FURTHER ALGORITHMS

LEVENSHTEIN DISTANCE

* The Levenshtein distance algorithm shows how similar 2 strings are, by representing a number of the combined additions, deletions and substitutions needed.
* The higher the number, the more different the strings are

Example

“kitten” => “sitting” = **3**

**Why?**

* **K**itten -> **S**itting (substitute *s* for *k*)
* Sitt**e**n -> Sitt**i**ng (substitute *I* for *e*)
* Sittin**g** -> Sittin**g** (Insertion of *g*)

3 operations

1. Substitute
2. Insertion
3. Deletion

Matrix visual

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **#** | **K** | **I** | **T** | **T** | **E** | **N** |
| **#** | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
| **S** | **1** | 1 | 2 | 3 | 4 | 5 | 6 |
| **I** | **2** | 2 | 1 | 2 | 3 | 4 | 5 |
| **T** | **3** | 3 | 2 | 1 | 2 | 3 | 4 |
| **T** | **4** | 4 | 3 | 2 | 1 | 2 | 3 |
| **I** | **5** | 5 | 4 | 3 | 2 | 2 | 3 |
| **N** | **6** | 6 | 5 | 4 | 3 | 3 | 2 |
| **G** | **7** | 7 | 6 | 5 | 4 | 4 | *3* |

* (1,1) – to transform *k* to *s* requires 1 operation (substitution)
* (3, 4) – *kit* to *sitt* requires 2 operations:
  + Substitute *k* with *s*
  + Insert *t* at the end
* (6, 7) – This is the full string comparisons

Typescript implementation

const from = ‘tset’;

const to = ‘test’;

const matrix: number[][] = [];

// these for loops populate the matrices rows & columns

for (let i = 1; i <= from.length; i++) {  
 matrix[i] = i;

}

for(let i = 1; i <= to.length; i++) {

matrix[0][i] = i;

}

// populate full matrix

for (let i = 1; i <= from.length; i++) {

for(let j = 1; j <= to.length; i++) {

const substitutionCost = (from[i-1] === to[j-1]) ? 0 : 1;

matrix[i][j] = Math.min(

matrix[I - 1][j] + 1,

matrix[i][j – 1] + 1,

matrix[i – 1][j – 1] + substitutionCost

);

}

}

console.table(matrix)