# Bi/BE/CS183: Homework One

Eli Sorey

February 4, 2017

## Problem 1

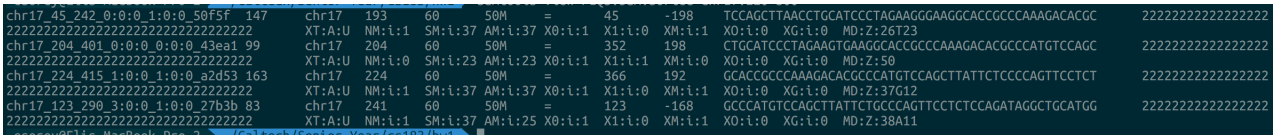
The time complexity of this algorithm is $\mathcal{O}(n^2)$, where $n$ is the length of the array being sorted. The algorithm works by iterating through the array in order. At each element, the algorithm performs another loop that propagates the element backwards until it is less than or equal to the element to its left. In the worst-case, the array is sorted in reverse (descending order), and each element must propagate back the entire length of the array. This is $\mathcal{O}(n)$ computations, and since it can happen for each of the $n$ locations in the array, the overall complexity is as stated.

This image shows an implementation of mergesort, a sorting algorithm that runs in $\mathcal{O}(n \log n)$.

```python
def merge_sort(arr):
    n = len(arr)
    # Base case
    if n <= 1:
        return arr

    # Recursive case
    split_idx = int(n / 2) # Round down for odd n

    # Recursively sort the left and right halves of the array
    left_sorted = merge_sort(arr[:split_idx])
    right_sorted = merge_sort(arr[split_idx:])

    # Join the sorted halves in order
    res = []
    i, j = 0, 0
    while i < len(left_sorted) and j < len(right_sorted):
        if left_sorted[i] <= right_sorted[j]:
            res.append(left_sorted[i])
            i += 1
        else:
            res.append(right_sorted[j])
            j += 1
    while i < len(left_sorted):
        res.append(left_sorted[i])
        i += 1
    while j < len(right_sorted):
        res.append(right_sorted[j])
        j += 1
    return res
```

## Problem 2

Here are the results:

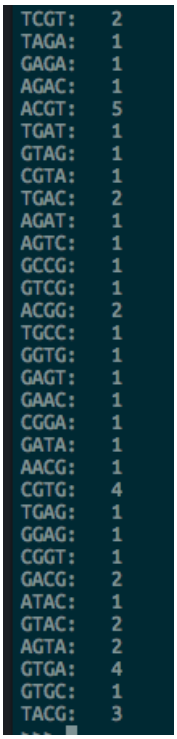



## Problem 3

### a.

Given these assumptions, the expected frequency for any 4-mer is just the probability of a given 4-mer times the number of "4-mer frames" in this set of sequences. There are 10 4-mer frames in each sequence and there are 5 frames, so the expected frequency is $50 * \left(\frac{1}{4}\right)^4 = 0.1953$.

The observed frequencies are shown in the following image:




**b.**

Obviously, even a single occurrence of a given 4-mer results in an observed frequency that is greater than the expected frequency. However, given 50 4-mer frames, there must be some sequences that appear. The sequence 'ACGT' appears 5 times, however, and this is much more frequently than I'd expect.

**c.**

Here's an image with 'ACGT' highlighted:

AGTCGTACGTGAC
AGTAGACGTGCCG
ACGTGAGATACGT
GAACGGAGTACGT
TCGTGACGGTGAT

## Problem 4

### a.

Code attached.

### b.

The scores are 'TTGTAGG' : 40, 'GAGGACC' : 43, 'TATACGG' : 44, 'CCGCAGG' : 28, and 'CAGCAGG' : 11.

### c.

The pattern most likely to be the implanted motif is 'CAGCAGG'. It achieves the lowest Hamming distance score.

### d.

Code attached. The entropy of 'CAGCAGG' is 0.421.

## Problem 5

We will make several assumptions in order to attack this problem. Assume that each of the $t$ sequences have the same length $N$. Then each sequence contains $N-k+1$ k-mers, and the probability of randomly selecting the best k-mer in a sequence is $\frac{1}{N-k+1}$. Our second simplifying assumption is that after each iteration of the algorithm, each of the $N-k+1$ k-mers in each sequence is selected uniformly at random with probability $\frac{1}{N-k+1}$. This is not the case in reality, but it makes the following calculations tractable and is a reasonable assumption for approximation.

Let $X_i \sim Geom\left(\frac{t}{N-k+1}\right)$ be the geometric random variable with success probability $\frac{1}{N-k+1}$ that is the number of algorithm iterations needed for the ith correct kmer to be selected. Note that the ith correct kmer need not be the kmer from the ith sequence; it is simply the ith correct kmer to be chosen in order. The probability that all $t$ sequences find their best kmers within $z$ iterations is:

$$\begin{aligned}
\Pr(t \text{ convergences in } \leq z \text{ iterations}) &= \Pr(X_1 \leq z, X_2 \leq z, ..., X_t \leq z) \\
&= \Pr(X_1 \leq z), \Pr(X_1 \leq z), ..., \Pr(X_t \leq z) \\
&= \Pr(X_i \leq z)^t \\
&= \left(1 - \left(1 - \frac{1}{N-k+1}\right)^z\right)^t
\end{aligned}$$

where the second equality follows from our assumption that all $X_i$ are independently and identically distributed, and the last equality is a simple application of the CDF of the geometric distribution.

The expected number of iterations for all $t$ sequences to find their best kmer (and therefore for the algorithm to converge is

$$\begin{aligned}\mathbf{E}[\sum_{i=1}^{t} X_i] &= \sum_{i=1}^{t} \mathbf{E}[X_i] \\ &= t\left(\frac{N-k+1}{t}\right) \\ &= N-k+1\end{aligned}$$

Thus, for $z$ iterations, the expected number of restarts needed to converge $\frac{N-k+1}{z}$.

## Problem 6

Code attached. Here are the PWMs I got:

| | 1st position | 2nd position | 3rd position | 4th position |
|---|---|---|---|---|
| G | 0.4 | 0 | 0.88 | 0 |
| A | 0 | 1 | 0.04 | 0 |
| C | 0.96 | 0 | 0.08 | 1 |
| T | 0 | 0 | 0 | 0 |

| | 1st position | 2nd position | 3rd position | 4th position |
|---|---|---|---|---|
| G | 0.92 | 0 | 0 | 0.84 |
| A | 0.08 | 0 | 0.96 | 0 |
| C | 0 | 1 | 0 | 0 |
| T | 0 | 0 | 0.04 | 0.16 |

| | 1st position | 2nd position | 3rd position | 4th position |
|---|---|---|---|---|
| G | 0 | 0 | 0 | 0.8 |
| A | 0.16 | 0.32 | 0 | 0 |
| C | 0.8 | 0.68 | 0.96 | 0.16 |
| T | 0.04 | 0 | 0.04 | 0.04 |

## Problem 7

Code attached. I ran the algorithm for 10, 100, and 1000 iterations. Here are the PWMs this yields:

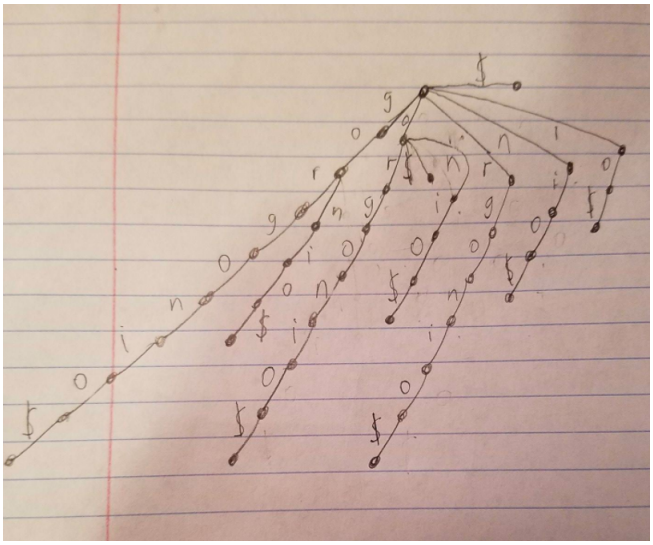
| | 1st position | 2nd position | 3rd position | 4th position |
|---|---|---|---|---|
| G | 0.16 | 0.36 | 0.28 | 0.08 |
| A | 0.24 | 0.12 | 0.4 | 0.24 |
| C | 0.32 | 0.44 | 0.2 | 0.36 |
| T | 0.28 | 0.08 | 0.12 | 0.32 |

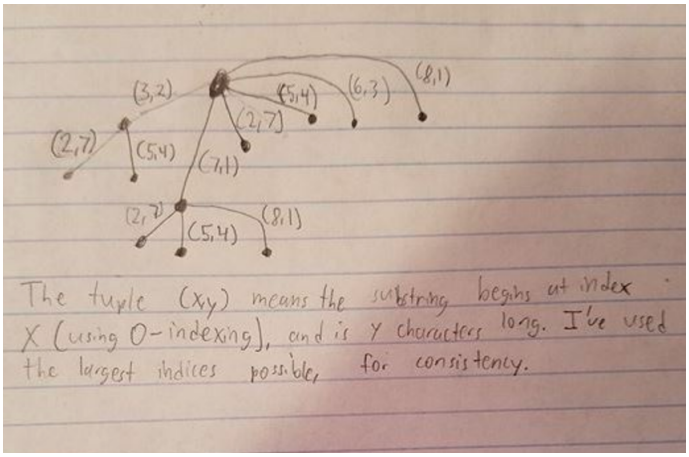
| | 1st position | 2nd position | 3rd position | 4th position |
|---|---|---|---|---|
| G | 0.24 | 0.28 | 0.24 | 0.28 |
| A | 0.28 | 0.16 | 0.24 | 0.28 |
| C | 0.2 | 0.2 | 0.2 | 0.24 |
| T | 0.28 | 0.36 | 0.32 | 0.2 |

| | 1st position | 2nd position | 3rd position | 4th position |
|---|---|---|---|---|
| G | 0.28 | 0.24 | 0.2 | 0.28 |
| A | 0.08 | 0.4 | 0.2 | 0.32 |
| C | 0.28 | 0.08 | 0.24 | 0.24 |
| T | 0.36 | 0.28 | 0.36 | 0.16 |

These are not the same motifs as problem 6. This is because gibbs sampling has no guarantee of monotonically decreasing error, and is therefore not sure to converge.

## Problem 8




(3,2)
(5,4)
(6,3)
(8,1)
(2,7)
(5,4)
(2,7)
(7,1)
(2,2)
(8,1)
(5,4)
The tuple (x,y) means the substring begins at index x (using 0-indexing), and is y characters long. I've used the largest indices possible, for consistency.

## Problem 9

### a.

Code attached.

```
...
('$', 8)
('Gorgonio$', 0)
('gonio$', 3)
('io$', 6)
('nio$', 5)
('o$', 7)
('onio$', 4)
('orgonio$', 1)
('rgonio$', 2)
```

## Problem 10

The entropy metric gives more information than a simple difference score. Say our PWM is generated by ATTT, ATTT, CTTT, CTTT, and we're comparing the pattern GTTT to it. Changing the first element of the PWM-generating set to TTTT does not change the difference score of the pattern, but it does change the entropy score. This is because the entropy score encodes something about the probability of different base pairs in a given location, while the simple difference score does not.