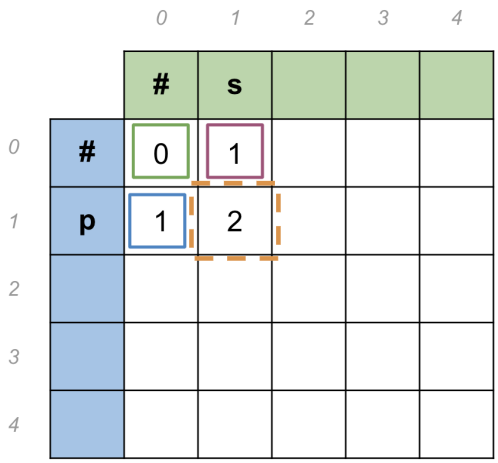
Minimum edit distance algorithm

When computing the minimum edit distance, you would start with a *source word* and transform it into the *target word*. Let's look at the following example:

Source: play → Target: stay
Cost: insert: 1, delete: 1, replace: 2

p → s

insert + *delete*:	p → ps → s:	2
delete + *insert*:	p → # → s:	2
replace:	p → s:	2



To go from $\#$ → $\#$ you need a cost of 0. From p → $\#$ you get 1, because that is the cost of a delete. p → s is 2 because that is the minimum cost one could use to get from **p** to **s**. You can keep going this way by populating one element at a time, but it turns out there is a faster way to do this. You will learn about it next.