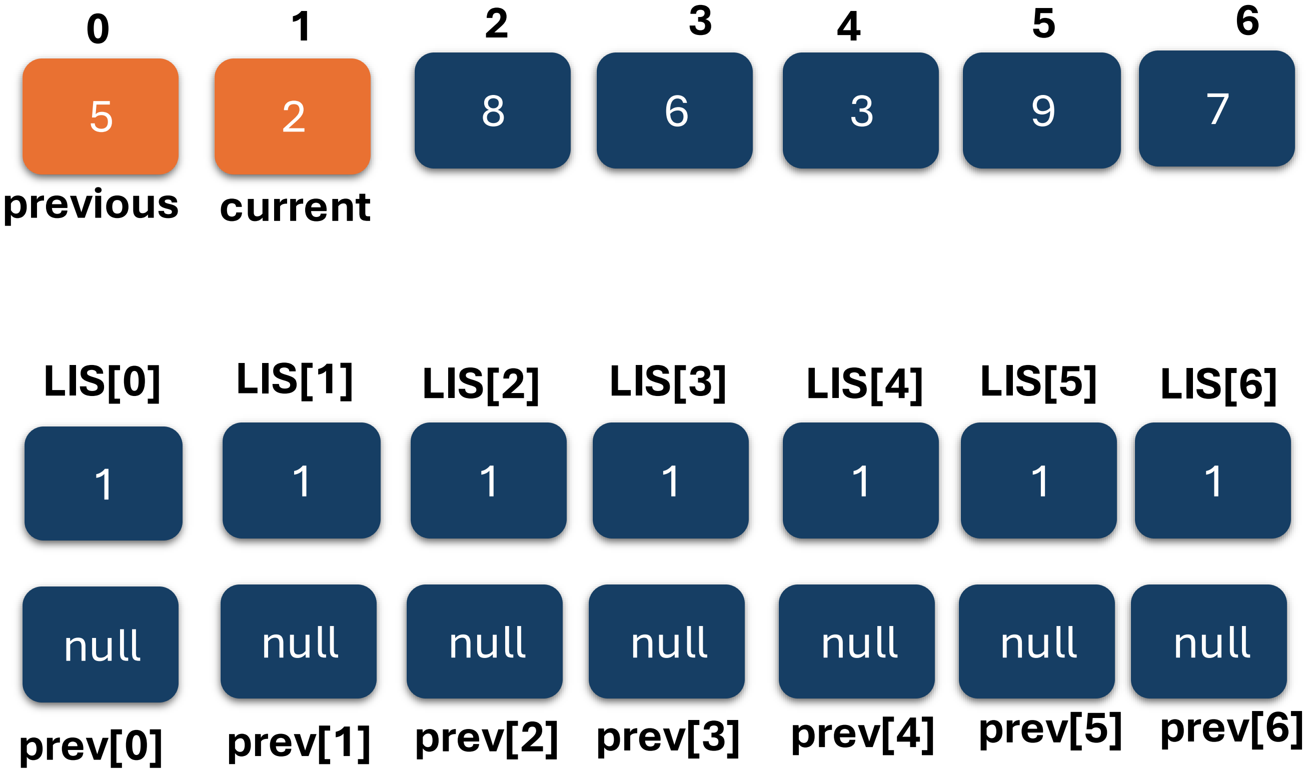
DP Idea

- The LIS of the shorter sequence can also be part of the LIS of the longer sequence
- This is our optimal substructure.



0	1	2	3	4	5	6
5	2	8	6	3	9	7
current						
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	1	1	1	1	1
null	null	null	null	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

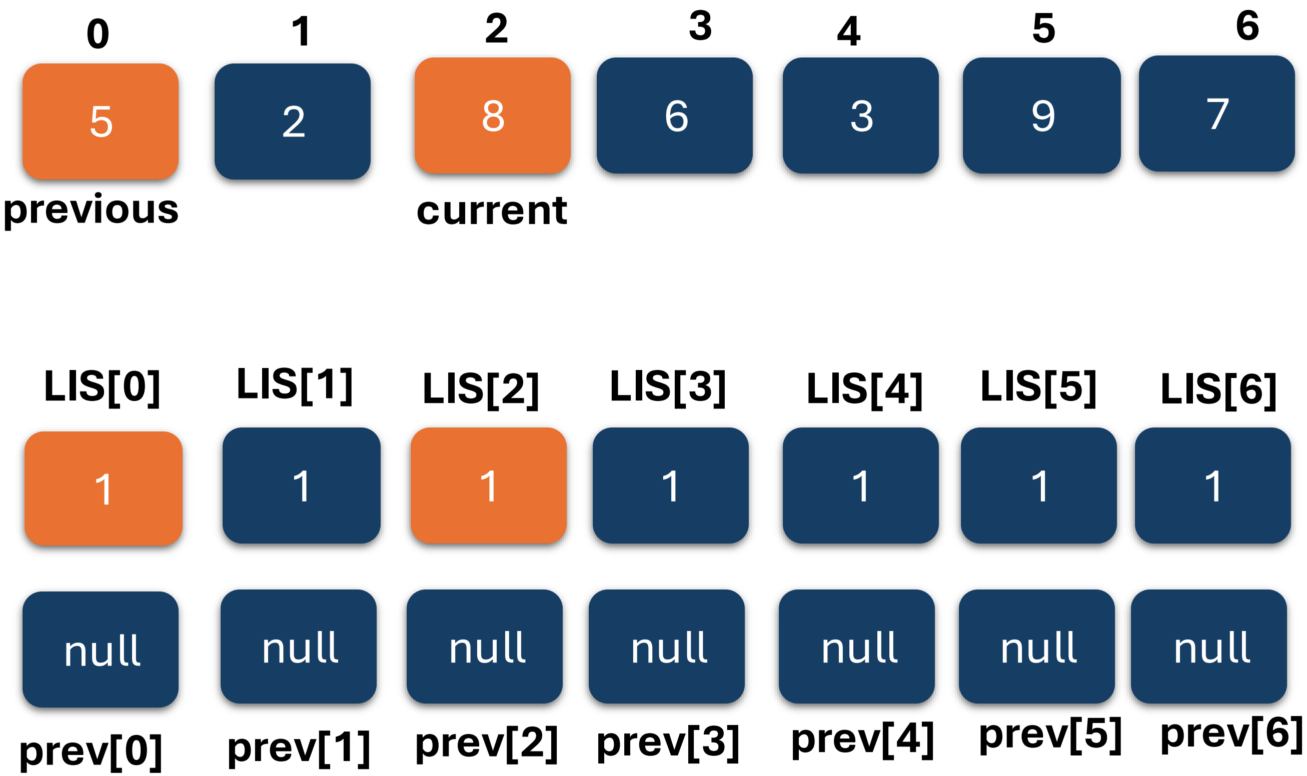
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous	current					

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	1	1	1	1	1
null	null	null	null	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

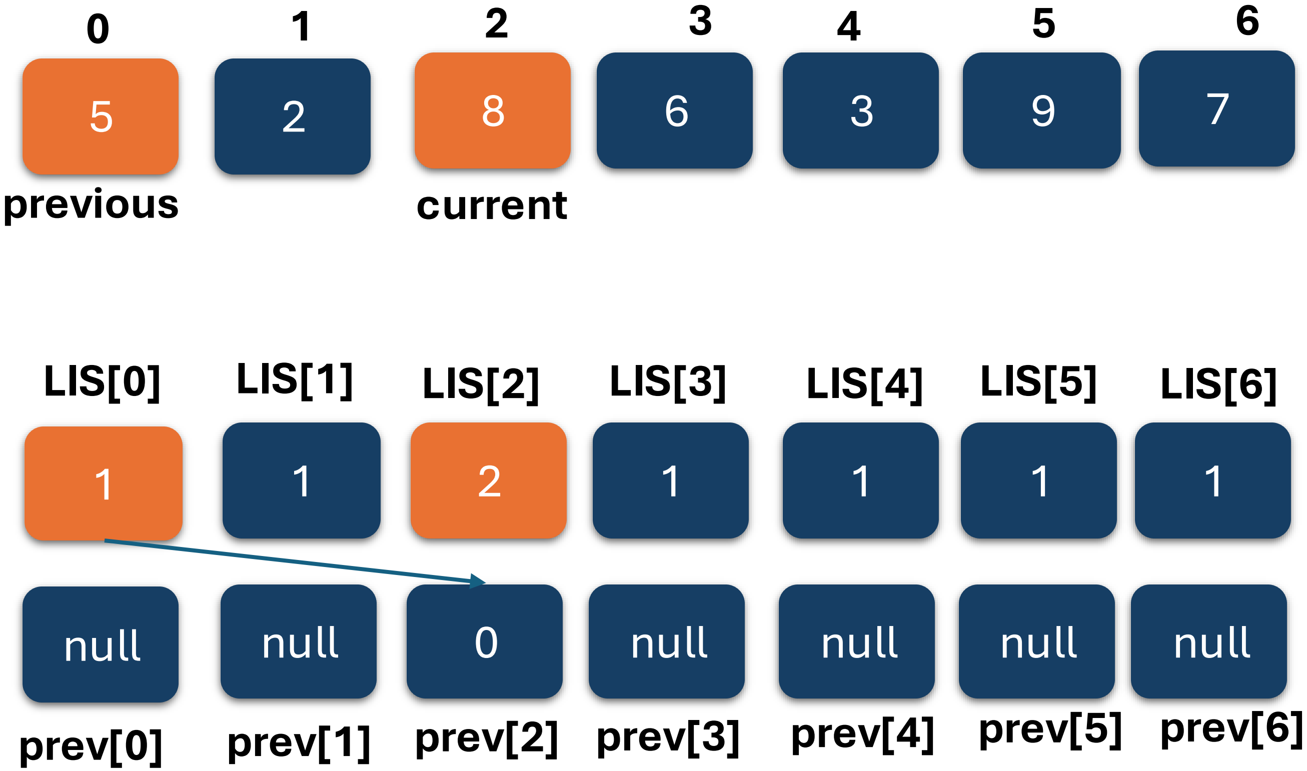


0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous		current				

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	1	1	1	1	1

prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	null	null	null	null	null

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

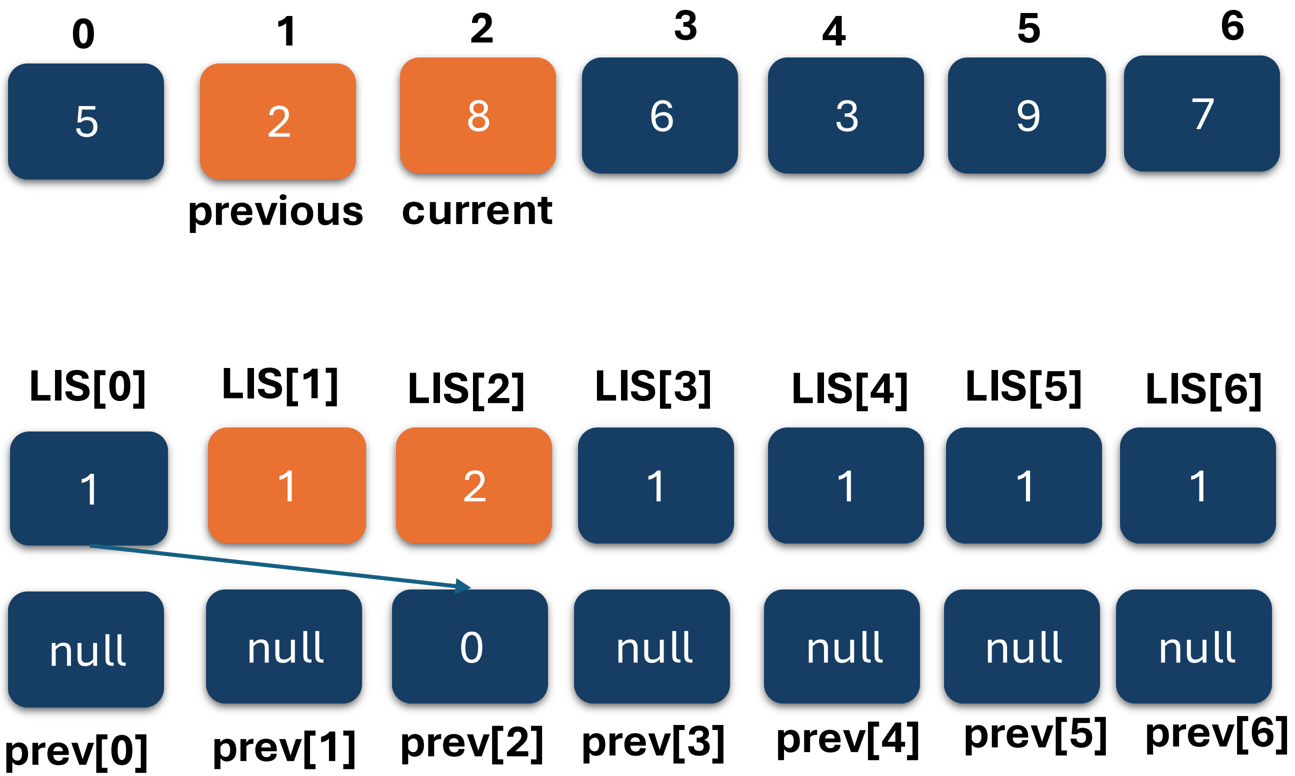


0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous		current				

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	1	1	1	1
null	null	0	null	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

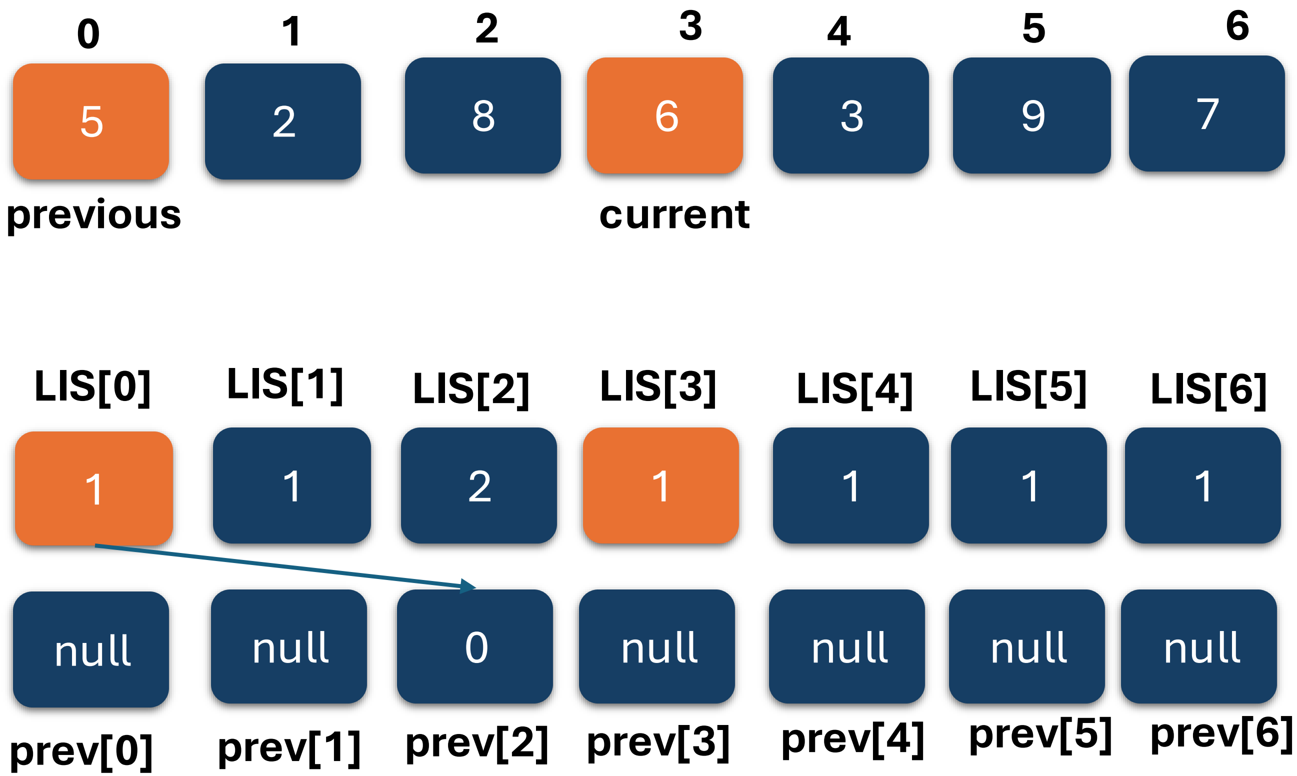
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
                if A[current] > A[previous]:
                        if LIS[previous] + 1 > LIS[current]:
                                LIS[current] = LIS[previous] + 1
                                PREV[current] = previous

    return max(LIS)
```



	0	1	2	3	4	5	6
	5	2	8	6	3	9	7
		previous	current				
	LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
	1	1	2	1	1	1	1
	null	null	0	null	null	null	null
	prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

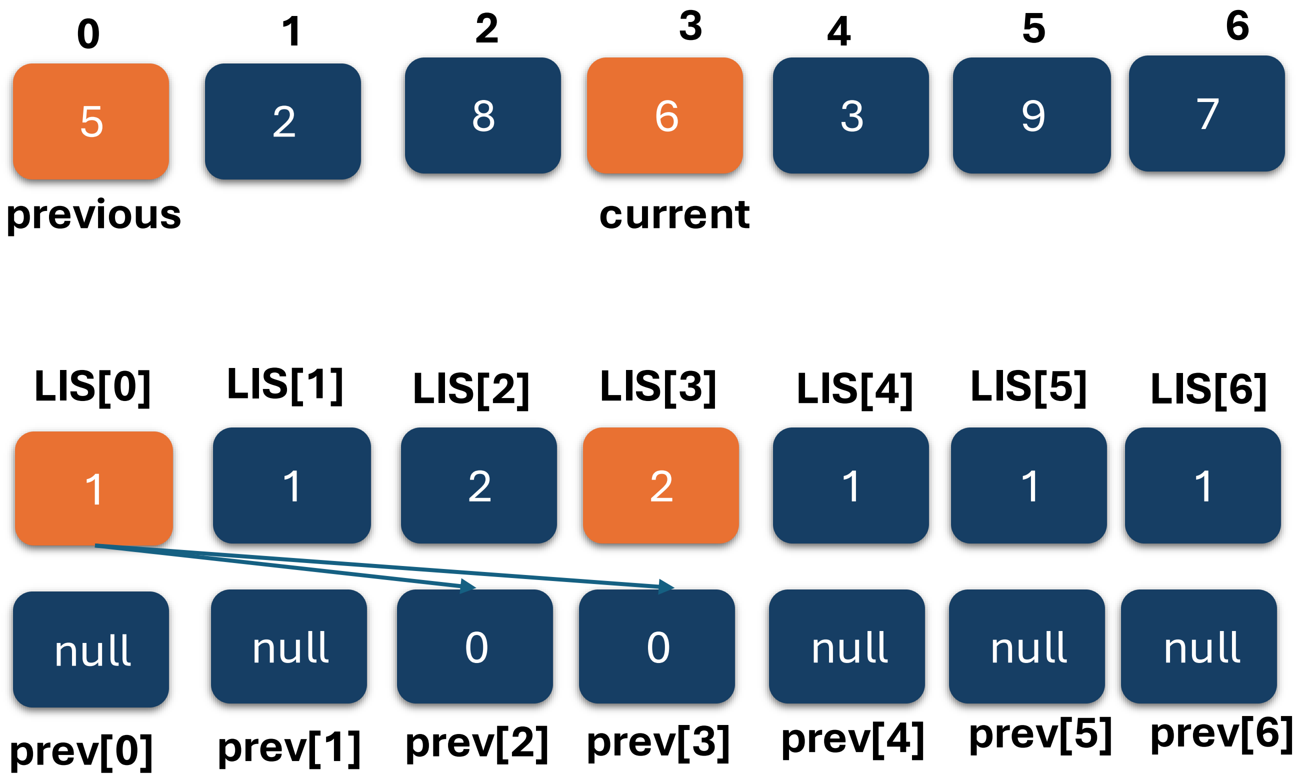
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous			current			

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	1	1	1	1
null	null	0	null	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

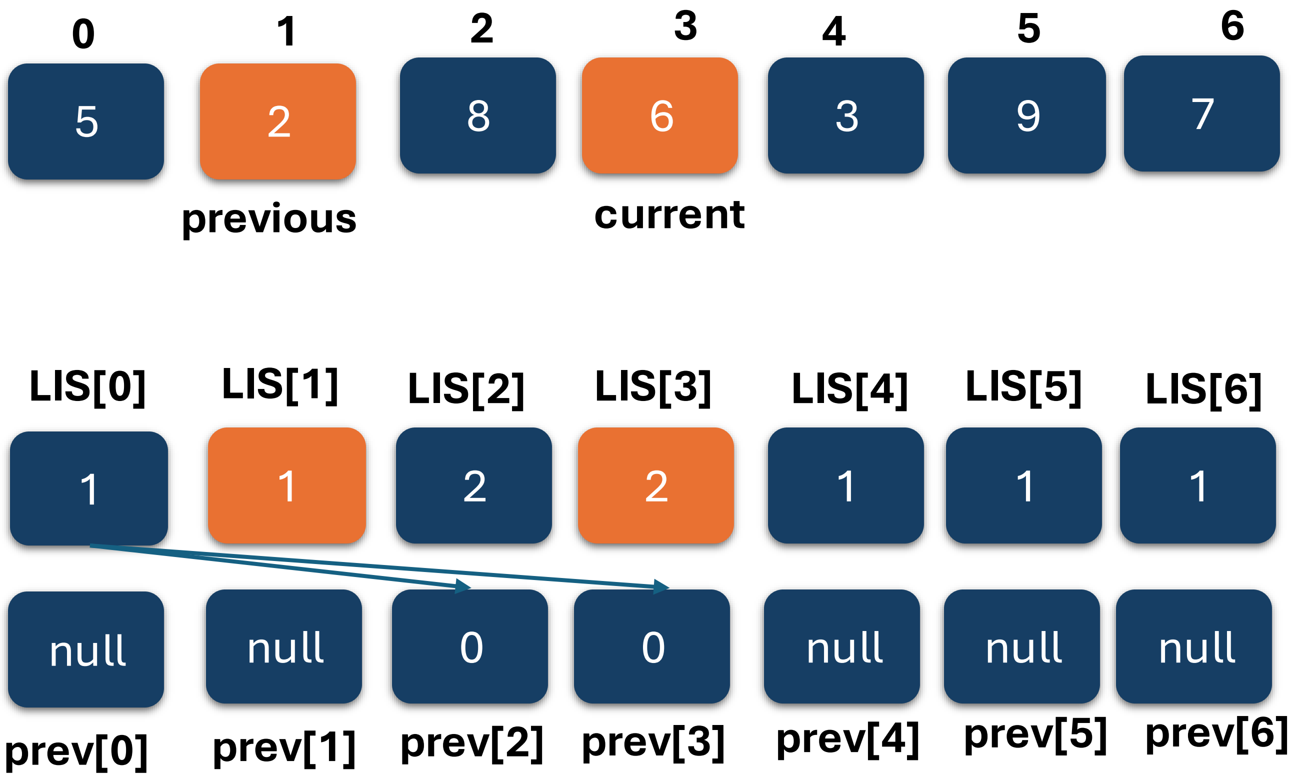


0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous			current			

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	1	1	1

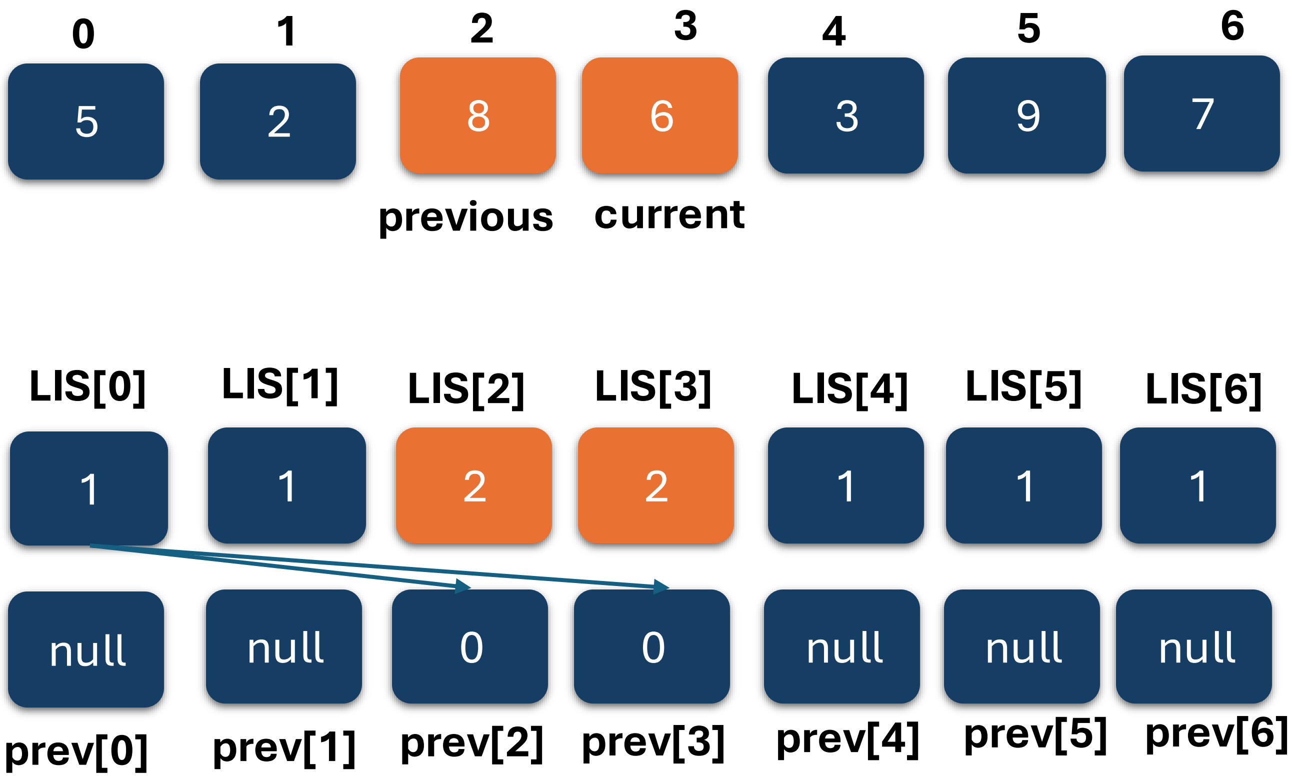
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	0	0	null	null	null

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
	previous		current			
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	1	1	1
null	null	0	0	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

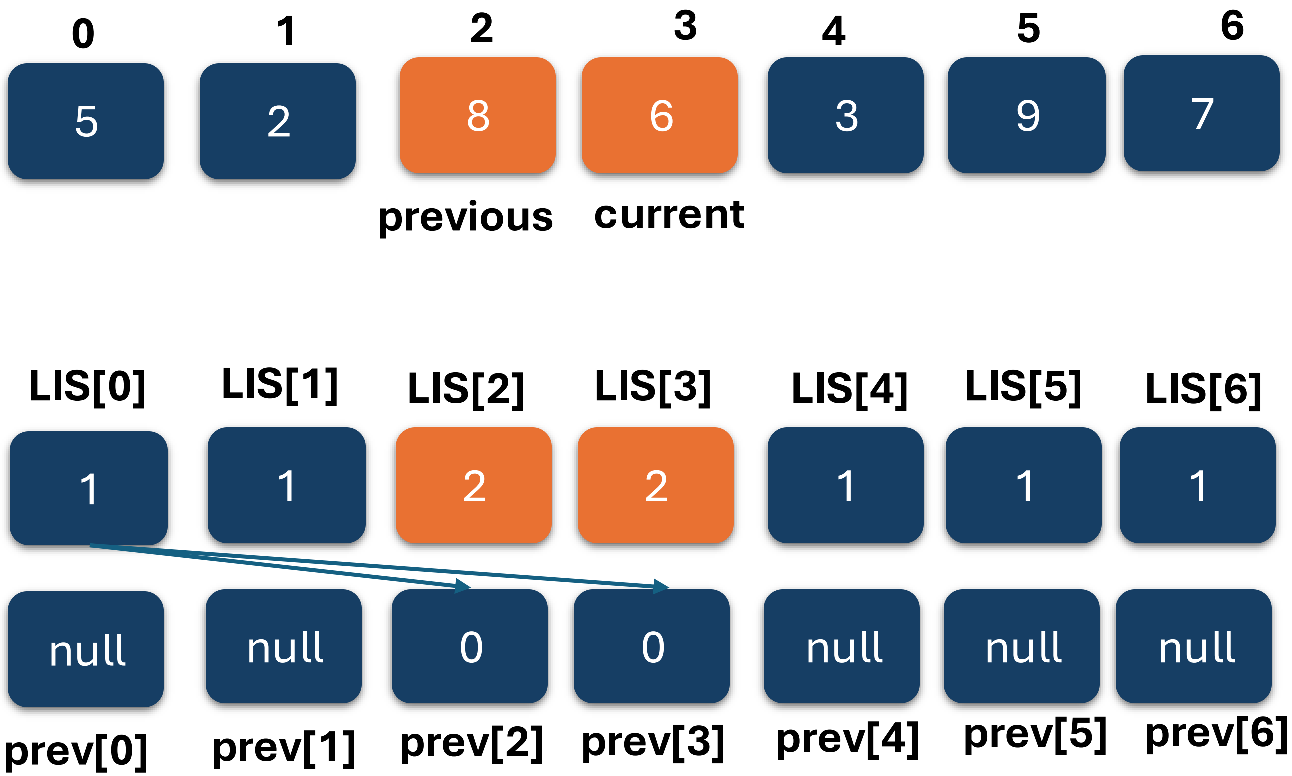
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
		previous	current			

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	1	1	1
null	null	0	0	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

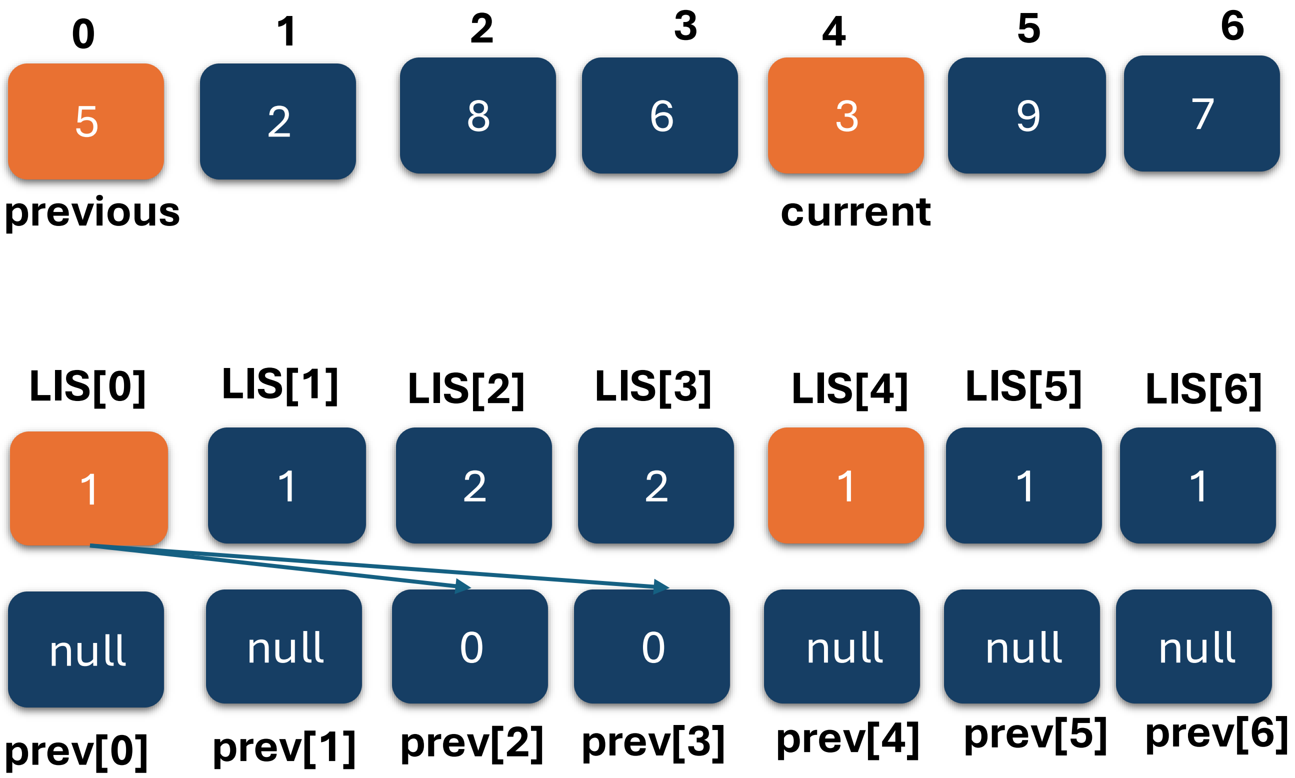
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
		previous	current			

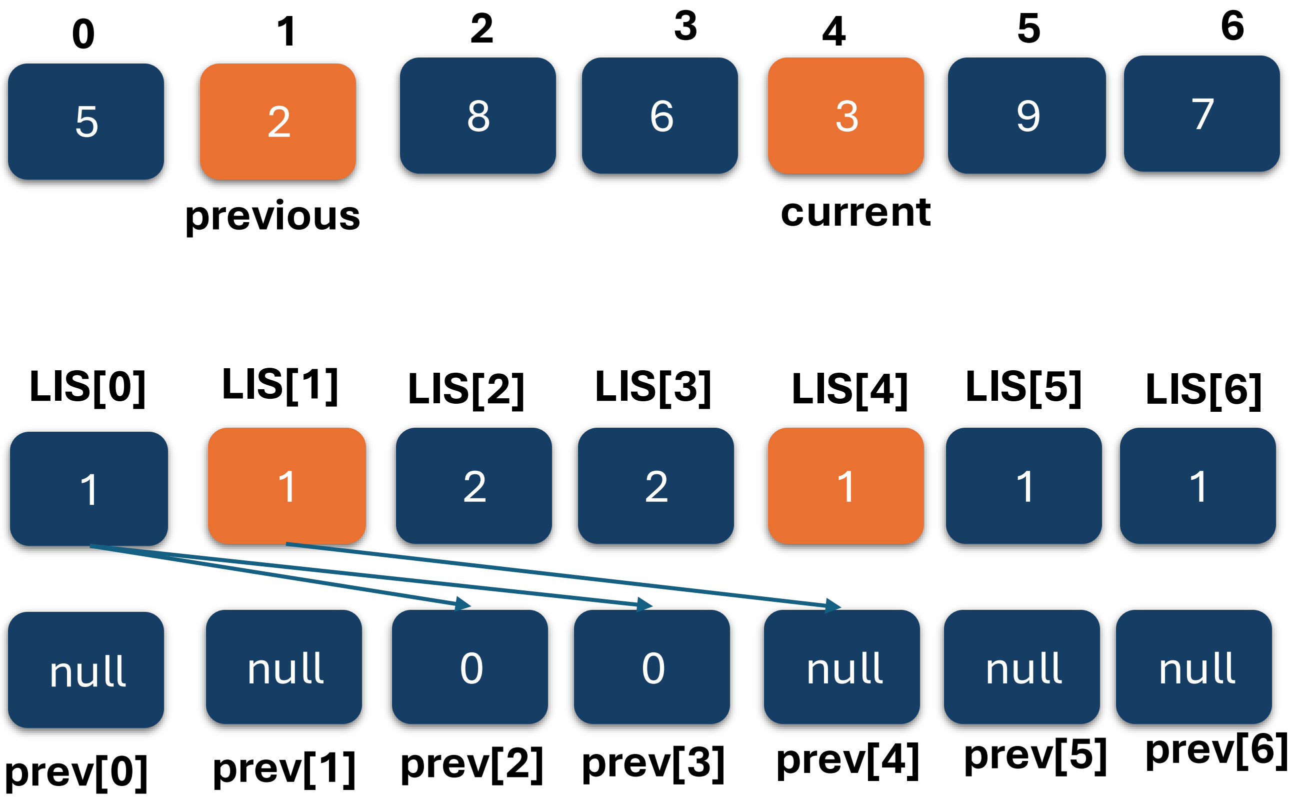
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	1	1	1
null	null	0	0	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



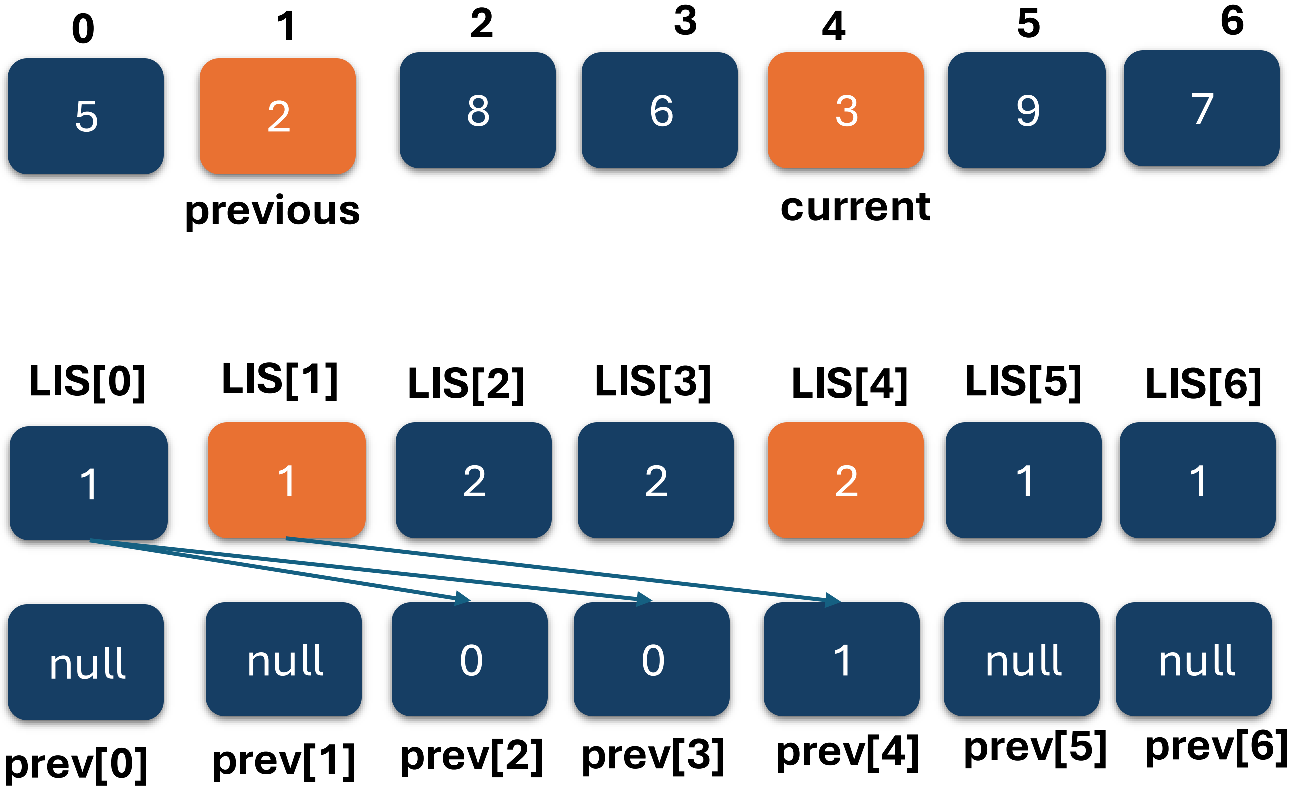
0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous				current		
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	1	1	1
null	null	0	0	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	1	1	1
null	null	0	0	null	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

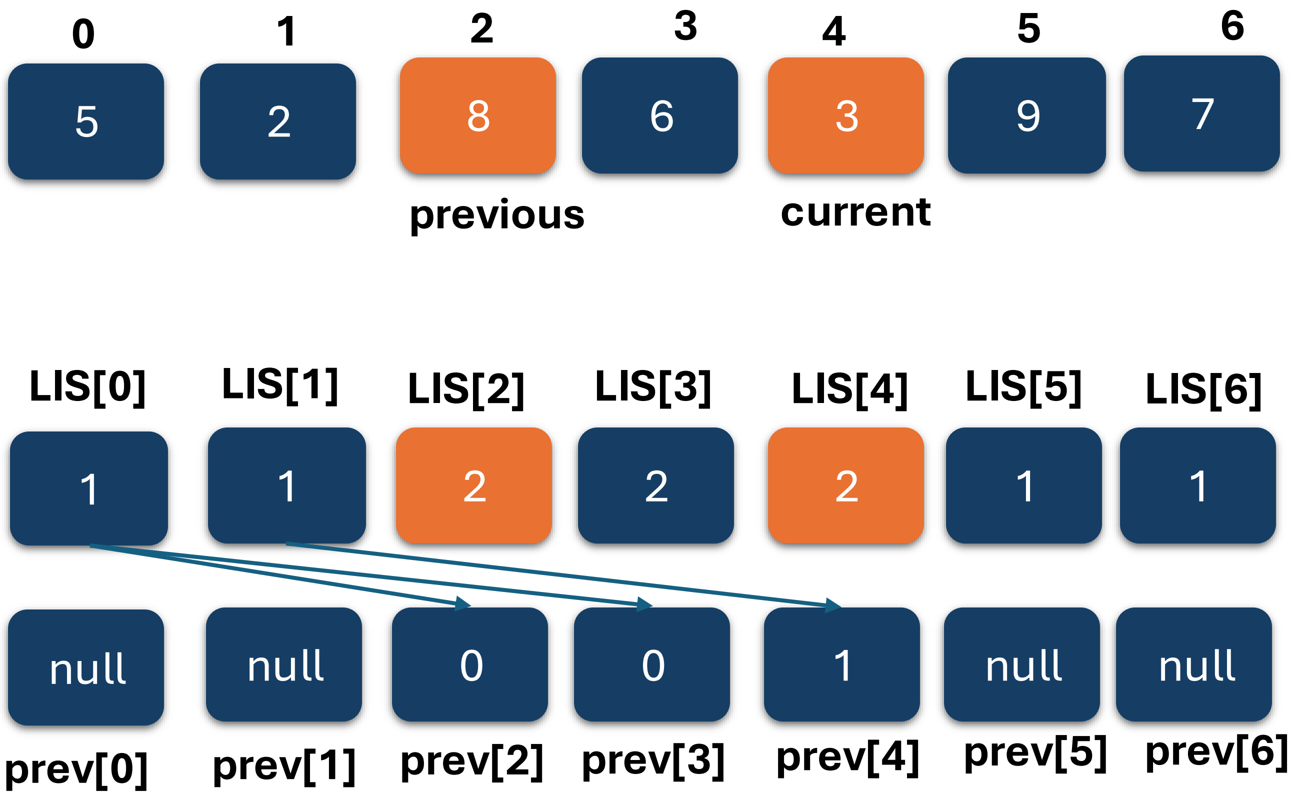
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
	previous			current		

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	1	1
null	null	0	0	1	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

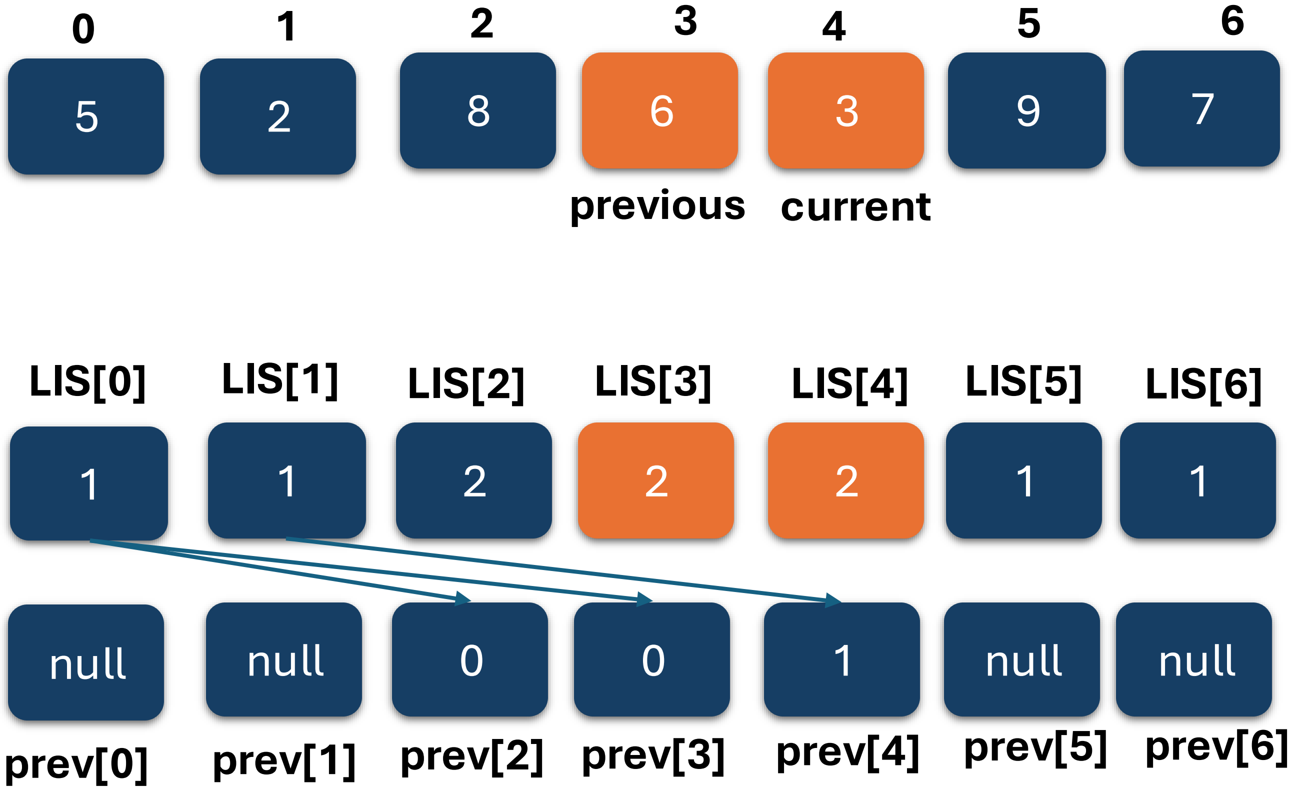
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
		previous		current		

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	1	1
null	null	0	0	1	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

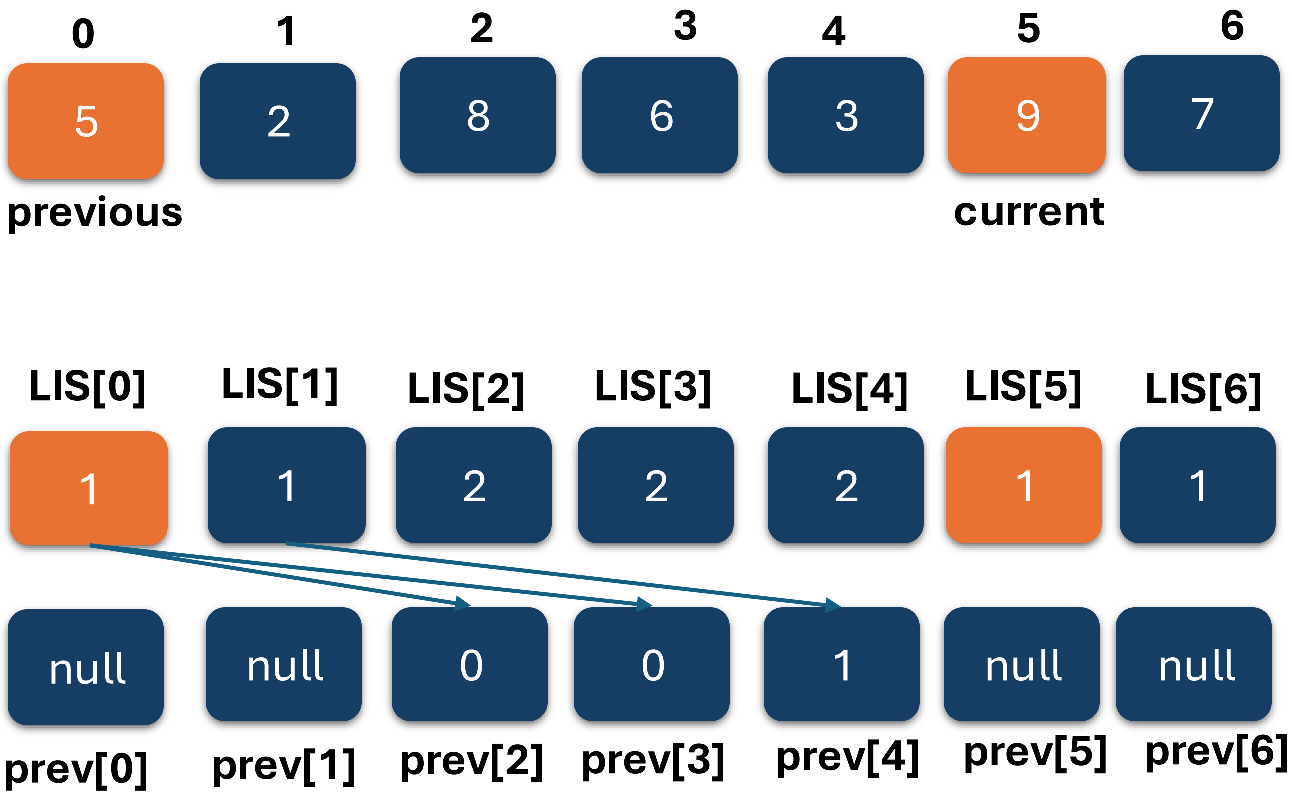


0	1	2	3	4	5	6
5	2	8	6	3	9	7
			previous	current		

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	1	1

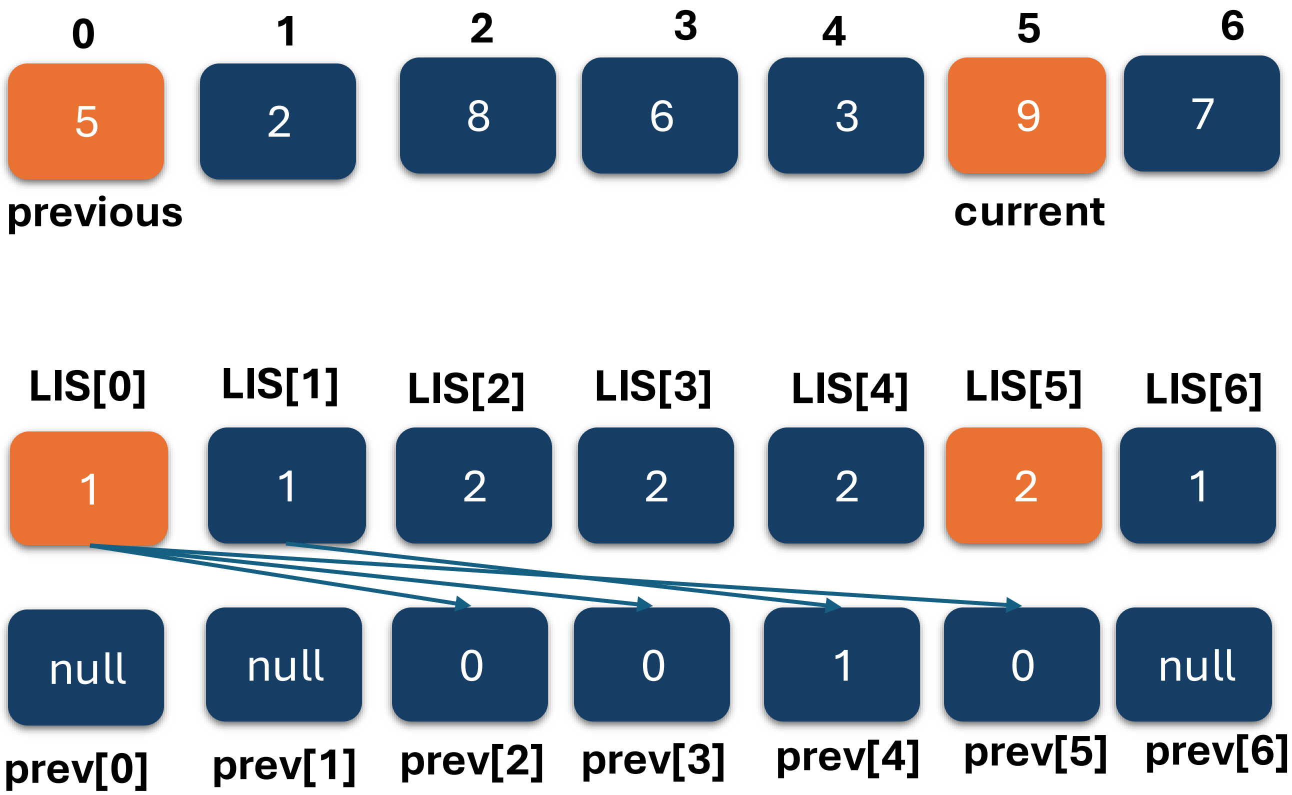
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	0	0	1	null	null

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous					current	
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	1	1
null	null	0	0	1	null	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

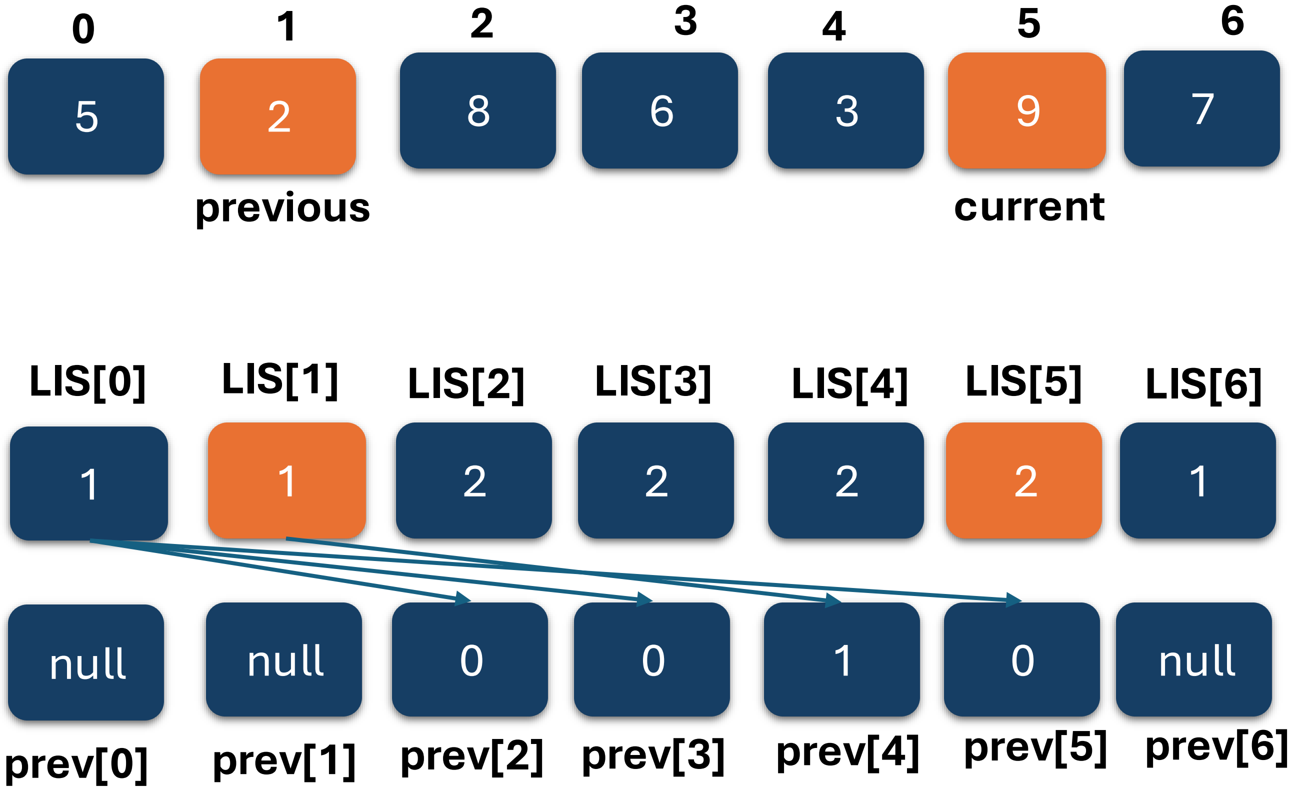


0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous					current	

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	2	1

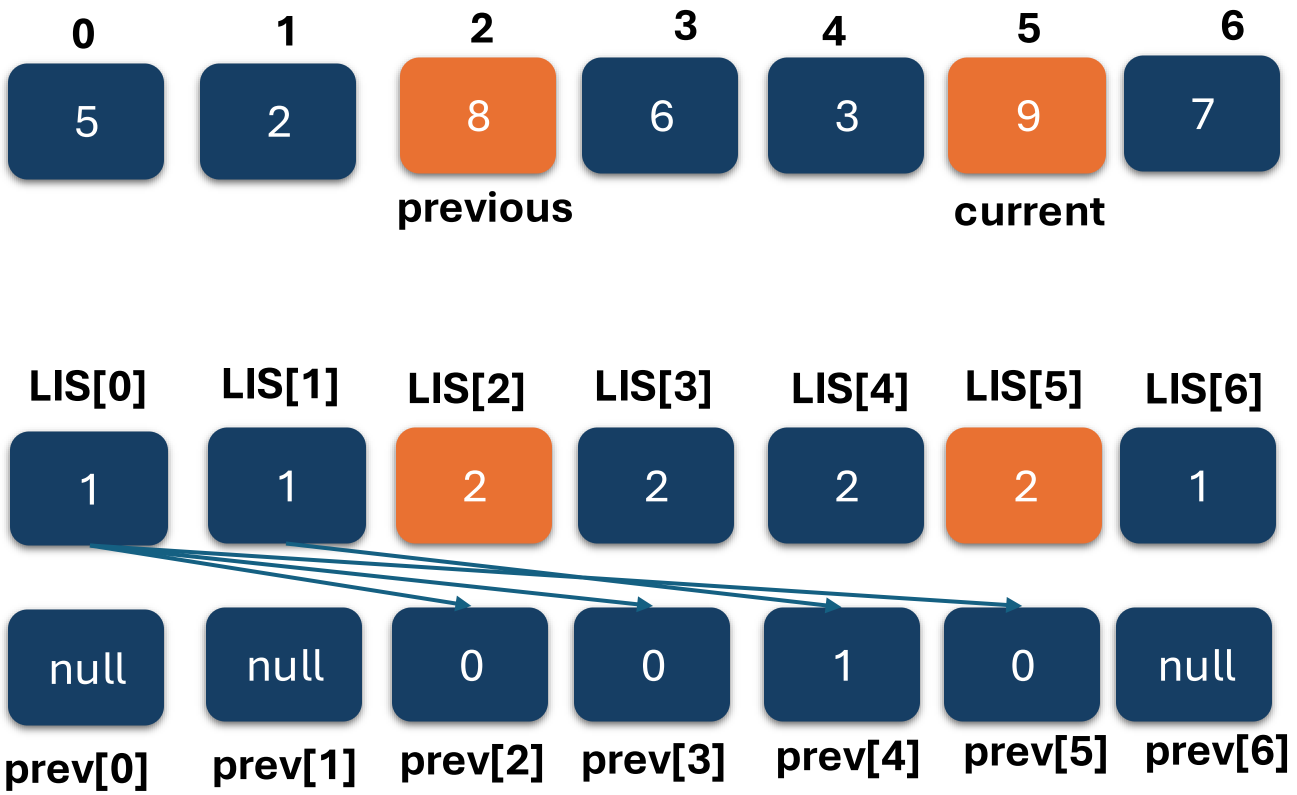
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	0	0	1	0	null

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



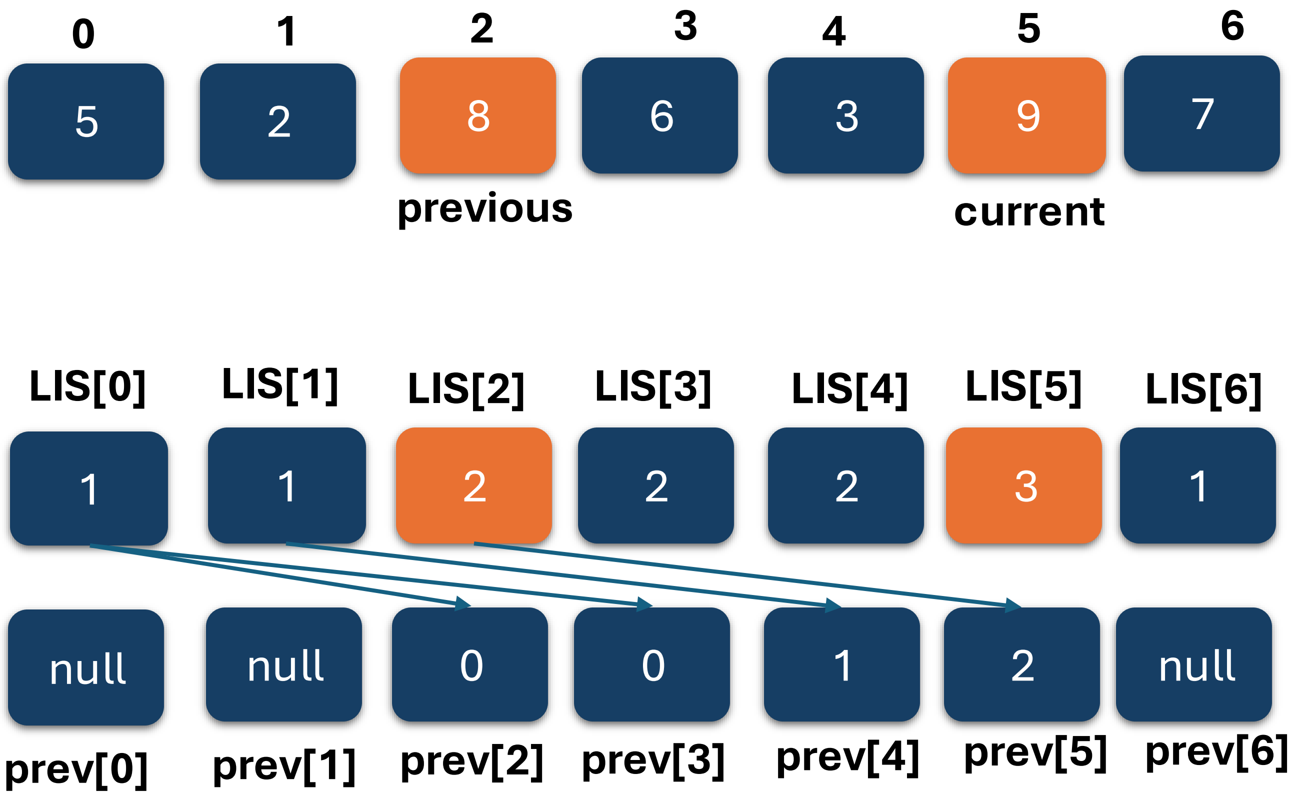
0	1	2	3	4	5	6
5	2	8	6	3	9	7
	previous				current	
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	2	1
null	null	0	0	1	0	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
		previous			current	
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	2	1
null	null	0	0	1	0	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

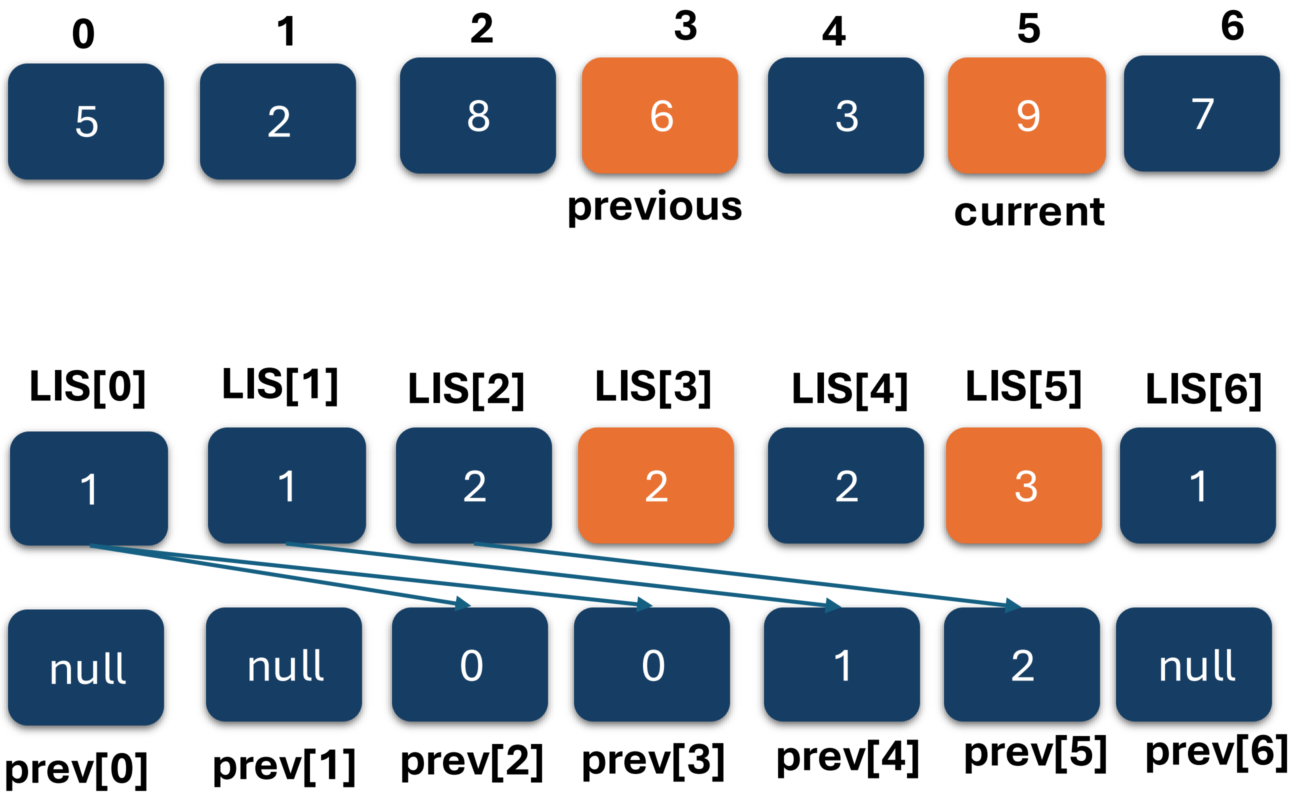


0	1	2	3	4	5	6
5	2	8	6	3	9	7
		previous			current	

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	1

prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	0	0	1	2	null

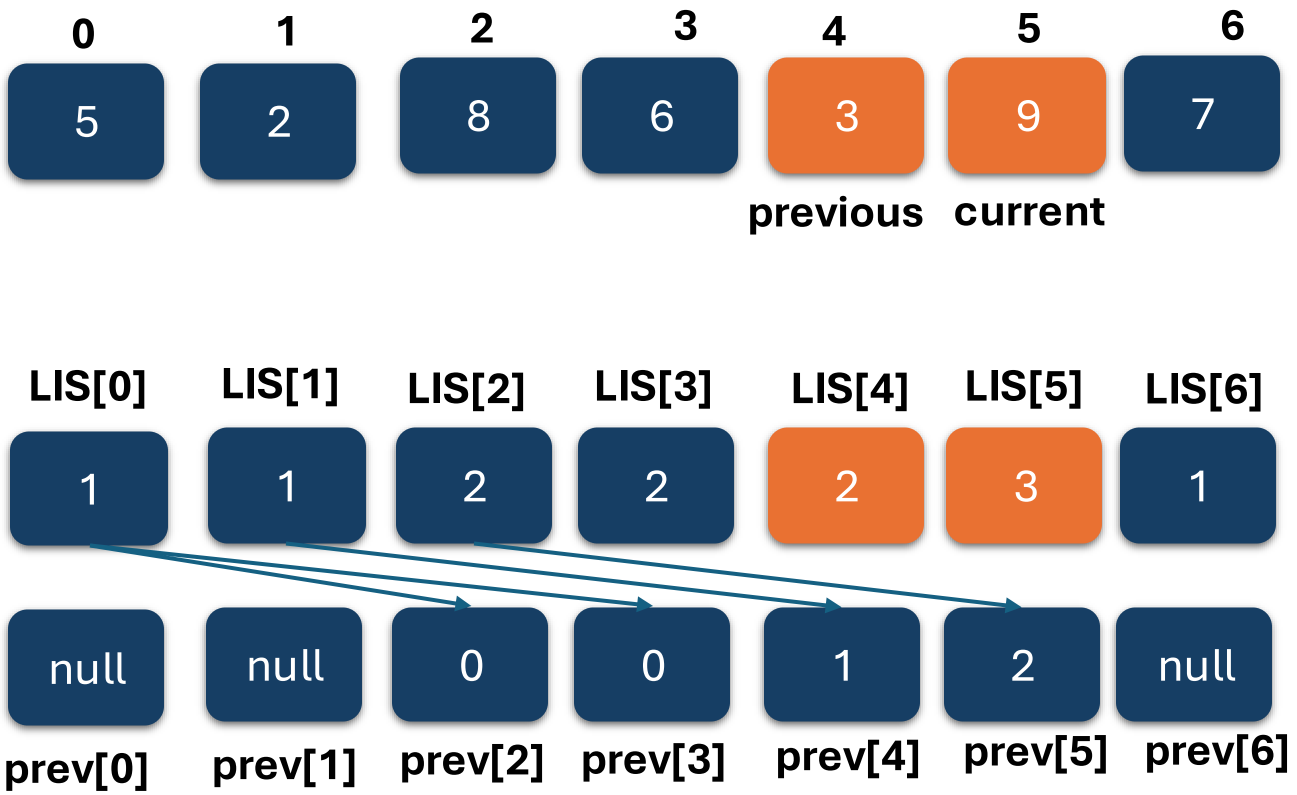
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
			previous		current	

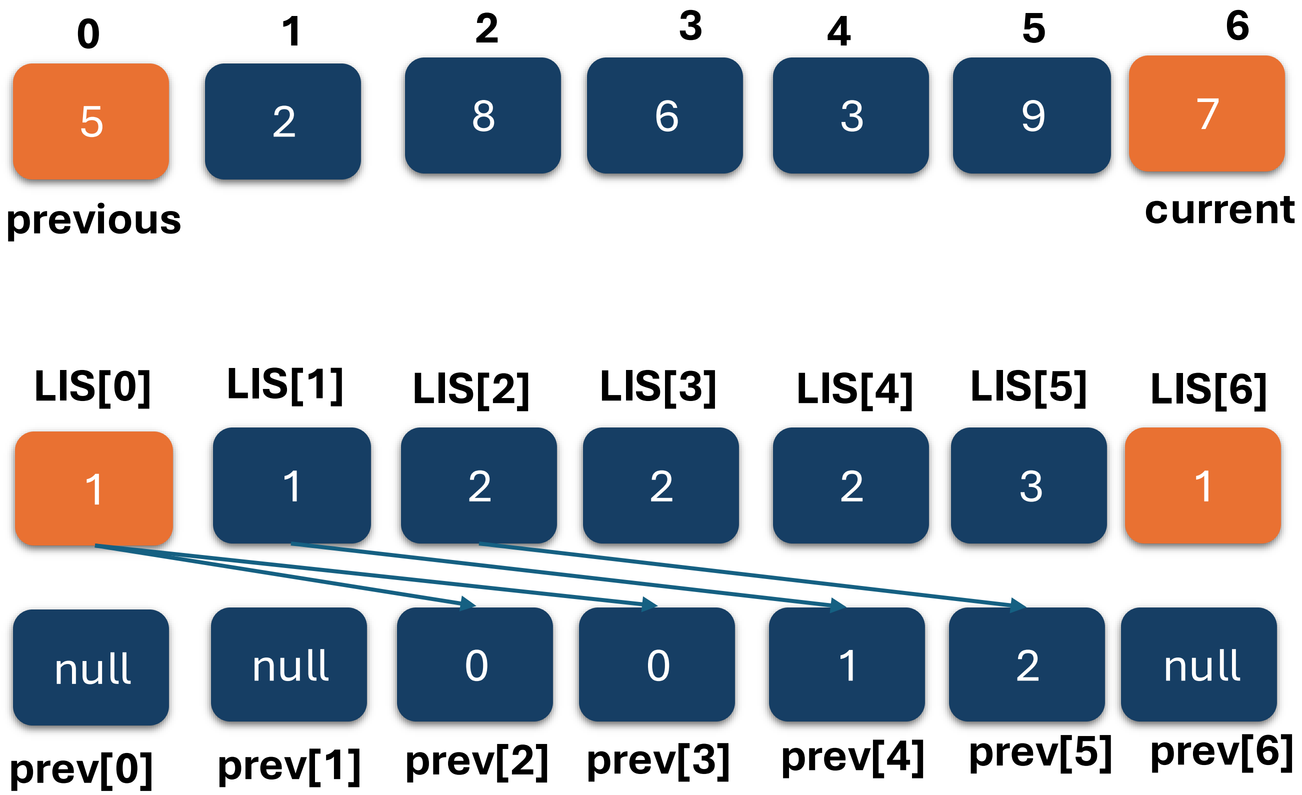
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	1
null	null	0	0	1	2	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



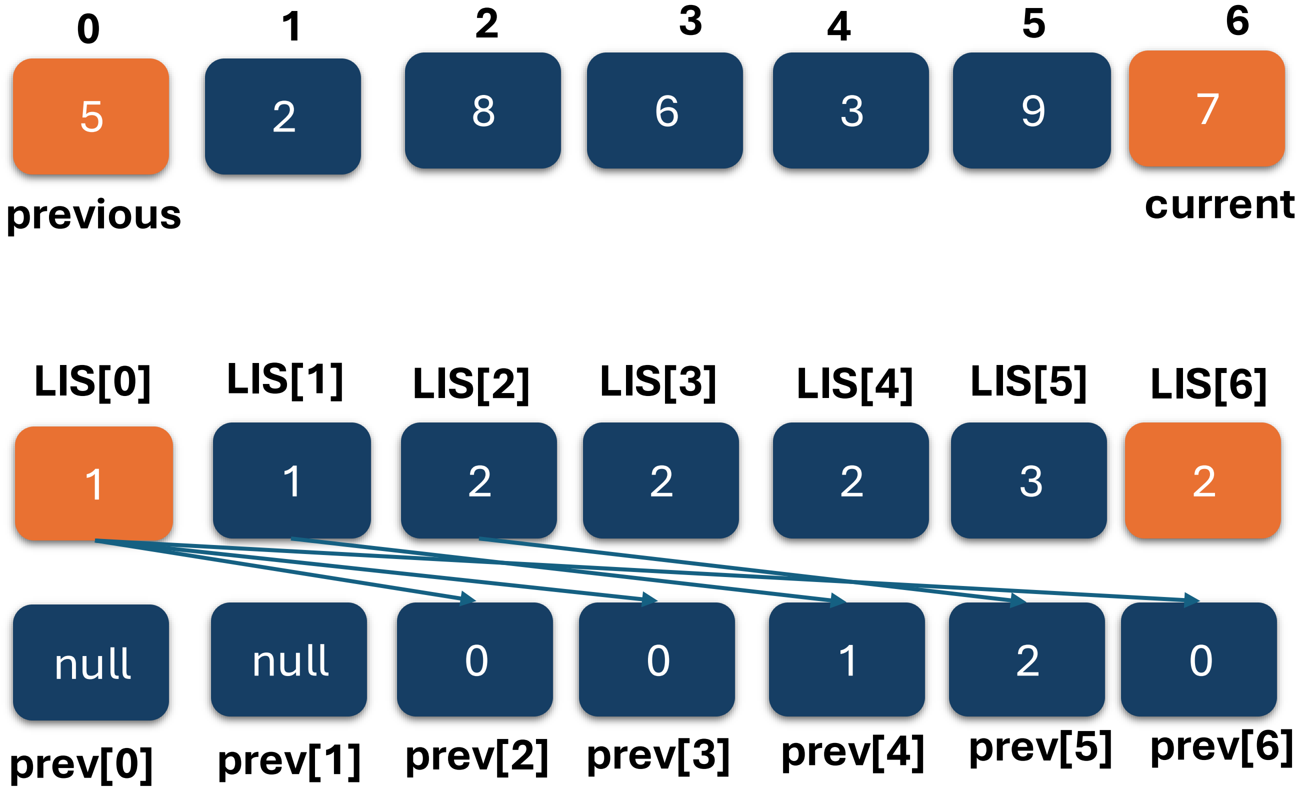
0	1	2	3	4	5	6
5	2	8	6	3	9	7
				previous	current	
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	1
null	null	0	0	1	2	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous						current
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	1
null	null	0	0	1	2	null
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

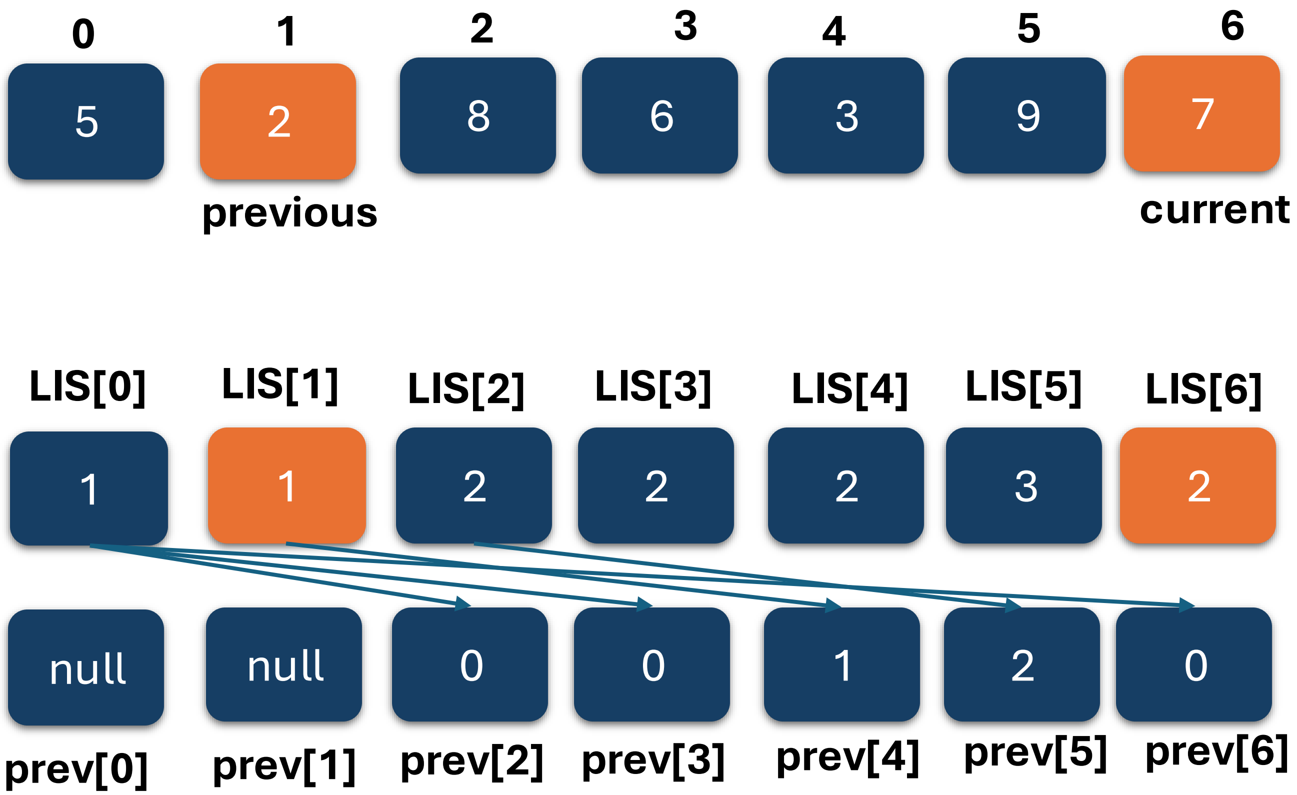
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
previous						current

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	2
null	null	0	0	1	2	0
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

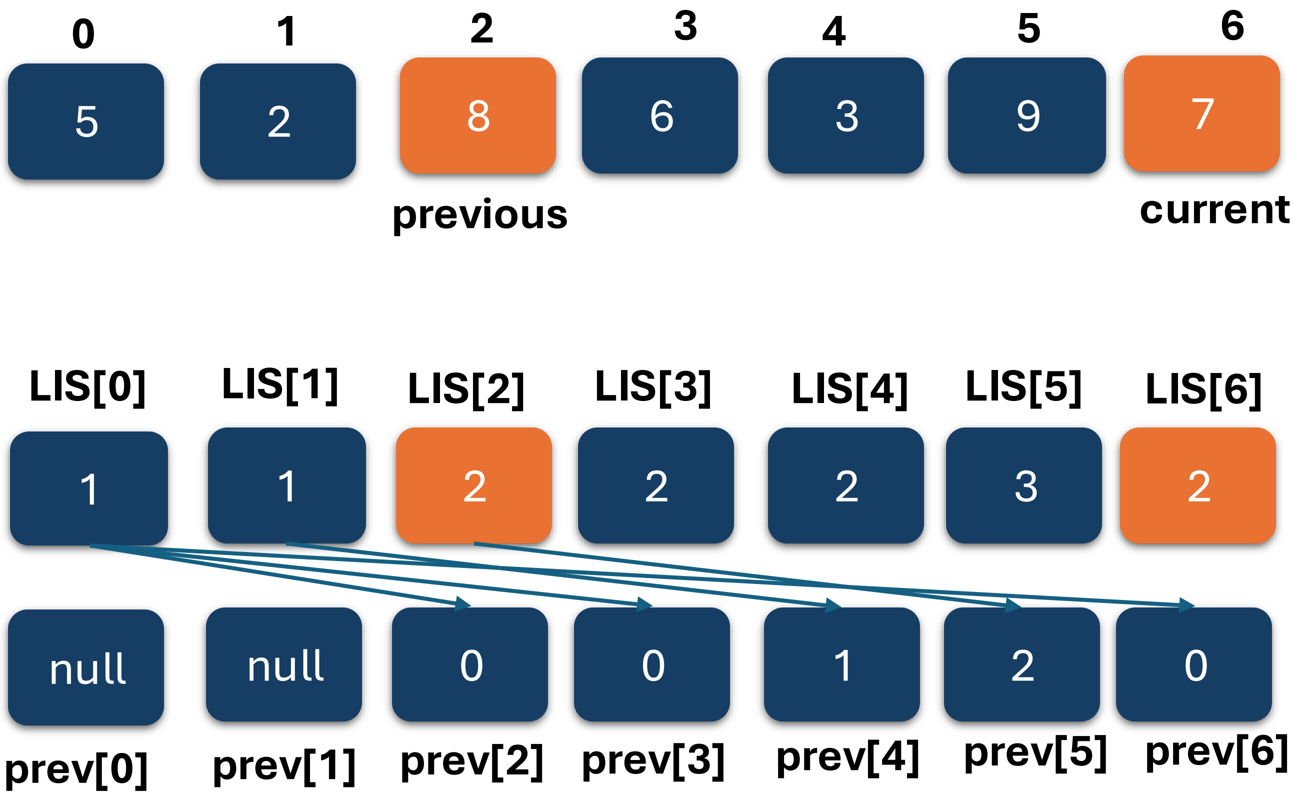
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



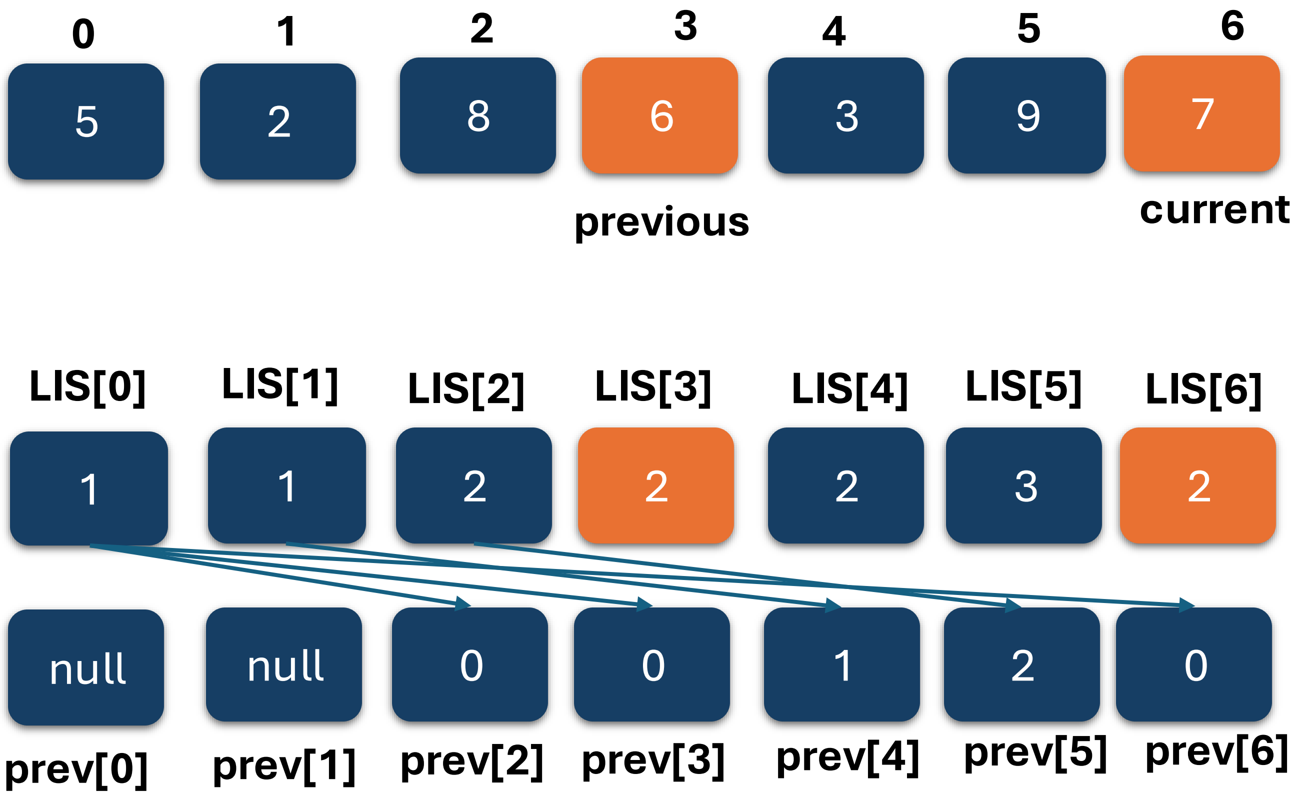
0	1	2	3	4	5	6
5	2	8	6	3	9	7
	previous					current

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	2
null	null	0	0	1	2	0
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

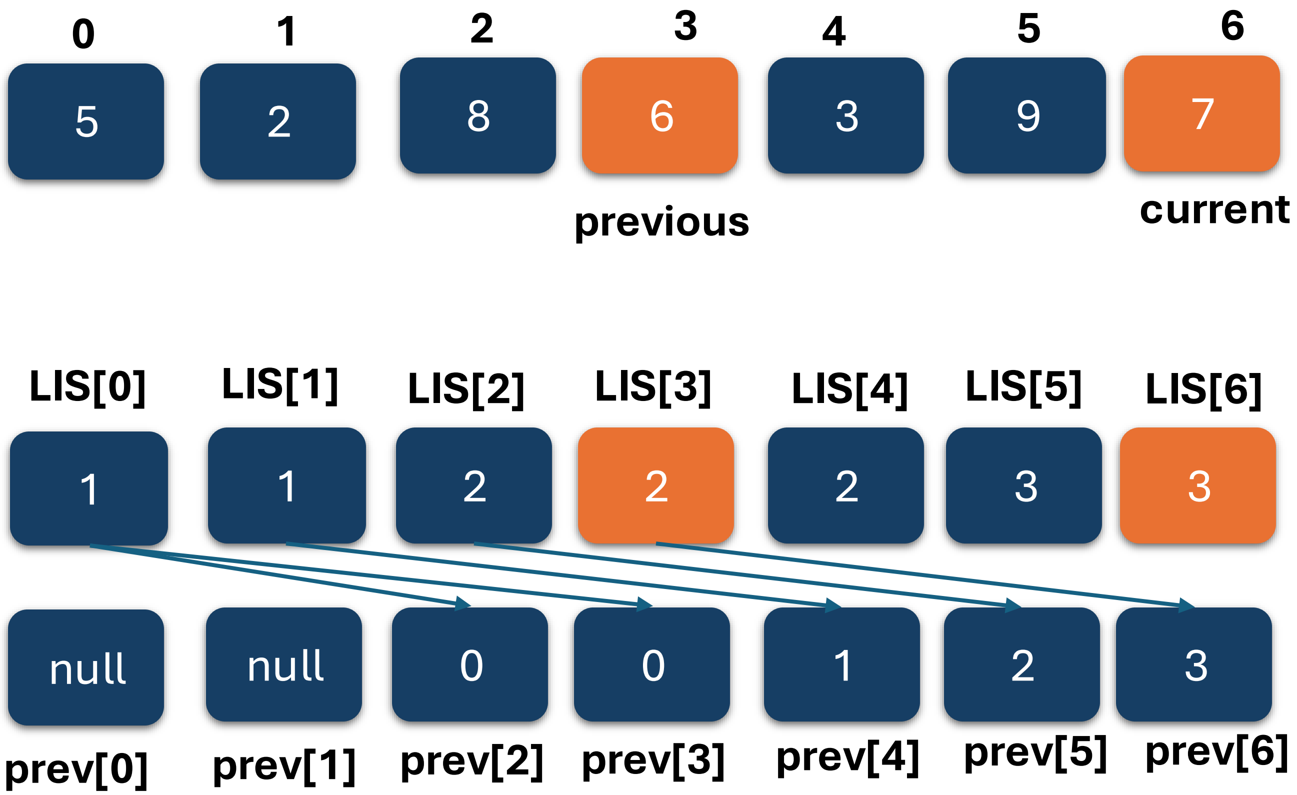


0	1	2	3	4	5	6
5	2	8	6	3	9	7
			previous			current

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	2

prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	0	0	1	2	0

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```

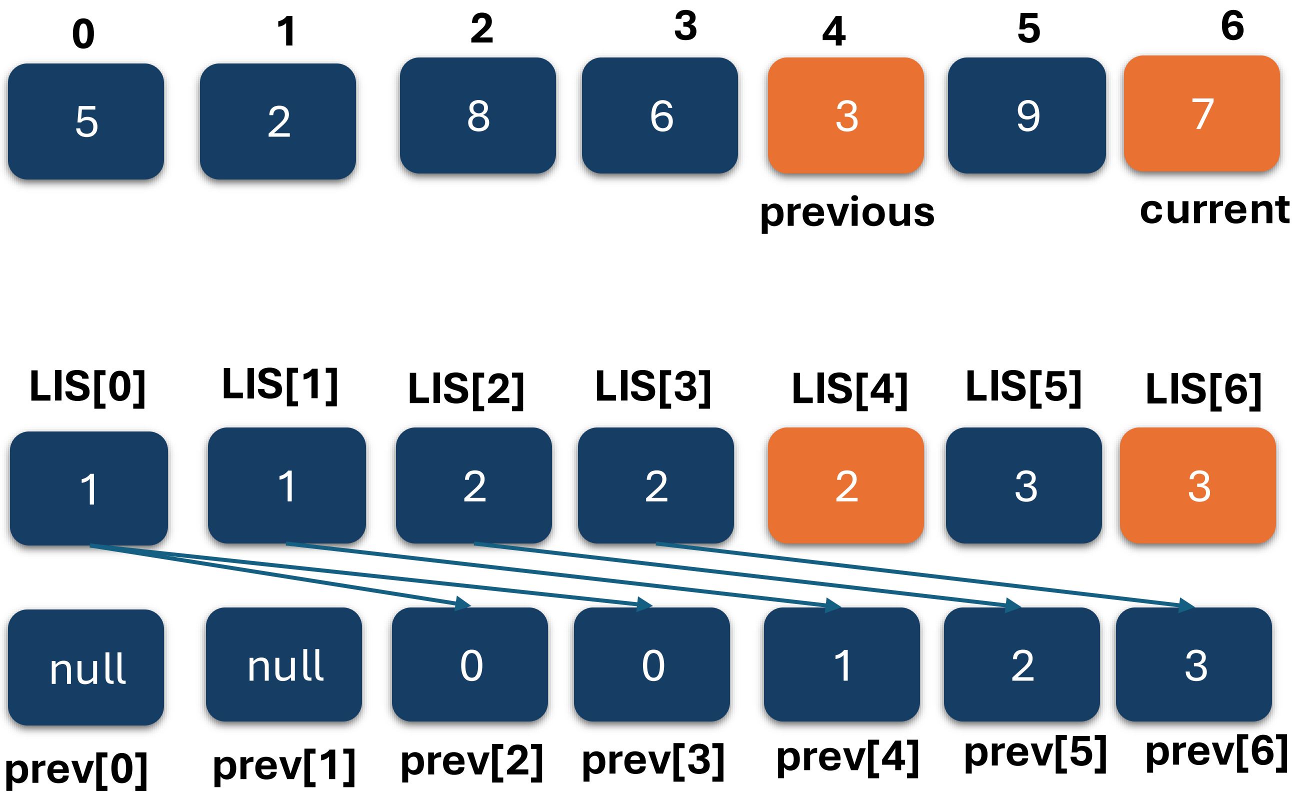


0	1	2	3	4	5	6
5	2	8	6	3	9	7
			previous			current

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	3

null	null	0	0	1	2	3
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

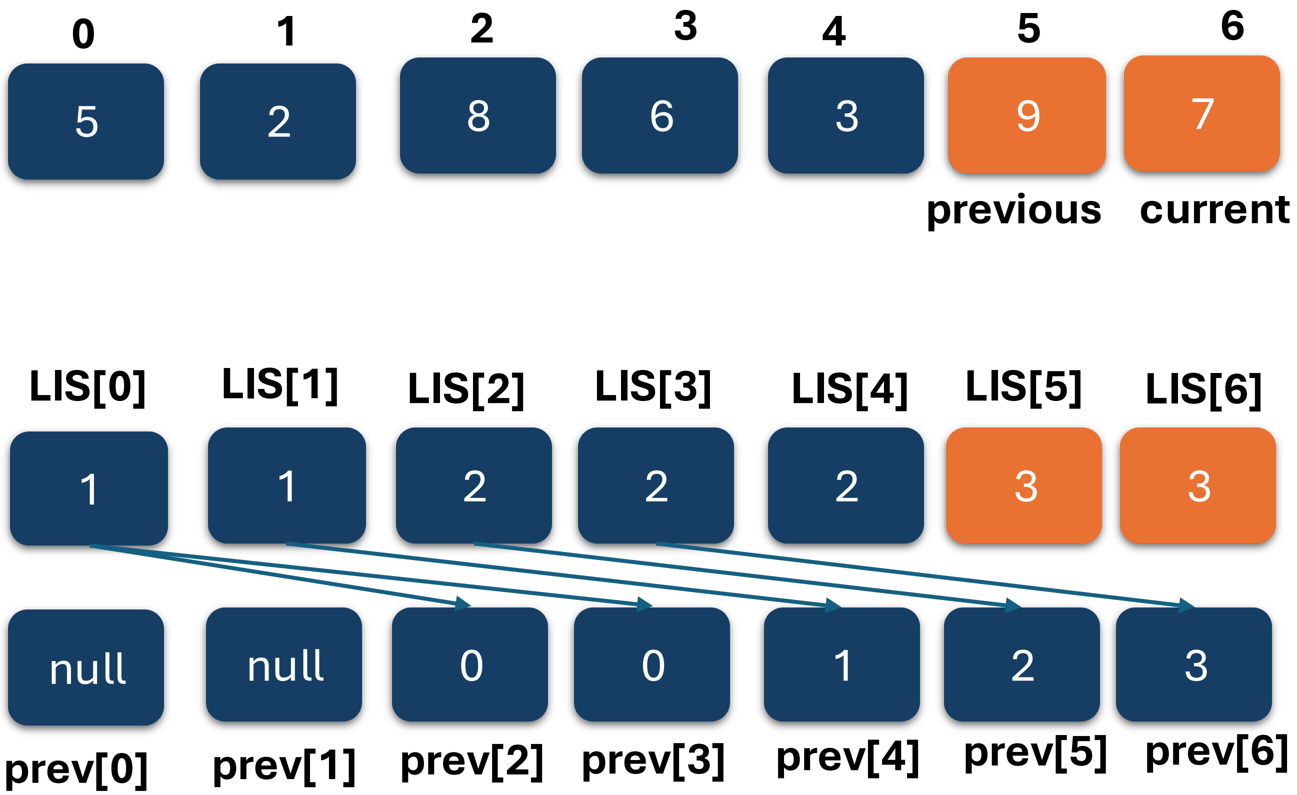
```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



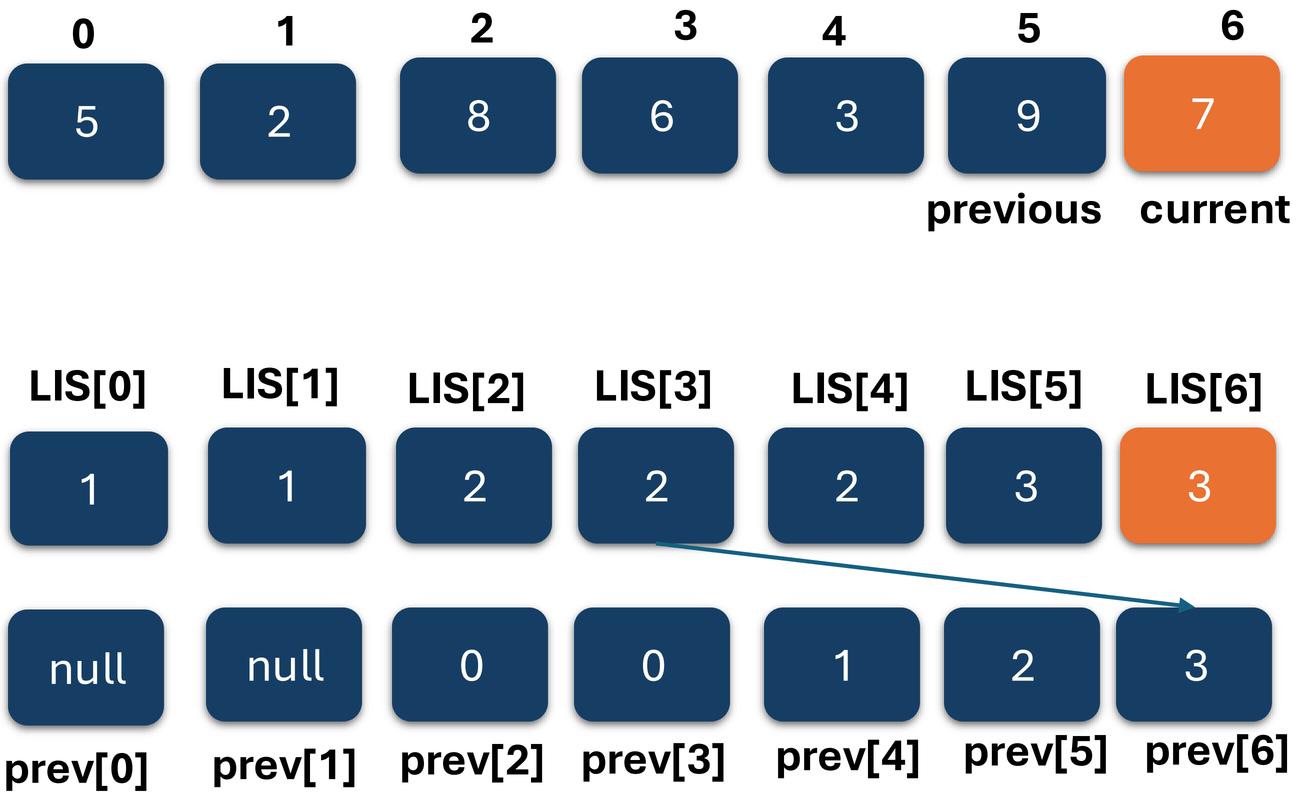
0	1	2	3	4	5	6
5	2	8	6	3	9	7
				previous		current

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	3
null	null	0	0	1	2	3
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



```
find_LIS_length(sequence A):
    N = length(A)
    LIS = array of 1s, length N
    PREV = array of nulls, length N
    for current = 0 to N-1:
        for previous = 0 to current-1:
            if A[current] > A[previous]:
                if LIS[previous] + 1 > LIS[current]:
                    LIS[current] = LIS[previous] + 1
                    PREV[current] = previous
    return max(LIS)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	3

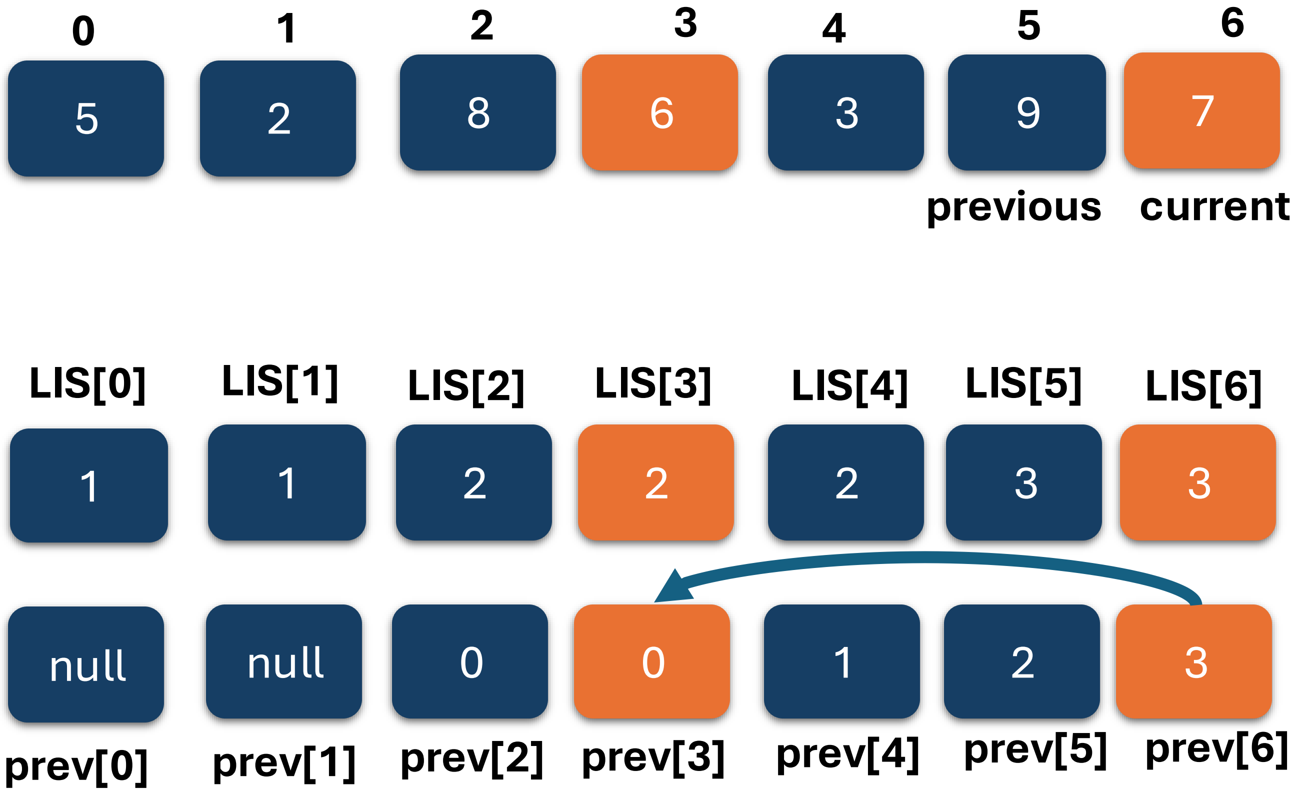
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]
null	null	0	0	1	2	3

```
print_LIS(sequence A, array LIS, array PREV):
    max_count = max(LIS)
    current = LIS.indexOf(max_count)
    numbers = [ ]
    while current != null:
        add A[current] to numbers
        current = PREV[current]
    return reversed(numbers)
```

0	1	2	3	4	5	6
5	2	8	6	3	9	7
					previous	current

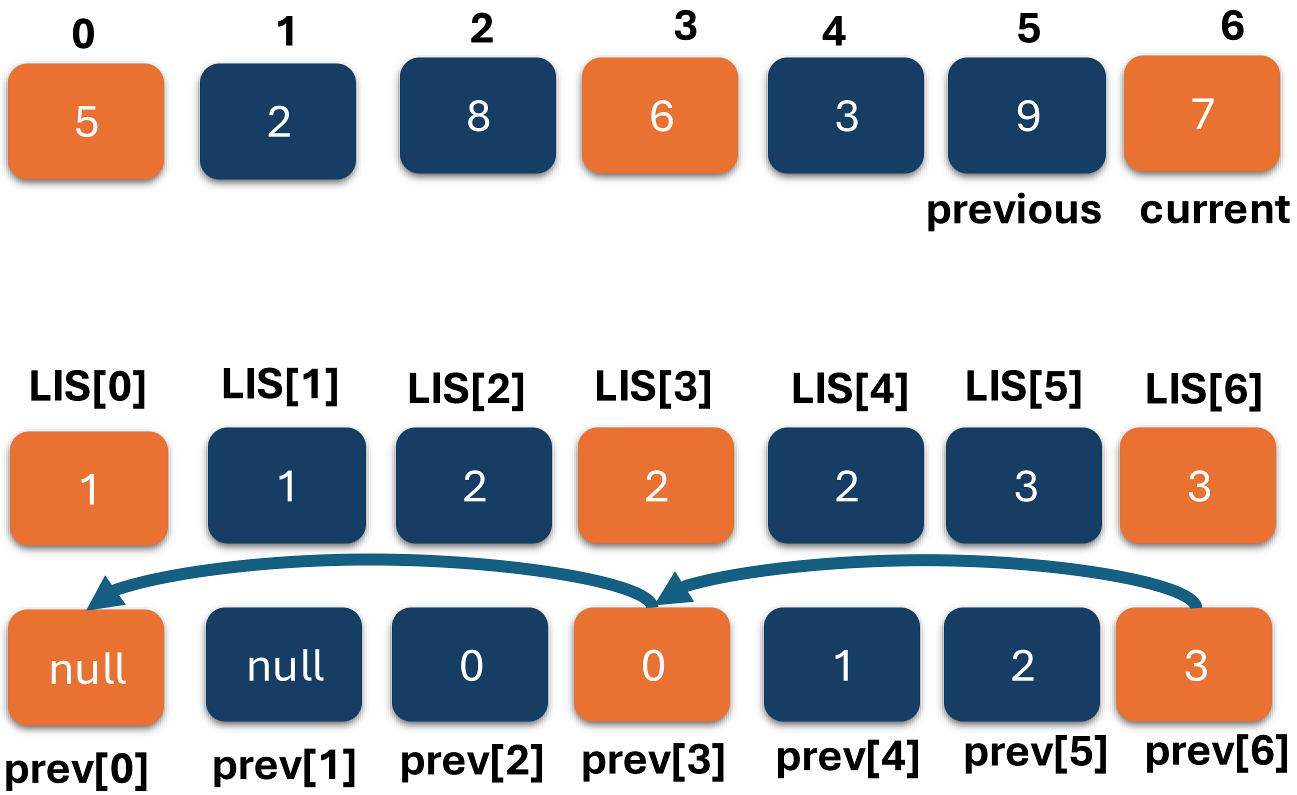
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	3
null	null	0	0	1	2	3
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
print_LIS(sequence A, array LIS, array PREV):
    max_count = max(LIS)
    current = LIS.indexOf(max_count)   6
    numbers = [ ]
    while current != null:
        add A[current] to numbers   A[6]
        current = PREV[current]   Current = PREV[6]=3
    return reversed(numbers)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
					previous	current
LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	3
null	null	0	0	1	2	3
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

```
print_LIS(sequence A, array LIS, array PREV):
        max_count = max(LIS)
        current = LIS.indexOf(max_count)    6
        numbers = [ ]
        while current != null:
                add A[current] to numbers    A[3]
                current = PREV[current]    Current = PREV[3]=0
        return reversed(numbers)
```



0	1	2	3	4	5	6
5	2	8	6	3	9	7
					previous	current

LIS[0]	LIS[1]	LIS[2]	LIS[3]	LIS[4]	LIS[5]	LIS[6]
1	1	2	2	2	3	3
null	null	0	0	1	2	3
prev[0]	prev[1]	prev[2]	prev[3]	prev[4]	prev[5]	prev[6]

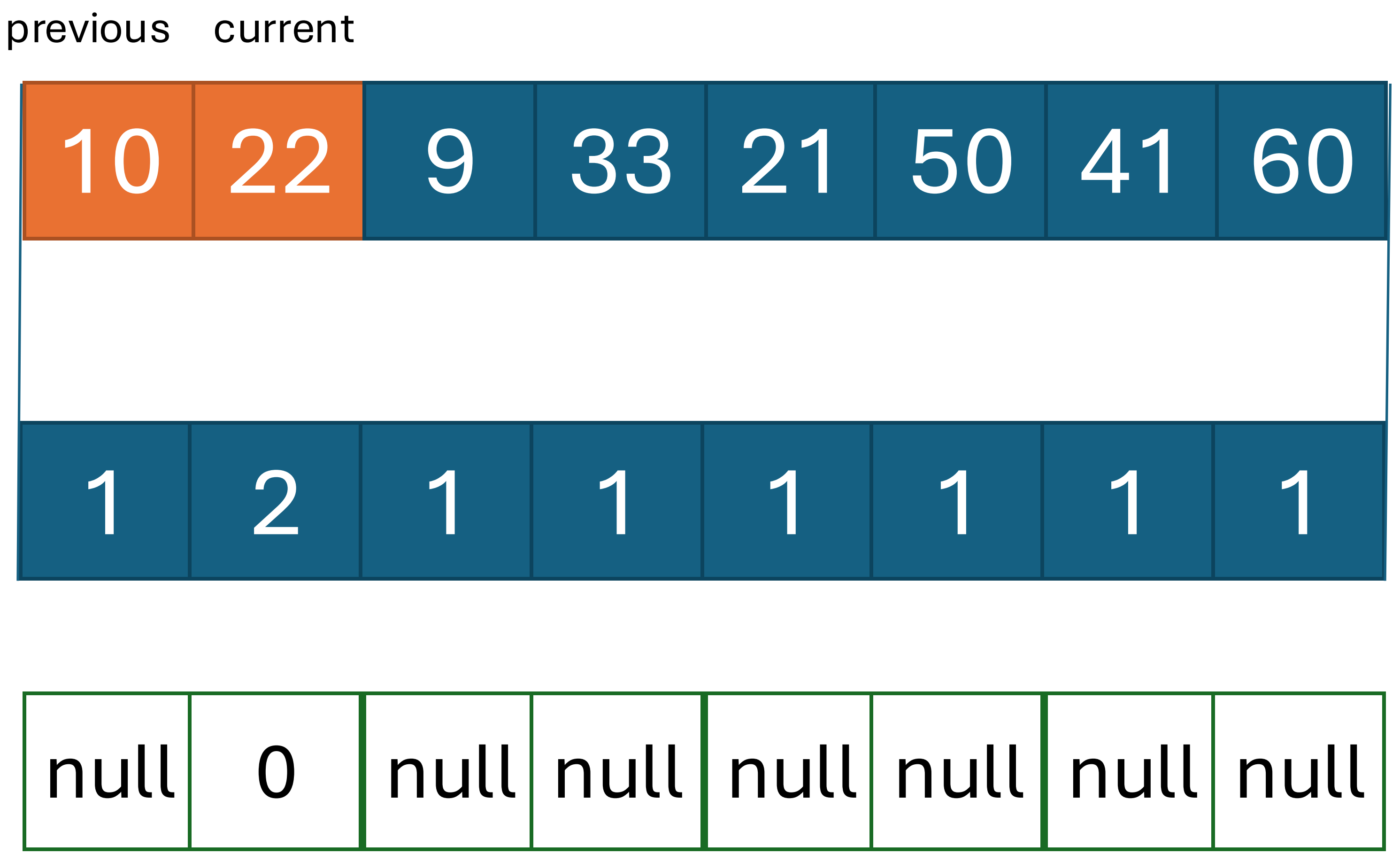
```
print_LIS(sequence A, array LIS, array PREV):
    max_count = max(LIS)
    current = LIS.indexOf(max_count)   6
    numbers = [ ]
    while current != null:
        add A[current] to numbers   A[0]
        current = PREV[current]   Current = PREV[0]=null
    return reversed(numbers)
```

Another Example

previous	current						
10	22	9	33	21	50	41	60
1	1	1	1	1	1	1	1

null	null	null	null	null	null	null	null

Example



previous	current						
10	22	9	33	21	50	41	60
1	2	1	1	1	1	1	1
null	0	null	null	null	null	null	null

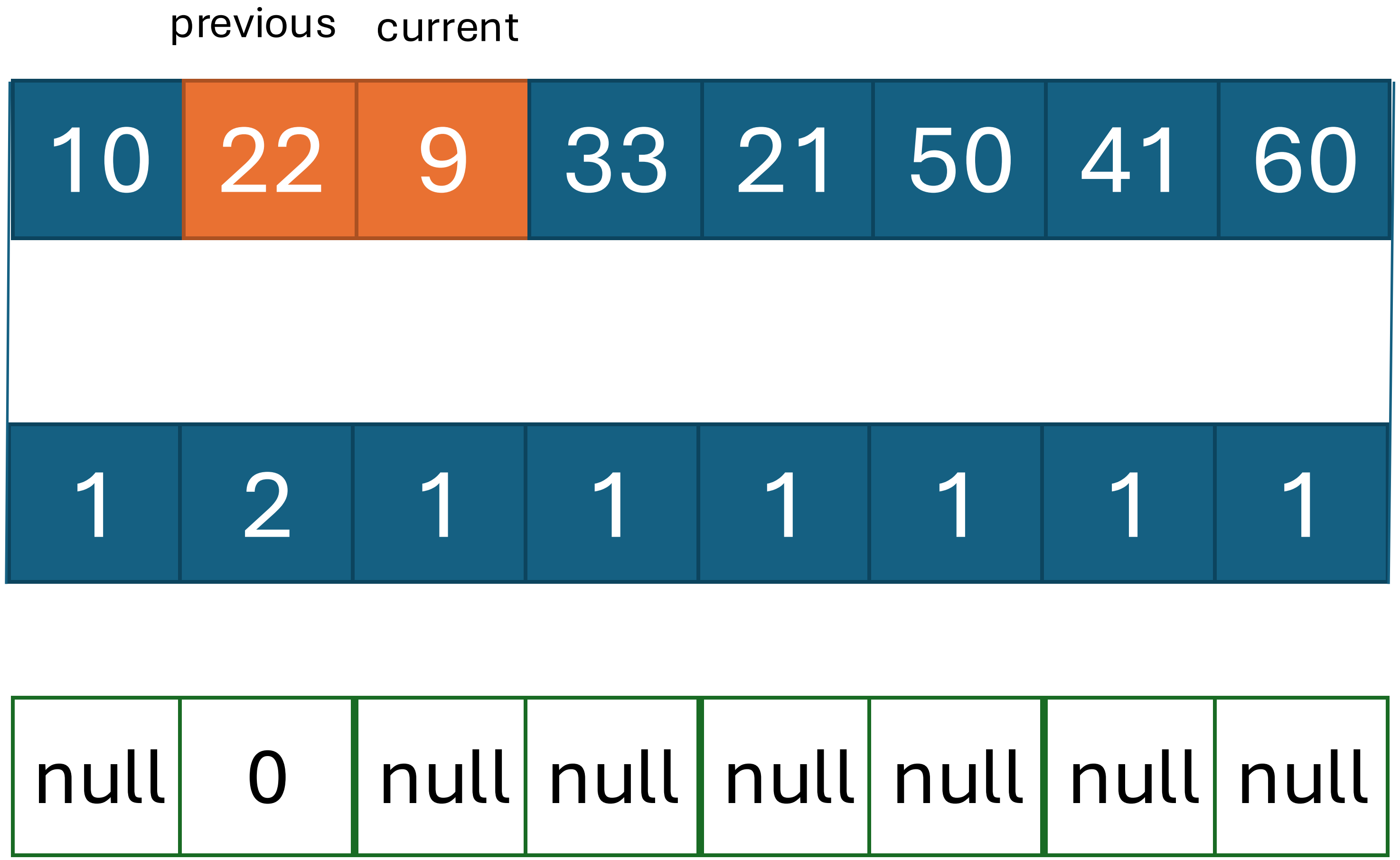
Example

previous		current					
10	22	9	33	21	50	41	60
1	2	1	1	1	1	1	1
null	0	null	null	null	null	null	null

Example

previous		current					
10	22	9	33	21	50	41	60
1	2	1	1	1	1	1	1
null	0	null	null	null	null	null	null

Example

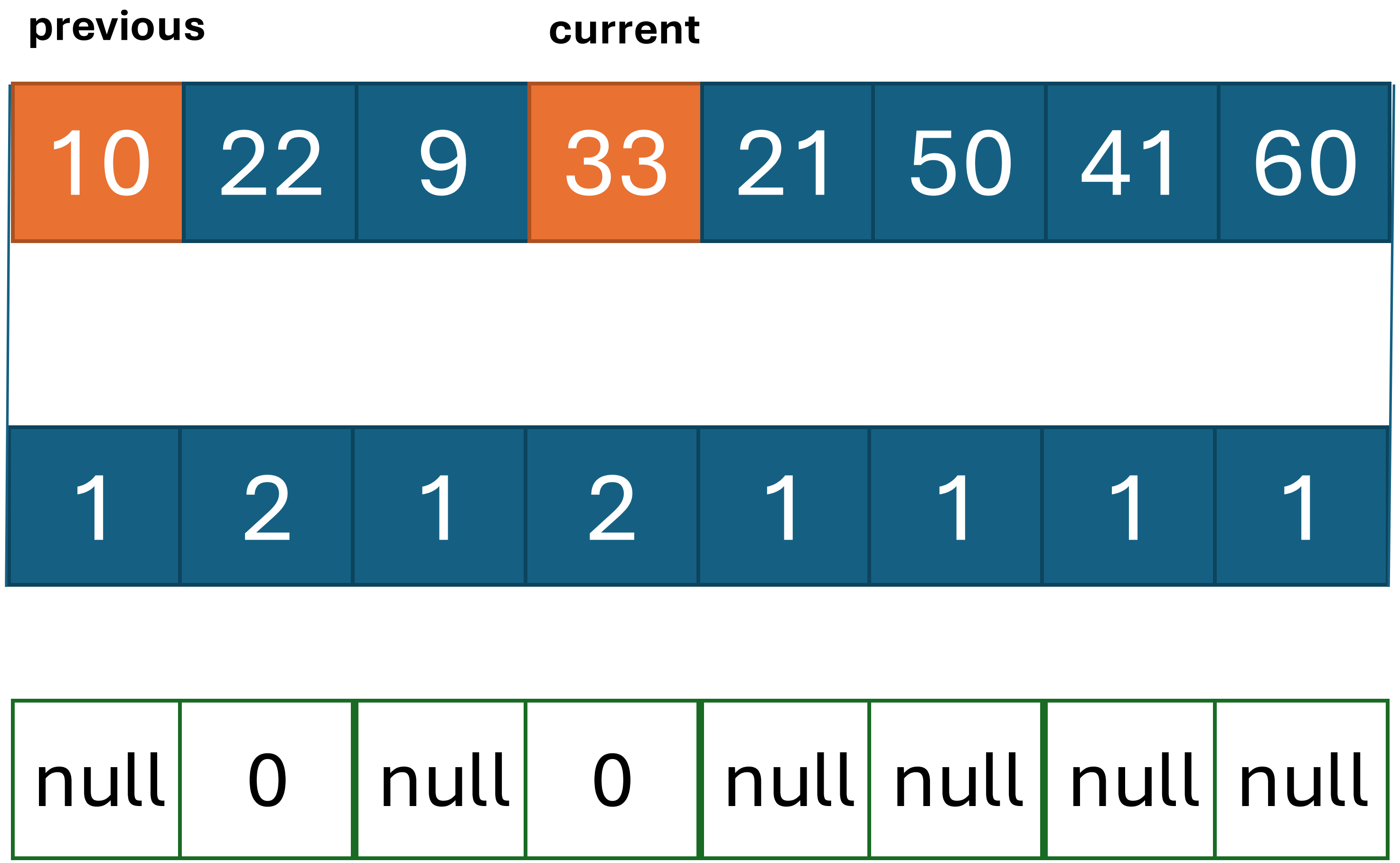


	previous	current					
10	22	9	33	21	50	41	60
1	2	1	1	1	1	1	1
null	0	null	null	null	null	null	null

Example

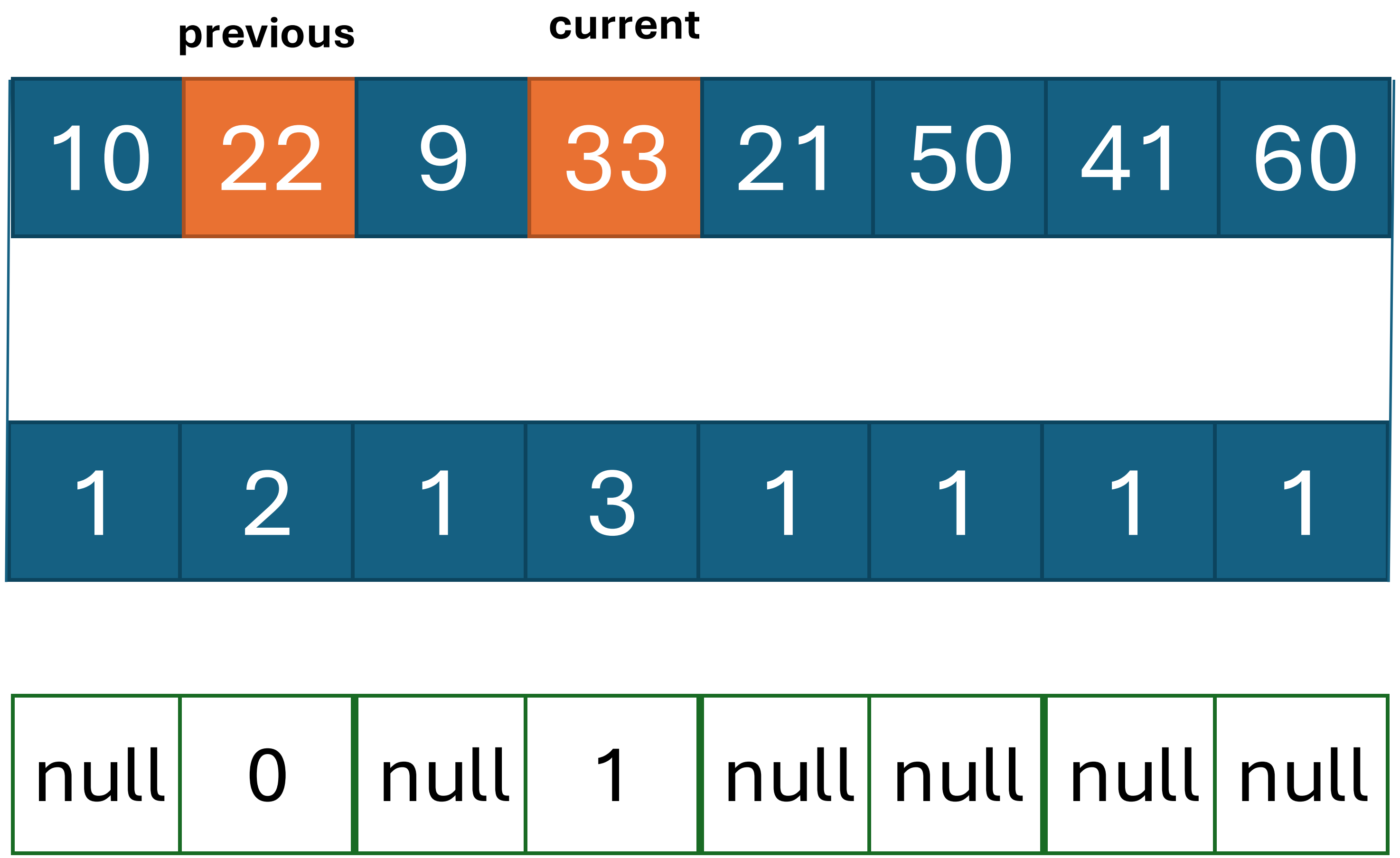
previous			current				
10	22	9	33	21	50	41	60
1	2	1	1	1	1	1	1
null	0	null	null	null	null	null	null

Example



previous			current				
10	22	9	33	21	50	41	60
1	2	1	2	1	1	1	1
null	0	null	0	null	null	null	null

Example



	previous		current				
10	22	9	33	21	50	41	60
1	2	1	3	1	1	1	1
null	0	null	1	null	null	null	null

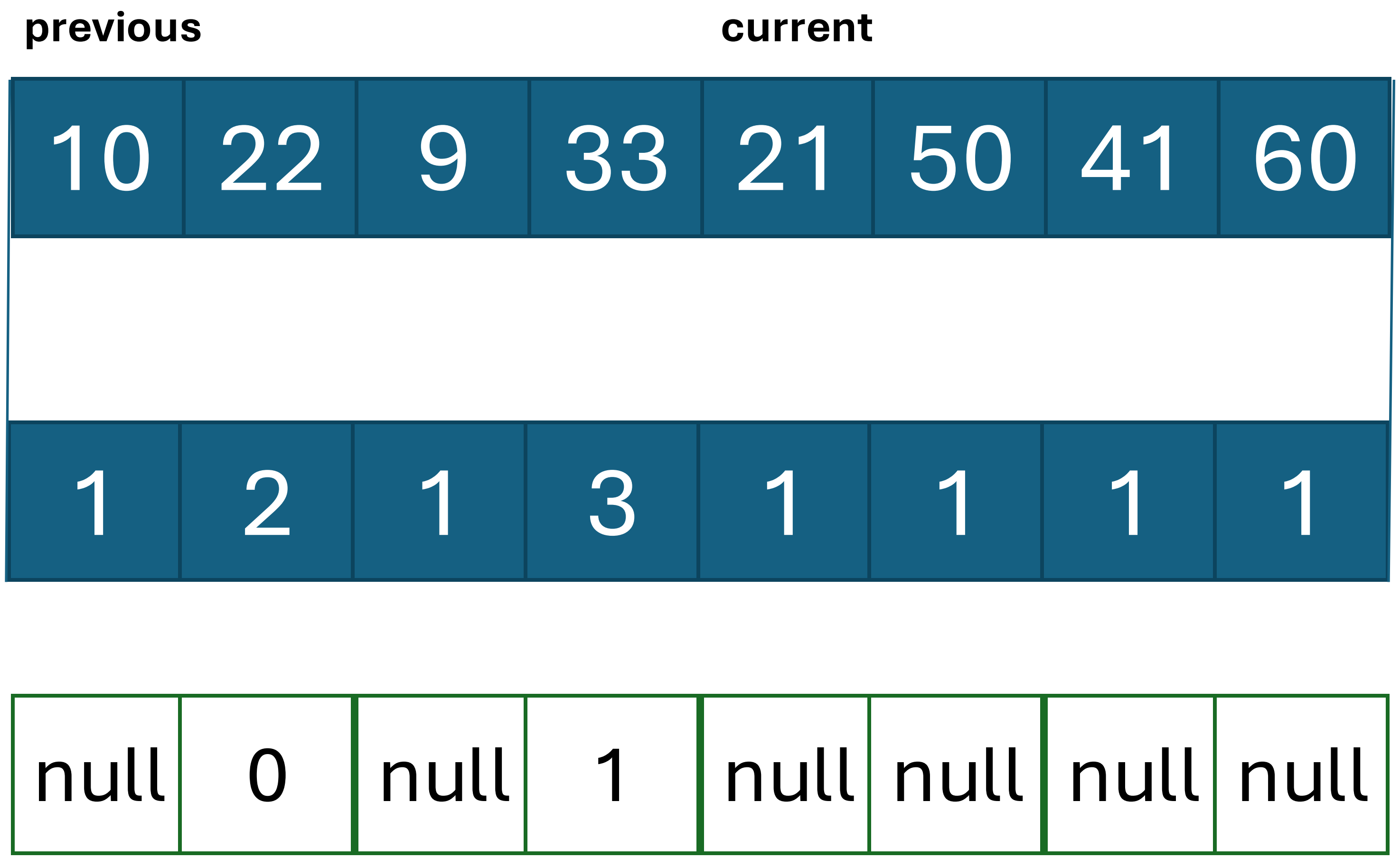
Example

		previous	current				
10	22	9	33	21	50	41	60

1	2	1	3	1	1	1	1

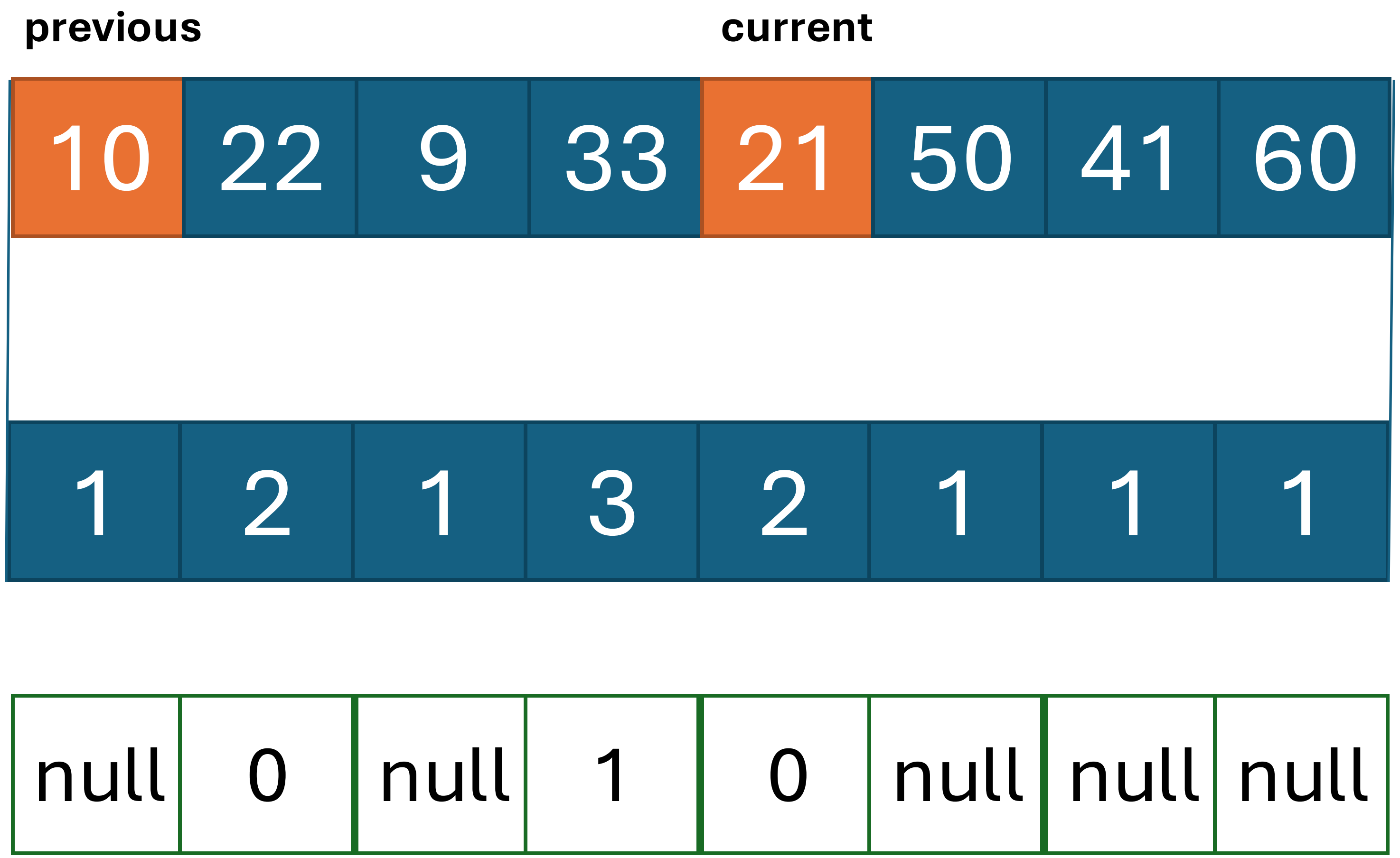
null	0	null	1	null	null	null	null

Example



previous				current			
10	22	9	33	21	50	41	60
1	2	1	3	1	1	1	1
null	0	null	1	null	null	null	null

Example



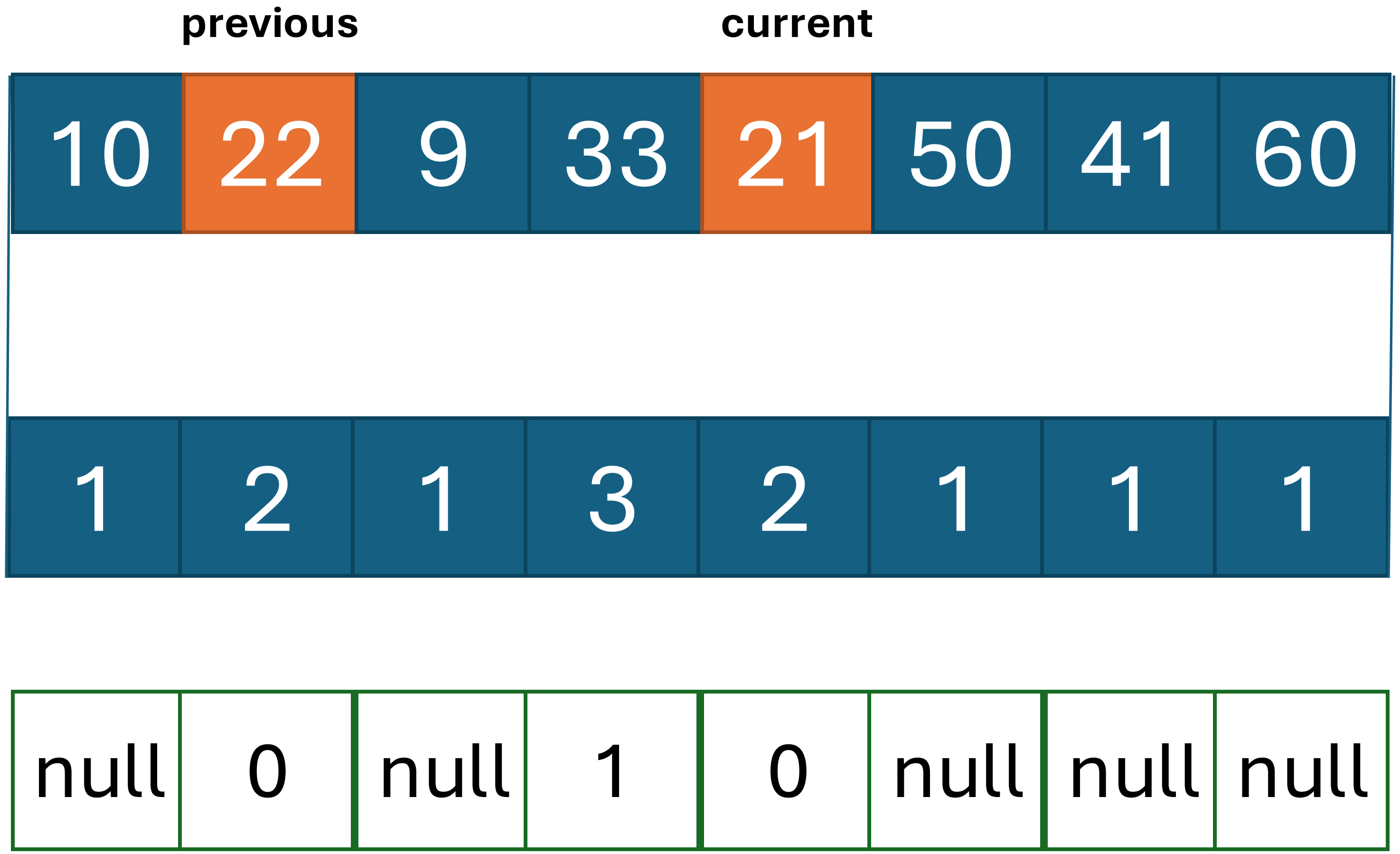
previous

current

10	22	9	33	21	50	41	60
1	2	1	3	2	1	1	1

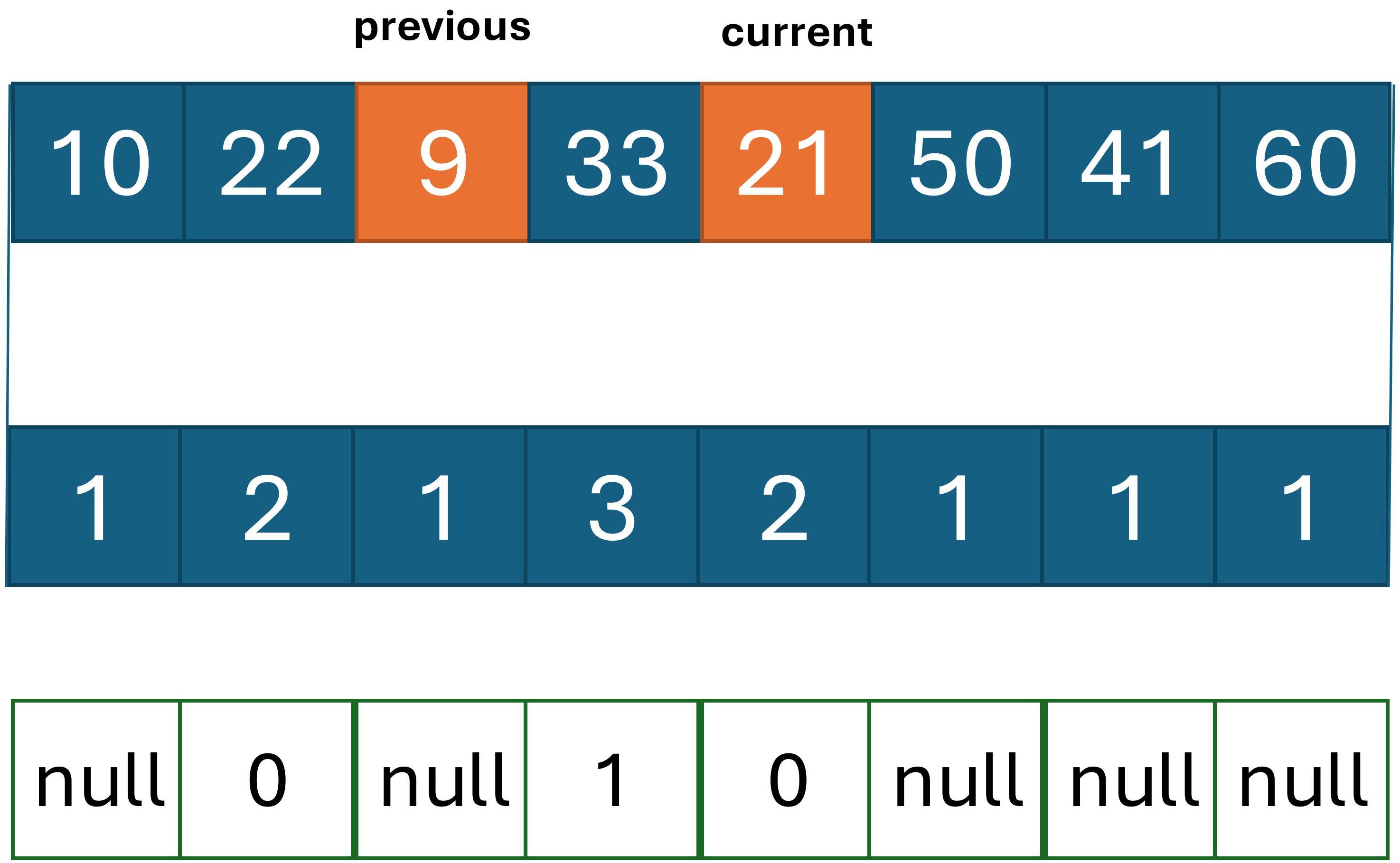
null	0	null	1	0	null	null	null

Example



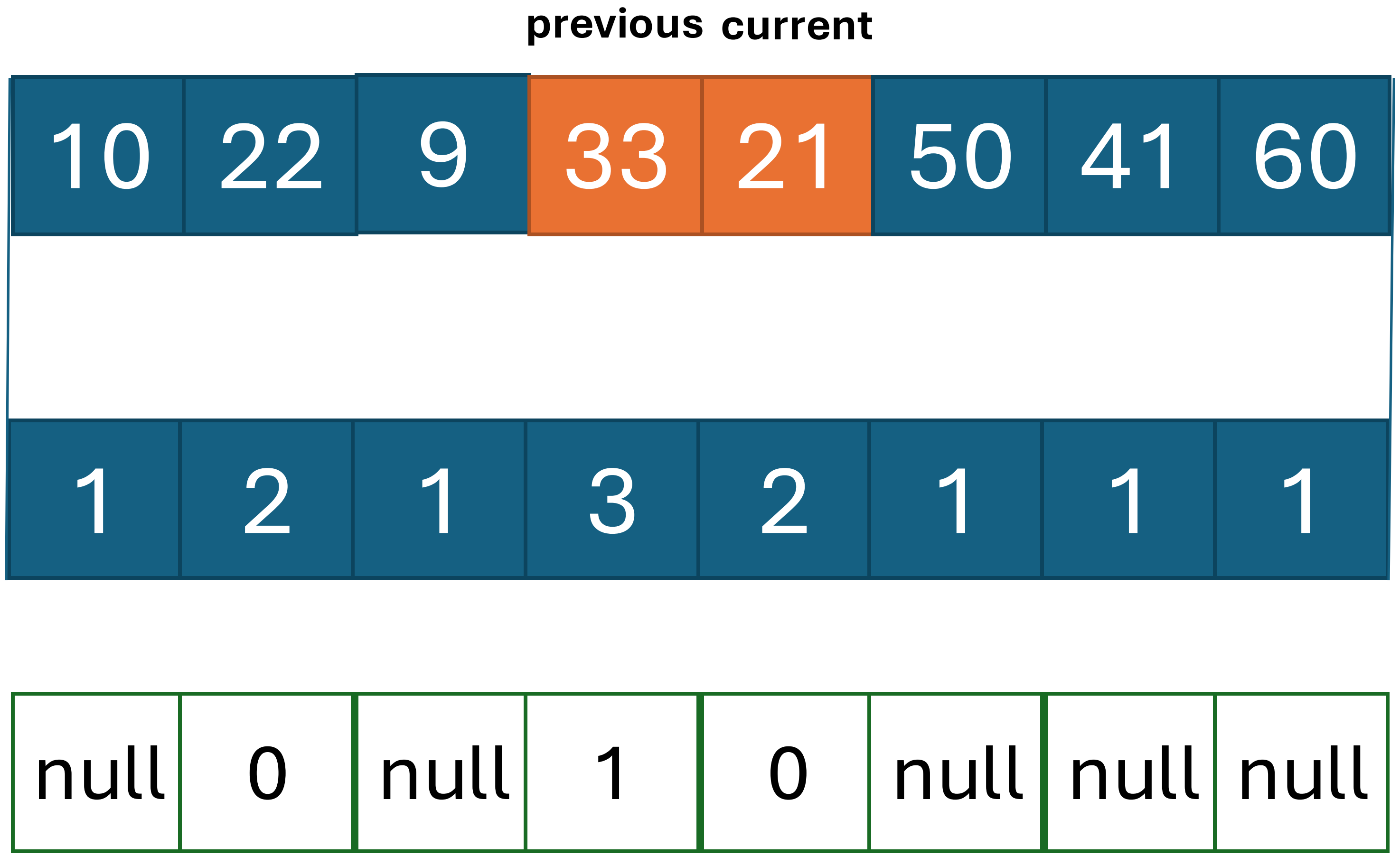
	previous			current			
10	22	9	33	21	50	41	60
1	2	1	3	2	1	1	1
null	0	null	1	0	null	null	null

Example



		previous		current			
10	22	9	33	21	50	41	60
1	2	1	3	2	1	1	1
null	0	null	1	0	null	null	null

Example

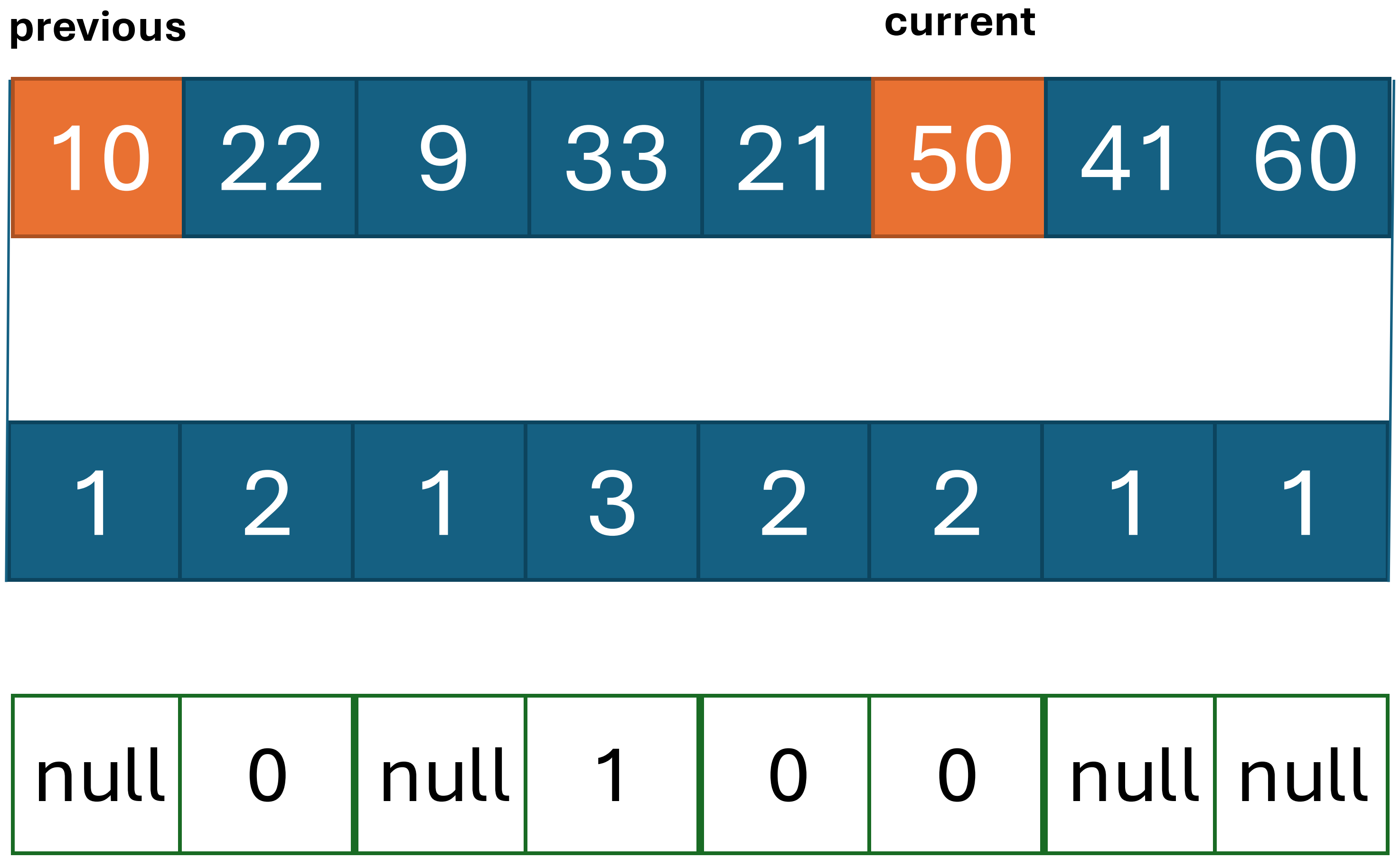


			previous	current			
10	22	9	33	21	50	41	60
1	2	1	3	2	1	1	1
null	0	null	1	0	null	null	null

Example

previous					current		
10	22	9	33	21	50	41	60
1	2	1	3	2	1	1	1
null	0	null	1	0	null	null	null

Example



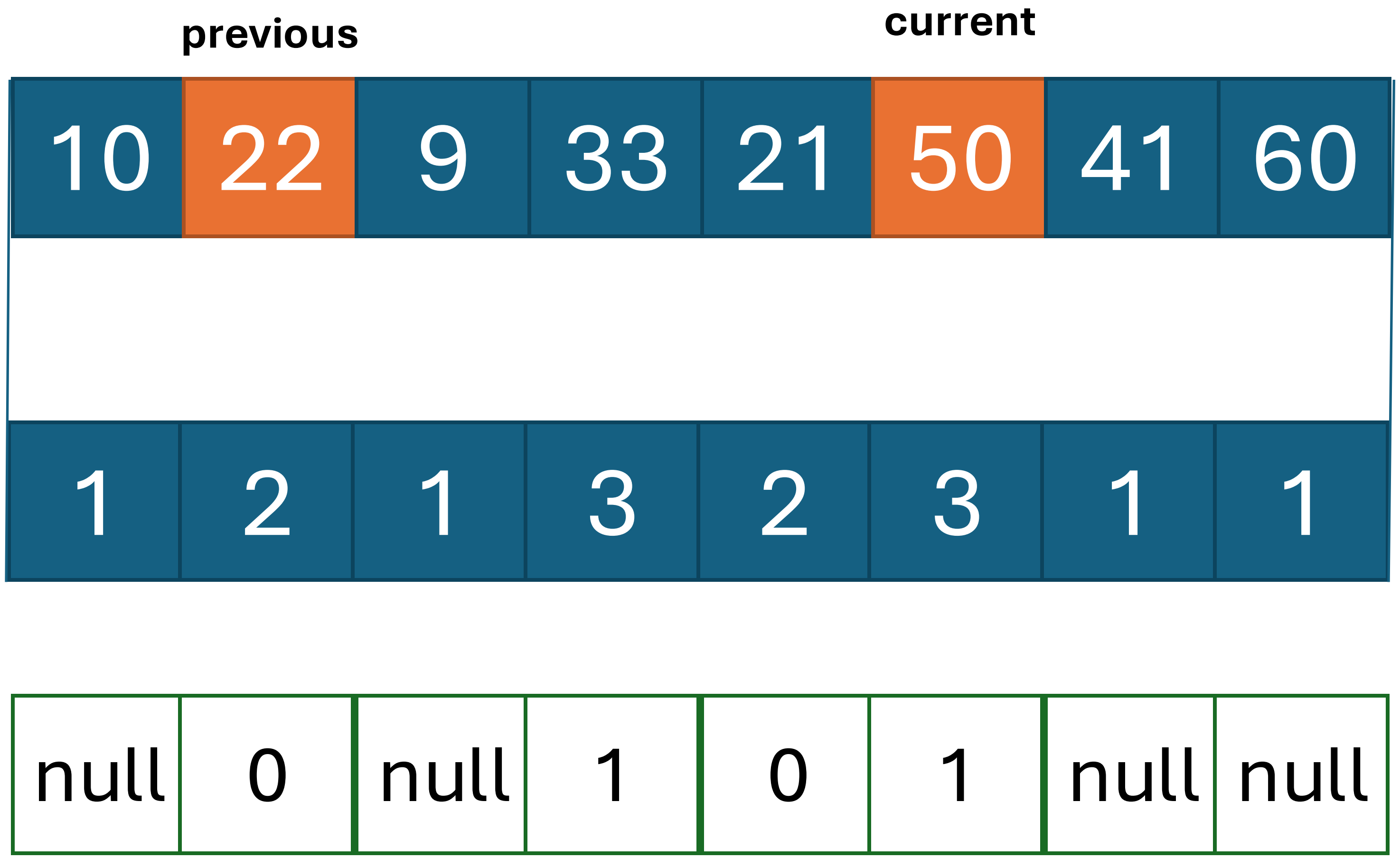
previous

current

10	22	9	33	21	50	41	60
1	2	1	3	2	2	1	1

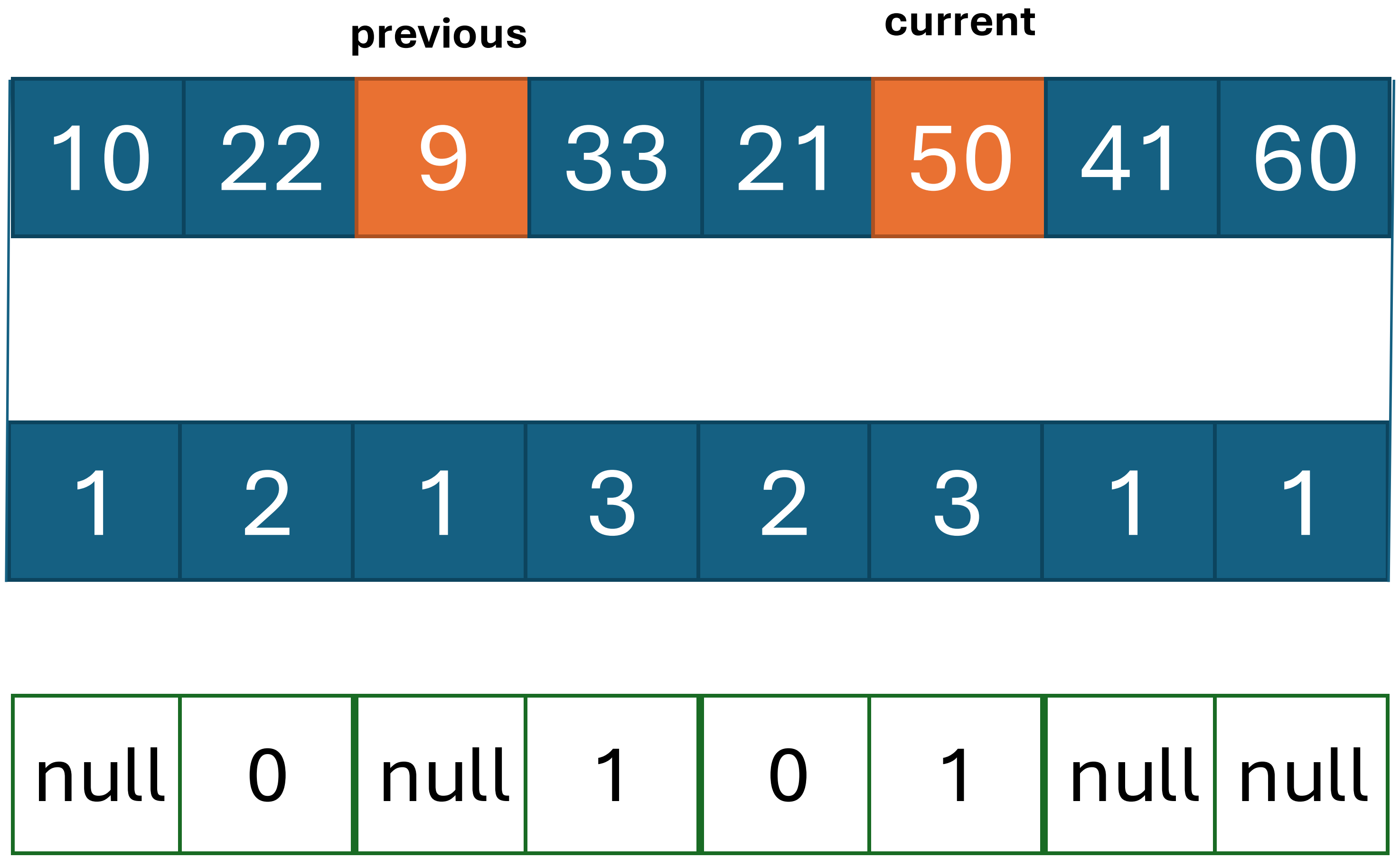
null	0	null	1	0	0	null	null

Example



	previous				current		
10	22	9	33	21	50	41	60
1	2	1	3	2	3	1	1
null	0	null	1	0	1	null	null

Example



		previous			current		
10	22	9	33	21	50	41	60

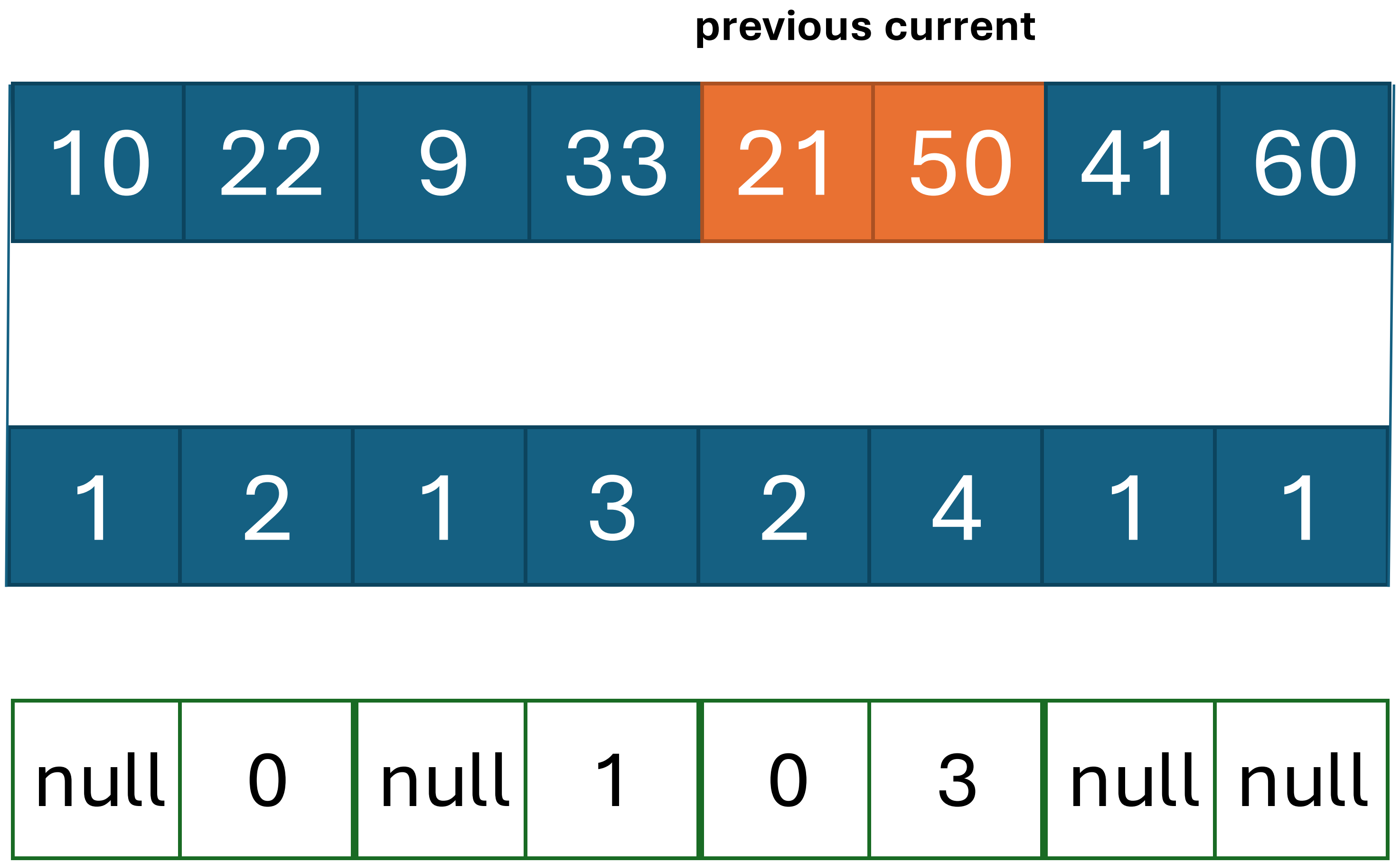
1	2	1	3	2	3	1	1

null	0	null	1	0	1	null	null

Example

			previous		current		
10	22	9	33	21	50	41	60
1	2	1	3	2	4	1	1
null	0	null	1	0	3	null	null

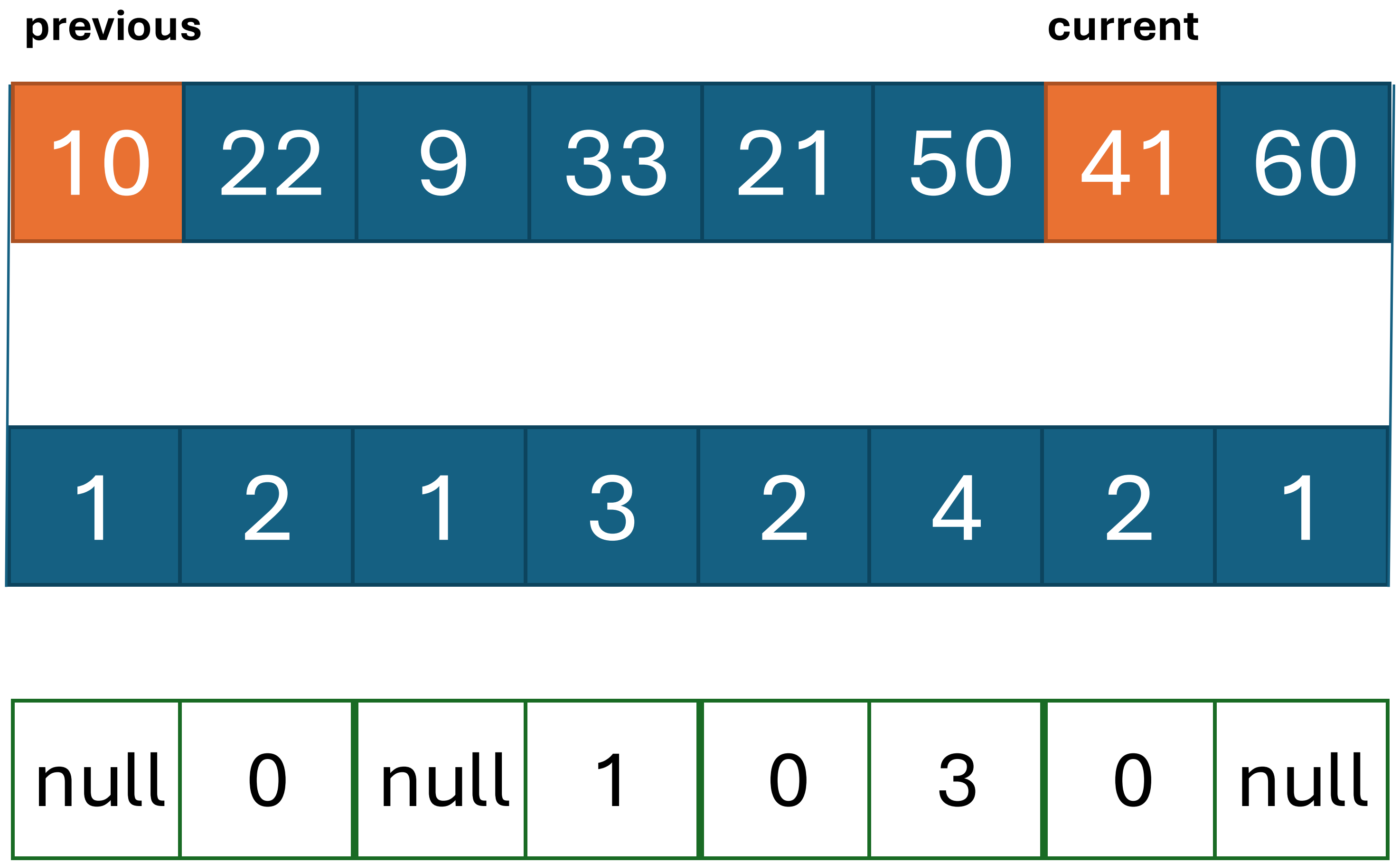
Example



previous current

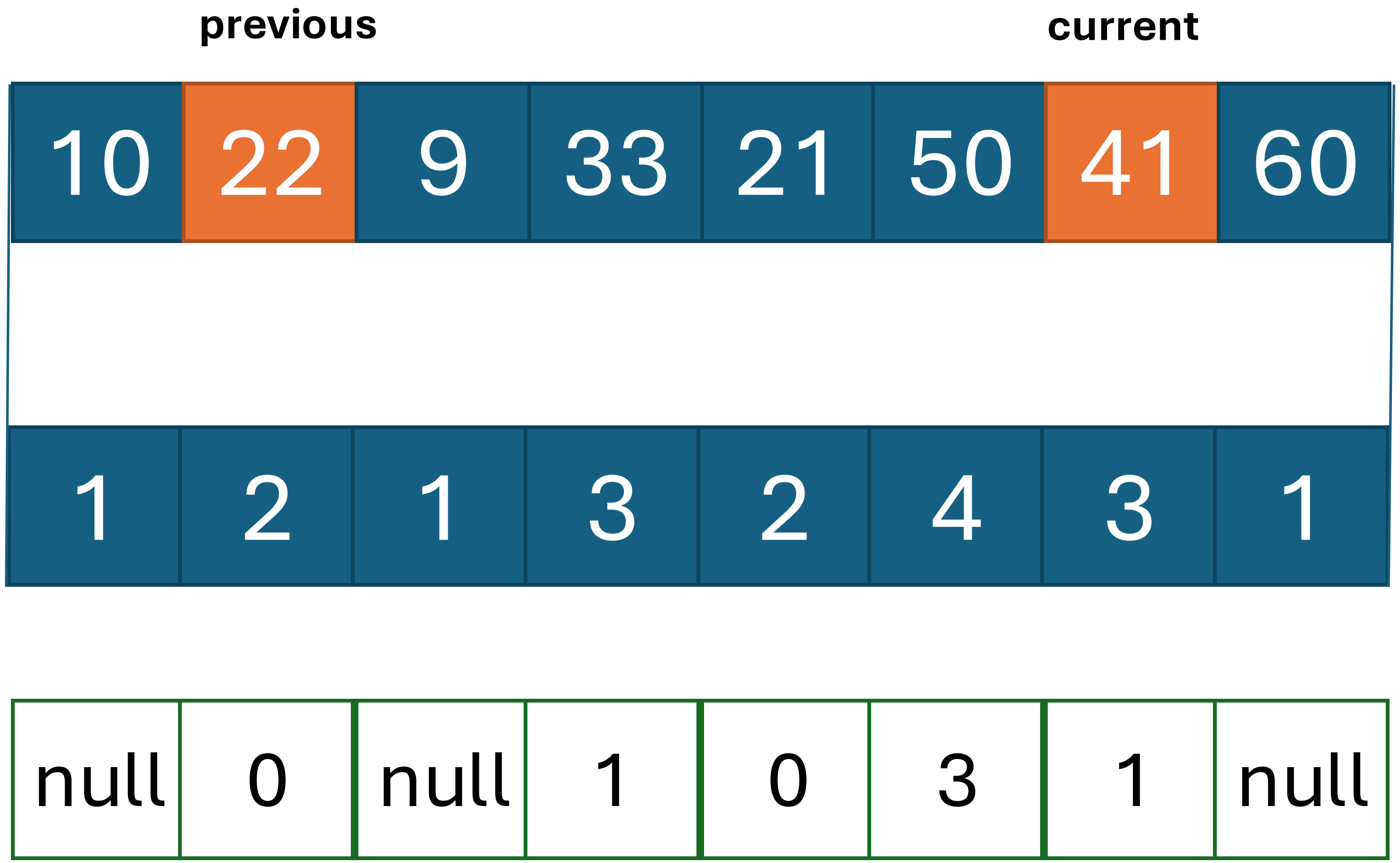
10	22	9	33	21	50	41	60
1	2	1	3	2	4	1	1
null	0	null	1	0	3	null	null

Example



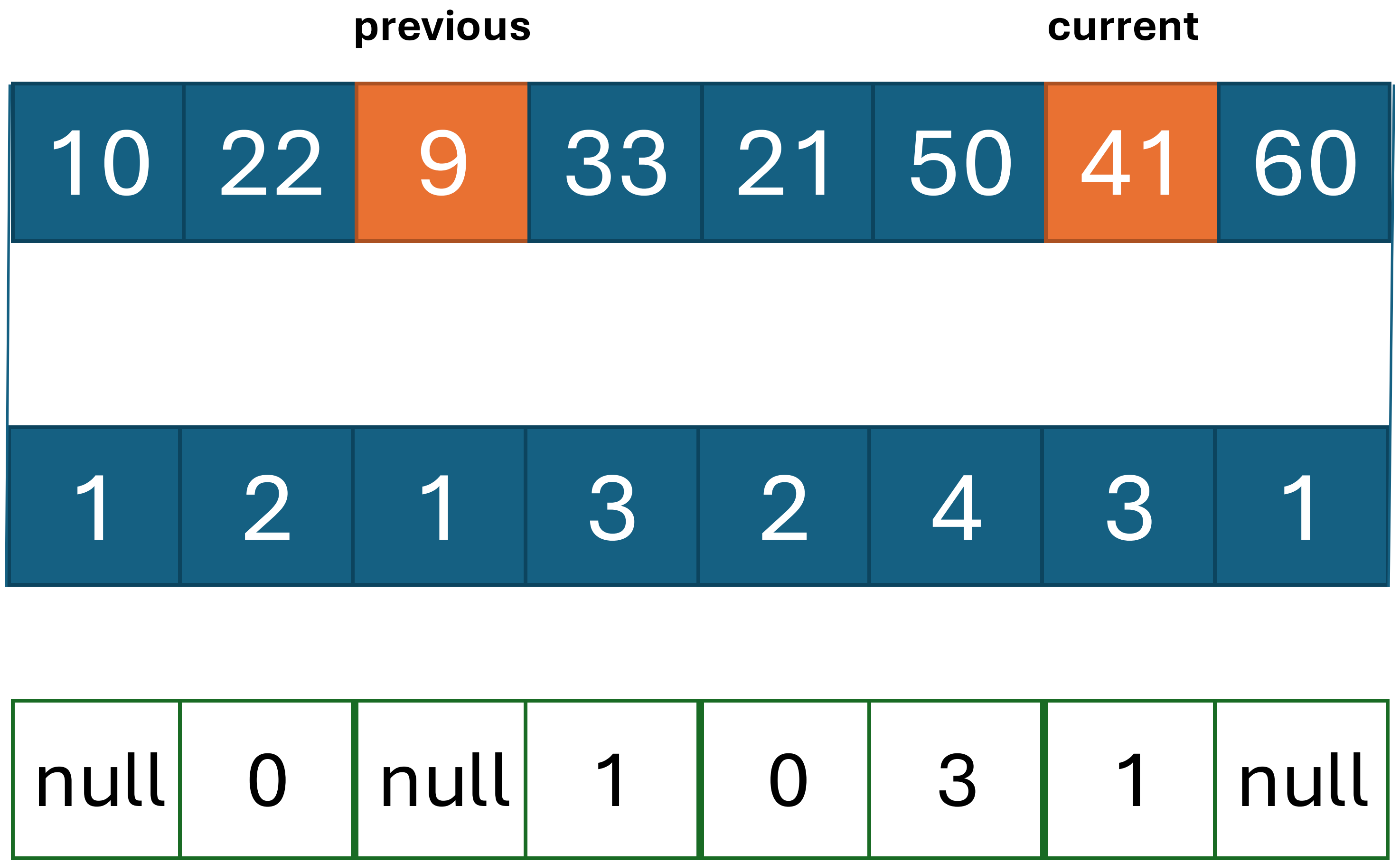
previous						current	
10	22	9	33	21	50	41	60
1	2	1	3	2	4	2	1
null	0	null	1	0	3	0	null

Example



10	22	9	33	21	50	41	60
1	2	1	3	2	4	3	1
null	0	null	1	0	3	1	null

Example



10	22	9	33	21	50	41	60
1	2	1	3	2	4	3	1
null	0	null	1	0	3	1	null

Example

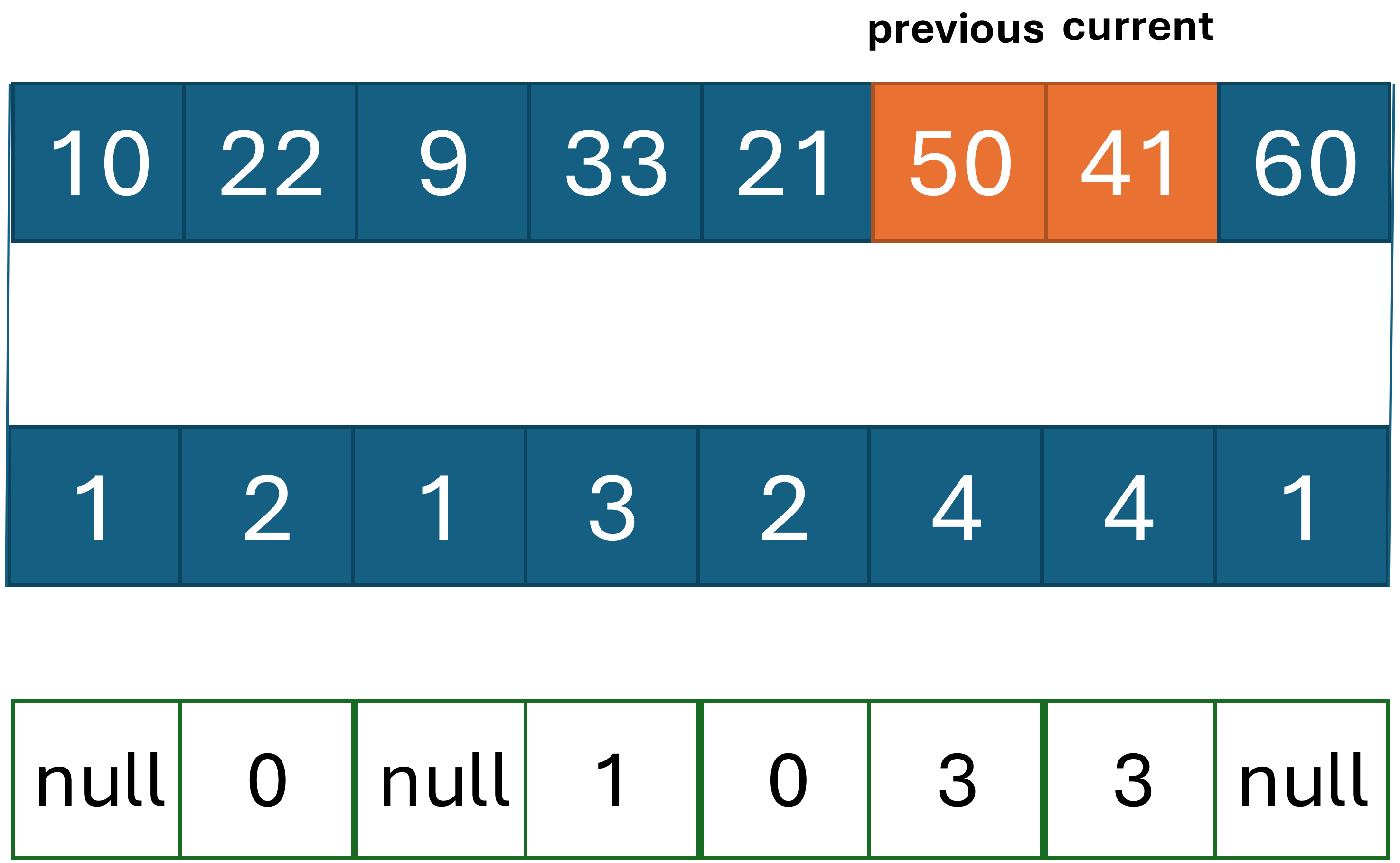
			previous			current	
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	1
null	0	null	1	0	3	3	null

Example

				previous		current	
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	1

null	0	null	1	0	3	3	null

Example



					previous	current	
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	1
null	0	null	1	0	3	3	null

Example

previous current

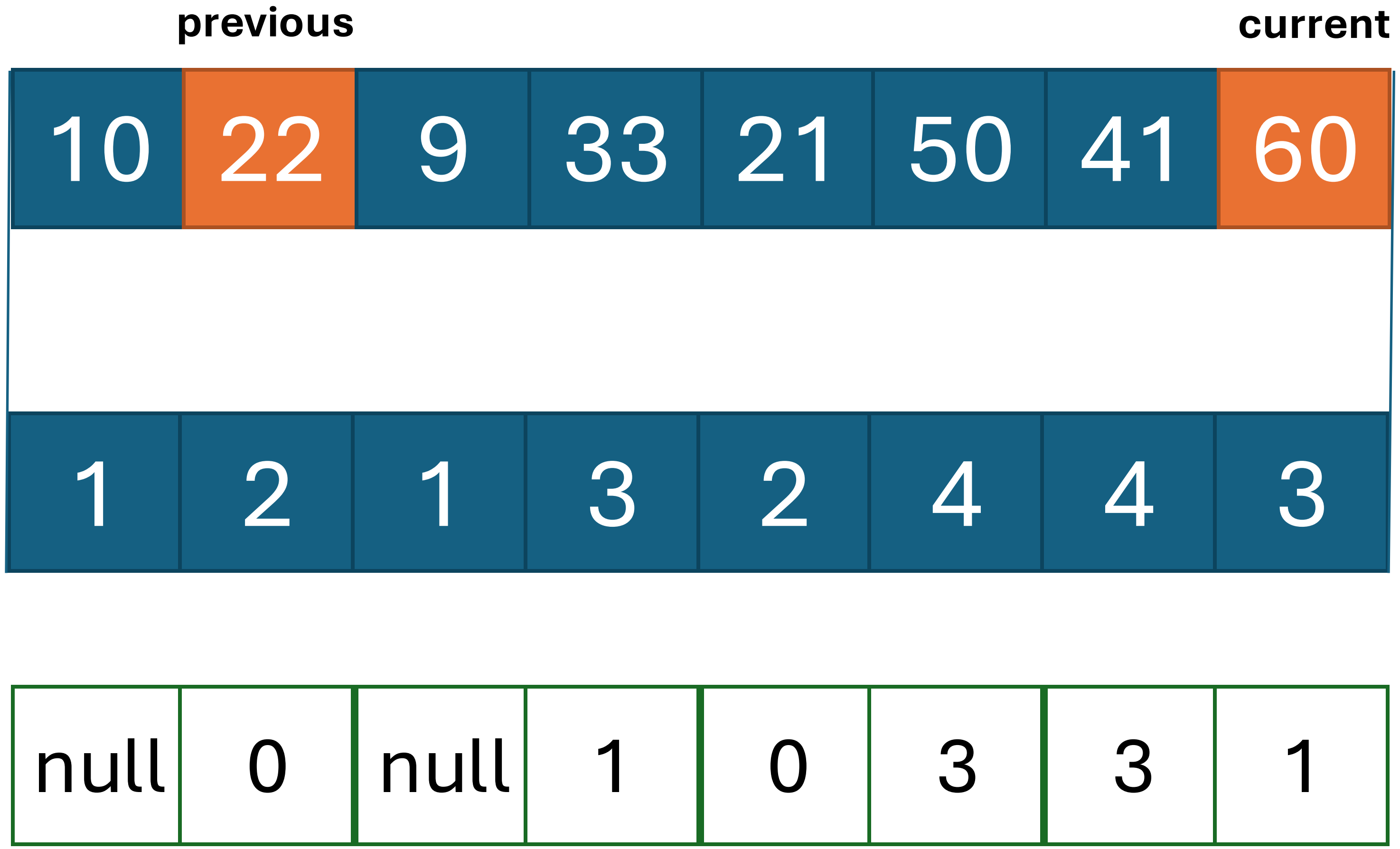
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	1
null	0	null	1	0	3	3	null

Example

previous							current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	2

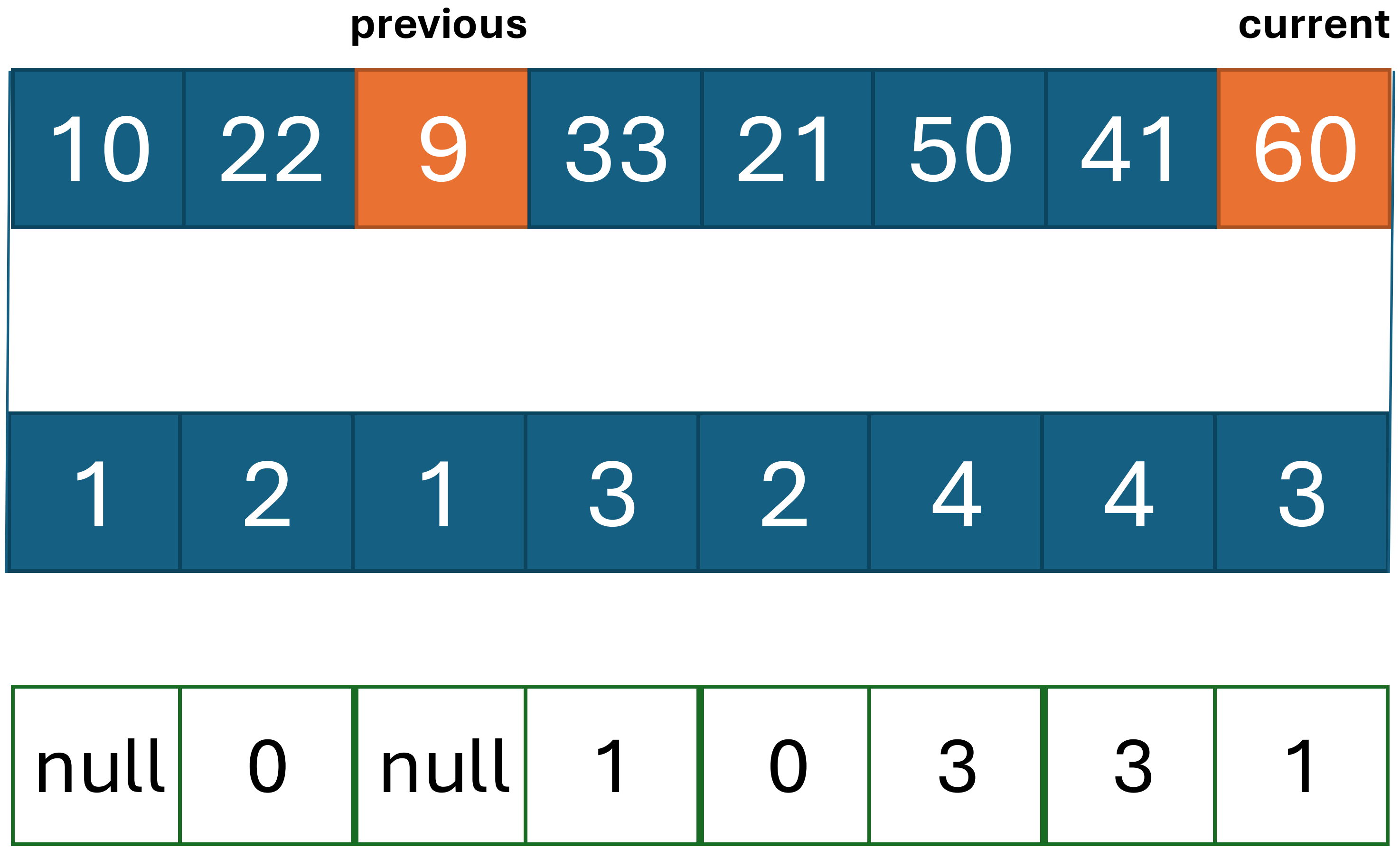
null	0	null	1	0	3	3	0

Example



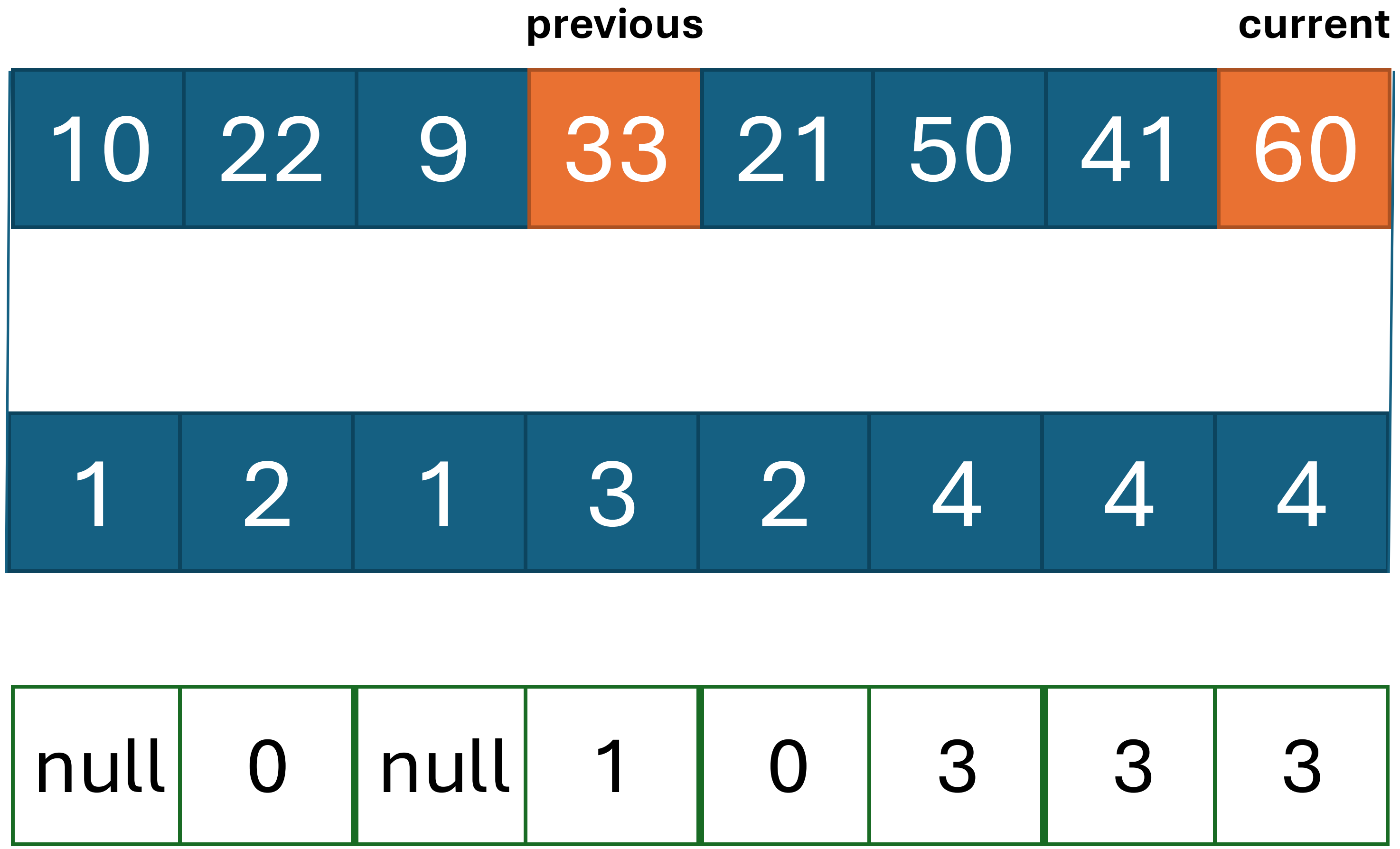
	previous						**current**
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	3
null	0	null	1	0	3	3	1

Example



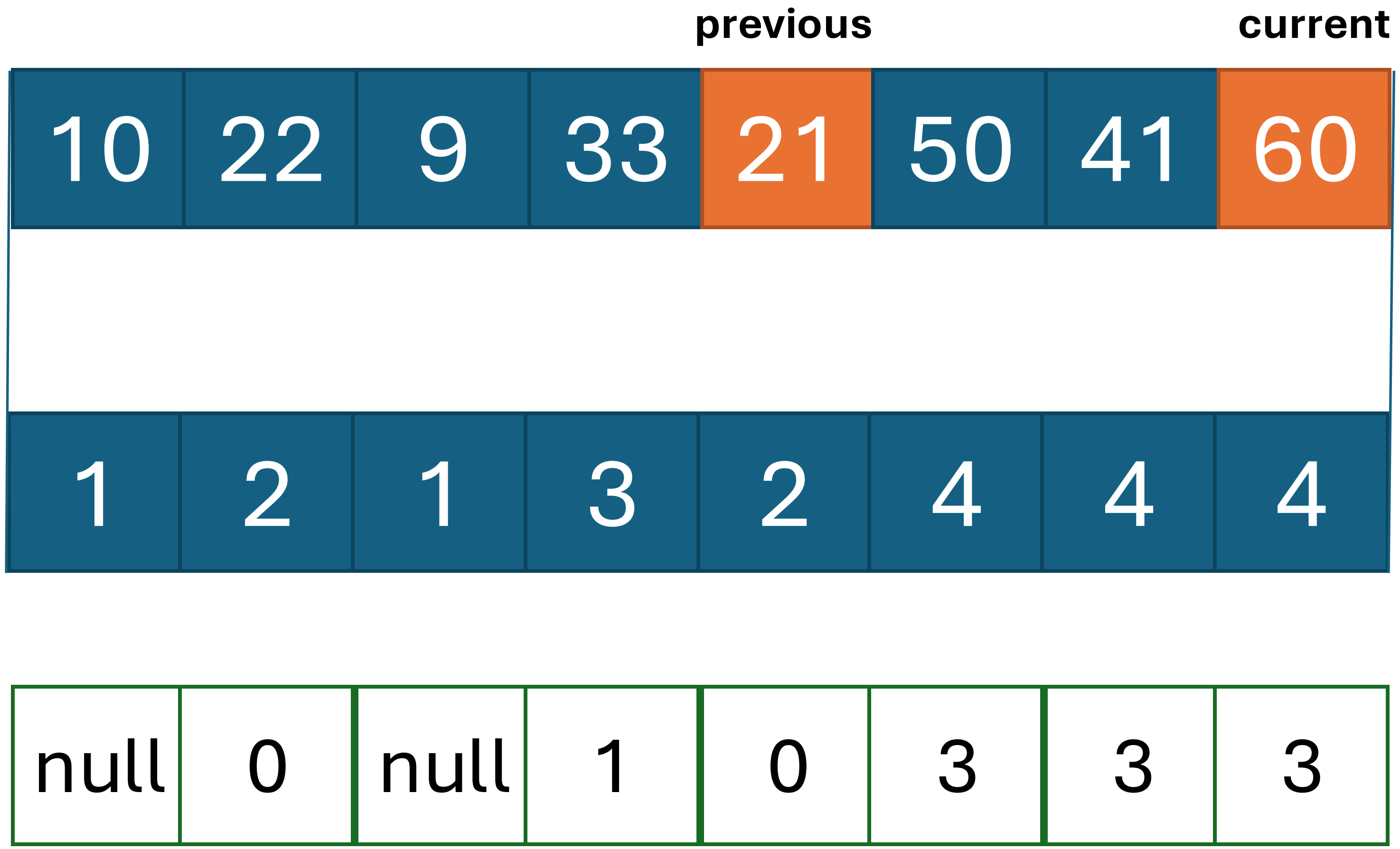
		previous					current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	3
null	0	null	1	0	3	3	1

Example



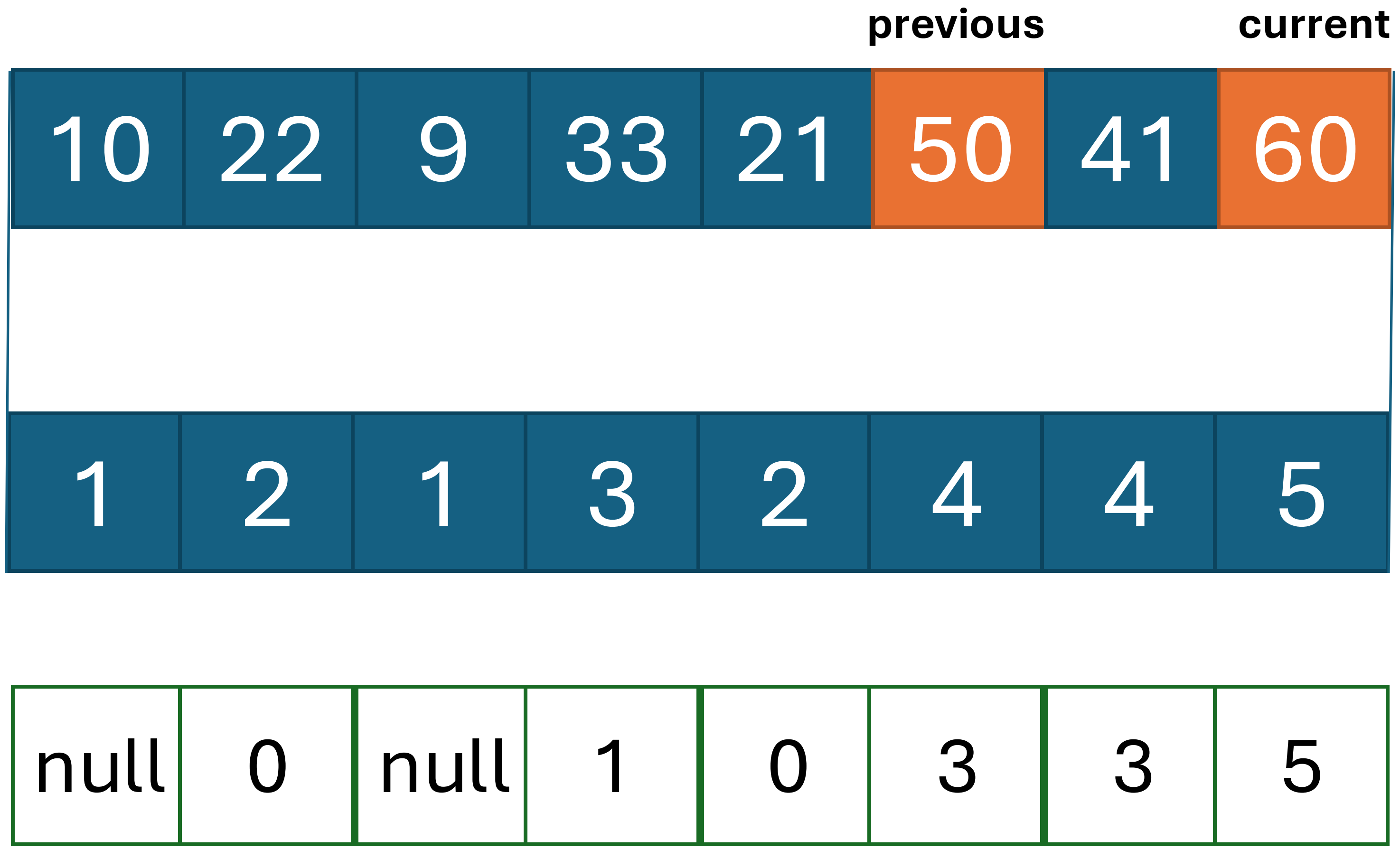
			previous				current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	4
null	0	null	1	0	3	3	3

Example



				previous			current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	4
null	0	null	1	0	3	3	3

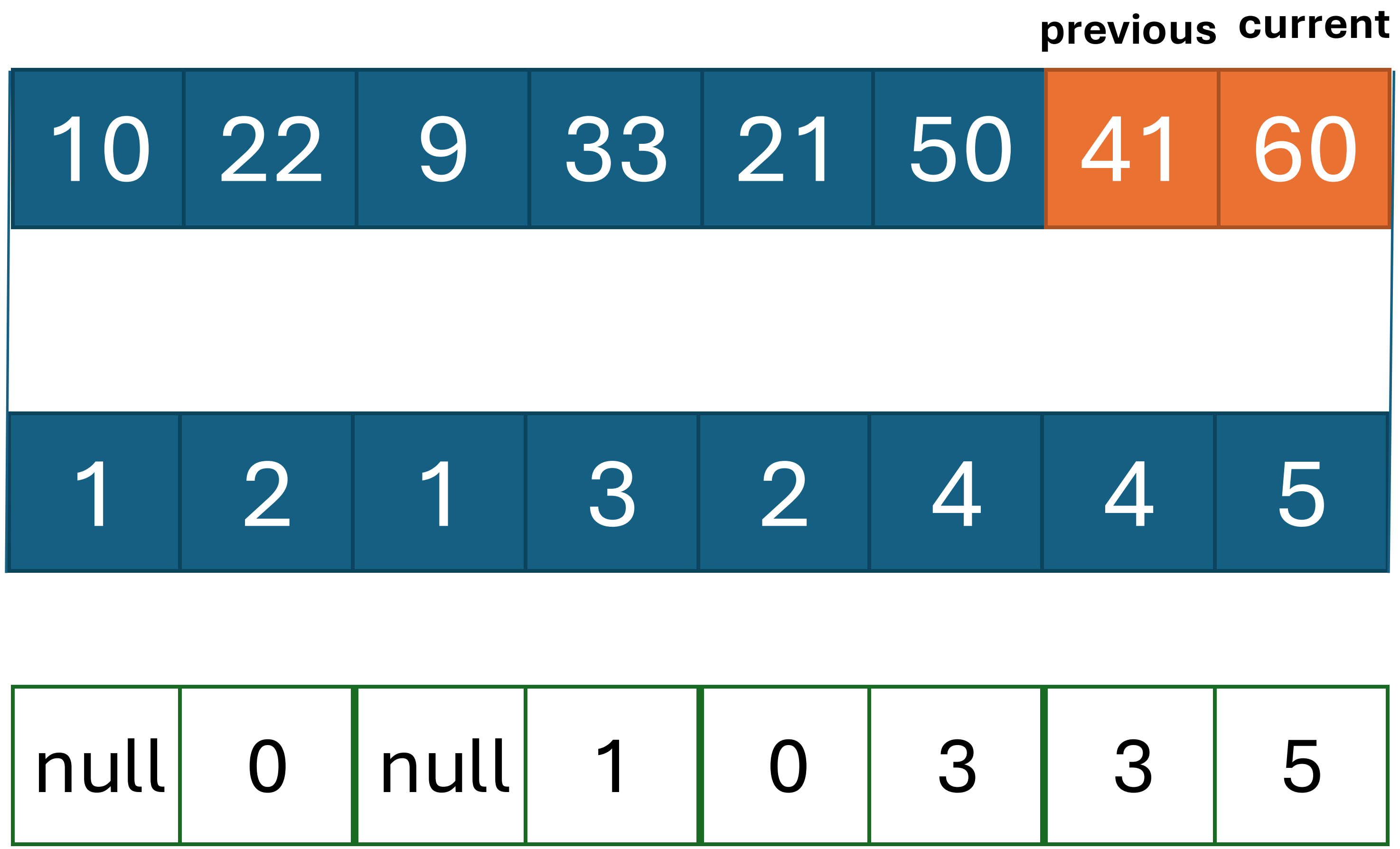
Example



					previous		current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	5

null	0	null	1	0	3	3	5

Example

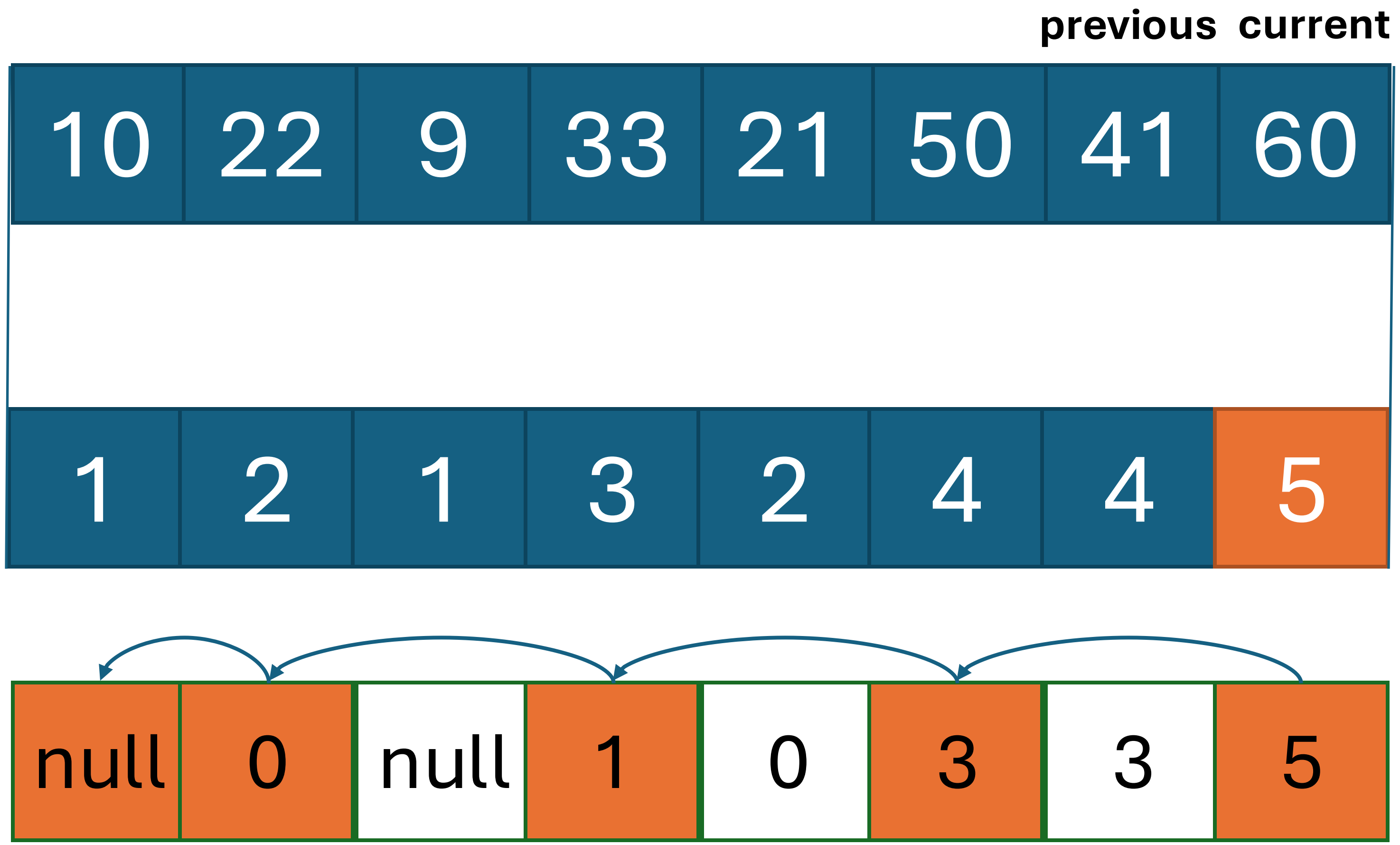


						previous	current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	5
null	0	null	1	0	3	3	5

Example

						previous	current
10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	5
null	0	null	1	0	3	3	5

Example



previous current

10	22	9	33	21	50	41	60
1	2	1	3	2	4	4	5
null	0	null	1	0	3	3	5

DP Solution

- Initial Invocation: LIS_LENGTH
- Algorithm takes $O(n^2)$ time

End of Lecture