**PROFESSOR:** All right. So today, we're going to briefly review classical sequencing and next-gen or second-gen sequencing, which sort of provides a lot of the data that the analytical methods we'll be talking about work on. And we'll then introduce local alignment a la BLAST and some of the statistics associated with that.

So just a few brief items on topic one. All right. So today, we're going to talk about sequencing first. Conventional-- or Sanger sequencing-- then next-gen or second-gen sequencing briefly. And then talk about local alignments.

So background for the sequencing part, the Metzger review covers everything you'll need. And for the alignment-- we'll talk about local alignment today, global alignment on Tuesday-- then chapters four and five of the text cover it pretty well. So here's the text. If you haven't decided whether to get it or not, I'll have it up here. You can come flip through it after class.

Sequencing is mostly done at the level of DNA. Whether the original material was RNA or not, usually convert to DNA and sequence at the DNA level. So we'll often think about DNA as sort of a string. But it's important to remember that it actually has a three dimensional structure as shown here. And often, it's helpful to think of it in sort of a two dimensional representation where you think about the bases and their hydrogen bonding and so forth as shown in the middle.

My mouse is not working today for some reason, but hopefully, we won't need it.

So the chemistry of sequencing is very closely related to the chemistry of the individual bases. And there are really three main types that are going to be relevant here. Ribonucleotides, deoxyribonucleotides, then for Sanger sequencing, dideoxyribonucleotides.

So who can tell me which of these structures corresponds to which of those names? And also, please let me know your name and I'll attempt to remember some of your names toward the end of the semester probably. So, which are which? Yes, what's your name?

**AUDIENCE:** I'm Simona. So the ribonucleotide is the top right. The deoxy is the one below it. And the dideoxy is the one to the left.

**PROFESSOR:** OK, so that is correct. So one way to keep these things in mind is the numbering of the bases. So the carbons in the ribo sugar are numbered one, so carbon 1 is the one where the base is attached. Two is here, which has an OH in RNA and just an H in DNA. And then three is very important. Four, and then five. So five connects to the phosphates, which then will connect the base to the sugar phosphate backbone. And three is where you extend. That's where you're going to add the next base in a growing chain.

And so what will happen if you give DNA polymerase a template and some dideoxy nucleotides? It won't be able to extend because there's no 3-prime OH. And all the chemistry requires the OH. And so that's the basis of classical or Sanger sequencing, which Fred Sanger got the Nobel Prize for in the 1980s-- I think it was developed in the '70s-- and it's really the basis of most of the sequencing, or pretty much all the DNA sequencing up until the early 2000s before some newer technologies came about. And it takes advantage of this special property of dideoxy nucleotides that they terminate the growing chain.

So imagine we have a template DNA. So this is the molecule whose sequence we want to determine shown there in black. We then have a primer. And notice the primer's written in 5-prime to 3-prime direction. The ends would be primer sequences and then primer complimentary sequences in the template. So you typically will have your template cloned-- this is in conventional sequencing-- cloned into some vector like a phage vector for sequencing so you know the flanking sequences.

And then you do four sequencing reactions in conventional Sanger sequencing. And I know some of you have probably had this before. So let's take the first chemical reaction. The one here with a DDGTP. So what would you put in that reaction? What are all the components of that reaction if you wanted to do conventional sequencing on, say, an acrylonitrile? Anyone? What do you need and what does it accomplish? Yeah, what's your name?

**AUDIENCE:** I'm Tim.

**PROFESSOR:** Tim? Oh yeah, I know you, Tim. OK, go ahead.

**AUDIENCE:** So you need the four nucleotides-- the deoxynucleotides. You will need the dideoxy P nucleotides. In addition, you need all the other [INAUDIBLE]. You need polymerase. Generally, you need a buffer of some sort, [INAUDIBLE], to [INAUDIBLE].

**PROFESSOR:** Yeah, primary template. Yeah. Great. That's good. It sounds like Tim could actually do this experiment. And what ratio would you put in? So you said you're going to put in all four conventional deoxynucleotides and then one dideoxynucleotide. So let's say dideoxy G just for simplicity here. So in what ratio would you put the dideoxynucleotide compared to the conventional nucleotides?

**AUDIENCE:** To lower the concentration.

**PROFESSOR:** Lower? Like how much lower?

**AUDIENCE:** Like, a lot lower.

**PROFESSOR:** Like maybe 1%?

**AUDIENCE:** Yeah.

**PROFESSOR:** Something like that. You want to put it a lot lower. And why is that so important?

**AUDIENCE:** Because you want the thing to be able to progress. Because you need enough of the ribonucleotide concentration so that [INAUDIBLE] every [INAUDIBLE] equivalent

or excess and you're going to terminate [INAUDIBLE].

**PROFESSOR:** Right. So if you put equamolar deoxy G and dideoxy G, then it's going to be a 50% chance of terminating every time you hit a C in the template. So you're going to have half as much of the material at the second G, and a quarter as much as the third, and you're going to have vanishingly small amounts. So you're only going to be able to sequence the first few C's in the template. Exactly. So that's a very good point.

So now let's imagine you do these four separate reactions. You typically would have radiolabeled primer so you can see your DNA. And then you would run it on some sort of gel. This is obviously not a real gel, but an idealized version. And then in the lane where you put dideoxy G, you would see the smallest products. So you read these guys from the bottom up.

And in this lane there is a very small product that's just one base longer than the primer here. And that's because there was a C there and it terminated there. And then the next C appears several bases later. So you have sort of a gap here.

And so you can see that the first base in the template would be a complement of T, or C. And the second base would be, you can see, the next smallest product in this dideoxy T lane, therefore it would be A. And you just sort of snake your way up through the gel and read out the sequence. And this works well.

So what does it actually look like in practice? Here are some actual sequencing gels. So you run four lanes. And on big polyacrylamide gels like this. Torbin, you ever run one of these?

**AUDIENCE:** Yes.

**PROFESSOR:** Yes? They're a big pain to cast. Run for several hours, I think. And you get these banding patterns. And what limits the sequence read length? So we normally call the sequence generated from one run of a sequencer as a read. So that one attempt to sequence the template is called a read.

And you can see it's relatively easy to read the sequence toward the bottom, and then it gets harder as you go up. And so that's really what fundamentally limits the read length, is that the bands get closer and closer together. So they'll run inversely proportional to size with the small ones running faster. But then the difference between a 20 base product and a 21 might be significant. But the difference between a 500 base product and a 501 base product is going to be very small. And so you basically can't order the lanes anymore. And therefore, that's sort of what fundamentally limits it.

All right. So here we had to run four lanes of a gel. Can anyone think of a more efficient way of doing Sanger sequencing? Is there any way to do it in one lane? Yeah, what's your name?

**AUDIENCE:** Adrian. You can use four different types of the entities. Maybe like four different colors.

**AUDIENCE:** Four different colors. OK, so instead of using radio labeling on the primary, you use fluorophore on your dideoxy entities, for example. And then you can run them. Depending where that strand terminated, it'll be a different color. And you can run them all in one lane. OK, so that looks like that.

And so this was an important development called terminator sequencing in the '90s. That was the basis of the ABI 3700 machine, which was really the workhorse of genome sequencing in the late '90s and early 2000s. Really what enabled the human genome to be sequenced.

And so one of the other innovations in this technology was that instead of having a big gel, they shrunk the gel. And then they just had a reader at the bottom. So the gel was shrunk to as thin as these little capillaries. I don't know if you can see these guys. But basically it's like a little thread here. And so each one of these is effectively-- oops! Oh no. No worries, this is not valuable. Ancient technology that I got for free from somebody.

So the DNA would be loaded at the top. There would be a little gel in each of these--

it's called capillary sequencing. And then it would run out the bottom and there would be a detector which would detect the four different flours and read out the sequence.

So this basically condensed the volume needed for sequencing. Any questions about conventional sequencing? Yes?

**AUDIENCE:** Where are the [INAUDIBLE] where you'd put the fluorescent flags? Like the topic from the [INAUDIBLE]?

**PROFESSOR:** Yeah, that's a good question. I don't actually remember. I think there are different options available. And sometimes with some of these reactions, you need to use modified polymerases that can tolerate these modified nucleotides. Yeah, so I don't remember that. It's a good question. I can look that up.

So how long can a conventional sequencer go? What's the read length? Anyone know? It's about, say, 600 or so. And so that's reasonably long. How long is a typical mammalian mRNA? Maybe two, three kb? So you have in a typical exon, maybe 150 bases or so. So you have a chunk. You don't generally get full length cDNA. But you get a chunk of a cDNA that's say, three, four exons in length. And that is actually generally sufficient to uniquely identify the gene locus that that read came from.

And so that was the basis of EST sequencing-- so-called Expressed Sequence Tag sequencing. And millions of these 600 base chunks of cDNA were generated and they have been quite useful over the years.

All right. So what is next-gen sequencing? So in next-gen sequencing, you only read one base at a time. So it's often a little bit slower. But it's really massively parallel. And that's the big advantage. And it's orders of magnitude cheaper per base than conventional sequencing. Like when it first came out it, it was maybe two orders of magnitude cheaper. And now it's probably another four orders of magnitude.

So it really blows away conventional sequencing if the output that you care about is mostly proportional to number of bases sequence. If the output is proportional to the

quality of the assembly or something, then there are applications where conventional sequencing still is very useful Because the next-gen sequencing tends to be shorter. But in terms of just volume, it generates much, much more bases in one reaction.

And so the basic ideas are that you have your template DNA molecules. Now typically, tens of thousands for technologies like PacBio or hundreds of millions for technologies like Illumina that are immobilized on some sort of surface-- typically a flow cell-- and there are either single molecule methods where you have a single molecule of your template or there are methods that locally amplify your template and produce, say, hundreds of identical copies in little clusters. And then you use modified nucleotides, often with fluorophores attached, to interrogate the next base at each of your template molecules for hundreds and hundreds of millions of them.

And so there are several different technologies. We won't talk about all of them. We'll just talk about two or three that are interesting and widely used. And they differ depending on the DNA template, what types of modified nucleotides are used, and to some extent, in the imaging and the image analysis, which differs for single molecule methods, for example, compared to the ones that sequence a cluster.

So there's a table in the Metzger review. And so I've just told you that next-gen sequencing is so cheap. But then you see how much these machines cost and you could buy lots of other interesting things with that kind of money. And I also want to emphasize that that's not even the full cost. So if you were to buy an Illumina GA2-- this would be like a couple years ago when the GA2 was the state of the art-- for half a million dollars, the reagents to run that thing, if you're going to run it continuously throughout the year, the reagents to run it would be over a million. So this actually underestimates the cost.

However, the cost per base is super, super low. Because they generate so much data at once. All right, So we'll talk about a couple of these.

The first next-gen sequencing technology to be published and still used today was from 454-- now Roche-- and it was based on what's called emulsion PCR. So they

have these little beads, the little beads have adapter DNA molecules covalently attached. You incubate the beads with DNA, and you actually make an emulsion. So it's an oil water emulsion.

So each bead, which is hydrophilic, is in the little bubble of water inside oil. And the reason for that is so that you do it at a template concentration that's low enough that only a single molecule of template is associated with each bead. So the oil then provides a barrier so that the DNA can't get transferred from one bead to another. So each bead will have a unique template molecule. You do sort of a local PCR-like reaction to amplify that DNA molecule on the bead, and then you do sequencing one base at a time using a luciferase based method that I'll show you on the next slide.

So Illumina technology differs in that instead of an emulsion, you're doing it on the surface of a flow cell. Again, you start with a single molecule of template. Your flow cell has these two types of adapters covalently attached. The template anneals to one of these adapters. You extend the adapter molecule with dNTPs and polymerase. Now you have the complement of your template, your denature.

Now you have the inverse complement of your template molecule covalently attached to the cell surface. And then at the other end there's the other adapter. And so what you could do is what's called bridge amplification where that now complement of the template molecule will bridge over hybridized to the other adapter, and then you can extend that adapter. And now you've regenerated your original template. And so now you have the complementary strand, and the original strand, your denature. And then each of those molecules can undergo subsequent rounds of bridge amplification to make clusters of typically several hundred thousand molecules. Is that clear? Question. Yeah, what's your name?

**AUDIENCE:** Stephanie. How do they get the adapters onto the template molecules?

**PROFESSOR:** How do you get the adapters onto the template molecules? So that's typically by DNA ligation. So we may cover that in later steps. It depends. There's a few different protocol. So for example, if you're sequencing microRNAs, you typically

would isolate the small RNAs and use RNA litigation to get the adapters on. And then you would do an RT step to get DNA.

With most other applications like RNA-seq or genome sequencing-- so with RNA-seq, you're starting from mRNA, you typically will isolate total RNA, do poly(A) selection, you fragment your RNA to reduce the effects of secondary structure, you random prime with, like, random hexamers RT enzyme. So that'll make little bits of cDNA 200 bases long. You use second strand synthesis. Now you have double stranded cDNA fragments. And then you do, like, blunt end ligation to add the adapters. And then you denature so you have single strand.

**AUDIENCE:** I guess my question is how do you make sure that the two ends sandwiching the DNA are different as opposed to--

**PROFESSOR:** That the two ends are different. Yeah, that's a good question. I'll post some stuff about-- It's a good question. I don't want to sweep it under the rug. But I kind of want to move on. And I'll post a little bit about that.

All right so we did 454 Illumina. Helicos is sort of like Illumina sequencing except single molecule. So you have your template covalently attached to your substrate. You just anneal primer and just start sequencing it And there's major pros and cons of single molecule sequencing, which we can talk about.

And then the PacBio technology is fundamentally different in that the template is not actually covalently attached to the surface. The DNA polymerase is covalently attached to the surface and the template is sort of threaded into the polymerase. And this is a phage polymerase that's highly processive and strand displacing. And the template is often a circular molecule. And so you can actually read around the template multiple times, which turns out to be really useful in PacBio because the error rate is quite high for the sequencing.

So in the top, in the 454, you're measuring luciferase activity-- light. In Illumina, you're measuring fluorescence. Four different fluorescent tags, sort of like the four different tags we saw in Sanger sequencing. Helicose, it's single tag one base at a

time. And in PacBio, you actually have a fluorescently labeled dNTP that has the label on-- it's actually hexaphosphate-- it's got the label on the sixth phosphate.

So the dNTP is labeled. It enters the active site of the DNA polymerase. And the residence time is much longer if the base is actually going to get incorporated into that growing chain. And so you measure how much time you have a fluorescent signal. And if it's long, that means that that base must have incorporated into the DNA.

But then, the extension reaction itself will cleave off the last five phosphates and the fluorophore tag. And so you'll regenerate native DNA. So that's another difference. Whereas in Illumina sequencing, as we'll see, there's this reversible terminator chemistry. So the DNA is not native that you're synthesizing.

So this is just a little bit more on 454. Just some pretty pictures. I think I described that before. The key chemistry here is that you add one dNTP at a time. So only a subset of the wells-- perhaps a quarter of them-- that have that next base, the complementary base free-- as the next one after the primer-- will undergo synthesis. And when they undergo synthesis, you release pyrophosphate.

And they have these enzymes attached to these little micro beads-- the orange beads-- sulfurylase and luciferase, that use pyrophosphate to basically generate light. And so then you have one of these beads in each well. You look at which wells lit up when we added dCTP. And they must have had G as the next base and so forth.

And there's no termination here. The only termination is because you're only adding one base at a time. So if you have a single gene in the template, you'll add one base. But if you have two Gs in the template, you'll add two Cs. And in principle, you'll get twice as much light.

But then you have to sort of do some analysis after the fact to say, OK how much light do we have? And was that one G, two G, and so forth. And the amount of light is supposed to be linear up to about five or six Gs. But that's still a more error-prone

step. And the most common type of error in 454 is actually insertions and deletions. Whereas in Illumina sequencing, it's substitutions.

David actually encouraged me to talk more about sequencing errors and quality scores. And I need to do a little bit more background. But I may add that a little bit later in the semester.

OK, so in Illumina sequencing, you add all four dNTPs at the same time. But they're non-native. They have two major modifications. So one is that they're three prime blocked. That means that the OH is not free, I'll show the chemical structure in a moment.

So you can't extend more than one base. You incorporate that one base, and the polymerase can't do anything more. And they're also tagged with four different fluors. So you add all four dNTPs at once. You let the polymerase incorporate them. And then you image the whole flow cell using two lasers and two filters.

So basically, to image the four fluors. So you have to sort of take four different pictures of each portion of the flow cell and then the camera moves and you scan the whole cell. And so then, those clusters that incorporated a C, let's say, they will show up in the green channel as spots. And those incorporated in A, and so forth.

So you basically have these clusters, each of them represents a distinct template, and you read one base at a time. So, first you read the first base after the primer. So it's sequencing downwards into the template. And you read the first base so you know what the first base of all your clusters is. And then you reverse the termination. You cleave off that chemical group that was blocking the 3-prime OH so now it can extend again. And then you add the four dNTPs again, do another round of extension, and then image again, and so forth.

And so it takes a little while. Each round of imaging takes about an hour. So if you want to do 100 base single and Illumina sequencing, it'll be running on the machine for about four days or so. Plus the time you have to build the clusters, which might be several hours on the day before.

So what is this? So actually the whole idea of blocking termination-- basically Sanger's idea-- is carried over here in Illumina sequencing with a little twist. And that's that you can reverse the termination. So if you look down here at the bottom, these are two different 3-prime terminators. Remember your base counting. Base one, two, three. So this was the 3-prime OH, now it's got this methyl [INAUDIBLE], or whatever that is. I'm not much of a chemist, so you can look that one up.

And then here's another version. And this is sort of chemistry that can cleave this off when you're done. And then this whole thing here, hanging off the base, is the fluor. And you cleave that off as well. So you add this big complicated thing, you image it, and then you cleave off the fluor and cleave off the 3-prime block.

These are some actual sequencing images you would image in the four channels. They're actually black and white. These are pseudocode. And then you can merge those and you can see then all the clusters on the flow cell. So this is from a GA2 with the recommended cluster density back in the day, like a few years ago. And nowadays, the image now since the software has gotten a lot better, so you can actually load the clusters more densely and therefore get more sequence out of the same area.

But imagine just millions and millions of these little clusters like this. Notice the clusters are not all the same size. Basically, you're doing PCR in situ, and so some molecules are easier to amplify by PCR than others. And that probably accounts for these variations in size.

So what is the current throughput? These data are accurate as of about, maybe, last year. So the HiSeq 2000 instrument is the most high performance, widely used instrument. Now there's a 2500, but I think it's roughly similar. You have one flow cell. So a flow cell looks sort of like a glass slide, except that it has these tunnels carved in it like eight little tubes inside the glass slide. And on the surfaces of those tubes is where the adapters are covalently attached. And so you have eight lanes and so you can sequence eight different things in those eight lanes. You could do yeast genome in one and fly RNA-seq in another, and so forth.

And these days, a single lane will produce something like 200 million reads. And this is typically routine to get 200 million reads from a lane. Sometimes you can get more. You can do up to 100 bases. You can do 150 these days on a MiSeq, which is a miniature version. You can do maybe 300 or more. And so that's a whole lot of sequence. So that's 160 billion bases of sequence from a single lane. And that will cost you-- that single lane-- maybe $2,000 to $3,000, depending where you're doing it. And the cost doesn't include the capital cost, that's just the reagent cost for running that.

So 160 billion-- the human genome is 3 billion, so you've now sequenced the human genome over many times there.

You can do more. So you can do paired-end sequencing, where you sequence both ends of your template. And that'll basically double the amount of sequence you get. And you can also, on this machine, do two flow cells at once. So you can actually double it beyond that.

And so for many applications, 160 billion bases is overkill. It's more than you need. Imagine you're doing bacterial genome sequencing. Bacterial genome might be five megabases or so. This is complete overkill. So you can do bar coding where you add little six base tags to different libraries, and then mix them together, introduce them to the machine, sequence the tags first or second, and then sequence the templates. And then you effectively sort them out later. And then do many samples in one lane. And that's what people most commonly do.

So, questions about next-gen sequencing? There's a lot more to learn. I'm happy to talk about it more. It's very relevant to this class. But I'm sure it'll come up later in David's sections, so I don't want to take too much time on it.

So, now once you generate reads from an Illumina instrument or some other instrument, you'll want to align them to the genome to determine, for example, if you're doing RNA-seq mapping reads that come from mRNA, you'll want to know what genes they came from. So you need to map those reads back to the genome. What are some other reasons you might want to align sequences? Just in general,

why is aligning sequences-- meaning, matching them up and finding individual bases or amino acid residues that match-- why is that useful? Diego?

**AUDIENCE:** You can assemble them if you want.

**PROFESSOR:** You can assemble them? Yes. So if you're doing genome sequencing, if you align them to each other and you find a whole stack that sort of align this way, you can then assemble and infer the existence of a longer sequence. That's a good point.

Yes, your name?

**AUDIENCE:** Julianne. Looking at homologs.

**PROFESSOR:** Looking at homologs. Right. So if you, for example, are doing disease gene mapping, you've identified a human gene of unknown function that's associated with a disease. Then you might want to search it against, say, the mouse database and find a homolog in mouse and then that might be what you would want to study further. You might want to then knock it out in mouse or mutate it or something. So those are some good reasons. There's others.

So we're going to first talk about local alignment, which is a type of alignment where you want to find shorter stretches of high similarity. You don't require alignment of the entire sequence. So there are certain situations where you might want to do that.

So here's an example. You are studying a recently discovered human non-coding RNA. As you can see, it's 45 bases. You want to see if there's a mouse homolog. You run it through NCBI BLAST, which as we said is sort of the Google search engine of mathematics-- and you're going get a chance to do it on pump set one, and you get a hit that looks like this.

So notice, this is sort of BLAST notation. It says Q at the top. Q is for "query," that's the sequence you put in. S is "subject," that's the database you were searching against. You have coordinates, so 1 to 45. And then, in the subject, it happened to be base 403 to 447 in some mouse chromosome or something. And you can see

that it's got some matching. But it also has some mismatches. So in all, there are 40 matches and five mismatches in the alignment.

So is that significant? Remember, the mouse genome is 2.7 billion bases long. It's big. So would you get a match this good by chance? So the question is really, should you trust this? Is this something you can confidently say, yes mouse is a homolog, and that's it? Or should you just be like, well, that's not better than I get by chance so I have no evidence of anything? Or is it sort of somewhere in between? And how would you tell? Yeah, what's your name?

**AUDIENCE:** Chris. You would want to figure out a scoring function for the alignment. And then, with that scoring function, you would find whether or not you have a significant match.

**PROFESSOR:** OK. So Chris says you want to define a scoring system and then use the scoring system to define statistical significance. Do want to suggest a scoring system? What's the simplest one you can think of?

**AUDIENCE:** Just if there's a match, you add a certain score. If it's a mismatch, you subtract a certain score.

**PROFESSOR:** So let's do that scoring system. So the notation that's often used is Sii. So that would be a match between nucleotide i and then another copy of nucleotide i. We'll call that 1, plus 1 for a match. And sij, where i and j are different, we'll give that a negative score. Minus 1. So this is i not equal to j.

So that's a scoring matrix. It's a four by four matrix with 1 on the diagonal and minus 1 everywhere else. And this is commonly used for DNA. And then there's a few other variations on this that are also used. So good, a scoring system. So then, how are we going to do the statistics? Any ideas? How do we know what's significant?

**AUDIENCE:** The higher score would probably be a little more significant than a lower score. But the scale, I'm not sure--

**PROFESSOR:** The scale is not so obvious. Yes, question?

**AUDIENCE:** My name is Andrea. So if you shuffled the RNA, like permute the sequence, then we'll get the [INAUDIBLE] genome you get with that shuffled sequence. And the score is about the same as you'd get with the non-shuffled sequence [INAUDIBLE] about very significant scores.

**PROFESSOR:** Yeah, so that's a good idea. BLAST, as it turns out-- is pretty fast. So you could shuffle your RNA molecule, randomly permute the nucleotides many times, maybe even like 1,000 times, search each one against the mouse genome, and get a distribution of what's the best score-- the top score-- that you get against a genome, look at that distribution and say whether the score of the actual one is significantly higher than that distribution or just falls in the middle of that somewhere. And that's reasonable.

You can certainly do that, and it's not a bad thing to do. But it turns out there is an analytical theory here that you can use. And so that you can determine significance more quickly without doing so much computation. And that's what we'll talk about. But another issue, before we get to the statistics, is how do you actually find that alignment? How do you find the top scoring match in a mouse genome?

So let's suppose this guy is your RNA. OK, of course, we're using T's, but that's just because you usually sequences it at the DNA level. But imagine this is your RNA. It's very short. This is like 10 or so, I think. And this is your database. But it goes on a few billion more. Several more blackboards. And I want to come up with an algorithm that will find the highest scoring segment of this query sequence against this database.

Any ideas? So this would be like our first algorithm. And it's not terribly hard, so that's why it's a good one to start with. Not totally obvious either. Who can think of an algorithm or something, some operation that we can do on this sequence compared to this sequence-- in some way-- that will help us find the highest scoring match? I'm sorry. Yeah?

**AUDIENCE:** You have to consider insertion and deletion.

**PROFESSOR:** Yeah, OK. So we're going to keep it simple. That's true, in general. But we're going to keep it simple and just say no insertions and deletions. So we're going to look for an ungapped local alignment. So that's the algorithm that I want. First, no gaps. And then we'll do gaps on Tuesday. Tim?

**AUDIENCE:** You could just compare your [INAUDIBLE] to [INAUDIBLE] all across the database and turn off all the [INAUDIBLE] on that [INAUDIBLE], and then figure out [INAUDIBLE].

**PROFESSOR:** Yeah, OK. Pretty much. I mean, that's pretty much right. Although it's not quite as much of a description as you would need if you want to actually code that. Like, how would you actually do that? So, I want a description that is sort of more at the level of pseudocode. Like, here's how you would actually organize your code.

So, let's say you entertain the hypothesis that the alignment can be in different registers. The alignment can correspond to base one of the query and base one of the subject. Or it could be shifted. It could be an alignment where base 1 of the query matches these two, and so forth. So there's sort of different registers. So let's just consider one register first. The one where base 1 matches.

So let's just look at the matches between corresponding bases. I'm just going to make these little angle bracket guys here. Hopefully I won't make any mistakes. I'm going to take this. This is sort of implementing Tim's idea here. And then I'm going to look for each of these-- so consider it going down here. Now we're sort of looking at an alignment here. Is this a match or a mismatch?

That's a mismatch. That's a match. That's a mismatch. That's a mismatch. That's a match. Match Match. Mismatch. Mismatch. Mismatch. So where is the top scoring match between the query and the subject? Tim? Anyone?

**AUDIENCE:** 6, 7, 8.

**PROFESSOR:** 6, 7, 8. Good. Oh--

**AUDIENCE:** 5, 6, 7.

**PROFESSOR:** 5, 6, 7. Right. Right here. You can see there's three in a row. Well, what about this? Why can't we add this to the match? What's the reason why it's not 2, 3, 4, 5, 6, 7?

**AUDIENCE:** Because the score for that is lower.

**PROFESSOR:** Because the score for that is lower. Right. We defined top scoring segment. You sum up the scores across the map. So you can have mismatches in there, but this will have a score of 3. And if you wanted to add these three bases, you would be adding negative 2 and plus 1, so it would reduce your score. So that would be worse.

Any ideas on how to do this in an automatic, algorithmic way? Yeah? What's your name?

**AUDIENCE:** Simon. So if you keep shifting the entire database, [INAUDIBLE].

**PROFESSOR:** OK so you keep shifting it over, and you generate one of these lines. But imagine my query was like 1,000 or something. And my database is like a billion. How do I look along here? And here it was obvious what the top scoring match is. But if I had two matches here, then we would've actually had a longer match here.

So in general, how do I find that that top match? For each of those registers, if you will, you'll have a thousand long diagonal here with 1's and minus 1's on it. How do I process those scores to find the top scoring segment? What's an algorithm to do that?

It's kind of intuitively obvious, but I want to do something with, you define a variable and you update it, and you add to it, and subtract. Something like that. But, like a computer could actually handle. Yeah? What was your name? Julianne?

**AUDIENCE:** Could you keep track of what the highest total score is, and then you keep going down the diagonal, And then you update it?

**PROFESSOR:** OK. You keep track of what the highest total score was?

**AUDIENCE:** Yeah. The highest test score.

**PROFESSOR:** The highest segment score? OK. I'm going to put this up here. And we'll define max s. That's the highest segment score we've achieved to date. And we'll initialize that to zero, let's say. Because if you had all mismatches, zero would be the correct answer. If your query was A's and your subject was T's. And then what do you do?

**AUDIENCE:** As you go down the diagonal, you keep track of--

**PROFESSOR:** Keep track of what?

**AUDIENCE:** So you look at 1 in 1 first. And then you go 1 in 2, and you find a score of zero. But that's higher than negative 1.

**PROFESSOR:** But the score of the maximum segment at that point, after base 2, is not zero. It's actually 1. Because you could have a segment of one base alignment. The cumulative score is zero. I think you're onto something here that may be also something useful to keep track of.

Let's do the cumulative score and then you tell me more. We'll define cumulative score variable. We'll initialize that to zero. And then we'll have some for loops that, as some of you have said, you want to loop through the subject. All the possible registers of the subject. So that would be maybe j equals 1 to subject length minus query length. Something like that. Don't worry too much about this. Again, this is not real code, obviously. It's pseudocode.

So then this will be, say, 1 to query language. And so this will be going along our diagonal. And we're going to plot the cumulative score. So here you would you have an update where cumulative score plus equals the score of query position i matched against subject position j. And update that. So that's just cumulative score.

So what will it look like? So in this case, I'll just use this down here. So you have zero, 1, 2, minus 1, minus 2. So you'll start at position zero in the sequence. At position 1 you're down here at minus 1 because it was a mismatch.

Then at position 2, as you said, we're back up to zero. And then what happens? Go

down to minus 1, down to minus 2. Then we go up three times in a row until we're up here to 1. And then we go down after that.

So where is your highest scoring match in this cumulative score plot? People said it was from 5 to 7. Yeah, question?

**AUDIENCE:** So would it be from like a local minimum to a local maximum?

**PROFESSOR:** Yeah. Exactly. So, what do you want to keep track of?

**AUDIENCE:** You want to keep track of the minimum and the maximum. And look for the range which you maximize to different--

**PROFESSOR:** Yeah, so this is now sort of more what I was looking for in terms of-- so this was the local minimum, and that's the local maximum. This is the score. That's your mass s there. And you also want to keep track of where that happened in both the query and the subject. Does that make sense? So you would keep track of this running cumulative score variable. You keep track of the last minimum. The minimum that you've achieved so far. And so that would then be down here to minus 2.

And then when your cumulative score got up to plus 1, you always take that cumulative score, minus the last minimum cumulative score. That gives you a potential candidate for a high scoring segment. And if that is bigger than your current max high scoring segment, then you update it and you would update this. And then you would also have variables that would store where you are. And also, where did that last minimum occur.

So I'm not spelling it all out. I'm not going to give you all the variables. But this is an algorithm that would find the maximum score. Yeah, question?

**AUDIENCE:** So you're keeping track of the global maximum, local minimum, so that you can accept the most recent local minimum following the global maximum?

**PROFESSOR:** I'm not sure I got all that. But you're keeping track of the cumulative score. The minimum that that cumulative score ever got to. And the maximum difference, the maximum that you ever in the past have gone up. Where you've had a net

increment upwards.

Like here. So this variable here, this max s, it would be initialized to zero. When you got to here, your last minimum score would be minus 1. Your cumulative score would be zero. You would take the difference of those, and you'd be like, oh I've got a high scoring segment of score one. So I'm going to update that.

So now, that variable is now 1 at this point. Then you're going down, so you're not getting anything. You're just lowering this minimum cumulative score down to minus 2 here. And then when you get to here, now you check the cumulative score minus the last minimum. It's 1. That's a tie. We won't keep track of ties.

Now at here, that difference is 2. So now we've got a new record. So now we update this maximum score to 2 in the locations. And then we get here, now it's 3, and we update that. Does that make sense?

**AUDIENCE:** Imagine the first dip-- instead of going down to negative 1, it went down to negative 3.

**PROFESSOR:** Negative 3?

**AUDIENCE:** That first dip.

**PROFESSOR:** Right here? So we started back a little bit. So back here, like this?

**AUDIENCE:** No.

**PROFESSOR:** Down to negative 3?

**AUDIENCE:** No.

**PROFESSOR:** But how do we get to negative 3? Because our scoring is this way. You want this dip to minus 3?

**AUDIENCE:** No

**PROFESSOR:** This one minus 3? Imagine we are at minus 3 here?

**AUDIENCE:** Yeah. Imagine it dipped to minus 3. And then the next one dipped to higher than that, to minus 2. And then it went up to 1. And so, would the difference you look at be negative 2 to 1, or negative 3 to 1?

**PROFESSOR:** Like that, right? So, minus 3, let's say, minus 2, 1. Something like that. What do people think? Anyone want to--?

**AUDIENCE:** Minus 3 to 1.

**PROFESSOR:** Minus 3 to 1. It's the minimum that you ever got to. This might be a stronger match, but this is a higher scoring match. And we said we want higher scoring. So you would count that.

**AUDIENCE:** So you keep track of both the global minimum and the global maximum, and you take the difference between them.

**PROFESSOR:** You keep track of the global minimum and the current cumulative score, and you take the difference.

**AUDIENCE:** The global maximum--

**PROFESSOR:** It's not necessarily global maximum because we could be well below zero here. We could do like this. From here to here. So this is not the global maximum. This just happens to be, we went up a lot since our last minimum. So that's your high scoring segment. Does that make sense?

I haven't completely spelled it out. But I think you guys have given enough ideas here that there;s sort of the core of an algorithm. And I encourage you to think this through afterwards and let me know if there are questions. And we could add an optional homework where I ask you to do this, that we've sometimes had in the past. It is a useful thing to look at.

This is not exactly how the BLAST algorithm works. It uses some tricks for faster speed. But this is sort of morally equivalent to BLAST in the sense that it has the same order of magnitude running time.

So this algorithm-- what is the running time in Big-O notation? So just for those who are non-CS people, when you use this Big-O notation, then you're asking, how does the running time increase in the size of the input? And so what is the input? So we have two inputs. We have a query of length. And let's say subject of length n. So clearly, if those are bigger, it'll take longer to run. But when you compare different algorithms, you want to know how the run time depends on those lengths. Yes. What's your name?

**AUDIENCE:** Sally. m times n.

**PROFESSOR:** So with this, this is what you would call an order mn algorithm. And why is that? How can you see that?

**AUDIENCE:** You have two for loops And for each length, essentially, you're going through everything in the query. And then, for everything that you go through in the query, you would [INAUDIBLE].

**PROFESSOR:** Right. In this second for loop here, you're going through the query. And you're doing that nested inside of a for loop that's basically the length of the subject. And eventually you're going to have to compare every base in the query to every base in the subject. There's no way around that. And that takes some unit of time. And so the actual time will be proportional to that. So the bigger n gets and m gets, it's just proportional to the product. Does that make sense?

Or another way to think about it is, you're clearly going to have to do something on this diagonal. And then you're going to have to do something on this diagonal, and this one, and this one. And actually, you have to also check these ones here. And in the end, the total number of computations there is going to be this times that. You're basically doing a rectangle's worth of computations. Does that makes sense?

So that's not bad, right? It could be worse. It could be, like, mn squared or something like that. So that's basically why BLAST is fast.

So what do these things look like, in general? And what is the condition on our score

for this algorithm to work? What if I gave a score of plus 1 for a match, and zero for a mismatch? Could we do this? Joe, you're shaking your head.

**AUDIENCE:** It would just be going up.

**PROFESSOR:** Yeah. The problem is, it might be flat for a while, but eventually it would go up. And it would just go up and up and up. And so your highest scoring segment would, most of the time, be something that started very near the beginning and ended very near the end. So that doesn't work. So you have to have a net negative drift. And the way that's formalized is the expected score has to be negative.

So why is the expected score negative in this scoring system that has plus 1 for a match, and minus 1 for a mismatch? Why does that work?

**AUDIENCE:** It should be wrong three quarters of the time.

**PROFESSOR:** Yeah. You'll have a mismatch three quarters of the time. So on average, you tend to drift down. And then you have these little excursions upwards, and those are your high scoring segments. Any questions about that?

**AUDIENCE:** Question. Is there something better than m times n?

**PROFESSOR:** We've got some computer scientists here. David? Better than m times n? I don't think so, because you have to do all those comparisons. And so there's no way around that, so I don't think so. All right. But the constant-- you can do better on the constant than this algorithm thing.

**AUDIENCE:** With multiple queries--

**PROFESSOR:** With multiple queries, yeah. Then you can maybe do some hashing or find some-- to speed it up.

OK, so what about the statistics of this? So it turns out that Karlin and Altschul developed some theory for just exactly this problem. For searching a query sequence. It can be nucleotide or protein as long as you have integer scores and the average-- or the expected-- score is negative, then this theory tells you how

often the highest score of all-- across the entire query database comparison-- exceeds a cut off x using a local alignment algorithm such as BLAST.

And it turns out that these scores follow what's called an extreme value or Gumbel distribution. And it has this kind of double exponential form here. So x is some cut off. So usually x would be the score that you actually observed when you searched your query against the database. That's the one you care about.

And then you want to know, what's the probability we would've seen something higher than that? Or you might do x is one less than the score you observed. So what's the chance we observed something the same, as good as this, or better? Does that make sense? And so this is going to be your P value then.

So the probability of S. The score of the highest segment under a model where you have a random query against a random database of the same length is 1 minus e to the minus KMN e to the minus lambda x. Where M and N are the lengths of the query and the database. x is the score. And then K and lambda are two positive parameters that depend actually on the details of your score matrix and the composition of your sequences.

And it turns out that lambda is really the one that matters. And you can see that because lambda is up there in that exponent multiplying x. So if you double lambda, that'll have a big effect on the answer. And K, it turns out, you can mostly ignore it for most purposes.

So as a formula, what does this thing look like? It looks like that. Kind of a funny shape. It sort of looks like an umlauf a little bit, but then has a different shape on the right than the left. And how do you calculate this lambda? So I said that lambda is sort of the key to all this because of its uniquely important place in that formula, multiplying the score.

So it turns out that lambda is the unique positive solution to this equation here. So now it actually depends on the scoring matrix. So you see there's sij there. It depends on the composition of your query. That's the pi's. The composition of your

subject, that's the rj's. You sum over the i and j equal to each of the four nucleotides. And that sum has to be 1. So there's a unique positive solution to this equation.

So how would we solve an equation like this? First of all, what kind of equation is this, given that we're going to set the sij, and we're going to just measure the pi and the rj? So those are all known constants, and lambda is what we're trying to solve for here. So what kind of an equation is this in lambda? Linear? Quadratic? Hyperbolic? Anybody know what this is?

So this is called a transcendental equation because you have different powers. That sounds kind of unpleasant. You don't take a class in transcendental equations probably. So in general, they're not possible to solve analytically when they get complicated. But in simple cases, you can solve them analytically. And in fact, let's just do one.

So let's take the simplest case, which would be that all the pi's are a quarter. All the ri's are a quarter. And we'll use the scoring system that we came up with before, where sii is 1, and sij is minus 1. If i does not equal j.

And so when we plug those in to that sum there, what do we get? We'll get four terms that are one quarter, times one quarter, times e to the lambda. There's four possible types of matches, right? They have probability one quarter times a quarter. That's pi and rj. And the e to the lambda sii is just e to the lambda because sii is 1. And then there's 12 terms that are one quarter, one quarter, e to the minus lambda. Because there's the minus 1 score. And that has to equal 1.

So cancel this, we'll multiply through by 4, maybe. So now we get e to the lambda plus 3. e to the minus lambda equals 1. It's still a transcendental equation, but it's looking a little simpler. Any ideas how to solve this for lambda? Sally?

**AUDIENCE:** Wouldn't the 1 be 4?

**PROFESSOR:** I'm sorry. 4. Thank you. Yeah, what's your name?

**AUDIENCE:** [INAUDIBLE] I think [INAUDIBLE] quadratic equation. If you multiply both sides by [INAUDIBLE] then [INAUDIBLE].

**PROFESSOR:** OK, so the claim is this is basically a quadratic equation. So you multiply both sides by e to the lambda. So then you get e to the 2 lambda plus 3. And then it's going to move this over and do minus 4 e to the lambda equals zero. Is that good?

So how is it quadratic? What do you actually do to solve this?

**AUDIENCE:** Well, [INAUDIBLE].

**PROFESSOR:** Change the variable, x equals e to the lambda. Then it's quadratic in x. Solve for x. We all know how to solve quadratic equations. And then substitute that for lambda. OK, everyone got that?

If you use 16 different scores to represent all the different types of matches and mismatches, this will be very unpleasant. It's not unsolvable, it's just that you have to use computational numerical methods to solve it. But in simple cases where you just have a couple different types of scores, it will often be a quadratic equation.

All right. So let's suppose that we have a particular scoring system-- particular pi's, rj's-- and we have a value of lambda that satisfies those. So we've solved this quadratic equation for lambda. I think we get lambda equals natural log 3, something like that. Remember, it's a unique positive solution. Quadratic equations are two solutions, but there's going to be just one positive one. And then we have that value. It satisfies this equation.

So then, what if we double the scores? Instead of plus 1 minus 1, we use plus 2 minus 2? What would then happen? You can see that the original version of lambda wouldn't necessarily still satisfy this equation. But if you think about it a little bit, you can figure out what new value of lambda would satisfy this equation.

We've solved for the lambda that solves with these scores. Now we're going to have new scores. sii prime equals 2. sij prime equals minus 2. What is lambda prime? The lambda that goes with these scores? Yeah, go ahead.

**AUDIENCE:** Half of the original?

**PROFESSOR:** Half of the original? Right. So you're saying that lambda prime equals lambda over 2. And why is that? Can you explain?

**AUDIENCE:** Because of the [INAUDIBLE].

**PROFESSOR:** Yeah, if you think about these terms in the sum, the s part is all doubling. So if you cut the lambda apart, and the product will equal what it did before. And we haven't changed the pi's and rj's, so all those terms will be the same. So therefore, it will still satisfy that equation. So that's another way of thinking about it. Yes, you're correct.

So if you double the scores, lambda will be reduced by a factor of 2. So what does that tell us about lambda? What is it? What is its meaning? Yeah, go ahead, Jeff.

**AUDIENCE:** Scale of the distribution to the expectant score? Or the range score?

**PROFESSOR:** Yeah. It basically scales the scores. So we can have the same equation here used with arbitrary scoring. It just scales it. You can see the way it appears as a multiplicative factor in front of the score. So if you double all the scores, will that change what the highest scoring segment is? No, it won't change it because you'll have this cumulative thing. It just changes how you label the y-axis. It'll make it bigger, but it won't change what that is.

And if you look at this equation, it won't change the statistical significance. The x will double in value, because all the matches are now worth twice as much as what they were before. But lambda will be half as big, and so the product will be the same and therefore, the final probability will be the same. So it's just a scaling factor for using different scoring systems. Everyone got that?

All right. So what scoring matrix should we use for DNA? How about this one? So this is now a slight generalization. So we're going to keep 1 for the matches. You don't lose any generality by choosing 1 here for matches, because if you use 2, then lambda is just going to be reduced to compensate.

So 1 for matches. And then we're going to use m for mismatches. And m must be negative in order to satisfy this condition for this theory to work, that the average score has to be negative. Clearly, you have to have some negative scores.

And the question then is, should we use minus 1 like we used before? Or should we use like minus 2 or minus 5, or something else? Any thoughts on this? Or does it matter? Maybe it doesn't matter. Yeah, what's your name?

**AUDIENCE:** [INAUDIBLE]. Would it make sense to not use [INAUDIBLE], because [INAUDIBLE].

**PROFESSOR:** Yeah, OK. So you want to use a more complicated scoring system. What particular mismatches would you want to penalize more and less?

**AUDIENCE:** [INAUDIBLE] I think [INAUDIBLE] needs to be [INAUDIBLE].

**PROFESSOR:** Yeah, you are correct in your intuition. Maybe one of the biologists wants to offer a suggestion here. Yeah, go ahead.

**AUDIENCE:** So it's a mismatch between purine and pyrimidine [INAUDIBLE].

**PROFESSOR:** OK so now we've got purines and pyrimidines. So everyone remember, the purines are A and G. The pyrimidines are C and T. And the idea is that this should be penalized, or this should be penalized less than changing a purine to a pyrimidine. And why does that makes sense?

**AUDIENCE:** Well, structurally they're--

**PROFESSOR:** Structurally, purines are more similar to each other than they are to pyrimidines. And? More importantly, I think. In evolution?

**AUDIENCE:** [INAUDIBLE].

**PROFESSOR:** I'm sorry, can you speak up?

**AUDIENCE:** C to C mutations happen spontaneously in [INAUDIBLE] chemistry.

**PROFESSOR:** Yes. So C to C mutations happen spontaneously. So basically, it's easier because

they look more similar structurally. The DNA polymerase is more likely to make a mistake and substitute another purine. The rate of purine, purine or pyrimidine, pyrimidine to transversions which switch the type is about three to one, or two to one in different systems. So yeah, that's a good idea.

But for simplicity, just to keep the math simple, we're just going to go with one mismatch penalty. But that is a good point. In practice, you might want to do that.

So now, I'm saying I'm going to limit you to one mismatch penalty. But I'm going to let you choose any value you want. So what value should you choose? Or does it matter? Or maybe different applications? Tim, yeah?

**AUDIENCE:** I've just got a question. Does it depend on pi and ri? For example, we could use all these numbers. But if the overall wants to be negative, then you couldn't use negative .1.

**PROFESSOR:** Right, that's a good point. You can't make it too weak. It may depend on what your expected fraction of matches is, which actually depends on pi and ri. So if you have very biased sequences, like very AT rich, your expected fraction of matches is actually higher. When you're researching an AT rich sequence against another AT rich sequence, it's actually higher than a quarter.

So even minus one might not be sufficient there. You might need to go down more negative. So you may need to use a higher negative value just to make sure that the expected value is negative. That's true. And yeah, you may want to adjust it based on the composition.

So let's just do a bit more. So it turns out that the Karlin and Altschul theory, in addition to telling you what the p value is of your match-- the statistical significance-- it also tells you what the matches will look like in terms of what fraction of identity they will have. And this is the so-called target frequency equation.

The theory says that if I search a query with one particular composition, p, subject meta-composition r-- here, I've just assumed they're the same, both p just for simplicity-- with a scoring matrix sij, which has a corresponding of lambda. Then,

when I take those very high scoring matches-- the ones that are statistically significant-- and I look at those alignments of those matches, I will get values qij, given by this formula.

So look at the formula. So it's qij. So pipj e to the lambda sij. So it's basically the expected chance that you would have base i matching based j just by chance. That's pipj. But then weighted by e to the lambda sij. So we notice for a match, s will be positive, so e to the lambda will be positive. So that will be bigger than 1. And you'll have more matches and you'll have correspondingly less mismatches because the mismatch has a negative. So get the target value score.

And that also tells you that the so-called natural scores are actually determined by the fraction of matches that you want in your high scoring segments. If we want 90% matches, we just set qii to be 0.9, and use this equation here. Solve for sij.

For example, if you want to find regions with R% identities. Little r is just the r as a proportion. qii is going to be r over 4. This assumes unbiased base composition. A quarter of the matches are acgt. Qij, then, is 1 minus r over 12. 1 minus r is a fraction of non-matching positions. They're 12 different types.

Set sii equal to 1, that's what we said we normally do. And then you do a little bit of algebra here. m is sij. And you sort of plug in this equation twice here. And you get this equation. So it says that m equals log of 4 1 minus r over 3 over log 4 r.

And for this to be true, this assumes that both the query and the database have uniform composition of a quarter, and that r is between a quarter and 1. The proportion of matches in your high scoring segment-- you want it to be bigger than a quarter. A quarter is what you would see by chance. There's something wrong with your scoring system if you're considering those to be significant. So it's something above 25%.

And so it's just simple algebra-- you can check my work at home-- to solve for m here. And then this equation then tells you that if I want to find 75% identical matches in a nucleotide search, I should use a mismatch penalty of minus 1.

And if I want 99% identical matches, I should use a penalty of minus 3. Not minus 5, but minus 3. And I want you to think about, does that make sense? Does that not make sense? Because I'm going to ask you at the beginning of class on Tuesday to explain and comment on this particular phenomenon of how when you want higher percent identities, you want a more negative mismatch score. Any last questions? Comments?