PROFESSOR: All right, so let's get started. So today we're going to review local alignment, we talked about last time, and introduce global alignment, also talking about issues related to protein sequences, which include some more interesting scoring matrices. So just some info on topic one, which we are still in. So I will have an overview slide. It'll have a blue background, and there will be a review slide with a purple background in every lecture.

So last time we talked about local alignment and some of the statistics associated with that, and also a little bit about sequencing, technologies, both conventional Sanger DNA sequencing as well as second generation sequencing. And at the beginning of the local alignment section, we introduced a simple BLAST-like algorithm, and then we talked about statistics, target frequencies, mismatched penalties, that sort of thing.

So there were a couple questions at the end which I just wanted to briefly answer. So I believe it was Joe asked about how the dye is attached to the DNTP in dye terminator sequencing. And it appears that it's attached to the base, sort of the backside of the base, not the Watson-Crick face, obviously. That seems to be the common way that it's done.

And then there was another question from somebody in the back. I don't remember who asked about when you're making libraries, how do you make sure that each of your insert sequences has the two different adapters, one adaptor on one side and the other adapter on the other side? And there are at least three ways to do this. So simplest is in RNA ligation, when you take advantage of the different chemistry at the five prime and three prime ends of the small RNA that you're trying to clone. So you just use the phosphate and NLH to ligate two different adapters.

Another more complicated way occurs in ribosome footprint profiling, which is a method for mapping the precise locations of ribosomes along mRNAs, and involves polyA tailing, and then introducing the adapters together, the two adapters, with a polyT primer that primes off the polyA tail. And then you circularize, and then you PCR off the circles. And it's a little bit complicated, but you can look it up in the reference that's up here on the slide. It's working now.

And then, finally, the way that's actually most commonly used for protocols like RNA seq and genomic DNA sequencing is that after you make your double strand DNA, there's an enzyme that adds a single A to the three prime end of each strand. So now you have a symmetrical molecule. But then you add these funny Y-shaped adapters that have and overhanging T on, say, the red guy here.

And so what will happen is that each of these Y's can be ligated here. But each of the inserts, independent of which strand it is, will have a red adapter at the five prime end and a blue adaptor at the three prime end. Any questions about this or about sequencing technologies before we go to local alignments? OK, good. It was a good question, and that's the answer.

So we motivated our discussion of local alignments last time by talking about this example, where you have a non-coding RNA that you found, inhuman. You BLAST it against mouse, and you get this alignment. Is this significant? So is this really likely to be a homologous sequence? And how do you find the alignments?

And so we said that, well, there's this theory that's exact, at least exact in the asymptotic sense for large query and database sizes that tells us the statistical significance of the highest scoring ungapped local alignment. And it's given by this formula here, which is the extreme value or Gumbel distribution. And then we talked about the constraints or the expected score has to be negative, but positive scores have to be possible for this theory to work. And we also talked about an algorithm.

But if you remember, the algorithms was very simple. It involved-- this is zero-- keeping track of the cumulative score. So we have a mismatch and a match, mismatch, mismatch, mismatch, match, match, match. That is a high scoring segment, et cetera. So you keep track of the lowest point you've ever been to as well as the current score. And when the current score exceeds that lowest point you've ever been to by more than we've ever seen before, more than this, then that's your high scoring segment.

Now it turns out, if this is not intuitive to you, there's another algorithm which I find, personally, more intuitive. So I just want to tell you about that one as well. And it's basically the same thing, except whenever you go negative, you reset to zero.

So here, we were going to go negative, so we just reset to zero. That was on this mismatch here. Then we have a match. Now we're at plus 1. That's fine. Now we have a mismatch. Now we're down to zero. We don't need to do anything.

Now we have another mismatch. Here, we're still at zero. Remember, we just stay at zero. We were going to go negative, but we stayed at zero. Another mismatch, we still stay at zero. And now we have these three matches in a row. My line is not staying very flat.

But this should've been here flat at zero. The point is that now the highest scoring segment is the highest point you ever reach. So it's very simple. So this is actually slightly easier to implement. And that's sort of a little trick. So for local alignments, you can often reset to zero. Any questions about that?

So well, we talked about computational efficiency, this big O notation, where you consider the number of individual computations that are required to run an algorithm as a function of the size of the input, basically the number of units in the problem base pairs, amino acid residues, whatever. So computer scientists look at the asymptotic worst case running time. That's either because they're pessimistic, or perhaps because they want to guarantee things. They want to say, it's not going to be worse than this. Maybe it'll be faster, and then you'll be happy. But I can guarantee you, it's not going to be worse than this. And so in this case, the algorithm we talked about was order n times n, where that's the lengths of the two sequences.

So toward the end last time, we get we talked about this lambda parameter and said that lambda is the unique positive solution to this equation here, where sij are the scores and pi and rj are the nucleotide frequencies. And then there's this target frequency formula that comes up that says that if you use a scoring system sij to apply to sequences, and then you pull out just the high scoring segments, the ones that are unusually high scoring, they will have a frequency of matching nucleotides qij that's given by the product of the frequencies in the two sequences basically weighted by e to the lambda sij. So matches will occur more strongly, because that has a positive work, and mismatches less strongly.

And that then gives rise to this notion that there's an optimal mismatch penalty, if you just consider scoring systems that have plus 1 for a match and m for a mismatch, some negative number, that's given by this equation here, and here I've worked out a couple of values. So the theory says that to find matches that are 99% identical, you should use a mismatched score of minus 3, but for 75% identical, you should use minus 1. And I asked you to think about does that make sense, or how is that true?

So y is minus 3 better than minus 1 for finding nearly identical matches. Anyone have an idea or thought on this? There's some thoughts on the slide. But can anyone intuitively explain why this is true? Yeah, what's your name?

AUDIENCE: Eric.

PROFESSOR: Yeah, go ahead.

AUDIENCE: With a mismatch penalty of minus 3, you actually need more steps climbing back up to get back to some local maximum. And therefore, you do require a longer stretch [INAUDIBLE] matches in order to get a significant hit. That's my guess as to why a score of m equals minus 3, a penalty, [INAUDIBLE] why it would be better at finding the higher identity matches.

PROFESSOR: OK, because the minus 3 makes you go down faster, so it takes longer to recover, so you can only find nearly identical things with that kind of scoring system. Is that your point? OK, that's a good point. So yeah, when would you want to use a mismatched penalty of minus 1?

AUDIENCE: When you're trying to look for things that are [INAUDIBLE], but maybe not so close. Well, when you're looking for a [INAUDIBLE], you're looking for [INAUDIBLE]. That kind of situation.

PROFESSOR: And so let's say I'm using a mismatch penalty of minus 2. Can I find regions that are 66% identical?

AUDIENCE: Probably.

PROFESSOR: Probably.

AUDIENCE: But not guaranteed.

PROFESSOR: Anyone else have a comment on that? Match is plus 1. Mismatch is minus 2 regions of 66% identity. Yeah, with Levi, yeah.

AUDIENCE: No, since your score will just be zero.

PROFESSOR: Yeah. That's correct. So Levi's comment is your score will be zero. Well, I'll just say plus for match, plus, plus, minus, plus, plus, minus. I mean, it'll be interspersed. It doesn't have to be like this for every triplet. But but on average, you'll have two matches for every match. That's what 66% identity means. And so these will score a total of plus 2. And this will score minus 2. And so you basically will never rise much above zero.

And so you you can't really use that mismatch penalty. There's a limit. 66% is sort of at a point where you can no longer see. You could potentially see things that are 75% identical if they were incredibly long with that kind of mismatch penalty. But you just can't see anything below 2/3% identity with minus 2.

So to find those low things, you have to use the lower. You have to go down to minus 1 if you want to find the really weak matches. But they will have to be correspondingly very long in order to achieve statistical significance.

So correspondingly, the reason why it's better to use a harsher mismatch penalty of minus 3 to find the nearly identical regions is that, in this equation, when you go from having a plus 1, minus 1 scoring system to plus 1, minus 3, lambda will change. This equation will no longer satisfied so that a new value of lambda will be relevant. And that value will be larger.

That's not totally obvious from this equation because you sort of have one term, which is either the minus lambda in one term or either plus lambda. But it turns out that that making the mismatch penalty more negative will lead to a solution that's a bigger value of lambda.

So that means that the same score, x, will lead to a larger negative exponent here. and? How will that affect the p-value? Anyone take us through this? It's a little bit convoluted with all these negative exponentials and stuff, but can someone explain to us how that affects the p-value? Same x. We're going to increase lambda. What happens to the p-value?

This gets bigger, more negative. That means this e to the minus thing gets closer to 0. That means that this is inside an exponential. As that thing gets closer to 0, this whole term here gets closer to 1. Therefore you're subtracting it from 1. Therefore the p-value gets smaller, closer to 0, more significant. Does that made sense? So it's good to just work through how this equation works.

So that's all I wanted to say about mismatch penalties for DNA. Any questions about that? So how do you actually use this in practice? So if you just Google "BLAST end," you'll get to this website. It's been set up at NCBI for about 20 years or so. And of course, it's gone through various iterations and improvements over the years.

And if you look down at the bottom, there is a place where you can click and set the algorithm parameters. And there are a number of parameters you can set. Some of them affect the speed. But we're focused here mostly on parameters that will affect the quality, the nature of the alignments that you find.

And so here, you can set not arbitrary penalties, but you can set within a range of standard mismatch penalties. You can do 1 minus 1, 1 minus 2, et cetera.

So what about sequences that code for protein? So exons, for example. So you can search them with a nucleotide search, like BLAST. But it can often be the case that you'll do better if you first translate your exon into the corresponding amino acid sequence using a genetic code and then search that peptide.

Now you may or may not know the reading frame of your exon a priori, or even know that it is an exon, so BLAST automatically will do this translation for you. So for example, with this DNA sequence, it'll translate in all three of the reading frames, leading to essentially this bag of peptides here, where sometimes you'll hit a stop code on, like right here. And then it just treats it as, OK, there's a little PR dipeptide there. And then there's a longer peptide here, [INAUDIBLE], and so forth.

So it just makes these bags of peptides for each reading frame and searches all those peptides against some target, which can be approaching database or a DNA database, again, translated in all the reading frames. So the folks at NCBI have made all these different flavors of BLAST available. So BLASTP is for proteins. N is for nucleotides. And then the translating ones are called things like BLASTX for a nucleotide query against a protein database. TBLASTN for a protein query against a nucleotide database, which gets translated in all frames, or TBLASTX, where you translate both the nucleotide sequences in all frames.

And then there's a number of other versions of BLAST which we probably won't discuss but that are well described in the textbook and other accessible online sources. Let me ask you this. So remember ESTs. So ESTs are segments of cDNAs that typically correspond to one ABI 3700 Sanger sequenc off of that cDNA, so one read, like 600 bases or so.

So let's say you have some ESTs from chimp. And you don't have the chimp genome yet. So you're going to search them against human. What would you do? Would you use a translating search? Or would you use a BLASTN search? Or does it matter?

Chimp is a 98% identical human, very high. Any ideas? Yeah, Tim.

AUDIENCE: You could use a translating search, because you know that the cDNAs are at least coding for RNAs. And so if you just use a nucleotide search, then you're not going to have functional significance in terms of the alignment. But if it's going for a protein, then--

PROFESSOR: You mean you won't know whether it is the protein coding part of the cDNA or not?

AUDIENCE: So I just mean that if you're looking between chimp and human, then you're expecting some sort of mismatch. But it's possible that it could be a functional mismatch. Then you know that the cDNA is maybe coding for a protein. Therefore, if the mismatch is between two similar amino acids, then that would be picked up by a translating search, but it would be skewed against it in a nucleotide search.

PROFESSOR: OK, fair enough. But if you assume that the two genomes are, let's say, 97% identical, even in a non-coding region, which they're very high. I don't remember the exact percent, but very high. Then if you're searching 600 nucleotides against the genome, even if it's 95% identical, you'll easily find that under either. So either answer is correct, BLASTN or BLASTX. And the UTRs could only be found by-- if it happened that this was a sequence from a three prime UTR, you could only find that by BLASTN typically.

What if it's a human EST against the mouse genome? So mouse exons are about 80% identical to human exons at the nucleotide level, typically. Any ideas? What kind of search would you do? BLASTN, BLASTX, or something else? TBLASTX. Yeah, go ahead.

AUDIENCE: I have another question. What exactly is the kind of question we're trying to answer by doing this BLAST search?

PROFESSOR: Oh, well, I was assuming you're just trying to find the closest homologous cDNA or exons in the genome-- exons, I guess, yeah, the exons. of the homologous gene. Yeah, that's a good question. Exons of a homologous gene. We've got a human

EST going against the mouse genome. When do we do?

AUDIENCE: I suggest BLASTP because--

PROFESSOR: Well, BLASTP, that's protein. This is a nucleotide sequence against nucleotide. So we can do BLASTN or TBLASTX, let's say.

AUDIENCE: TBLASTX.

PROFESSOR: TBLASTX. You translate your EST, translate the genome, search those peptides. TBLASTX, why?

AUDIENCE: The nucleotide sequences may be only about 80% similarity, but the protein sequences functionally, due to the functional constraints, you might actually get higher similarities there.

PROFESSOR: Yeah. It's exactly right. So they are about, on average, about 80% identical. It varies by gene. But a lot of those variations that occur are at the third side of the codon that don't effect the amino acid, because there's a lot of constraint on protein sequence. And so you'll do better, in general, with a translating search than with a nucleotide search.

Although, they both may work. But you may find a more complete match with a translating search. That's good. Everyone got that? Sally, yeah.

AUDIENCE: Is there a reason why you wouldn't use BLASTX and instead you use TBLASTX?

PROFESSOR: Yeah, I just gave the example of searching against the genome. But you could search against the mouse proteome as well. You might or might not. It depends how well annotated that genome is. Mouse is pretty well annotated. Almost all the proteins are probably known. So you probably get it.

But if you were searching against some more obscure organism, the chameleon genome or something, and it wasn't well annotated, then you might do better with searching against the genome, because you could find a new x on there. OK, good. Question yeah, go ahead.

AUDIENCE: So when we do these translations, these nucleotide, amino acid things, do we get all frames? Do the algorithms to all frames?

PROFESSOR: Yeah, all six frames. So three frames on the plus strand, and three frames on the reverse strand. Yeah. All right, great.

So that's the end of local alignment, for the moment. And we're going to now move on to global alignment using two algorithms. For global alignment, Needleman-Wunch-Sellers, and then for gapped local alignment, Smith-Waterman. And toward the end, we're going to introduce the concept of amino acid substitution matrices.

So the background for today, the textbook does a pretty good job on these topics, especially the pages indicated are good for introducing the PAM series of matrices. We'll talked a little bit today and a little bit next time.

So why would we align protein sequences? So the most obvious reason is to find homologues that we might, then, want to investigate, or we might, for example, if you have a human protein and you find homologous mouse protein, and that mouse protein has known function from a knockout or from some biochemical studies, for example, then you can guess that the human protein will have similar function. So we often use this type of inference that sequence similarity implies similarity in function and/or structure.

So how true is this? So it turns out, from a wide body of literature, that this inference that sequence similarity implies functional and structural similarity is almost always true when the sequence similarity is more than about 30% identity over the whole length of a protein, over 300, 400 amino acids. That's a good inference.

Below that, sort of in the 20% to 30% sequence similarity, that's often referred to as the Twilight Zone, where sometimes it's a good inference, and sometimes it's not. So you need to be a little bit careful. And below that, it's deeper into the Twilight Zone, where most of the time you probably shouldn't trust it. But occasionally, you can see these very remote homologies. You might want to have additional information to support that kind of inference.

And I want to just point out that the converse is just not true in biology. So structural similarity does not imply sequence similarity or even derivation from a common ancestor. So you may think, well, every protein has a really complex, elaborate three dimensional structure, and there's no way that could ever evolve twice.

And it's true that probably that exact structure can never evolve twice. But a very similar structure, a similar fold even, in terms of the topology of alpha helices and beta strands, which Professor Frank will talk about later in the course, the identical fold can involve more than once. It's not that hard to evolve a pattern of alpha helices and beta strands.

And so this point about structural similarity not implying sequence similarity, the way I think about it is like this, like here are two organisms. This is a hummingbird, you've all seen. And some of you may have seen this. This is a hawk moth, which is an insect that is roughly two inches long, beats its wings very fast, has a long tongue that sips nectar from flowers. So it basically occupies the same ecological niche as a hummingbird, and looks very, very similar to a hummingbird at a distance. From 10 or more feet, you often can't tell.

This is an insect, and that's a bird. The last common ancestor was something that probably lived 500 million years ago, and certainly didn't have wings, and may not have had legs or eyes. And yet, they've independently evolved eyes and wings and all these things. So when there's selective pressure to evolve something, either a morphology or a protein structure, for example, evolution is flexible enough that it can evolve it many, many times.

So here's an example from the protein structure world. This is homophilous iron binding protein. This is just the iron coordination center. And this is now a eukaryotic protein called lactoferrin. Turns out these guys are homologous. But eukaryotes and bacteria diverged 2 million years ago or so, so their ancestry is very, very ancient.

And yet, you can see that in this iron coordination center, you have a tyrosine pointing into the iron here. And you have a histidine up here, and so forth. So the geometry has been highly conserved. It's not perfectly conserved. Like, here you have a carboxylate. And here you have a phosphate.

So there's been a little bit of change. But overall, this way of coordinating iron has basically evolved independently. So although these are homologous, the last common ancestor bound anions-- that's known from [INAUDIBLE] construction. So they independently evolved the ability to bind cations, like iron.

And here is actually my favorite example. So here's a protein called ribosome recycling factor. And that's its shape. So it's a very unusual shaped protein that's kind of shaped like an L.

Does this remind anyone of anything, this particular shape? Have you seen this in another biomolecule at some point?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Something like [INAUDIBLE]. OK, could be. Any other guesses? How about this? That's a tRNA. So the 3D structure of tRNA is almost identical, both in terms of the overall shape and in terms of the geometry. Sorry, I'm having issues with my animations here.

The geometry of these, they're both about 70 angstroms long. So why is that? Why would this protein evolve to have the same three dimensional shape as a tRNA? Any ideas?

AUDIENCE: [INAUDIBLE].

PROFESSOR: [INAUDIBLE]. Right, exactly. It fits into the ribosome, and it's involved, when the ribosome is stalled, and basically releasing the ribosome. So it's mimicking a tRNA in terms of structure.

And so the point about this is that, if you were to take a bunch of biomolecules and match them up using a structure comparison algorithm to find similar ones-- these two are clearly similar. And yet, they probably never had a common ancestor right, because one's an RNA in one's a protein.

OK. So now what we're going to talk about a few different types of alignments. So we talked about local alignments, where you don't try to align the entire sequence of your query or your database. You just find smaller regions of high similarity. Global alignment, where you try to align the two proteins from end to end, you assume that these two proteins are homologous, and actually that they haven't had major insertions or rearrangements of their sequence. And then semi-global, which is sort of a little twist on global.

And we'll talk about a few different scoring systems-- so ungapped, which we've been talking about until now, and then we'll introduce gaps of two types that are called linear and affine. And the nomenclature is a little bit confusing, as you'll see. They're both linear, in a sense.

So a common way to represent sequence alignments, especially in the protein alignment-- you can do it for protein or DNA-- is what's called a dot matrix. Now we've got two proteins. They might be 500 amino acids long each, let's say. You write sequence one along the x-axis, sequence two along the y-axis. And then you make a dot in this matrix whenever they have identical residues, although probably there would be a lot more dots in this.

So let's say, whenever you have three residues in a row that are identical-- OK, that's going to occur fairly rarely, since there's 20 amino acids. And you make that dot. And for these two proteins, you don't get any off diagonal dots. You just get these three diagonal lines here. So what does that tell you about the history of these two proteins? What's that right there? Sally.

AUDIENCE: An insertion or deletion.

PROFESSOR: An insertion or deletion. An insertion in which protein?

AUDIENCE: Sequence two.

PROFESSOR: Or a deletion in?

AUDIENCE: One.

PROFESSOR: OK. Everyone got that? OK, good. There's extra sequence in sequence two here that's not in sequence one. You don't know whether it's an insertion or deletion. It could be either one, based on this information. Sometimes you can figure that out from other information. So sometimes you call that an indel-- insertion or deletion.

And then, what is this down here? Someone else? Insertion, I heard, insertion in sequence one or deletion in sequence two. OK, good. All right, so what type of alignment would be most appropriate for this pair sequences, a local or a global?

AUDIENCE: I would do global, because they're very, very similar. [INAUDIBLE].

PROFESSOR: Yeah. They are quite similar across their entire lengths, just with these two major indels. So that's sort of the classical case where you want to do the global alignment.

All right. So what about these two proteins? Based on this dot matrix, what can you say about the relation between these two, and what type of alignment would you want to use when comparing these two proteins? Yeah, what's your name?

AUDIENCE: Sonia.

PROFESSOR: Go ahead, Sonia.

AUDIENCE: It looks like they've got similar domains, maybe. So local alignment might be better.

PROFESSOR: And why wouldn't you do a global alignment?

AUDIENCE: Local, because the local alignment might actually find those domains and tell you what they are.

PROFESSOR: So a local alignment should at least find these two guys here. And why do these two parallel diagonal lines, what does that tell you?

AUDIENCE: That the two different proteins have similar sequences, just in different parts of the protein, different areas relative to the start.

PROFESSOR: Right. Yeah, go ahead.

AUDIENCE: Doesn't it just basically mean that there's a section in sequence two that's in sequence one twice?

PROFESSOR: Yeah, exactly. So this segment of sequence two, here-- sorry, having trouble, there we go, that apart-- is present twice in sequence one. It's present once from about here over to here, and then it's present once from here over to here. So it's repeated.

So repeats and things like that will confuse your global alignment. The global alignment needs to align each residue-- or trying to align each residue in protein one to each residue in protein two. And here, it's ambiguous. It's not clear which part of sequence one to align to that part of sequence two.

So it'll get confused. It'll choose one or the other. But that may be wrong, and that really doesn't capture what actually happens. So yeah, so here a local alignment would be more suitable. Good.

So let's talk now about gaps, again, which can be called indels. In protein sequence alignments, or DNA, which many of you have probably seen, you often use maybe just a dash to represent a gap. So in this alignment here, you can see that's kind of a reasonable alignment, right? You've got pretty good matching on both sides.

But there's nothing in the second sequence that matches the RG in the first sequence. So that would be a reasonable alignment of those two. And so what's often used is what's called a linear gap penalty.

So if you have end gaps, like in this case two, you assign a gap penalty A, let's say. And A is a negative number. And then you can just run the same kinds of algorithms, where you add up matches, penalize mismatches, but then you have an additional penalty you apply when you introduce a gap.

And typically, the gap penalty is more severe than your average mismatch. But there's really no theory that says exactly how the gap penalty should be chosen. But empirically, in cases where you should know the answer, where you have, for example, a structural alignment, you can often find that a gap penalty that's bigger than your average mismatch penalty is usually the right thing to do.

So why would that be? Why would a gap penalty-- why would you want to set it larger than a typical mismatch? Any ideas? Yeah, what's your name?

AUDIENCE: I'm Chris.

PROFESSOR: Chris.

AUDIENCE: Because having mutations that shift the frame or that one insert would have insertions or deletions is far more uncommon than just having changing [INAUDIBLE].

PROFESSOR: Mutations that create insertions and deletions are less common than those that introduce substitutions of residues. Everyone got that? That's true. And do you know by what factor?

AUDIENCE: Oh, I couldn't give you a number.

PROFESSOR: So I mean, this varies by organism, and It varies by what type of insertion you're looking at. But even at the single nucleotide level, having insertions is about an order of magnitude less common than having a substitution in those lineages.

And here, in order to get an amino acid insertion, you actually have to have a triplet insertion, three or six or some multiple of three into the exon. And that's quite a bit less common. So they occur less commonly. A mutation occurs less commonly, and therefore the mutation is actually accepted by evolution even less commonly.

And an alternative is a so-called affine gap penalty, which is defined as G plus n lambda. So n is the number of gaps, and then G is what's called a gap opening penalty. So the idea here is that basically the gaps tend to cluster.

So having an insertion is a rare thing. You penalize that with G. But then, if you're going to have an insertion, sometimes you'll have a big insertion of two or three or four codons. A four codon insertion should not be penalized twice as much as a two codon insertion, because only one gap actually occurred. And when you have this insertion event, it can any variety of sizes.

You still penalize more for a bigger gap than for a smaller gap, but it's no longer linear. I mean, it's still a linear function, just with this constant thing added. So these are the two common types of gap penalties that you'll see in the literature.

The affine works a little bit better, but it is a little bit more complicated to implement. So sometimes you'll see both of them used in practice. And then, of course, by changing your definition of gamma, you could have a G plus n minus 1. So that first gap would be G, and then all the subsequent gaps would gamma. So you're not going to have to double score something.

All right. OK. You've got two proteins. How do you actually find the optimal global alignment? Any ideas on how to do this?

So we can write one sequence down one axis, one down the other axis. We can make this dot plot. The dot plot can give us some ideas about what's going on. But how do we actually find the optimal one where we want to start from the beginning? In the end, we're going to write the two sequences one above the other.

And if the first residue or the first sequence is n, and maybe we'll align it to here, then we have to write the entire sequence here all the way down to the end. And below it has to be either a residue in sequence two or a gap. And again, we can have gaps up here. So you have to do something. You have to make it all the way from the beginning to the end. And we're just going to sum the scores of all the matching residues, of all the mismatching residues, and of all the gaps. How do we find that alignment? Chris.

AUDIENCE: Well, since we're using dynamic programming, I'm guessing that you're going to have to fill out a matrix of some sort and backtrack.

PROFESSOR: And so when you see the term dynamic programming, what does that mean to you?

AUDIENCE: You're going to find solutions to sub problems until you find a smaller solution. Then you'd backtrack through what you've solved so far to find the global sequence.

PROFESSOR: Good. That's a good way of describing it. So what smaller problems are you going to break this large problem into?

AUDIENCE: The smaller sub-sequences.

PROFESSOR: Which smaller sub-sequences? Anyone else? You are definitely on the right track here. Go ahead.

AUDIENCE: I mean, it says at the top there, one sequence across the top and one down the side. You could start with just the gap versus the sequence and say your gap will increase as you move across. Basically, each cell there could be filled out with information from some of its neighbors. So you want to make sure that you fill out old cells in some order so that we can proceed to the next level with what we've [? written down. ?]

PROFESSOR: So if you had precisely to find a sub problem where you could see what the answer is, and then a slightly larger sub problem whose solution would build on the solution that first one, where would you start? What would be your smallest sub problem?

AUDIENCE: I'd start with the top row, because you could just the gap versus gap, and then move in the row, because you don't need anything above that.

PROFESSOR: And then what's the smallest actual problem where you actually have parts of the protein aligned?

AUDIENCE: One row in column two, basically. If it's a match, you have some score. And if it's a mismatch, you have some other score. And you want the best possible one in each block.

PROFESSOR: Yeah, OK. Yeah. That's good. So just to generalize this-- hopefully this is blank-- in general, you could think about we've got, let's say, 1 to n here, and a sequence 1 to n here. You could think about a position i here and a position j here. And we could say finding the optimal global alignment, that's a big problem. That's complicated.

But finding an alignment of just the sequence from 1 to i in the first protein against the sequence from 1 to j in the second protein, that could be pretty easy. If i is 2 and j is 2, you've got a dipeptide against a dipeptide. You could actually try all combinations and get the optimal alignment there.

And so the idea, then, is if you can record those optimal scores here in this matrix, then you could build out, for example, like this, and find the optimal alignments of increasingly bigger sub problems where you add another residue in each direction, for example. Does that makes sense to you?

The idea of a dynamic programming algorithm is it's a form of recursive optimization. So you first optimize something small, and then you optimize something bigger using the solution you got from that smaller piece. And the way that that's done for protein sequences in Neeleman-Wunsch is to, as we were saying, first consider that there might be a gap in one aligning to a residue in the other. So we need to put these gaps down across the top and down the side.

This is a linear gap penalty, for example. And so here would be how you start. And this is a gap penalty, obviously, of minus 8. So if you're the optimal solution that begins with this V in the top sequence aligned to this gap in the vertical sequence, there's one gap there, so it's minus 8. And then if you want to start with two gaps against this V and D, then it's minus 16.

So that's how you would start it. So you start with these problems where there's no options. If you have two gaps against two residues, that's minus 16. By our scoring system, it's unambiguous. So you just can fill those in.

And then you can start thinking about, what do we put right here? What score should we put right there? Remember, we're defining the entries in this matrix as the optimal score of the sub-sequence of the top protein up to position i against the vertical protein up to position j. So that would be the top protein position one up to the vertical protein position one. What score would that be? What's the optical alignment there that ends position one both sequences?

It'll depend on your scoring system. But for a reasonable scoring system, that's a match. That's going to get some positive score. That's going to be better than anything involving a gap in one against a gap in the other or something crazy like that. So that's going to get whatever your VV match score is. This is your Sij from your scoring matrix for your different amino acids.

And then, basically the way that this is done is to consider that when you're matching that position one against position one, you might have come from a gap before in one sequence or a gap in the other sequence, or from a match position in the other sequence. And that leads to these three arrows.

I think it gets clear if I write up the whole algorithm here. So Sij is the score of the optimal alignment ending at position i in sequence one and position j in sequence two. Requires that we know what's above, to the left, and diagonally above. And you solve it from the top and left down to the bottom and right, which is often called dynamic programming. And let's just look at what the recursion is.

So Needleman and Wunsch basically observed that you could find this optimal global alignment score by filling in the matrix by at each point taking the maximum of these three scores here. So you take the maximum of the score that you had above and to the left, so diagonally above, plus sigma of xi yj. Sigma, in this case, is the scoring matrix that you're using that's 20 by 20 that scores each amino acid against each other amino acid residue.

You add that score if you're going to move diagonally to whatever the optimal score was there, or if you're moving to the right or down, you're adding a gap in one sequence or the other. So you have to add A, which is this gap penalty, which is a negative number, to whatever the optimal alignment was before.

I think it's maybe easier if we do an example here. So here is the PAM250 scoring matrix. So this was actually developed by Dayhoff back in the '70s. This might be an updated version, but it's more or less the same as the original. Notice, it's a triangular matrix. Why is that?

AUDIENCE: It's symmetric.

PROFESSOR: It's symmetric, right. So it has a diagonal. But then everything below the diagonal, it would be mirrored above the diagonal, because it's symmetric. Because you don't know when you see a valine matched to a leucine, it's the same as a leucine matched to a valine, because it's a symmetrical definition of scoring.

And here are two relevant scores. So notice that VV has a score of plus 4 in this matrix. And over here, VD has a score of minus 2. So I'll just write those down.

Anyone notice anything else interesting about this matrix? We haven't said exactly where it comes from, but we're going to. Yeah, what's your name?

AUDIENCE: Michael.

PROFESSOR: Go ahead.

AUDIENCE: Not all the diagonal values are the same.

PROFESSOR: Not all diagonals are the same. In fact, there's a pretty big range, from 2 up to 17, so a big range. And anything else? OK, I'm sorry, go ahead. What's your name?

AUDIENCE: Tagius.

PROFESSOR: Tagius, yeah. Go ahead.

AUDIENCE: There are positive values for things that are not the same?

PROFESSOR: Yeah. So all the diagonal terms are positive. So a match of any particular residue type to its identical residue is always scored positively, but with varying scores. And there are also some positive scores in the off diagonal. And where are those positive scores occurring?

Notice they tend to be to nearby residues. And notice the order of residues is not alphabetical. So someone who knows a lot about amino acids, what can you see about these scores? Yeah, go ahead.

AUDIENCE: I think these amino acids [INAUDIBLE] based on their [INAUDIBLE].

PROFESSOR: So the comment was that the residues have been grouped by similar chemistry of their side chains. And that's exactly right. So the basic residues, histidine, arginine, and lysine, are all together. The acidic residues, aspartate and glutamate, are here, along with asparagine and glutamine.

And notice that D to E has a positive score here. It's 3. It's almost as good as D to D or E to E, which are plus 4. So recognizing that you can often substitute in evolution an aspartate for a glutamate.

So yeah, so it basically, to some extent, is scoring for similar chemistry. But that doesn't explain why, on the diagonal, you have such a large range of values. Why is a tryptophan more like a tryptophan than a serine is like a serine. Tim, you want to comment?

AUDIENCE: Perhaps it's because tryptophans occur very rarely in all proteins [INAUDIBLE]. So if you've got two [INAUDIBLE], that's a lot rarer of an occurrence and [INAUDIBLE].

PROFESSOR: So Tim's point was that tryptophans occur rarely, so when you see two tryptophans aligned, you should take note of it. It can anchor your alignment. You can be more confident in that. Sally.

AUDIENCE: Well, tryptophans are also incredibly bulky, and also have the ability to make electric interactions, electro-static interactions.

PROFESSOR: Not really electro-static, you would say, more--

[INTERPOSING VOICES]

AUDIENCE: Yes. But they do have a lot of abilities to interact with other side chains. And cysteines contribute very, very strongly to the three dimensional structure of the protein.

PROFESSOR: Why is that?

AUDIENCE: Well, because they can form [INAUDIBLE].

PROFESSOR: OK. Yeah. So maybe you don't put your tryptophans and your cysteines into your protein by chance, or you only put them when you want them, when there's enough space for a tryptophan. And when you substitute something smaller, it leaves a gap. It leaves a 3D spatial gap. And so you don't want that. You don't get good packing.

When you have cysteines, they form disulfide bonds. If you change it to something that's non-cysteine, it can form that anymore. That could be disruptive to the overall fold. So those ones tend to be more conserved in protein sequence alignments, absolutely.

Whereas, for example, if you look at these hydrophobics, the MILV group down here, they all have positive scores relative to each other. And that says that, basically, most the time when those are used-- I mean, there are sometimes when it went really matters. But a lot of time, if you just want a transmembrane segment, you can often substitute any one of those at several positions and it'll work equally well as a transmembrane segment.

So these are not random at all. There's some patterns here. So let's go back to this algorithm. So now, if we're going to implement this recursion, so we fill on the top row and the left column, and then we need to fill in this first. I would argue the first interesting place in the matrix is right here.

And we consider adding a gap here. When you move vertically or horizontally, you're not adding a match or adding a match. So from this position, this is sort of the beginning point. It doesn't actually correspond to a particular position in the protein. We're going to add now the score for VV.

And we said that VV, you look it up in that PAM matrix, and it's plus 4. So we're going to add 4 there to 0. And so that's clearly bigger than minus 16, which is what you get from coming above or coming from the left. So you put in the 4.

And then you also, in addition to putting that 4 there, you also keep the arrow. So there's that red arrow. We remember where we came from in this algorithm.

Because someone said something about backtracking-- I think Chris-- so that's going to be relevant later.

So we basically get rid of those two dotted arrows and just keep that red arrow as well as the score. And then we fill in the next position here. And so to fill in this, now we're considering going to the second position in sequence one, but we're still only at the first position in sequence two. So if we match V to V, then we'd have to add, basically, a gap in one of the sequences. Basically it would be a gap in sequence two. And that's going to be minus 8.

So you take 4, and then plus minus 8, so it's negative 4. Or you could do minus 8 and then plus negative 2, if you want to start from a gap and then add a DV mismatch there, because minus 2 was the score for a DV mismatch. Or again, you can start from a gap and then add another gap.

OK, does that make sense? So what is the maximum going to be? Negative 4. And the arrow is going to be horizontal, because we got some bonus points for that VV match, and now it's carrying over. We're negative, but that's OK. We're going to just keep the maximum, whatever it is.

All right, so it's minus 4, and the horizontal arrow. And then here's the entire matrix filled out. And you'll have a chance to do this for yourself on problem set one. And I've also filled in arrows. I haven't filled in all the arrows, because it gets kind of cluttered. But all the relevant arrows here are filled in, as well as some irrelevant arrows.

And so then, once I fill this in, what do I do with this information? How do I get an actual alignment out of this matrix? Any ideas? Yeah, what's your name?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah, so what he said is start at the bottom right corner and go backwards following the red arrows in reverse. Is that right? So why the bottom right corner? What's special about that?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah. It's a score of the optimal alignment of the entire sequence one against the entire sequence two. So that's the answer. That's what we define as the optimal global alignment.

And then you want to know how you got there. And so how did we get there? So the fact that there's this red arrow here, what does that red arrow correspond to specifically?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Right. In this particular case, for this particular red arrow, remember the diagonals are matches. So what match is that?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah, that's a Y to Y match, right? Can everyone see that? We added Y to Y, which was plus 10, to whatever this 13 was and got 23. OK, so now we go back to here. And then how do we get here? We came from up here by going this diagonal arrow.

What is that? What match was that? That's a cysteine-cysteine match. And then how do we get to this 1? We came vertically. And so what does that mean?

AUDIENCE: [INAUDIBLE].

PROFESSOR: We inserted a gap, in which sequence? The first one. The second one? What do people think? Moving down.

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah, the top one. And so that got us to here. Here's a match, plus 2 for having a serine-serine match. Here's a plus 3 for having a D to E mismatch. But remember, that's those are chemically similar, so they get a positive score. And then this is the V to V.

So can you see? I think I have the optimal alignment written somewhere here, hopefully, there. That's called the trace back. And then that is the alignment.

OK, we align the Y to the Y, the C to the C. Then we have basically a gap in this top sequence-- that's that purple dash there-- that's corresponding to that L. And you can see why we wanted to put that gap in there, because we want these S's to match, and we want the C's to match. And the only way to connect those is to have a gap in the purple.

And the purple was shorter than the green sequence anyway, so we kind of knew that there was going to be a gap somewhere. And good. And that's the optimal alignment. So that's just some philosophy on Needlemen-Wunsch alignments.

So what is semi-global alignment? You don't see that that commonly. It's not that big a deal. I don't want to spend too much time on it. But it is actually reasonable a lot of times that, let's say you have a protein that has a particular enzymatic activity, and you may find that the whole, the bulk of the protein is well conserved across species.

But then at the N and C termini, there's a little bit of flutter. You can add a few residues or delete a few residues, and not much matters at the N and C termini. Or it may matter not for the structure but for, you know, you're adding a single peptide so it'll be secreted, or you're adding some localization signal. You're adding some little thing that isn't necessarily conserved.

And so a semi-global alignment, where you use the same algorithm, except that you initialize the edges of the dynamic programming matrix to 0, instead of the minus 8, minus 16 whole gap, and go to 0. So we're not going to penalize for gaps of the edges.

And then, instead of requiring the trace back to begin at the bottom right, Smn, you allow it to begin at the highest score in the bottom row or the rightmost column. And when you do the trace back as before, these two changes basically find the optimal global alignment but allowing arbitrary numbers of gaps at the ends and just finding the core match.

It has to go basically to the end of one or the other sequences, but then you can have other residues hanging off the end on the other sequence, if you want, with no penalty. And this sometimes will give a better answer, so it's worth knowing about. And it's quite easy to implement.

Now what about gapped local alignments? So what if you have two proteins? Do you remember those two proteins where we had the two diagonal? I guess they were diagonal lines. How where they? Something like that. Anyway, diagonal lines like that.

So where in this protein on the vertical, there is a sequence here that matches two segments of the horizontal protein. So for those two, you don't want to do this global alignment. It'll get confused. It doesn't know whether to match this guy to this or this other one to the sequence. So you want to use a local alignment. So how do we modify this Needleman-Wunsch algorithm to do local alignment? Any ideas? It's not super hard. Yeah, go ahead.

AUDIENCE: If the score is going to go negative, instead of putting a negative score, you just put 0, and you start from where you get the highest total score, rather than the last column or last row. Start your traceback from the highest score.

PROFESSOR: So whenever you're going negative, you reset to 0. Now what does that remind you of? That's the same trick we did write previously with ungapped local alignment. So you reset to 0. And that's as a no penalty, because if you're going negative, it's better just to throw that stuff away and start over. We can do that because we're doing local alignment. We don't have to align the whole thing. So that's allowed.

And then, rather than going to the bottom right corner, you can be anywhere in the matrix. You look for that highest score and then do the traceback. That's exactly right. So it's not that different.

There are a few constraints, though, now on the scoring system. So if you think about the Needleman-Wunsch algorithm, we could actually use a matrix that had all positive scores. You could take the PAM250 matrix. And let's say the most negative score there is, I don't know, like minus 10 or something, and you could just add 10, or even add 20 to all those score. So they're all positive now. And you could still run that algorithm. And it would still produce more or less sensible results.

I mean, they t wouldn't be as good as the real PAM250, but you would still get a coherent alignment out of the other end. But that is no longer true when you talk about the Smith-Waterman algorithm, for the same reason that an ungapped local alignment, we had to require that the expected score be negative, because you have to have this negative drift to find small regions that go in the positive.

So if you have this rule, this kind of permissive that says, whenever we go negative we can just reset to 0, then you have to have this negative drift in order for positive scoring stuff to be unusual. All right, so that's another constraint there. You have to have negative values for mismatches-- I mean, not all mismatches. But if you took two random residues in alignment, the average score has to be negative. I should probably rephrase that, but more or less.

And here's an example of Smith-Waterman. So you right zeroes down the left side and across the top. And that's because, remember, if you go negative, you reset to 0. So we're doing that.

And then you take the maximum of four things. So coming from the diagonal and adding the score of the match, that's the same as before. Coming from the left and adding a gap in one sequence, coming from above and adding a gap in the other sequence, or 0. This "or 0" business allows us to reset to 0 if we ever go negative.

And when you have a 0, you still keep track of these arrows. But when you have a 0, there's no arrow. You're starting it. You're starting the alignment right there. So that's Smith-Waterman.

It's helpful. I think on problem set one, you'll have some experience thinking about both Needleman-Wunsch and Smith-Waterman. They sort of behave a little bit differently, but they're highly related. So it's important to understand how they're similar, how they're different.

And what I want to focus on for the remainder of this lecture is just introducing the concept of amino acid similarity matrices. We saw that PAM matrix, but where does it come from? And what does it mean? And does it work well or not, and are there alternatives?

So we could use this identity matrix. But as we've heard, there are a number of reasons why that may not be optimal. For example, the cysteines, we should surely score them more, because they're often involved in disulfide bonds, and those have major structural effects on the protein and are likely to be conserved more than your average leucine or alanine or whatever.

So clearly, scoring system should favor matching identical or related amino acids, penalize poor matches and for gaps. And there's also an argument that can be made that it should have to do with how often one residue is substituted for another during evolution. So that commonly substituted thing should have either positive scores or less negative scores than rarely substituted things.

And perhaps not totally obvious, but it is if you think about it for a while, is that any scoring system that you dream up carries with it an implicit model of molecular evolution for how often things are going to be substituted for each other. So it's going to turn out that the score is roughly proportional to a [INAUDIBLE] score for the occurrence of that pair of residues, divided by how often it would occur by chance, something like that.

And so that if you assign positive scores to things, to certain pairs of residues, you're basically implying that those things will commonly interchange during evolution. And so if you want to have realistic, useful scores, it helps to think about what the implicit evolutionary model is and whether that is a realistic model for how proteins evolve.

So I'm going to come to Dayhoff. And so unlike later matrices, she had an explicit evolutionary model, like an actual mathematical model, for how proteins evolve. And the idea was that there are going to be alignments of some proteins. And keep in mind, this was in 1978. So the protein database probably had like 1,000 proteins in it, or something. It was very, very small.

But there were some alignments that were obvious. If you see two protein segments of 50 residues long that are 85 identical, there's no way that occurred by chance. You don't even need to do statistics on that. So you're sure.

So she took these very high confidence protein sequence alignments, and she calculated the actual residue residue substitution frequencies, how often we have a valine in one sequence as a substitute for a leucine. And it's actually assumed it's symmetrical. Again, you don't know the direction. And calculated these substitution frequencies.

Basically estimated what she called a PAM1 one matrix, which is a matrix that implies 1% divergence between proteins. So there's, on average, only a 1% chance any given residue will change. And the real alignments had greater divergence than that. They had something like 15% divergence.

But you can look at those frequencies and reduce them by a factor of 15, and you'll get not exactly 15 but something like 15. And you'll get something where there's a 1% chance of substitution. And then once you have that model for what 1% sequence substitution looks like, turns out you can just represent that as a matrix and multiply it up to get a matrix that describes what 5% sequence substitution looks like, or 10% or 50% or 250%.

So that PAM250 matrix that we talked about before, that's a model for what 250% amino acid substitution looks like. How does that even make sense? How can you have more than 100%? Is anyone with me on this? Tim, yeah.

AUDIENCE: Because it can go back. So it's more likely, in some cases, that you revert rather than [INAUDIBLE].

PROFESSOR: Right. So a PAM10 matrix means, on average, 10% of the residues have changed. But a few of those residues might have actually-- so maybe about 90% won't have changed at all. Some will have changed once, but some might have even changed twice, even at 10%.

And when you get to 250%, on average, every residue has changed 2 and 1/2 times. But again, a few residues might have remained the same. And some residues that change-- for example, if you had an isoleucine that mutated to a valine, it might have actually changed back already in that time. So it basically accounts for all those sorts of things. And if you have commonly substituted residues, you get that type of evolution happening.

All right. So she took these protein sequence alignments-- it looks something like this-- and calculated these statistics. Again, I don't want to go through this in detail during lecture, because it's very well described in the text. But what I do want to do is introduce this concept of a Markov chain, because it's sort of what is underlying these Dayhoff matrices. So let's think about it.

We'll do more on this next time. But imagine that you were able to sequence the genomes of cartoon characters with some newly developed technology and you chose to analyze the complicated genetics of the Simpson lineage. I'm assuming you all know these people. This is Grandpa and Homer eating doughnut and his son, Bart.

So imagine this is Grandpa's genome at the apolipoprotein A locus. And a mutation occurred that he then passed on to Homer. So this mutation occurred in the germ line, passed on to Homer. And then when Homer passed on his genes to Bart, another mutation occurred here, changing this AT pair to a GC pair in Bart.

So this, I would argue, is a type of Markov chain. So what is a Markov chain? So it's just a stochastic process. So a stochastic process is a random process, is sort of the general meaning. But here we're going to be dealing with discrete stochastic processes, which is just a sequence of random variables.

So X1 here is a random variable that represents, for example, the genome of an individual, or it could represent the genotype, in this case, at a particular position, maybe whether it's an A, C, G, or T at one particular position in the genome. And now the index here-- one, two, three, and so forth-- is going to represent time.

So X1 might be the genotype in Grandpa Simpson at a particular position. And X2 might be the genotype of Homer Simpson. And X3 would be the genotype in the next generation, which would be Bart Simpson. And what a Markov chain is is it's a particular type of stochastic process that arises commonly in natural sciences, really, and other places all over the place.

So it's a good one to know that has what's called the Markov property. And that says that the probability that the next random variable, or the genotype at the next generation, if you will-- so Xn plus 1 equals some value, j, which could be any of the possible values, say any of the four bases, conditional on the values of X1 through Xn, that is the entire history of the process up to that time, is equal to the conditional probability that Xn plus 1 equals j given only that little xn equals some particular value.

So basically what it says that if I tell you what Homer's genotype was at this locus, and I tell you what Grandpa Simpson's genotype was at that locus, you can just ignore Grandpa Simpson's. That's irrelevant. It only matters what Homer's genotype was for the purpose of predicting Bart's genotype. Does that make sense?

So it really doesn't matter whether that base in Homer's genome was the same as it was in Grandpa Simpson's genome, or whether it was a mutation that's specific to Homer, because Homer is the one who passes on DNA to Bart. Does that make sense?

So you only look back one generation. It's a type of memoryless process, that you only remember the last generation. That's the only thing that's relevant. And so to understand Markov chains, it's very important that you all review your conditional probability.

So we're going to do a little bit more on Markov chains next time. P A given B, what does that mean? If you don't remember, look it up in the Probability and Statistics, because that's sort of essential to Markov chains.

So next time we're going to talk about comparative genomics, which will involve some applications of some of the alignment methods that we've been talking about. And I may post some examples of interesting comparative genomic research papers, which are going to be optional reading. You may get a little more out of the lecture if you read them, but it's not essential.