5.1 Overview

Last class, we analyzed the time complexity of the Needleman-Wunsch algorithm for global alignment. Today, we will analogously analyze the Smith-Waterman algorithm for local alignment.

Additionally, we discuss more general concepts of gapped alignment, and explore a plethora of dynamic programming algorithms to calculate them.

5.2 Time Complexity Analysis of Local Alignment

Here lines 3 and 6 require a single operation each, the first being executed \( m + 1 \) times and the second \( n \) times. In line 10, we maximize over 5 cases, requiring an additional comparison relative to the global alignment for a total of 10 operations, and we execute this line \( nm \) times. In line 13, we maximize over \( nm \) items. Let’s say each item requires a load and a comparison operation, for a total of \( 2nm \) operations.

The total number of operations is then \( m + n + 12nm + 1 \) operations, which again is in \( \Theta(nm) \). Again, this analysis is somewhat heuristic, as we weren’t concerned with getting the constants exactly right, but here we
5.3 Gaps, and how they are Scored

5.3.1 A Simplified Model of DNA Structure and Protein Encoding

Large regions of the genome consist of so-called “junk DNA.” This DNA does not code for protein, and often we aren’t sure of its function (or if it even has one).

Eukaryotic genes consist of exons that code for genes and introns that do not.

Most human genes consist of a few to 20 exons.

When a protein is synthesized, the DNA is *transcribed* from DNA to RNA. RNA is a similar macromolecule to DNA: the sugar deoxyribose is replaced by ribose, and the Thymine base (T) is replaced by Uracil (U).

Heteronuclear RNA is then lysed, and exons are then spliced together into *messenger RNA* (mRNA). In order to not complicated the picture with reverse-complementation, we often discuss *complementary DNA* rather than RNA, which refers to DNA strand that is itself the reverse complement of an RNA strand. See Figure 5.1 for a pictorial depiction of the process.

5.3.2 Gap Penalties

Now, let’s relate this situation to sequence alignment. What if we want to align the cDNA of a protein transcript to the nuclear DNA from which it was transcribed? The difficulty arises in that large section of nuclear DNA are excised before exons are spliced together to form mRNA, so we will need to be tolerant of
large gaps in our alignment. However, we don’t want there to be too many of these gaps, and we don’t want them to be too small.

In this case, we have a somewhat different notion of gap. A small gap and a big gap are in some sense equivalent, as we don’t care how long the introns are, rather we want to work under the assumption that there aren’t “too many” of them.

In order to handle this, we can use a nonlinear gap function. One type of function in popular use that we will discuss in detail today is the affine gap function. An affine function is a linear function with a constant added. More generally, we define gap functions of the form \( w(l) : \mathbb{Z} \to \mathbb{R} \). The affine gap function is then \( w(l) = \alpha + \beta l \) (note that this is the case we have already learned). A few different gap functions of varying degrees of usage are presented in Figure 5.2.

5.3.3 Gapped Alignment Recurrence Relationships

\[
V_{i,j} \triangleq \max(E_{i,j}, F_{i,j}, G_{i,j}) \\
G_{i,j} \triangleq V_{i-1,j-1} + \delta(x_i, y_j) \\
E_{i,j} \triangleq \max_{k \in \{0, 1, \ldots, i-1\}} V_{i,k} - w(j - k) \\
F_{i,j} \triangleq \max_{l \in \{0, 1, \ldots, j-1\}} V_{l,j} - w(i - l)
\]

Here \( V_{i,j} \) gives the gapped alignment cost of prefixes of lengths \( i, j \). \( V \) is defined in terms of \( G, E, \) and \( F \). \( G_{i,j} \) gives the optimal gapped alignment score of prefixes of \( x, y \) of lengths \( i, j \), conditional on the fact that the \( i \)th and \( j \)th characters are aligned. \( E \) and \( F \) on the other hand give optimal alignments of prefixes conditional on the last character containing a gap: \( E_{i,j} \) is conditional on the fact that the terminal position in the alignment contains a gap in sequence \( y \), and \( F_{i,j} \) requires that the alignment ends on a gap in sequence \( x \).

With these definitions, clearly \( V_{i,j} \) is correct: note that \( E_{i,j}, F_{i,j}, \) and \( G_{i,j} \) give the optimal gapped alignment cost under three mutually exclusive conditions, and \( V_{i,j} \) gives the optimal amongst these. \( G_{i,j} \) requires that the final character pair are aligned, so its score is identical to the aligned (no gap) case in linear gapped alignment. \( E_{i,j} \) represents the case where \( y \) has a gap. We consider all possible alignments that end in an \( y \) gap: such an alignment can start with an arbitrary amount of the \( y \) string consumed. The maximum over \( k \) represents the maximum score over alignments of the prefix of length \( k \) of \( x \) and length \( i \) in \( y \), and then the remainder of \( x \) is filled with gaps in the alignment. Similarly, we may flip \( x \) and \( y \) to make the same argument for \( F_{i,j} \).

5.3.3.1 Computational Costs

Using this recurrence, we can handle any gap penalty function. However, we pay an asymptotically significant cost in terms of time, and a constant factor cost in terms of memory, to use this technique. Because the maxima in \( E \) and \( F \) are over a non-constant number of cells, the time complexity increases. Each such maximum needs to examine no more than \( \max(m, n) \) cells of \( V \), and there are \( 2mn \) cells in \( E \) and \( F \). \( G \) on

\[1\] In other words, a function of the form \( f(x) = mx + b \). You may have learned in elementary school that such a function is linear, but this is an abuse of terminology. Technically, a linear function \( f \) is a function such that \( f(ax + by) = af(x) + bf(y) \), which takes the form \( f(x) = mx \) over the real numbers.
the other hand only needs to examine a constant number of cells, so evaluating it is efficient, requiring $O(mn)$ time, and by the same reasoning, evaluating $V$ also requires $O(mn)$ time. The total time to evaluate every cell in each of these four matrices is thus $O(mn) + O(mn) + O(mn(m+n)) + O(mn(m+n)) = O(mn(m+n))$, asymptotically.

Note also that the memory requirement increases. Rather than requiring $(m+1)(n+1)$ matrix cells, we now require $4(m+1)(n+1)$ matrix cells. However, this change is only by a constant factor, so there is no asymptotic difference.

### 5.3.4 Affine Gap Penalties

In the previous section, we saw that general gapped alignment is asymptotically inferior to linear gapped alignment. Here we derive the recurrence relationship for the special case of affine gap penalties and show that it is asymptotically equivalent to linear gapped alignment, again with a constant factor memory increase.

Let the following be so:

- $\alpha \doteq \text{match score.}$
- $\beta \doteq \text{match penalty.}$
- $\gamma \doteq \text{gap open penalty.}$
- $\tau \doteq \text{gap extension penalty.}$

Under these hypotheses, note that the affine gap penalty is given by $w(l) = \gamma + \tau l$. Now, we take the following recurrence relationships.

$$
V_{i,j} = \max(E_{i,j}, F_{i,j}, G_{i,j})
$$

$$
E_{i,j} = \max(E_{i,j-1} - \tau, V_{i,j-1} - \gamma - \tau)
$$

$$
G_{i,j} = \begin{cases}
  x_i = y_j & : V_{i-1,j-1} + \alpha \\
  x_i \neq y_j & : V_{i-1,j-1} - \beta
\end{cases}
$$

$$
F_{i,j} = \max(F_{i-1,j} - \tau, V_{i-1,j} - \gamma - \tau)
$$

Here $V, G, E,$ and $F$ all have the same interpretations as above.

Therefore, in order to prove this algorithm correct, all we need do is show that $V_{i,j}$ here is equivalent to $V_{i,j}$ in the general gapped alignment recurrence. Note that $V$ and $G$ are defined equivalently, but both are dependent on $E$ and $F$, thus we need only show that $V$ and $F$ are equivalent, and it follows that $G$, and thus $V$ as well, are also equivalent.

The only thing we changed was that we fixed $w(l) = -\gamma - \tau$. This algorithm does not work for general gapped alignment, so we will need to rely on this change in order to prove that it works for affine gapped alignment.

Intuitively, $E$ and $F$ make sense, as gaps cost $\gamma$ to open, and given that we are opening a gap, the previous pair did not have an equivalent gap, thus we take the score in $V_{i,j-1}$, subtract cost $\gamma$ to open a gap, and subtract $\tau$ to extend this gap to length 1. If, on the other hand, there is a highly scoring alignment that $\text{did}$

Note that $O(\max(m,n)) = O(m + n)$: to see this, consider the following: $\lim_{m,n \to \infty} \frac{\max(m,n)}{m+n} \leq \frac{\max(m,n)}{2\max(m,n)} = \frac{1}{2}$.

Here we write them in terms of match/mismatch/gap penalties rather than alignment score matrix $\delta$ and gap penalty function $w(\cdot)$. You should convince yourself that these formulae may be equivalently expressed in terms of $\delta$ and $w(\cdot)$, rather than $\alpha, \beta, \gamma,$ and $\tau$. 

---

2Note that $O(\max(m,n)) = O(m + n)$: to see this, consider the following: $\lim_{m,n \to \infty} \frac{\max(m,n)}{m+n} \leq \frac{\max(m,n)}{2\max(m,n)} = \frac{1}{2}$.

3Here we write them in terms of match/mismatch/gap penalties rather than alignment score matrix $\delta$ and gap penalty function $w(\cdot)$. You should convince yourself that these formulae may be equivalently expressed in terms of $\delta$ and $w(\cdot)$, rather than $\alpha, \beta, \gamma,$ and $\tau$. 

---
have an equivalent gap in the previous alignment, then we may take $E_{i,j-1} - \tau$ over $V_{i,j-1} - \gamma - \tau$. Here the cost is only $\tau$, as the gap has already been opened. Note that this may occur even if $E_{i,j-1} < V_{i,j-1}$: the gap open penalty $\gamma$ is used to control this behavior.

The above intuition could be turned into a formal proof, but we would require some technical details and essentially have to redo all the work from the general gapped alignment proof. I don’t give a formal proof here, but I give an alternative simple argument that could easily be turned into one. Instead of directly showing correctness, we need only show that $E_{i,j}$ and $F_{i,j}$ are equivalent between the two cases. An informal proof follows:

Let $E'$ denote the $E$ from the general recurrence, and $E$ denote the $E$ from the affine gap penalty recurrence. Also, we proceed without loss of generality, noting that the following applies to $F$ as well by symmetry.

The crux of the argument is the following simple observation: $E_{i,j-1} - \tau \geq E_{i,j-2} - 2\tau$ for any $i,j$. This follows by way of contradiction; if this were not true, $E_{i,j-1} = \max(V_{i-1,j-2}, E_{i,j-2} - \tau)$ would be violated, as this would imply that $E_{i,j-1} < E_{i,j-2} - \tau$. This fact implies a much stronger looking property of $E$: namely that $E_{i,j-1} \geq E_{i,k} - (\tau(j - 1) - k))$ for any $k \leq j - 1$. To see this, simply note that this is repeated application of the original observation: each time $k$ increases, $E_{i,k}$ decreases by no more than $\tau$.

Now, consider $E'_{i,j} = \max_{k \in \{0, 1, \ldots, i-1\}}(V_i, k - w(j - k))$. Using the scoring methodology of $E$, this is equivalent to $\max(V_{i,j-1} - \gamma - \tau, E_{i,j-1} - \tau, E_{i,j-2} - 2\tau, E_{i,j-3} - 2\tau, \ldots, E_{i,0} - j\tau)$. Using the stronger property above, we see that any term after the second term is no greater than the second term, so we may remove them from the consideration in the maximum. This only leaves the first two terms, namely, we have that $E'_{i,j} = \max(V_{i,j-1} - \gamma - \tau, E_{i,j-1} - \tau) = E_{i,j}$.

You should try to formally understand the above argument on its own, but it may help to understand it from a less formal intuition based perspective as well. The key observation translates to the following: “The optimally scoring alignment that ends in a double gap on string $y$ over prefixes of $x$ and $y$ of lengths $i$ and $j - 1$ and $j - 2$ scores at least as well as the optimally scoring alignment that ends in a triple gap on string $y$ over prefixes of $i$ and $j - 2$.” The stronger form of this is basically that “extending a gap of any length has score no more than extending the optimal sequence ending in a gap by 1.” We then translate $E'$ to use the same scoring mechanism as $E$, and then using this property, show that the maximum is equivalent to a maximum over two values: $V_{i,j-1} - \gamma - \tau$ and $E_{i,j-1} - \tau$, or in other words, the cost of creating a new gap and extending an existing gap.

Now that we see the algorithm works, the natural next question to ask is how much time it takes? Have we beaten the $O(mn(m + n))$ cost of general gap alignment? Hopefully by now you are getting the hang of asymptotic analysis (if not, a great exercise would be to repeat the analysis for the general gapped alignment for this case), so I will skip a few steps. Each of the 4 matrices has $(m+1)(n+1)$ cells, and each cell requires constant time to calculate. Therefore, we may conclude that the algorithm requires only $O(mn)$ time.

### 5.3.4.1 Heuristic Alignment Statistics of Affine Gap Penalties

In Lecture 3, we discussed informal alignment statistics for linear gapped global alignment. Today we will generalize this model to cover affine gap penalties.

Suppose we have $a$ matches, $b$ mismatches, $c$ gap openings, and $d$ gap positions. Then, the alignment cost is $\alpha a - \beta b - \gamma c - \tau d$, and the optimal alignment maximizes over these.
5.4 Global vs Local Assignment

Generally speaking, local alignment allows us to focus in on regions of high similarity, whereas global alignment forces us to align both sequences in their entirety. The optimal global alignment does not necessarily contain the optimal local alignment (see Figure 5.3 for an example), so when two sequences have low overall similarity, but a region in common, local alignment may be necessary to identify it.