Sequence Alignment from Scratch with BWA

Barış Salman

February 9, 2023

Contents

Abstract

The burrow wheeler algorithm is used to perform short read alignment, a cornerstone process in bioinformatics, especially next-generation sequencing. Short sequence alignment or mapping is one of the primary steps in re-sequencing projects where short reads from a fragmented genome are aligned back to a reference genome. We can use this aligned data to discover small or structural variations in clinical or population projects.

1 Intro

Try it out yourself at this notebook.

Living organisms have long strings of DNA in their cells and are made out of 4 bases `A, T, C, G`. The order that makes up the long stretches of DNA determines the characteristics of the cell. In the case of the human genome, the order of 3.25 billion of these bases decides the makeup of our cells. Over the last few decades, we have had technologies that can reveal this order of bases called *sequencing*. However, the majority of these sequencing technologies can only sequence around 100 bases which are not very long. However, the strength of these sequencing methods lies in their massive capacity to perform this sequencing operation millions of times in parallel. To get a more comprehensive view, we need to put these millions of small 100-base long sequences back together. It's a wide puzzle with small and many pieces. We have sequenced genomes lots of organisms and now have picture for the puzzle that we can use as reference. However, not every individual and their genome is the same so when we are doing resequencing pieces come from a different picture. Having pieces from another picture makes solving the puzzle bit more challenging. We don't know what is different in these

pieces and we need to account for that unknown when putting the pieces together. Burrows-Wheeler Transform makes it possible to put these pieces to their place despite their variation. Our end goals is to identify these differences (variation) which makes that organism unique. There are few approaches but in this tutorial we are gonna use FM-index which is used by Burrows-Wheeler Aligner. There are lots of terminology here so lets explain while showing.

2 Implementation

2.1 What is Burrows Wheeler Transform

Burrows-wheeler algorithm relies on burrows wheeler transform. This transform is performed once on the reference sequence before the alignment.

1. Burrows-Wheeler Transform(BWT) is performed by first getting the all the possible rotations (or circular shifts) of the sequence. Lets say we have example reference sequence string `OMICSSBS`. We can get the possible rotations with string splicing below.

We loop over the length of the string and for every loop we add slices of the string from start and end as the index increments we get the shifting effect.

```
from pprint import pprint

example_seq = "OMICSSBS$"
sequence_rotations = []
for index in range(len(example_seq)):
    sequence_rotations.append(example_seq[index:] + example_seq[:index])
pprint(sequence_rotations)
```

```
['OMICSSBS$',
 'MICSSBS$O',
 'ICSSBS$OM',
 'CSSBS$OMI',
 'SSBS$OMIC',
 'SBS$OMICS',
 'BS$OMICSS',
 'S$OMICSSB',
 '$OMICSSBS']
```

2. Then we sort rotations by a property we desire. In this case we are going to sort them alphabetically (or lexicographically if you're fancy).

```
sorted_sequence_rotations = sorted(sequence_rotations)
pprint(sorted_sequence_rotations)
```

```
['$OMICSSBS',
 'BS$OMICSS',
 'CSSBS$OMI',
 'ICSSBS$OM',
 'MICSSBS$O',
 'OMICSSBS$',
 'S$OMICSSB',
 'SBS$OMICS',
 'SSBS$OMIC']
```

3. We get the last column of this sorted rotation.

```
for sequence in sorted_sequence_rotations:
    pprint(sequence[-1])
```

```
'S'
'S'
'I'
'M'
'O'
'$'
'B'
'S'
'C'
```

And that's it. We done the BWT. To utilize this let's move to next step creating the FM-index.

2.2 What is FM-index

To utilize BWT we need to keep a bit more information. This information is the initial order of the sequences before sorting. Here we use the itemgetter to sort the enumerated rotations by the sequences themselves.

```
from operator import itemgetter
initial_indices, sorted_sequence_rotations = zip(*sorted(enumerate(sequence_rotations),

for index, sequence in zip(initial_indices, sorted_sequence_rotations):
    print(f"Initial index: {index} of the: {sequence[-1]}")

Initial index: 8 of the: S
Initial index: 6 of the: S
Initial index: 3 of the: I
Initial index: 2 of the: M
Initial index: 1 of the: O
Initial index: 0 of the: $
Initial index: 7 of the: B
Initial index: 5 of the: S
Initial index: 4 of the: C
```

Suffix array is the sequences that are up character $ which is used by bowtie.

```
for index, sequence in zip(initial_indices, sorted_sequence_rotations):
    print(f"Initial index: {index} of the sequence: {sequence.split('$')[0]}")

Initial index: 8 of the sequence:
Initial index: 6 of the sequence: BS
Initial index: 3 of the sequence: CSSBS
Initial index: 2 of the sequence: ICSSBS
Initial index: 1 of the sequence: MICSSBS
Initial index: 0 of the sequence: OMICSSBS
Initial index: 7 of the sequence: S
Initial index: 5 of the sequence: SBS
Initial index: 4 of the sequence: SSBS
```

We can keep the last column in a sequence.

```
last_column = ""
for sequence in sorted_sequence_rotations:
    last_column += sequence[-1]
print(last_column)

SSIMO$BSC
```

This transformation is reversible and we can get our original sequence back by back tracking the characters in the sequence. To make this back tracking work we need a function that maps the character in last back to first. This is possible because index of an *occurrence* of a character in the first column will be the same in the last column. e.g. We have three `S` characters in our string `OMICSSBS` where the first occurrence of the string `S` in last column is the same `S` in the first column. We can keep track of the occurrences of `S` like below. You can watch the index beside the `S` characters and confirm they're the same by the whole sequence besides them.

```
occurrence_of_s_in_first_column = 0
occurrence_of_s_in_last_column = 0
for index, sequence in zip(initial_indices, sorted_sequence_rotations):
    if sequence[0] == 'S' and sequence[-1] == 'S':
        print(f"{index=} sequence: {sequence} first: {sequence[0]} {occurrence_of_s_in_
        occurrence_of_s_in_first_column += 1
        occurrence_of_s_in_last_column += 1
    elif sequence[0] == 'S':
        print(f"{index=} sequence: {sequence} first: {sequence[0]} {occurrence_of_s_in_
        occurrence_of_s_in_first_column += 1
    elif sequence[-1] == 'S':
        print(f"{index=} sequence: {sequence} first: {sequence[0]} 0 last: {sequence[-1
        occurrence_of_s_in_last_column += 1
    else:
        print(f"{index=} sequence: {sequence} first: {sequence[0]} 0 last: {sequence[-1
```

```
index=8 sequence: $OMICSSBS first: $ 0 last: S 0
index=6 sequence: BS$OMICSS first: B 0 last: S 1
index=3 sequence: CSSBS$OMI first: C 0 last: I 0
index=2 sequence: ICSSBS$OM first: I 0 last: M 0
index=1 sequence: MICSSBS$O first: M 0 last: O 0
index=0 sequence: OMICSSBS$ first: O 0 last: $ 0
index=7 sequence: S$OMICSSB first: S 0 last: B 0
index=5 sequence: SBS$OMICS first: S 1 last: S 2
index=4 sequence: SSBS$OMIC first: S 2 last: C 0
```

Instead of counting them characters one by one like above lets count all the characters and keep it in a lookup index using the last column. The `totals` is how many times we see a character in total and `tallymatrix` is how many we saw up to a given point.

```
totals = {k: 0 for k in "".join(set(last_column))}
tallymatrix = {k: [] for k in "".join(set(last_column))}
for c in last_column:
    totals[c] += 1
    for k in tallymatrix.keys():
        if k != c and tallymatrix[k]:
            tallymatrix[k].append(tallymatrix[k][-1])
        elif k == c:
            tallymatrix[k].append(totals[c])
        else:
            tallymatrix[k].append(0)
print(totals)
print(tallymatrix)
```

```
{'O': 1, 'S': 3, 'M': 1, 'C': 1, 'B': 1, '$': 1, 'I': 1}
{'O': [0, 0, 0, 0, 1, 1, 1, 1, 1], 'S': [1, 2, 2, 2, 2, 2, 2, 3, 3], 'M': [0, 0, 0, 1,
```

Using this we can rebuild a index of where we start and stop seeing the characters in the first column.

```
first = {}
totc = 0
for c, count in sorted(totals.items()):
    first[c] = (totc, totc + count)
    totc += count
print(first)
```

```
{'$': (0, 1), 'B': (1, 2), 'C': (2, 3), 'I': (3, 4), 'M': (4, 5), 'O': (5, 6), 'S': (6
```

This simplified representation of the first column and BWT is the FM-index

2.3 Last to First Mapping

Using the we can map a given character with an index `i` back to first column. e.g `S` with second occurrence should map to row with initial index 4 witch is the 8th row.

```
print(first["S"][0] + tallymatrix["S"][2])
```

```
8
```

We can than reverse our transformation to build back our initial sequence. We remove one from the counts so we can use it as an index.

```
i = 0
t = "$"
while last_column[i] != "$":
    c = last_column[i]
    t = c + t
    i = first[c][0] + tallymatrix[c][i] - 1
print(t)
```

```
OMICSSBS$
```

To get a better grasp of whats happening lets track the index jumps too.

```
i = 0
t = "$"
j = 0
print(f"loop\tloopindex\tindex\tinitial_ind\tfirst_column\tlast_column\tc\tt\n")
while last_column[i] != "$":
    c = last_column[i]
    t = c + t
    i = first[c][0] + tallymatrix[c][i] - 1
    j += 1
    print(f"END{j:13} {i:8} {initial_indices[j]:8}{' '*14}{[seq[0] for seq in sorted_se
```

```
loop loopindex index initial_ind first_column last_column c t

END          1          6          6                B               S           S
END          2          1          3                C               I           B
END          3          7          2                I               M           S
END          4          8          1                M               O           S
END          5          2          0                O               $           C
END          6          3          7                S               B           I
END          7          4          5                S               S           M
END          8          5          4                S               C           O
```

Here we are building up the initial sequence backwards. Observe below the "index", "last_column" and "c" columns. We initiate the `index=0` which points us to character `S` in the last column, this `S` in turn with its `index=6` points to preceding character `B`.

2.4 Building a class

Keeping track of the intermediately data and passing them through functions can become confusing to keep all this in one place lets put them in a `class`. Classes are a way of bundling data and functionality in python.

Our class aptly named `BWA` takes a reference sequence and creates the BWT and FM-index when initialized.

```
from operator import itemgetter


class BWA:
    def __init__(self, ref):
        """TODO: to be defined1."""
        self.ref = ref + "$"

    @property
    def last(self):
        '''
        Get all the rotations of given sequence, sort them alphabetically and
        return the last character from each rotation
        '''
        sa = [self.ref[i:] + self.ref[:i] for i in range(len(self.ref))]
        return "".join(i[-1] for i in sorted(sa))

    @property
    def occ(self):
        '''
        Get the occurrences and total counts for each character in bwt
        '''
        totals = {}
        tallymatrix = {k: [] for k in "".join(set(self.last))}
        for c in self.last:
            if c not in totals:
                totals[c] = 0
            totals[c] += 1
            for k in tallymatrix.keys():
                if k != c and tallymatrix[k]:
                    tallymatrix[k].append(tallymatrix[k][-1])
                elif k == c:
                    tallymatrix[k].append(totals[c])
```

```
                else:
                    tallymatrix[k].append(0)
        return tallymatrix, totals

    @property
    def first(self):
        '''
        Get the lexicographical order of the characters
        '''
        first = {}
        totc = 0
        for c, count in sorted(self.occ[1].items()):
            first[c] = totc
            totc += count
        return first

    def lf(self, c, i):
        """
        The i-th occurrence of character ‘c’ in last is the same text character
        as the i-th occurrence of ‘c’ in the first
        """
        return self.first[c] + self.occ[0][c][i] - 1

    @property
    def rebwt(self):
        '''
        rebuild the original sequence
        '''
        i = 0
        t = "$"
        while self.last[i] != "$":
            c = self.last[i]
            t = c + t
            i = self.lf(c, i)
        return t[:-1]
```

We can initialize our function with our reference and access the BWT and FM-index as its properties.

```
example_seq = "OMICSSBS"
```

```
bwa = BWA(example_seq)
print(f"{bwa.last=}")
print(f"{bwa.first=}")
print(f"{bwa.occ=}")
print(f"{bwa.rebwt=}")
```

```
bwa.last=('SSIMO$BSC', (8, 6, 3, 2, 1, 0, 7, 5, 4), ('$OMICSSBS', 'BS$OMICSS', 'CSSBS$O
bwa.first={'$': (0, 0), 'B': (1, 1), 'C': (2, 2), 'I': (3, 3), 'M': (4, 4), 'O': (5, 5)
bwa.occ={'O': [0, 0, 0, 0, 1, 1, 1, 1, 1], 'S': [1, 2, 2, 2, 2, 2, 2, 3, 3], 'M': [0,
bwa.rebwt='OMICSSBS'
```

2.5 Mapping

2.5.1 Exact Matching

We want to align the small sequences back to references. If we have a `read` sequence with no variations. We can do a backward search to match a substring. We start by getting the minimum and maximum indices of the last character. Than we map these using our `lf` function in a loop.

```
low, high = self.last[0].find(read[-1]), self.last[0].rfind(read[-1])

i = len(read) - 1
while low <= high and i >= 0:
    low = self.lf(read[i], low)
    high = self.lf(read[i], high)
    i -= 1
return low, high
```

Let's add this to our class too.

```
from operator import itemgetter


class BWA:
    .
    .
    .

    def exact_match(self, read):
        low, high = self.last[0].find(read[-1]), self.last[0].rfind(read[-1])
```

```
        i = len(read) - 1
        while low <= high and i >= 0:
            low = self.lf(read[i], low)
            high = self.lf(read[i], high)
            i -= 1
        return low, high

read = "OMI"
bwa = BWA(example_seq)
low, high = bwa.exact_match(read)
initial_ind = bwa.last[1][low]
print(low, high)
print(initial_ind)
print(example_seq[initial_ind:initial_ind + len(read)])
```

```
5 5
0
OMI
```

2.5.2 Inexact Matching

Most crucial part is of course matching the reads with variations. Backtracking algorithm described by (Li and Durbin 2009) accounts for insertions, deletions and mismatches.

Precalculation:

Calculate BWT string B for reference string X
Calculate array $C(\cdot)$ and $O(\cdot,\cdot)$ from B
Calculate BWT string B' for the reverse reference
Calculate array $O'(\cdot,\cdot)$ from B'

Procedures:

```
    InexactSearch(W, z)
        CalculateD(W)
        return InexRecur(W, |W|−1, z, 1, |X|−1)

    CalculateD(W)
        k ← 1
        l ← |X|−1
        z ← 0
        for i = 0 to |W|−1 do
            k ← C(W[i]) + O'(W[i], k−1) + 1
            l ← C(W[i]) + O'(W[i], l)
            if k > l then
                k ← 1
                l ← |X|−1
                z ← z+1
            D(i) ← z

    InexRecur(W, i, z, k, l)
        if z < D(i) then
            return ∅
        if i < 0 then
            return {[k, l]}
        I ← ∅
*       I ← I ∪ InexRecur(W, i−1, z−1, k, l)
        for each b ∈ {A, C, G, T} do
            k ← C(b) + O(b, k−1) + 1
            l ← C(b) + O(b, l)
            if k ≤ l then
*               I ← I ∪ InexRecur(W, i, z−1, k, l)
                if b = W[i] then
                    I ← I ∪ InexRecur(W, i−1, z, k, l)
                else
                    I ← I ∪ InexRecur(W, i−1, z−1, k, l)
```

Inexact search calls the function `calculate_diff` initially which calculates the lower bound of differences for each character in the read string. Then we call the `inexact_match` which in turns calls itself recursively. This function has two stop conditions either we reach the end of the string or we run out of allowed differences.

```
class BWA:
    .
    .
    .
    def inexact_search(self, read, z):
        self.calculate_diff(read)
        return self.inexact_match(read, len(read) - 1, z, 1, len(self.last[0]) - 1)


    def calculate_diff(self, read):
        '''
        Estimate the lower bound of allowed z for every character.
        '''
        low = 1
        high = len(self.last[0]) - 1
        diffs = []
        z = 0
        for c in read:
            low = self.lf(c, low - 1) + 1
            high = self.lf(c, high)
            if low > high:
                low = 1
                high = len(self.last[0]) - 1
                z +=  1
            diffs.append(z)
        self.diffs = diffs


    def inexact_match(self, read, i, z, low, high):
        '''
        This function has two stop condition. Either we run out of the allowed
        differences or we find a match.
        '''
        if z < self.diffs[i]:
```

```
            return []
        if i < 0:
            return [[low,high]]

        matches = []
        matches.extend(self.inexact_match(read, i - 1, z - 1, low, high)) # Insertion
        for c in set(self.last[0]):
            low_ = self.lf(c, low - 1) + 1
            high_ = self.lf(c, high)
            if low <= high:
                matches.extend(self.inexact_match(read, i, z - 1, low_, high_)) # Delet
                if c == read[i]:
                    matches.extend(self.inexact_match(read, i - 1, z, low_, high_)) # M
                else:
                    matches.extend(self.inexact_match(read, i - 1, z - 1, low_, high_))
        return matches


example_seq="OMICSSBS"
bwa = BWA(example_seq)
read = "OMC"

matches = bwa.inexact_search(read, 2)
print( matches)
for match in matches:
    low, high = match
    initial_ind = bwa.last[1][low]
    print(example_seq[initial_ind:initial_ind + len(read)])
```

```
Hİ 6 [[5, 5], [5, 4], [4, 3], [5, 5], [5, 4], [5, 5]]
OMI
OMI
MIC
OMI
OMI
OMI
```

Visualizing the recursion

We can use some help to better visualize the recursion. Lets yank this snippet: `https://github.com/arpitbbhayani/recviz/blob/master/src/recviz/rec.py` and use a decorator.

```
class BWA:
    .
    .
    .

    @recviz
    def inexact_match(self, read, i, z, low, high):
        .
        .
        .


example_seq="OMICSSBS"
bwa = BWA(example_seq)
read = "OMC"

matches = bwa.inexact_search(read, 2)
print( matches)
for match in matches:
    low, high = match
    initial_ind = bwa.last[1][low]
    print(example_seq[initial_ind:initial_ind + len(read)])

 -> inexact_match('OMC', 2, 2, 1, 8)
    -> inexact_match('OMC', 1, 1, 1, 8, 'ins')
       -> inexact_match('OMC', 0, 0, 1, 8, 'ins')
          -> inexact_match('OMC', -1, -1, 1, 8, 'ins')
          <- []
          -> inexact_match('OMC', 0, -1, 3, 3, 'del')
          <- []
          -> inexact_match('OMC', -1, -1, 3, 3, 'missmatch')
          <- []
          -> inexact_match('OMC', 0, -1, 0, 0, 'del')
          <- []
          -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
          <- []
          -> inexact_match('OMC', 0, -1, 7, 8, 'del')
          <- []
          -> inexact_match('OMC', -1, -1, 7, 8, 'missmatch')
          <- []
```

```
   -> inexact_match('OMC', 0, -1, 4, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 5, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 2, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 1, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 3, 3, 'del')
<- []
-> inexact_match('OMC', 0, 0, 3, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 3, 3, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
```

```
   -> inexact_match('OMC', -1, 0, 5, 4, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, 0, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, 0, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, 0, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 2, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 9, 6, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 9, 6, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 4, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 3, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 3, 1, 'missmatch')
   <- []
```

```
   -> inexact_match('OMC', 0, -1, 2, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 7, 8, 'del')
<- []
-> inexact_match('OMC', 0, 0, 7, 8, 'missmatch')
   -> inexact_match('OMC', -1, -1, 7, 8, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 8, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 8, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 2, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 4, 'del')
```

```
<- []
-> inexact_match('OMC', 0, 1, 4, 4, 'match')
   -> inexact_match('OMC', -1, 0, 4, 4, 'ins')
   <- []
   -> inexact_match('OMC', 0, 0, 4, 3, 'del')
      -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 0, -1, 'del')
      -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 8, 7, 'del')
      -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 5, 4, 'del')
      -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 5, 5, 'del')
      -> inexact_match('OMC', -1, -1, 5, 5, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 0, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 7, 'del')
```

```
      <- []
      -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 6, 5, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 6, 5, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
   <- []
   -> inexact_match('OMC', -1, 1, 5, 5, 'match')
   <- [[5, 5, 'match']]
   -> inexact_match('OMC', 0, 0, 2, 1, 'del')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 1, 0, 'del')
      -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 1, 0, 'missmatch')
   <- []
<- [[5, 5, 'match']]
-> inexact_match('OMC', 1, 0, 5, 5, 'del')
<- []
-> inexact_match('OMC', 0, 0, 5, 5, 'missmatch')
   -> inexact_match('OMC', -1, -1, 5, 5, 'ins')
   <- []
```

```
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 2, 2, 'del')
<- []
-> inexact_match('OMC', 0, 0, 2, 2, 'missmatch')
   -> inexact_match('OMC', -1, -1, 2, 2, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 3, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 3, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, -1, 'del')
   <- []
```

```
   -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 5, 4, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 1, 1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 1, 1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 1, 1, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 3, 2, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 3, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 7, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 7, 7, 'missmatch')
   <- []
```

```
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 5, 4, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
   <- []
<- [[5, 5, 'match']]
-> inexact_match('OMC', 2, 1, 3, 3, 'del')
   -> inexact_match('OMC', 1, 0, 3, 3, 'ins')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 4, 'missmatch')
```

```
   <- []
   -> inexact_match('OMC', 2, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 1, 2, 1, 'match')
      -> inexact_match('OMC', 0, 0, 2, 1, 'ins')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
   <- []
   -> inexact_match('OMC', 2, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 3, 3, 'missmatch')
   -> inexact_match('OMC', 0, 0, 3, 3, 'ins')
      -> inexact_match('OMC', -1, -1, 3, 3, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 0, -1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 7, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 4, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 5, 4, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
```

```
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 3, 'del')
<- []
-> inexact_match('OMC', 0, 0, 4, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, -1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 8, 7, 'del')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'missmatch')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 4, 'del')
<- []
-> inexact_match('OMC', 0, 1, 4, 4, 'match')
   -> inexact_match('OMC', -1, 0, 4, 4, 'ins')
   <- []
   -> inexact_match('OMC', 0, 0, 4, 3, 'del')
      -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 0, -1, 'del')
      -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
      <- []
   <- []
```

```
-> inexact_match('OMC', -1, 0, 0, -1, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'del')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 8, 7, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 5, 4, 'del')
   -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 5, 4, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 5, 5, 'del')
   -> inexact_match('OMC', -1, -1, 5, 5, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
```

```
            <- []
            -> inexact_match('OMC', 0, -1, 1, 0, 'del')
            <- []
            -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
            <- []
         <- []
         -> inexact_match('OMC', -1, 1, 5, 5, 'match')
         <- [[5, 5, 'match']]
         -> inexact_match('OMC', 0, 0, 2, 1, 'del')
            -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
            <- []
         <- []
         -> inexact_match('OMC', -1, 0, 2, 1, 'missmatch')
         <- []
         -> inexact_match('OMC', 0, 0, 1, 0, 'del')
            -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
            <- []
         <- []
         -> inexact_match('OMC', -1, 0, 1, 0, 'missmatch')
         <- []
      <- [[5, 5, 'match']]
      -> inexact_match('OMC', 1, 0, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', 0, 0, 5, 4, 'missmatch')
         -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
         <- []
      <- []
      -> inexact_match('OMC', 1, 0, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
      -> inexact_match('OMC', 1, 0, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
         -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
         <- []
      <- []
<- [[5, 5, 'match']]
```

```
-> inexact_match('OMC', 2, 1, 0, 0, 'del')
   -> inexact_match('OMC', 1, 0, 0, 0, 'ins')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 2, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 1, -1, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 1, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 9, 6, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 9, 6, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 6, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 6, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 3, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 1, 3, 1, 'match')
      -> inexact_match('OMC', 0, 0, 3, 1, 'ins')
         -> inexact_match('OMC', -1, -1, 3, 1, 'ins')
         <- []
      <- []
   <- []
   -> inexact_match('OMC', 2, 0, 2, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 2, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 0, 0, 'missmatch')
   -> inexact_match('OMC', 0, 0, 0, 0, 'ins')
      -> inexact_match('OMC', -1, -1, 0, 0, 'ins')
      <- []
```

```
   -> inexact_match('OMC', 0, -1, 4, 2, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 9, 6, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 9, 6, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 4, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 3, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 3, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 2, 'del')
<- []
-> inexact_match('OMC', 0, 0, 4, 2, 'missmatch')
   -> inexact_match('OMC', -1, -1, 4, 2, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 1, -1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 1, -1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 1, -1, 'ins')
   <- []
```

```
   <- []
   -> inexact_match('OMC', 1, 0, 9, 6, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 9, 6, 'missmatch')
      -> inexact_match('OMC', -1, -1, 9, 6, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 5, 3, 'del')
   <- []
   -> inexact_match('OMC', 0, 1, 5, 3, 'match')
      -> inexact_match('OMC', -1, 0, 5, 3, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 6, 4, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 6, 4, 'missmatch')
      -> inexact_match('OMC', -1, -1, 6, 4, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 3, 1, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 3, 1, 'missmatch')
      -> inexact_match('OMC', -1, -1, 3, 1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 2, 0, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 2, 0, 'missmatch')
      -> inexact_match('OMC', -1, -1, 2, 0, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 2, 1, 7, 8, 'del')
   -> inexact_match('OMC', 1, 0, 7, 8, 'ins')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 1, 0, 'del')
```

```
<- []
-> inexact_match('OMC', 1, 0, 1, 0, 'missmatch')
<- []
-> inexact_match('OMC', 2, 0, 8, 8, 'del')
<- []
-> inexact_match('OMC', 1, 0, 8, 8, 'missmatch')
<- []
-> inexact_match('OMC', 2, 0, 5, 4, 'del')
<- []
-> inexact_match('OMC', 1, 0, 5, 4, 'missmatch')
<- []
-> inexact_match('OMC', 2, 0, 6, 5, 'del')
<- []
-> inexact_match('OMC', 1, 0, 6, 5, 'missmatch')
<- []
-> inexact_match('OMC', 2, 0, 2, 2, 'del')
<- []
-> inexact_match('OMC', 1, 1, 2, 2, 'match')
   -> inexact_match('OMC', 0, 0, 2, 2, 'ins')
      -> inexact_match('OMC', -1, -1, 2, 2, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 3, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 3, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 0, -1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 7, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 5, 4, 'match')
```

```
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 3, 3, 'del')
<- []
-> inexact_match('OMC', 0, 0, 3, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 3, 3, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 5, 4, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
```

```
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, -1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 8, 7, 'del')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'missmatch')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 3, 'del')
<- []
-> inexact_match('OMC', 0, 1, 4, 3, 'match')
   -> inexact_match('OMC', -1, 0, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 5, 4, 'del')
<- []
-> inexact_match('OMC', 0, 0, 5, 4, 'missmatch')
   -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 2, 1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 1, 0, 'del')
<- []
-> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
   -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
   <- []
<- []
```

```
   <- []
   -> inexact_match('OMC', 2, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 2, 1, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 7, 8, 'missmatch')
   -> inexact_match('OMC', 0, 0, 7, 8, 'ins')
      -> inexact_match('OMC', -1, -1, 7, 8, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 8, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 8, 8, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 6, 5, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 6, 5, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 2, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 2, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'del')
```

```
<- []
-> inexact_match('OMC', 0, 0, 4, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 1, 0, 'del')
<- []
-> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
   -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 8, 8, 'del')
<- []
-> inexact_match('OMC', 0, 0, 8, 8, 'missmatch')
   -> inexact_match('OMC', -1, -1, 8, 8, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 9, 8, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 9, 8, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 2, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 2, 'missmatch')
   <- []
```

```
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 5, 4, 'del')
<- []
-> inexact_match('OMC', 0, 1, 5, 4, 'match')
   -> inexact_match('OMC', -1, 0, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 6, 5, 'del')
<- []
-> inexact_match('OMC', 0, 0, 6, 5, 'missmatch')
   -> inexact_match('OMC', -1, -1, 6, 5, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 2, 2, 'del')
<- []
-> inexact_match('OMC', 0, 0, 2, 2, 'missmatch')
   -> inexact_match('OMC', -1, -1, 2, 2, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 3, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 3, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
```

```
      -> inexact_match('OMC', -1, 0, 5, 4, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 2, 1, 4, 4, 'del')
   -> inexact_match('OMC', 1, 0, 4, 4, 'ins')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 5, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 5, 'missmatch')
```

```
   <- []
   -> inexact_match('OMC', 2, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 1, 2, 1, 'match')
      -> inexact_match('OMC', 0, 0, 2, 1, 'ins')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
   <- []
   -> inexact_match('OMC', 2, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 4, 4, 'missmatch')
   -> inexact_match('OMC', 0, 0, 4, 4, 'ins')
      -> inexact_match('OMC', -1, -1, 4, 4, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 0, -1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 7, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 5, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 5, 5, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
```

```
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 3, 'del')
<- []
-> inexact_match('OMC', 0, 0, 4, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, -1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 8, 7, 'del')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'missmatch')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 5, 4, 'del')
<- []
-> inexact_match('OMC', 0, 1, 5, 4, 'match')
   -> inexact_match('OMC', -1, 0, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 5, 5, 'del')
<- []
-> inexact_match('OMC', 0, 0, 5, 5, 'missmatch')
   -> inexact_match('OMC', -1, -1, 5, 5, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
```

```
      -> inexact_match('OMC', 0, -1, 0, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 7, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 6, 5, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 6, 5, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
      -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 2, 1, 5, 5, 'del')
   -> inexact_match('OMC', 1, 0, 5, 5, 'ins')
```

```
   <- []
   -> inexact_match('OMC', 2, 0, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 0, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 0, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 6, 5, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 1, 2, 1, 'match')
      -> inexact_match('OMC', 0, 0, 2, 1, 'ins')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
   <- []
   -> inexact_match('OMC', 2, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 5, 5, 'missmatch')
   -> inexact_match('OMC', 0, 0, 5, 5, 'ins')
      -> inexact_match('OMC', -1, -1, 5, 5, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
```

```
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 3, 'del')
<- []
-> inexact_match('OMC', 0, 0, 4, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, 0, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, 0, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, 0, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 2, 'del')
   <- []
```

```
   -> inexact_match('OMC', -1, -1, 4, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 9, 6, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 9, 6, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 4, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 3, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 3, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 8, 7, 'del')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'missmatch')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 5, 4, 'del')
<- []
-> inexact_match('OMC', 0, 1, 5, 4, 'match')
   -> inexact_match('OMC', -1, 0, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 6, 5, 'del')
```

```
   <- []
   -> inexact_match('OMC', 0, 0, 6, 5, 'missmatch')
      -> inexact_match('OMC', -1, -1, 6, 5, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
      -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 2, 1, 2, 2, 'del')
   -> inexact_match('OMC', 1, 0, 2, 2, 'ins')
   <- []
   -> inexact_match('OMC', 2, 0, 3, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 3, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 4, 'missmatch')
```

```
   <- []
   -> inexact_match('OMC', 2, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 1, 2, 1, 'match')
      -> inexact_match('OMC', 0, 0, 2, 1, 'ins')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
   <- []
   -> inexact_match('OMC', 2, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 2, 2, 2, 'match')
   -> inexact_match('OMC', 0, 1, 2, 2, 'ins')
      -> inexact_match('OMC', -1, 0, 2, 2, 'ins')
      <- []
      -> inexact_match('OMC', 0, 0, 3, 3, 'del')
         -> inexact_match('OMC', -1, -1, 3, 3, 'ins')
         <- []
         -> inexact_match('OMC', 0, -1, 4, 3, 'del')
         <- []
         -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
         <- []
         -> inexact_match('OMC', 0, -1, 0, -1, 'del')
         <- []
         -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
         <- []
         -> inexact_match('OMC', 0, -1, 8, 7, 'del')
         <- []
         -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
         <- []
         -> inexact_match('OMC', 0, -1, 4, 4, 'del')
         <- []
         -> inexact_match('OMC', -1, -1, 4, 4, 'missmatch')
         <- []
         -> inexact_match('OMC', 0, -1, 5, 4, 'del')
         <- []
         -> inexact_match('OMC', -1, 0, 5, 4, 'match')
```

```
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 3, 3, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'del')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 0, -1, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'del')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 8, 7, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 4, 3, 'del')
   -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 4, 3, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 5, 4, 'del')
   -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 1, 5, 4, 'match')
<- [[5, 4, 'match']]
-> inexact_match('OMC', 0, 0, 2, 1, 'del')
   -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
   <- []
<- []
```

```
   -> inexact_match('OMC', -1, 0, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 1, 0, 'del')
      -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 1, 0, 'missmatch')
   <- []
<- [[5, 4, 'match']]
-> inexact_match('OMC', 1, 1, 3, 3, 'del')
   -> inexact_match('OMC', 0, 0, 3, 3, 'ins')
      -> inexact_match('OMC', -1, -1, 3, 3, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 0, -1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 8, 7, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 4, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 5, 4, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
```

```
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 3, 'del')
<- []
-> inexact_match('OMC', 0, 0, 4, 3, 'missmatch')
   -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, -1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 8, 7, 'del')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'missmatch')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 4, 4, 'del')
<- []
-> inexact_match('OMC', 0, 1, 4, 4, 'match')
   -> inexact_match('OMC', -1, 0, 4, 4, 'ins')
   <- []
   -> inexact_match('OMC', 0, 0, 4, 3, 'del')
      -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 0, -1, 'del')
      -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 8, 7, 'del')
      -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
      <- []
```

```
<- []
-> inexact_match('OMC', -1, 0, 8, 7, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 5, 4, 'del')
   -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 5, 4, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 5, 5, 'del')
   -> inexact_match('OMC', -1, -1, 5, 5, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
```

```
      <- []
      -> inexact_match('OMC', -1, 1, 5, 5, 'match')
      <- [[5, 5, 'match']]
      -> inexact_match('OMC', 0, 0, 2, 1, 'del')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
      -> inexact_match('OMC', -1, 0, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, 0, 1, 0, 'del')
         -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
         <- []
      <- []
      -> inexact_match('OMC', -1, 0, 1, 0, 'missmatch')
      <- []
   <- [[5, 5, 'match']]
   -> inexact_match('OMC', 1, 0, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 5, 4, 'missmatch')
      -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
      -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
      <- []
   <- []
<- [[5, 5, 'match']]
-> inexact_match('OMC', 0, 1, 3, 3, 'missmatch')
   -> inexact_match('OMC', -1, 0, 3, 3, 'ins')
   <- []
   -> inexact_match('OMC', 0, 0, 4, 3, 'del')
      -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
```

```
   <- []
<- []
-> inexact_match('OMC', -1, 0, 4, 3, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'del')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 0, -1, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 8, 7, 'del')
   -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', -1, 0, 8, 7, 'missmatch')
<- []
-> inexact_match('OMC', 0, 0, 4, 4, 'del')
   -> inexact_match('OMC', -1, -1, 4, 4, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 7, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 5, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
```

```
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 4, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 5, 4, 'del')
      -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 1, 5, 4, 'match')
   <- [[5, 4, 'match']]
   -> inexact_match('OMC', 0, 0, 2, 1, 'del')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 2, 1, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, 0, 1, 0, 'del')
      -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
      <- []
   <- []
   -> inexact_match('OMC', -1, 0, 1, 0, 'missmatch')
   <- []
<- [[5, 4, 'match']]
-> inexact_match('OMC', 1, 1, 0, -1, 'del')
   -> inexact_match('OMC', 0, 0, 0, -1, 'ins')
      -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 0, 1, 0, -1, 'missmatch')
   -> inexact_match('OMC', -1, 0, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 8, 7, 'del')
```

```
   -> inexact_match('OMC', 0, 0, 8, 7, 'ins')
      -> inexact_match('OMC', -1, -1, 8, 7, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 0, 1, 8, 7, 'missmatch')
   -> inexact_match('OMC', -1, 0, 8, 7, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 4, 3, 'del')
   -> inexact_match('OMC', 0, 0, 4, 3, 'ins')
      -> inexact_match('OMC', -1, -1, 4, 3, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 0, 2, 4, 3, 'match')
   -> inexact_match('OMC', -1, 1, 4, 3, 'ins')
   <- [[4, 3, 'ins']]
<- [[4, 3, 'ins']]
-> inexact_match('OMC', 1, 1, 5, 4, 'del')
   -> inexact_match('OMC', 0, 0, 5, 4, 'ins')
      -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 0, 1, 5, 4, 'missmatch')
   -> inexact_match('OMC', -1, 0, 5, 4, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 2, 1, 'del')
   -> inexact_match('OMC', 0, 0, 2, 1, 'ins')
      -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
      <- []
   <- []
<- []
-> inexact_match('OMC', 0, 1, 2, 1, 'missmatch')
   -> inexact_match('OMC', -1, 0, 2, 1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 1, 0, 'del')
```

```
      -> inexact_match('OMC', 0, 0, 1, 0, 'ins')
         -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
         <- []
      <- []
   <- []
   -> inexact_match('OMC', 0, 1, 1, 0, 'missmatch')
      -> inexact_match('OMC', -1, 0, 1, 0, 'ins')
      <- []
   <- []
<- [[5, 4, 'match'], [5, 5, 'match'], [5, 4, 'match'], [4, 3, 'ins']]
-> inexact_match('OMC', 2, 1, 1, 1, 'del')
   -> inexact_match('OMC', 1, 0, 1, 1, 'ins')
   <- []
   -> inexact_match('OMC', 2, 0, 3, 2, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 3, 2, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 0, -1, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 0, -1, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 7, 7, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 7, 7, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 2, 0, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', 1, 1, 2, 1, 'match')
      -> inexact_match('OMC', 0, 0, 2, 1, 'ins')
         -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
         <- []
      <- []
```

```
   <- []
   -> inexact_match('OMC', 2, 0, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', 1, 0, 1, 0, 'missmatch')
   <- []
<- []
-> inexact_match('OMC', 1, 1, 1, 1, 'missmatch')
   -> inexact_match('OMC', 0, 0, 1, 1, 'ins')
      -> inexact_match('OMC', -1, -1, 1, 1, 'ins')
      <- []
      -> inexact_match('OMC', 0, -1, 3, 2, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 3, 2, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 0, -1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 0, -1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 7, 7, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 7, 7, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 4, 3, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 5, 4, 'del')
      <- []
      -> inexact_match('OMC', -1, 0, 5, 4, 'match')
      <- []
      -> inexact_match('OMC', 0, -1, 2, 1, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
      <- []
      -> inexact_match('OMC', 0, -1, 1, 0, 'del')
      <- []
      -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
      <- []
   <- []
   -> inexact_match('OMC', 1, 0, 3, 2, 'del')
```

```
<- []
-> inexact_match('OMC', 0, 0, 3, 2, 'missmatch')
   -> inexact_match('OMC', -1, -1, 3, 2, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 0, -1, 'del')
<- []
-> inexact_match('OMC', 0, 0, 0, -1, 'missmatch')
   -> inexact_match('OMC', -1, -1, 0, -1, 'ins')
   <- []
<- []
-> inexact_match('OMC', 1, 0, 7, 7, 'del')
<- []
-> inexact_match('OMC', 0, 0, 7, 7, 'missmatch')
   -> inexact_match('OMC', -1, -1, 7, 7, 'ins')
   <- []
   -> inexact_match('OMC', 0, -1, 4, 3, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 4, 3, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 1, 0, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 1, 0, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 8, 8, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 8, 8, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 5, 4, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 5, 4, 'missmatch')
   <- []
   -> inexact_match('OMC', 0, -1, 6, 5, 'del')
   <- []
   -> inexact_match('OMC', -1, 0, 6, 5, 'match')
   <- []
   -> inexact_match('OMC', 0, -1, 2, 1, 'del')
   <- []
   -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
   <- []
```

```
          -> inexact_match('OMC', 0, -1, 2, 1, 'del')
          <- []
          -> inexact_match('OMC', -1, -1, 2, 1, 'missmatch')
          <- []
       <- []
       -> inexact_match('OMC', 1, 0, 4, 3, 'del')
       <- []
       -> inexact_match('OMC', 0, 1, 4, 3, 'match')
          -> inexact_match('OMC', -1, 0, 4, 3, 'ins')
          <- []
       <- []
       -> inexact_match('OMC', 1, 0, 5, 4, 'del')
       <- []
       -> inexact_match('OMC', 0, 0, 5, 4, 'missmatch')
          -> inexact_match('OMC', -1, -1, 5, 4, 'ins')
          <- []
       <- []
       -> inexact_match('OMC', 1, 0, 2, 1, 'del')
       <- []
       -> inexact_match('OMC', 0, 0, 2, 1, 'missmatch')
          -> inexact_match('OMC', -1, -1, 2, 1, 'ins')
          <- []
       <- []
       -> inexact_match('OMC', 1, 0, 1, 0, 'del')
       <- []
       -> inexact_match('OMC', 0, 0, 1, 0, 'missmatch')
          -> inexact_match('OMC', -1, -1, 1, 0, 'ins')
          <- []
       <- []
    <- []
 <- [[5, 5, 'match'], [5, 5, 'match'], [5, 4, 'match'], [5, 5, 'match'], [5, 4, 'match
[[5, 5, 'match'], [5, 5, 'match'], [5, 4, 'match'], [5, 5, 'match'], [5, 4, 'match'],
OMI match
OMI match
OMI match
OMI match
OMI match
MIC ins
```

2.6 Putting it all together

```
from operator import itemgetter


class BWA:
    def __init__(self, ref):
        """todo: to be defined1."""
        self.ref = ref + "$"

    @property
    def last(self):
        sa = [self.ref[i:] + self.ref[:i] for i in range(len(self.ref))]
        idx, sa = zip(*sorted(enumerate(sa), key=itemgetter(1)))
        return "".join(i[-1] for i in sa), idx, sa

    @property
    def occ(self):
        '''
        get the occurrences and total counts for each character in bwt
        '''
        totals = {}
        tallymatrix = {k: [] for k in "".join(set(self.last[0]))}
        for c in self.last[0]:
            if c not in totals:
                totals[c] = 0
            totals[c] += 1
            for k in tallymatrix.keys():
                if k != c and tallymatrix[k]:
                    tallymatrix[k].append(tallymatrix[k][-1])
                elif k == c:
                    tallymatrix[k].append(totals[c])
                else:
                    tallymatrix[k].append(0)
        return tallymatrix, totals

    @property
    def first(self):
        '''
        because first column is alphabetically sorted
```

```
        its enough to just to get start and end position for each character.
        '''
        first = {}
        totc = 0
        for c, count in sorted(self.occ[1].items()):
            first[c] = (totc, totc + count - 1)
            totc += count
        return first

    def lf(self, c, i):
        """
        the i-th occurrence of character ‘c’ in last
        column is the same text character as the i-th
        occurrence of ‘c’ in the first column
        """
        return self.first[c][0] + self.occ[0][c][i] - 1

    @property
    def rebwt(self):
        i = 0
        t = "$"
        while self.last[0][i] != "$":
            c = self.last[0][i]
            t = c + t
            i = self.lf(c, i)
        return t[:-1]

    def exact_match(self, read):
        low, high = self.last[0].find(read[-1]), self.last[0].rfind(read[-1])

        i = len(read) - 1
        while low <= high and i >= 0:
            low = self.lf(read[i], low - 1) + 1
            high = self.lf(read[i], high)
            i -= 1
        return low, high


    def inexact_search(self, read, z):
        self.calculate_diff(read)
```

```
    return self.inexact_match(read, len(read) - 1, z, 1, len(self.last[0]) - 1)


def calculate_diff(self, read):
    '''
    estimate the lower bound of allowed z for every character.
    '''
    low = 1
    high = len(self.last[0]) - 1
    diffs = []
    z = 0
    for c in read:
        low = self.lf(c, low - 1) + 1
        high = self.lf(c, high)
        if low > high:
            low = 1
            high = len(self.last[0]) - 1
            z +=  1
        diffs.append(z)
    self.diffs = diffs


def inexact_match(self, read, i, z, low, high):
    '''
    this function has two stop condition. either we run out of the allowed
    differences or we find a match.
    '''
    if z < self.diffs[i]:
        return []
    if i < 0:
        return [[low, high]]

    matches = []
    matches.extend(self.inexact_match(read, i - 1, z - 1, low, high)) # insertion
    for c in set(self.last[0]):
        low_ = self.lf(c, low - 1) + 1
        high_ = self.lf(c, high)
        if low <= high:
            matches.extend(self.inexact_match(read, i, z - 1, low_, high_)) # delet
            if c == read[i]:
```

```
                matches.extend(self.inexact_match(read, i - 1, z, low_, high_)) # r
            else:
                matches.extend(self.inexact_match(read, i - 1, z - 1, low_, high_))
    return matches
```

2.7 Trying out with dummy data

2.7.1 Sequence generation

Lets generate our reference genome first. We are going to use the same approach as we did in motif finding post. We are going to create our reference sequence with length `1200` with same probability of `.25` for all the bases. We add `$` character at the end to denote the end of the string which will be useful when we do the burrows wheeler below.

```
import random
random.seed(0)
alphabet = ['A', 'C', 'G', 'T']
seq_weights = [.25, .25, .25, .25]
seq_length = 120

ref_seq = "".join(
    random.choices(population=alphabet, weights=seq_weights, k=seq_length)
)
print(ref_seq)
```

```
TTCCGCTCCGTGCTGCTTTTCGTGCACGTTCTCTGAGCTGACTACTAGATTCACGTAGGGTGTGGCGCGCAAGGCATTTTTTGCGCT
```

2.7.2 Mutating the sequence

We want our reads to be different from our reference so, before sampling our samples we are going to create a mutated sequence first. Here with the `mutated` function we decide if mutation occurs with the given percentage and if it does type of the mutation. With the `mutate_read` function we either insert a single base, substitute a base or remove a base.

```
def mutated(mutation_rate):
    r = random.randint(1,100)
    if r <= mutation_rate:
```

```
        r = random.randint(1,100)
        if r <= 25:
            return 'del'
        elif 25 < r <= 50:
            return 'ins'
        else:
            return 'sub'
    else:
        return False


def mutate(sequence, mutation_rate):
    mutated_seq = ''
    mutated_once = False
    for base in sequence:
        mutatition_type = mutated(mutation_rate)
        match mutatition_type:
            case 'del':
                pass
            case 'ins':
                possible_mutations = "".join(['A', 'T', 'C', 'G'])
                mutantion = random.choice(possible_mutations)
                mutated_seq += base
                mutated_seq += mutantion
            case 'sub':
                possible_mutations = "".join(['A', 'T', 'C', 'G'])
                possible_mutations = possible_mutations.replace(base, '')
                mutantion = random.choice(possible_mutations)
                mutated_seq += mutantion
            case _:
                mutated_seq += base
    return mutated_seq
```

After mutation with rate .5 we got our new mutated string some indels and substitutions.

```
mt_seq = mutate(ref_seq, 5)
print(f"{ref_seq=}")
print(f" {mt_seq=}")
```

```
ref_seq='TTCCGCTCCGTGCTGCTTTTCGTGCACGTTCTCTGAGCTGACTACTAGATTCACGTAGGGTGTGGCGCGCAAGGCAT
```

```
mt_seq='TTCCGCTCCGTGCTGCTTTTCGTGCACGTTCTCTGAGCTGACTACTAATTCATGTAGGGTGTGGCGCGCAAGGCATTT
```

2.7.3 Getting the reads

Lets randomly sample 600 reads with length 6 from our mutated sequence.

```
number_of_reads = 600
read_length = 6
random_indices = [random.randrange(len(mt_seq) - read_length) for _ in range(number_of_
mt_reads = [mt_seq[i:i + read_length] for i in random_indices]
print(mt_reads[:20])
```

```
['AGAGTG', 'TCATGT', 'GCGTAG', 'GTAGGG', 'CTCCGT', 'TGAGCT', 'GCGCAA', 'GGCGCG', 'ATGTA
```

We can now try to align our reads from the `mutated_sequence` back to the our reference. Since BWA returns many matches we are just gonna get the most common one using the snippet here.

```
# Code from: https://stackoverflow.com/a/1520716/7141527
# Get the most common match
import itertools
import operator

def most_common(L):
  # get an iterable of (item, iterable) pairs
  SL = sorted((x, i) for i, x in enumerate(L))
  # print 'SL:', SL
  groups = itertools.groupby(SL, key=operator.itemgetter(0))
  # auxiliary function to get "quality" for an item
  def _auxfun(g):
    item, iterable = g
    count = 0
    min_index = len(L)
    for _, where in iterable:
      count += 1
      min_index = min(min_index, where)
    # print 'item %r, count %r, minind %r' % (item, count, min_index)
    return count, -min_index
  # pick the highest-count/earliest item
  return max(groups, key=_auxfun)[0]
```

2.7.4 Performing the alignment

First lets initialize our BWA class with the `ref_seq`. Then we are going to align our mutated reads and keep track of the most common matches. Then we are going the get the `initial_ind` from `bwa.last[1]` using the `low` value.

```
bwa = BWA(ref_seq)
alignment_indices = []
for mt_read in mt_reads:
    matches=bwa.inexact_search(mt_read, 2)
    if matches:
        # match = most_common(matches)
        match = matches[0]
        alignment_indices.append(bwa.last[1][match[0]])
    else:
        alignment_indices.append(0)

print(f"{   random_indices[:20]=}")
print(f"{alignment_indices[:20]=}")
```

```
   random_indices[20:]=[96, 12, 79, 102, 32, 68, 90, 103, 77, 18, 39, 12, 93, 9, 108, 8
alignment_indices[20:]=[57, 12, 80, 39, 32, 69, 98, 109, 78, 18, 39, 12, 79, 9, 110, 88
```

2.7.5 Outputing to SAM

Lets output our alignmets as Sequence Alignment/Map (SAM) format. SAM has header lines starting with '@' and 11 fields denoting various alignment properties. We are going to fill in just the QNAME, POS, TLEN and SEQ fields.

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,254}	Query template NAME
2	FLAG	Int	[0, 2^{16} 1]	bitwise FLAG
3	RNAME	String	*|[:rname:]*=][:rname:]*	Reference sequence NAME11
4	POS	Int	[0, 2^{31} 1]	1-based leftmost mapping POSition
5	MAPQ	Int	[0, 2^{8} 1]	MAPping Quality
6	CIGAR	String	*|([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	*|=|[:rname:]*=][:rname:]*	Reference name of the mate/next read
8	PNEXT	Int	[0, 2^{31} 1]	Position of the mate/next read
9	TLEN	Int	[$2^{31}+1, 2^{31}$1]	observed Template LENgth
10	SEQ	String	*|[A-Za-z=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

For header we are going to print a version number, reference sequence name and length. For our alignments we are going to create a name for them and sort them and print their alignment index with their sequence. Ideally, reads that haven't aligned should be flagged with 0x4. However we are just gonna skip them. Another thing we are skipping over is the CIGAR entries which says where are the indels and mismatches we are just gonna write everything is matched.

```
sam_contents = ""
sam_header = f"@HD\tVN:1.6\tSO:coordinate\n@SQ\tSN:REF\tLN:{len(ref_seq)}\n"
sam_contents += sam_header

mapped_reads = {i: read for i, read in sorted(zip(alignment_indices, mt_reads))}
for i, read in mapped_reads.items():
    #                QNAME  FLAG RNAME POS MAPQ CIGAR RNEXT PNEXT TLEN SEQ     QUAL
    if i == 0:
        continue
    else:
        sam_contents += f"read{i}\t0\tREF\t{i+1}\t30\t6M\t*\t0\t{len(read)}\t{read}\t{

print(sam_contents)
```

```
@HD VN:1.6 SO:coordinate
@SQ SN:REF LN:120
read1 0 REF 2 30 6M * 0 6 TCCGCT KKKKKK
read2 0 REF 3 30 6M * 0 6 CCGCTC KKKKKK
read3 0 REF 4 30 6M * 0 6 CGCTCC KKKKKK
```

```
read4 0 REF 5 30 6M * 0 6 GCTCCG KKKKKK
read5 0 REF 6 30 6M * 0 6 CTCCGT KKKKKK
read6 0 REF 7 30 6M * 0 6 TCCGTG KKKKKK
read7 0 REF 8 30 6M * 0 6 CCGTGC KKKKKK
read9 0 REF 10 30 6M * 0 6 GTGCTG KKKKKK
read10 0 REF 11 30 6M * 0 6 TGCTTT KKKKKK
read11 0 REF 12 30 6M * 0 6 GCTGCT KKKKKK
read12 0 REF 13 30 6M * 0 6 CTGCTT KKKKKK
read16 0 REF 17 30 6M * 0 6 TTTTTT KKKKKK
read17 0 REF 18 30 6M * 0 6 TTTCGT KKKKKK
read18 0 REF 19 30 6M * 0 6 TTCGTG KKKKKK
read19 0 REF 20 30 6M * 0 6 TCGTGC KKKKKK
read20 0 REF 21 30 6M * 0 6 CGTGCT KKKKKK
read21 0 REF 22 30 6M * 0 6 GTGCAC KKKKKK
read22 0 REF 23 30 6M * 0 6 TGCACG KKKKKK
read23 0 REF 24 30 6M * 0 6 GCACGT KKKKKK
read25 0 REF 26 30 6M * 0 6 ACGTTC KKKKKK
read26 0 REF 27 30 6M * 0 6 CGTTCT KKKKKK
read27 0 REF 28 30 6M * 0 6 GTTCTC KKKKKK
read28 0 REF 29 30 6M * 0 6 TTCTCT KKKKKK
read30 0 REF 31 30 6M * 0 6 CTCTGA KKKKKK
read31 0 REF 32 30 6M * 0 6 TCTGAG KKKKKK
read32 0 REF 33 30 6M * 0 6 CTGAGC KKKKKK
read33 0 REF 34 30 6M * 0 6 TGAGCT KKKKKK
read34 0 REF 35 30 6M * 0 6 GAGCTG KKKKKK
read35 0 REF 36 30 6M * 0 6 AGCTGA KKKKKK
read36 0 REF 37 30 6M * 0 6 GCTGAC KKKKKK
read37 0 REF 38 30 6M * 0 6 CTGACT KKKKKK
read38 0 REF 39 30 6M * 0 6 TGACTA KKKKKK
read39 0 REF 40 30 6M * 0 6 GACTAC KKKKKK
read40 0 REF 41 30 6M * 0 6 ACTACT KKKKKK
read41 0 REF 42 30 6M * 0 6 CTACTA KKKKKK
read42 0 REF 43 30 6M * 0 6 TACTAA KKKKKK
read44 0 REF 45 30 6M * 0 6 CTAATT KKKKKK
read48 0 REF 49 30 6M * 0 6 ATTCAT KKKKKK
read51 0 REF 52 30 6M * 0 6 CATGTA KKKKKK
read52 0 REF 53 30 6M * 0 6 AATTCA KKKKKK
read54 0 REF 55 30 6M * 0 6 GTAGGG KKKKKK
read55 0 REF 56 30 6M * 0 6 TAGGGT KKKKKK
read56 0 REF 57 30 6M * 0 6 AGGGTG KKKKKK
```

```
read57 0 REF 58 30 6M * 0 6 GGGTGT KKKKKK
read58 0 REF 59 30 6M * 0 6 GGTGTG KKKKKK
read59 0 REF 60 30 6M * 0 6 GTGTGG KKKKKK
read60 0 REF 61 30 6M * 0 6 TGTGGC KKKKKK
read61 0 REF 62 30 6M * 0 6 GTGGCG KKKKKK
read62 0 REF 63 30 6M * 0 6 TGGCGT KKKKKK
read63 0 REF 64 30 6M * 0 6 GGCGCG KKKKKK
read64 0 REF 65 30 6M * 0 6 GCGCGC KKKKKK
read65 0 REF 66 30 6M * 0 6 CGCGCA KKKKKK
read66 0 REF 67 30 6M * 0 6 GCGCAA KKKKKK
read67 0 REF 68 30 6M * 0 6 CGCAAG KKKKKK
read68 0 REF 69 30 6M * 0 6 GCAAGG KKKKKK
read69 0 REF 70 30 6M * 0 6 CAAGGC KKKKKK
read70 0 REF 71 30 6M * 0 6 AAGGCA KKKKKK
read71 0 REF 72 30 6M * 0 6 AGGCAT KKKKKK
read72 0 REF 73 30 6M * 0 6 GGCATT KKKKKK
read73 0 REF 74 30 6M * 0 6 GCATTT KKKKKK
read74 0 REF 75 30 6M * 0 6 CATTTT KKKKKK
read75 0 REF 76 30 6M * 0 6 ATTTTT KKKKKK
read77 0 REF 78 30 6M * 0 6 TTTTTG KKKKKK
read78 0 REF 79 30 6M * 0 6 TTTTGC KKKKKK
read79 0 REF 80 30 6M * 0 6 TTTGGC KKKKKK
read80 0 REF 81 30 6M * 0 6 TTGCGC KKKKKK
read81 0 REF 82 30 6M * 0 6 TGCGCT KKKKKK
read82 0 REF 83 30 6M * 0 6 GCGCTT KKKKKK
read83 0 REF 84 30 6M * 0 6 CGCTTT KKKKKK
read84 0 REF 85 30 6M * 0 6 GCTTTT KKKKKK
read85 0 REF 86 30 6M * 0 6 CTTTTC KKKKKK
read86 0 REF 87 30 6M * 0 6 TTTGTG KKKKKK
read87 0 REF 88 30 6M * 0 6 TTGTGC KKKKKK
read88 0 REF 89 30 6M * 0 6 TGTGCG KKKKKK
read89 0 REF 90 30 6M * 0 6 GTGCGT KKKKKK
read90 0 REF 91 30 6M * 0 6 TGCGTT KKKKKK
read92 0 REF 93 30 6M * 0 6 CGTTTG KKKKKK
read98 0 REF 99 30 6M * 0 6 GCGTTT KKKKKK
read99 0 REF 100 30 6M * 0 6 CGTAGA KKKKKK
read100 0 REF 101 30 6M * 0 6 GTAGAA KKKKKK
read101 0 REF 102 30 6M * 0 6 TAGAAC KKKKKK
read102 0 REF 103 30 6M * 0 6 AGAACT KKKKKK
read107 0 REF 108 30 6M * 0 6 CTAAGA KKKKKK
```

```
read108 0 REF 109 30 6M * 0 6 TAAGAG KKKKKK
read109 0 REF 110 30 6M * 0 6 AAGAGT KKKKKK
read110 0 REF 111 30 6M * 0 6 AGAGTG KKKKKK
read111 0 REF 112 30 6M * 0 6 GAGTGG KKKKKK
read112 0 REF 113 30 6M * 0 6 AGTGGA KKKKKK
read113 0 REF 114 30 6M * 0 6 GTGGAG KKKKKK
```

Lets write our alignments to a file as well as our reference sequence.

```
with open('myalignments.sam', 'w') as f:
    f.write(sam_contents)
with open('ref.fasta', 'w') as f:
    f.write('>REF\n')
    f.write(ref_seq)
```

2.7.6 Visualizing our alignments with IGV

With SAM and FASTA files we can visualize our alignmets with IGV. We can convert our SAM to BAM and index both BAM and FASTA files with samtools.

```
samtools view ./myalignments.sam -bh > ./myalignments.bam
samtools index ./myalignments.bam

samtools faidx ./ref.fasta
```

[width=.9]./igv

3 Referanslar

Li, Heng, and Richard Durbin. 2009. "Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform." *Bioinformatics (Oxford, England)* 25 (14): 1754–60. https://doi.org/10.1093/bioinformatics/btp324.