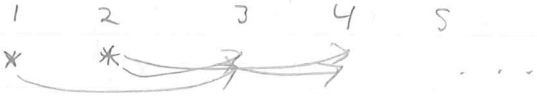
Proof techniques

It's one thing for something to be true, or for it to work - it's quite another to argue rigorously for for it - the essense of proof. There are of course various levels of proof: the most stringent would be a set of sentences starting from axioms and employing only rules of logic to build the conclusion. Less stringent is an informal argument where we can assume the audience can use previously shown results and make certain jumps in logic. Generally, demonstrating a pattern for a subset of specific cases does not constitute an acceptible proof.

We've seen one proof technique already - proof by induction, wherein we argue for truth (or correctness of an algorithm) by starting w/ a base case and arguing for the correctness of higher levels based on preceeding levels:



By arguing generally, all levels can be shown to be correct/true.

Proof by contradiction

Another beautiful proof technique - arguably the most powerful and common, is proof by contradiction. Here, we <u>assume</u> the <u>opposite</u> of what we wish to prove, show this implies a contradiction, and therefore our assumption must be wrong.

<u>Ex:</u> There is no smallest integer.
<u>Proof:</u> Assume there <u>IS</u> a smallest integer, let it be i. $i-1$ is also an integer, but is smaller than i; a contradiction in our hypothetical world, so our assumption is false.

The technique is a bit trickier though, when we want to prove something but we make multiple assumptions, only one of which is our opposing assumption.

Ex IF a cars battery is dead and its key is turned, it won't start.

Proof (by contradiction)

Assume the opposite: it ① starts, even though its key is turned ② and the battery is dead (3)

Since it starts, it must use the starter - an electrical device. This needs energy - from the only source of eergy on the car, the battery. But we assuned the battery is dead (3), thus we have a contradiction.

From this proof we can conclude 3 things: either the car can't start, or the battery can't be dead, or the key can't have been turned. But we want the not starting proof - we can get if we continue to assume the battery is dead and the key is turned.

The beauty and power of a proof by contradiction is there are many facts one can bring to bear to 'search out' a contradiction, even if we eventually find one that has seemingly little to do with what we want to prove. We just need to show the 'hypothetical world' is absurd in some way.

Sequence Alignment (Global)

Sequence alignment is the problem of: given two similar strings (DNA, in our case), of potentially different lengths, add "-" (gap) characters to make them the same length and maximize the modified strings' similarity:

Sequence X (length n): ACTAGC ⇒ $\dot{A}CT\dot{A}GC$
Sequence Y (length m): ATACC $\quad\dot{A}-\dot{T}\dot{A}C\dot{C}$
+2 -4 +2 +2 -3 +2 =

Of course, in order to do this we need some way to quantify the similarity of the modified strings. To start with, we'll do this with a set of simple scoring rules, like:

matching bases: +2
mismatching bases: -3
gap character: -4

We sum these up over the alignment to produce the total score (1, in this case). So, more formally, we want to produce an alignment that maximizes the alignment score according to the scoring parameters.

Here, we should note that this process only really makes sense for real DNA sequences if mathing is worth more than mismatching, and usually this is worth more than gaps:

gap score < mismatch score < match score.

—//—

In some very simple cases, this problem is quite easy: if the two strings are only 1 base long, the optimal alignment is simply the strings:

```
x:   A  =>  A      or      A  =>  A
y:   A      A              T      T
```

(since mismatches are less costly than gaps). Although a bit odd to consider, if one of the strings is of length ∅, then the optimal alignment must simply fill it out w/ gaps:

```
x:   A    =   A      or     ""    =>   -
y:   ""       -             T          T

                     or     ACT   =>   ACT
                            ""         ---
```

(This is our 'base case' - hah, base case! Get it?)

What about other cases? Consider some optimal alignments:

```
x:   ACTAGC   =>   ACTAG|C|
y:   ATACC         A-TAC|C|

x:   ACTAGC   =>   ACTAG|C|
y:   ACAG          AC-AG|-|

x:   ACTAGC   =>   ACTAGC|-|
y:   CAGCT         -C-ACC|T|
```

In all cases, the end of the alignment strings has either a gap, -, or the last base of the input string. After all, how could it have anything else? (If it were to have some other base, that wouldn't be part of the input string).

Let's make some definitions. Given two strings x and y, let p_x be the 'prefix' of x (all except the last base), and e_x be the end (last base) of x. Similarly for y.

$$x: \overbrace{ACTAG}^{p_x}\overbrace{C}^{e_x}$$

$$y: \underbrace{ATAC}_{p_y}\underbrace{C}_{e_y}$$

Based on the observation above, there are three 3 possibilities for the alignment of x and y, and their scores:

A

$$\begin{pmatrix} p_x \text{ aligned w/} \\ p_y \end{pmatrix} \begin{matrix} e_x \\ e_y \end{matrix}, \qquad \text{score} = S(A) + S(e_x, e_y)$$

B

$$\begin{pmatrix} p_x e_x \text{ aligned w/} \\ p_y \end{pmatrix} \begin{matrix} - \\ e_y \end{matrix} \qquad \text{score} = S(B) + S(-, e_y)$$

C

$$\begin{pmatrix} p_x \text{ aligned w/} \\ p_y e_y \end{pmatrix} \begin{matrix} e_x \\ - \end{matrix} \qquad \text{score} = S(C) + S(e_x, -)$$

The best alignment, then, is the best scoring of these three options:

This relationship is often written in a more obtuse mathematical notation:

$$S(X_{1,i}, Y_{1,i}) = \max \begin{cases} S(X_{1,i-1}, Y_{1,i-1}) + s(X_i, Y_i), \\ S(X_{1,i}, Y_{1,i-1}) + s(\text{"-"}, Y_i), \\ S(X_{1,i-1}, Y_{1,i}) + s(X_i, \text{"-"}) \end{cases}$$

Where $S()$ is the scoring function, $X_{a,b}$ represents the sequence of X from a to b, and X_i is just the base in X at position i.

Now, there's a similarity here between solving the alignment problem and our bees problem - we've identified the patterns that we think we can use to solve the problem - in a recursive fashion - via base cases and a recursive case. Certainly, the argumentation we've used is proof-like, and it nearly *is* a proof. But how do we know there isn't some outside possibility that this method won't produce optimal alignments? We can finish the argument w/ a, you guessed it, proof by induction. Let's dot those i's and cross those t's.

Proof of recursive definition/solution

Let the base cases be defined as before - sequences where the length of one is Ø, or both are exactly 1 base long. For both one base long, the optimal alignment is clearly just the two aligned together. (So long as the gap score is smaller than the mismatch and match scores). For the other case, the *optimal* alignment is just the Ø-length sequence padded w/ "-"s.

Any other pair of sequences X and Y must have one of length at least 1 and the other of length at least 2, and so can be handled by the recursive/inductive case.

Inductive case: Assume that the 3 subalignments A, B, C have been computed and and are <u>optimal</u> (ie, S(A) is the minimum score aligning P_x and P_y, etc.)

For a (sub) proof by contradiction, assume that the optimal alignment Z is <u>not</u> one of $A\begin{matrix} e_x \\ e_y \end{matrix}$ or $B\begin{matrix} - \\ e_y \end{matrix}$ or $C\begin{matrix} e_x \\ - \end{matrix}$, and that the score is smaller than any of these.

However, we know that Z must end in either $\begin{matrix} e_x \\ e_y \end{matrix}$ or $\begin{matrix} - \\ e_y \end{matrix}$ or $\begin{matrix} e_x \\ - \end{matrix}$: any other possibility would mean its not actually an alignment for X and Y! Without loss of generality (ie - the same argumentation will work for other cases), suppose Z is

$$\overset{Z'}{\begin{pmatrix} P_x \text{ aligned w/} \\ P_y \end{pmatrix}} \begin{matrix} e_x \\ e_y \end{matrix}$$

The claim is that the score of Z is less than the score for A:

$$S(Z') + S(e_x, e_y) < S(A) + S(e_x, e_y)$$

This would imply that S(Z') - Z's alignment P_x and P_y, is <u>better</u> than the alignment of these in A, which <u>contradicts</u> the assumption that S(A) is <u>optimal</u>.

Q.E.D.