Project3 Dynamic Programming

Shengyuan Wang
Apr 3, 2020

Adapt the editDistance function we saw in practical (copied below) to answer questions 1 and 2 below. Your function should take arguments p (pattern), t (text) and should return the edit distance of the match between P and T with the fewest edits.

In [10]:
```
def editDistance(x, y):
    # Create distance matrix
    D = []
    for i in range(len(x)+1):
        D.append([0]*(len(y)+1))
    # Initialize first row and column of matrix
    for i in range(len(x)+1):
        D[i][0] = i
    for i in range(len(y)+1):
        D[0][i] = i
    # Fill in the rest of the matrix
    for i in range(1, len(x)+1):
        for j in range(1, len(y)+1):
            distHor = D[i][j-1] + 1
            distVer = D[i-1][j] + 1
            if x[i-1] == y[j-1]:
                distDiag = D[i-1][j-1]
            else:
                distDiag = D[i-1][j-1] + 1
            D[i][j] = min(distHor, distVer, distDiag)
    # Edit distance is the value in the bottom right corner of the matrix
    return D[-1][-1]
```

Q1. What is the edit distance of the best match between pattern GCTGATCGATCGTACG and the excerpt of human chromosome 1?

Q2. What is the edit distance of the best match between pattern GATTTACCAGATTGAG and the excerpt of human chromosome 1?

In [11]:
```
def readGenome(filename):
    genome = ''
    with open(filename, 'r') as f:
        for line in f:
            # ignore header line with genome information
            if not line[0] == '>':
                genome += line.rstrip()
    return genome
```

In [13]:
```
genome_filename = 'chr1.GRCh38.excerpt.fasta'

t = readGenome(genome_filename)

# Question1
p = 'GCTGATCGATCGTACG'
print(editDistance(t, p))
```

```
799984
```

In [14]:
```
# Question2
p = 'GATTTACCAGATTGAG'
print(editDistance(t, p))
```

```
799984
```

Find all pairs of reads with an exact suffix/prefix match of length at least 30. Don't overlap a read with itself; if a read has a suffix/prefix match to itself, ignore that match. Picture the overlap graph corresponding to the overlaps just calculated.

Q3. How many edges are in the graph? In other words, how many distinct pairs of reads overlap?

In [18]:
```
def readFastq(filename):
    sequences = []
    qualities = []
    with open(filename) as fh:
        while True:
            fh.readline()  # skip name line
            seq = fh.readline().rstrip()  # read base sequence
            fh.readline()  # skip placeholder line
            qual = fh.readline().rstrip() # base quality line
            if len(seq) == 0:
                break
            sequences.append(seq)
            qualities.append(qual)
    return sequences, qualities


def overlap(a, b, min_length=3):
    """ Return length of longest suffix of 'a' matching
        a prefix of 'b' that is at least 'min_length'
        characters long.  If no such overlap exists,
        return 0. """
    start = 0  # start all the way at the left
    while True:
        start = a.find(b[:min_length], start)  # look for b's prefix in a
        if start == -1:  # no more occurrences to right
            return 0
        # found occurrence; check for full suffix/prefix match
        if b.startswith(a[start:]):
            return len(a)-start
        start += 1  # move just past previous match


def smart_overlap_map(reads, k):
    olaps = {}
    result = {}
    for read in reads:
        for i in range(len(read)-k+1):
            if read[i:i+k] not in olaps:
                olaps[read[i:i+k]] = [read]
            else:
                olaps[read[i:i+k]].append(read)

    count = 0
    for read in reads:
        read_suffix = read[-k:]
        for possible_read in olaps[read_suffix]:
            if possible_read != read:
                olen = overlap(read, possible_read, k)
                if olen > 0:
                    count += 1
                    result[(read, possible_read)] = olen

    return result, count
```

In [19]:
```
reads_filename = 'ERR266411_1.for_asm.fastq'
reads, _ = readFastq(reads_filename)
pairs_olen, pairs_count = smart_overlap_map(reads, 30)

# Question 3
print("Number of pairs of reads overlap:", pairs_count)
```

```
Number of pairs of reads overlap: 904746
```

In [20]:
```
# Question 4
reads_involved = []
for key, value in pairs_olen:
    reads_involved.append(key)
print("Number of reads have a suffix involved in an overlap:", len(set(reads_involved)))
```

```
Number of reads have a suffix involved in an overlap: 7161
```