

One Away: There are three types of edits - insertion, deletion, and replacement. Given two strings, write a function to check if they are one or zero edits away.

Example:

- pale, ple \rightarrow T
- pales, pale \rightarrow T
- pale, bale \rightarrow T
- pale, bake \rightarrow F

First Thoughts:

- Replacement checking seems simple by following this pattern:

ex)

p	a	l	e
b	a	l	e

0 1 2 3

iterate the strings at the same time & check if each character matches

- Insertion & deletion is a bit trickier, and the same technique cannot be applied

ex)


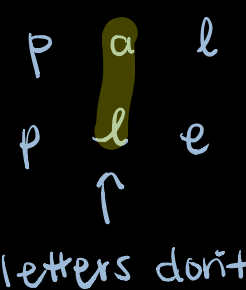
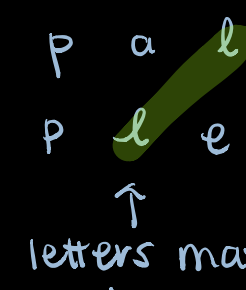
p	a	l	e
p		l	e

0 1 2 3

- But the difference for insertion/deletion is the length of the strings.

If the lengths differ by ± 1 , we would be checking for insertion/deletion, not replacement. order also matters

↳ if the letters do not match, we can iterate to the next letter for the longer word, but stay on the same letter for the shorter word.

ex)   

- whether its insertion or deletion, it doesn't matter - we just need to know the lengths of the strings & if they are the same (or not.)

function **isOneAway**(str1, str2) {

let counter = 0;

if (str1.length === str2.length) {

for (let i = 0; i < str1.length; i++) {

if (str1[i] !== str2[i]) {

counter++;

}

}

if (counter > 1) {

return false;

}

return true;

}

let shortStr, longStr;

if (str1.length === str2.length - 1) {

shortStr = str1;

longStr = str2;

} else if (str1.length === str2.length + 1) {

shortStr = str2;

longStr = str1;

} else {

```
return false;
```

```
}
```

```
let i, j = 0; // originally to be a for loop, but realized I needed 2 separate index trackers
```

```
while (i < longstr.length) {
```

```
  if (longstr[i] !== shortstr[j]) {
```

```
    counter++;
```

```
  } else {
```

```
    j++;
```

```
  }
```

```
  i++;
```

```
}
```

```
if (counter > 1) {
```

```
  return false;
```

```
}
```

```
return true;
```

```
}
```

- instead of a counter, using a boolean value would also work

ex) let foundDiff = false

```
for ...
```

```
  if (not a match) {
```

```
    if (foundDiff) {
```

```
      return false;
```

```
    }
```

```
    foundDiff = true;
```

```
  }
```

- making shortstr / longstr wasn't necessary, could just defer to setting 1 to be short or 2 to be long, and passing them into functions as (str1, str2) or (str2, str1), using order

* while loop can be modified to check against the length of the shortstr as well

Complexity

space: $O(1)$, no additional structures used

Time: $O(n)$, where n is the length of the shorter string (assuming the * modification)