Next Generation Sequencing Technologies: Data Analysis and Applications

Overlap- Layout-Consensus Approach

Dr. Riddhiman Dhar, Department of Biotechnology

Indian Institute of Technology Kharagpur

Good day, everyone. Welcome to the course on Next Generation Sequencing Technologies, Data Analysis, and Applications. In the last couple of classes, we have talked about genome assembly, and we have started discussing algorithms that could allow us to assemble the reference genome from a set of reads. So, in the first class, we talked about the assembly problem and the challenges that are there when you want to do an assembly. So, then we actually talked about different approaches, so that one could take them, and we mentioned three methods that we will be discussing in this course. So, in the last class, we talked about one such approach, which is the shortest common superstring approach, or the SCS approach.

So, we actually discussed the SCS approach. We saw that the SCS approach is not really solvable within a reasonable amount of time, right? So, it works through an overlap graph, and thus finding the optimal path in that graph is an NP-hard problem, right? So, we talked about something called the greedy SCS approach, which could give us a solution, but we have found that this is actually not an optimal solution in many cases.

So, in this class, what we will be talking about is the overlap layout consensus approach. This is a different approach to genome assembly, and this is what we will be discussing in this class, ok? So, we will talk about the idea. We will actually apply this approach again to a set of reads for a small reference genome, and then we will see how this method works and what the limitations of this method are. So, these are the keywords for today's class.

The first is inferred edges, and the second is repeated. So, just to summarize the drawbacks of the SCS approach that we discussed in the last couple of classes, So, one of the challenges is that finding the shortest common superstring from the overlap graph is NP-hard. So, we have not talked about what NP-hard or NP-complete is, but what I meant is that it will take a long time if we have to deal with millions or billions of reads that we get

from current generation sequences, ok? So, this is kind of you do the genome sequencing within 7 days and you would have to wait for a few months to get assembly or maybe a year to get assembly, right?

So, that is something that we do not want. We want this assembly to be done within another week or so, or faster is better, right? So, we also talked about the greedy approach, which could give us a workaround to this problem. So, what does greedy approach do? It takes the best possibility that is available at that point, and it will try to find a solution, ok? So, with a greedy approach to the shortest common superstring, it will start merging reads that show the most overlap, ok?

And it will go on with that approach, and it will try to find the reference genome. What we have seen is that this might give us wrong results at times, and in some cases, for example, when you have repeats, it will miss certain parts and give us an incorrect reference sequence. So, again, this is not optimal. It can get us close to the original reference sequence, but in many cases, we might not get the right reference sequence. So, maybe we need to think about some other approach, right?

And what we have also seen is that repeats lead to incorrect assembly. In a greedy approach, this is actually a problem, right? It will not just say I cannot solve the repeat region. It will actually give you a solution and say, "Okay, this is the solution that I found," which might be the incorrect solution. So, we need to think about other approaches, and the second approach that we are going to talk about is called the overlap layout consensus approach, or the OLC approach.

So, as the name suggests, there are three steps in this method. As you can probably understand, the first step is the overlap step. So, this overlap step is exactly identical to the SCS approach, right? So, when you start with the SCS approach, you first build the overlap graph, right? So, we do a pairwise comparison between the set of reads, and we build the overlap graph based on their overlaps.

Again, this is a directed graph. The graph that we build is a directed graph, right? So, we have these directions as well as edge weights, right? So, in the last class, we talked about this. The edge weights are determined by the length of the overlap.

So, we built this for some examples in the last couple of classes, ok? So, this step is common between SCS and the OLC approach. The second step is the layout approach, ok? So, we will talk about this layout approach in subsequent slides, ok, in much more detail, and this is where the difference starts, ok. This is where it differs from the SCS approach.

So, the SCS approach tries to find the best path, right? So, it tries to find an optimal path, ok? So, we talked about what the optimal path is. So, we want to find a path that goes through each node of the overlap graph exactly once and maximizes the total score. So, and then we also showed that this is equivalent to the traveling salesman problem, which is why it is NP-hard.

So, the layout approach is different in that it does not try to do that. It actually takes a different approach, and we will see with some examples how this actually works. And the third step is the consensus in the name, as you can see. So, this is where we find the consensus sequences for the reference genome. So, let us begin, right?

So, we will actually start with the overlap finding. We have discussed this in a lot of detail. We have built an overlap graph, but we have not mentioned the methods that we can use for building this graph. So, for example, we can start with a name approach, right? So, for example, we can say we will have a sliding window approach.

So, we slide one read against another and see if there are matches or overlaps of certain leads, right? Again, we use a cutoff for the overlap, right? So, let us say if we are working with some reads, right? So, if we have a certain length, then we use a cutoff, right. So, we have to have that amount of overlap before we can add those nodes with an edge, ok?

So, in the sliding window, we simply slide these reads against each other and try to find a

match, ok? So, this is actually a bit cumbersome, right? Again, this sliding part The second approach is dynamic programming, ok? So, we will not go into this in detail in this class.

We will not have time to do that. So, some of you might be familiar with dynamic programming. This is applied in cases of global alignment and local alignment, the Needleman-Wunch algorithm, or the Smith-Waterman algorithm. We apply dynamic programming, right, where we build this matrix, right? So, if you have these two sequences, right?

So, take sequence 1, for example. So, in dynamic programming, we build a matrix something like this, and in one direction, we have sequence 1, right? So, these are the bases, which may be ATG, CC, etcetera, and so on. In the other direction, maybe we have this sequence 2, right? So, we have these two sequences, one along the rows and one along the columns, and then we build up this matrix with this match mismatch score and then also the gap opening penalty or gap extension penalty. So, those who are familiar with alignment, local alignment, and global alignment algorithms will know what dynamic programming is.

Now, for each pair-wise comparison, we would have to build a matrix, right? So, which will be of length m into n, right, where m and n are the lengths of the sequences, right? So, sequence 1 is of length m, and sequence 2 might be of length n. So, for each comparison, you have to calculate these many values, right, and then find the overlaps between these two sequences, right. So, as you can probably imagine this is going to take time, ok?

This is not an easy process, ok? So, the third approach that you can take is maybe a suffix tree or a prefix tree, right? So, in suffix trees, you remember we built suffix trees in earlier

classes, and we can utilize those structures to actually find matches between two reaches. So, if you remember, when we are looking for overlaps, we are looking for overlaps between the suffix of a read and the prefix of another read, right? So, we can utilize that property, right? using a suffix tree and try to find a match, and it will immediately tell us, ok, where this match is and how much the overlap is, right? And as you can imagine, each of these approaches requires some sort of elaborate data structure, maybe a matrix calculation or maybe sliding on the approach.

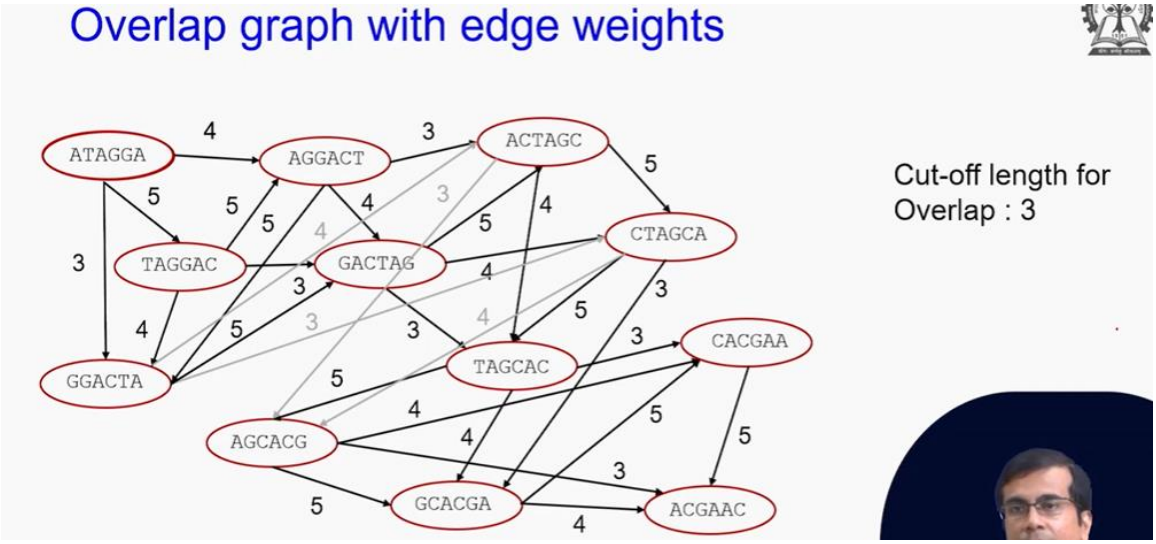
So, whichever approach you take, this is a slow process, and in fact, this is the slowest process of the full method. So, if you are talking about SCS, this is actually quite a time-consuming step. If you are talking about OLC, this is also the most time-consuming step, right? So, as you can imagine, right, if you are working with 10, 15, 20 reads, or 100 reads, ok, you will be able to do this in a reasonable amount of time. But the moment you are working with billions of reads, right, as you get today, you will have to do those pairwise comparisons—all pairwise comparisons for this overlap graph, ok, and that is time-consuming, ok.

So, remember, this is a very time-consuming step. The second part is that the overlap graphs are quite big, right? So, each node is a read, right? So, the graph that you construct from this overlap analysis will contain billions of nodes, right? So, it will be identical to the number of reads that you work with, ok?

So, this is a time-consuming step, and this is also quite a big structure because of the number of reads that you are working with. So, I will go straight to this example because we have built this overlap graph in the last couple of classes, right? So, hopefully you remember or you can go back, right? So, this is with the specific sequence and small reference sequence we used, and we built this overlap graph, ok? So, and we also added the edge weights to this graph, ok?

So, as you can see, the numbers 4, 5, and 3 actually denote the overlap length between these two nodes. So, this is something we have discussed. Now what can you do with this

overlap graph? So, can we apply this layout approach to this overlap graph to actually find the reference sequence? Instead of this SCS approach or greedy SCS approach, how will the OLC approach find the reference graph from this overlap graph?



So, the layout approach is okay. So, what happens in the layout approach is that we merge an ensemble of reads into contigs. So, this is the idea behind this layout approach, ok? And the major step here is that we start to remove the edge from the overlap graphs that can be inferred. So, the edges that can be inferred are called the transitively inferred edges. We will give an example, and you will realize how these edges are inferred, ok?

Now, just before we go there, right, I just wanted to also define contigs, right? So, a contig is a contiguous genomic region in which the sequence of bases is known, right? So, we know the order of this ATGC in this region, and after the assembly, you can get multiple contigs, ok? And you can have gaps between these contigs, ok? So, ideally, we would like to get the contig number to be exactly the same as the chromosome number, right?

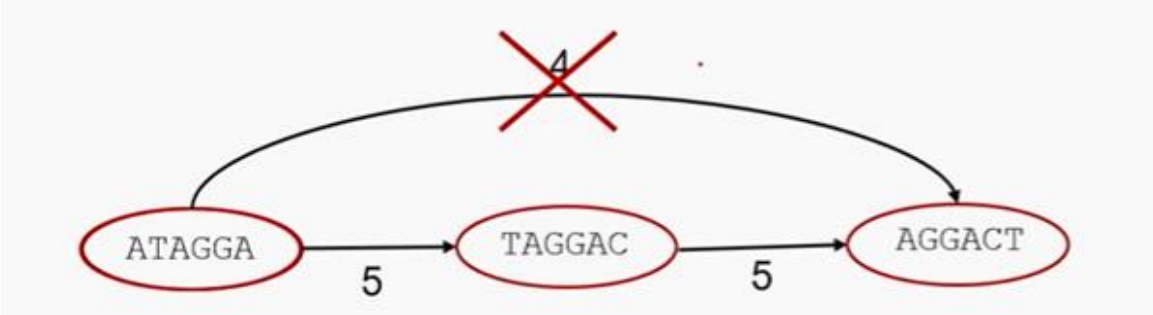
For example, if you are working with mammalian genomes, in an ideal world, we would like the contig number to be the same as the chromosome number, but that is not the case. So, you will get more contigs than the number of chromosomes, ok? And this happens again because of the presence of repeat sequences, which would not be possible and which cannot perhaps be properly inferred through these assembly methods. So, we can see these gaps between contigs, and we will come back to this later on to see how we can fill up

these gaps and actually get the full genome sequence. So, let us now move back to the layout approach.

We will take a very simple example, and then we will talk about the layout approach before we apply this approach to the big graph, right, the overlap graph that I just showed you. So, here are three reads, ok, and they have some overlap, as you can see. So, the first one and the second one has here 5 base overlap; the second and third one share 5 base overlap, ok? So, that means the first and third ones are likely to share four base overlaps because of the properties, right? We have this suffix and the prefix structure. So, if we represent this overlap in this overlap graph, it will look something like this, ok?

So, this is the first one, right? In that sequence, this is the second one, the third one, okay? So, and also, we have added the edge weights, as you can see in 5 5 4, right, again depending on the number of overlaps, or the number of bases that overlap with each other. Now, in this graph, one thing you probably notice immediately, right, is that this edge here, ok, this can be inferred if we have the information that these two reads have 5 base overlap, and if these two reads have 5 base overlap, you can infer that these two reads here will have 4 base overlap, right? Why? Because of this nature that we or the constraint that we impose when you are building the overlap graph, right? We require the suffix of this node, right, to align or overlap with the prefix of this node 2, ok?

So, if that is the case, then the suffix of node 2 will align 5 bases with the prefix of 5 bases of node 3, ok? So, what it means is that now if you align this node 1 with node 3, you are going to see 4 bases overlap, ok? This is something you can infer from the structure of the graph because of the constraints that we have imposed, ok? So, this means this edge becomes an inferred edge or a transitively inferred edge, ok? And in the layout approach, we start to remove these inferred edges, which can be inferred from the other connections in the graph.

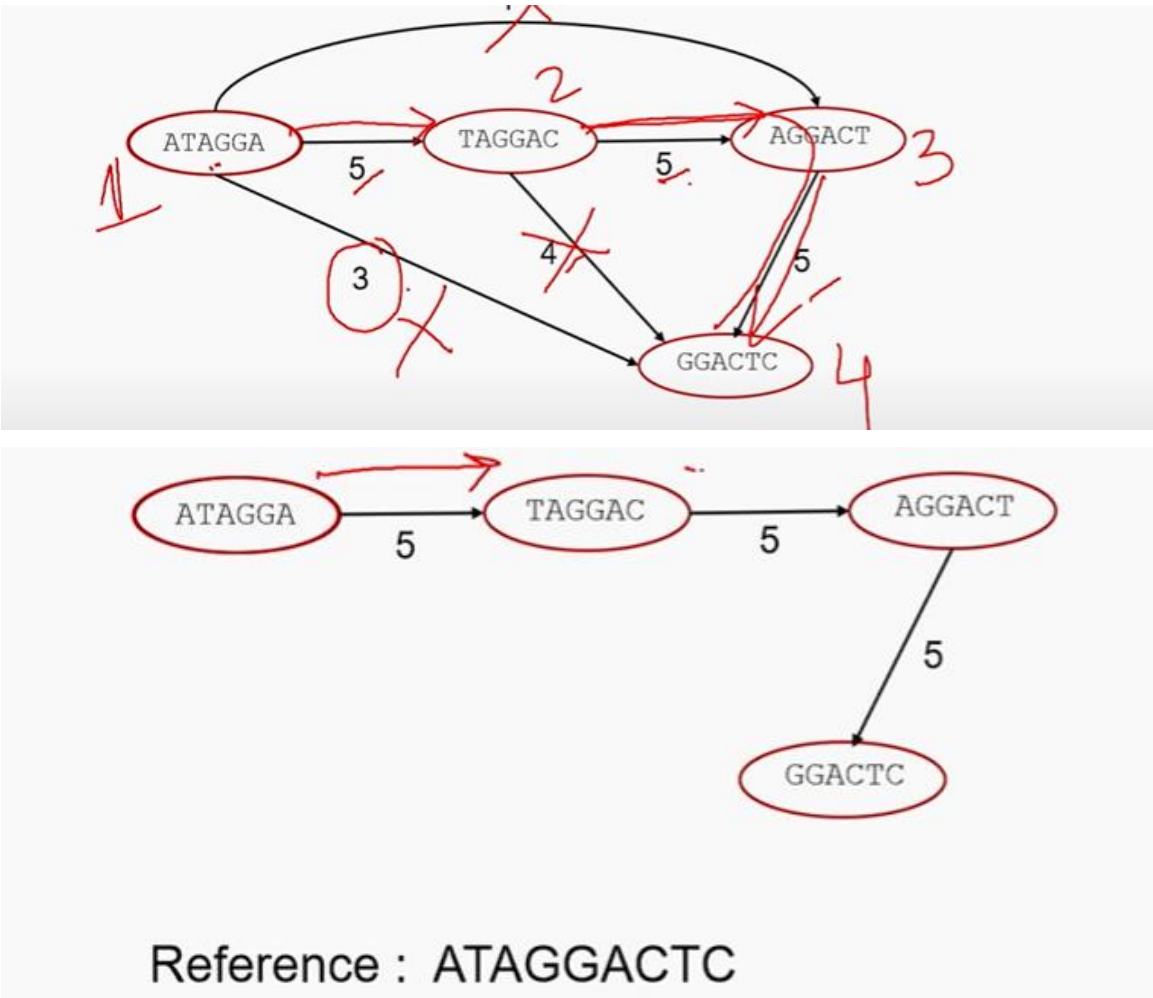


So, this edge is then removed, ok? Let us now take a slightly bigger example, right? We have added one more read, ok, to the same data, and we build the overlap graph for the same for this set of reads, ok. And it looks a bit more complex, right? So, we have now also added the weights. So, here you have node 1, node 2, node 3, and node 4, ok?

So, what you see is that again, in node 1 and 2, the overlap is 5 bases; in nodes 2 and 3, 5 bases; in nodes 3 and 4, 5 bases Now, can you tell me which of these are inferred edges? So, which can be described as inferred edges or transitively inferred edges? So, as before, if we apply the same criteria, right, this one is an inferred edge, right? We have already discussed this just now. For the other ones, you can see for this one, right, the connection between 2 and 4, which can also be inferred if we know that 2 and 3 have overlap of 5 and 3 and 4 have overlap of 5.

So, from the information here, we can infer this edge. So, this edge can also be removed. And finally, we have one more edge, which is this one here, ok? So, this one we can also infer, right? So, what you see is that if we know there is this overlap here 5, then you have 5 and 5 here. It will be the case that this node and node 1 and node 4 will share 3 base overlap, ok?

If you can see again because of the constraints, this is this will that can be inferred from this graph, ok. So, this also becomes a transitively inferred edge. So, we can remove this edge from the overlap graph. So, what you see here is that, in essence, we are now removing these edges that can be inferred from other edges in the graph.



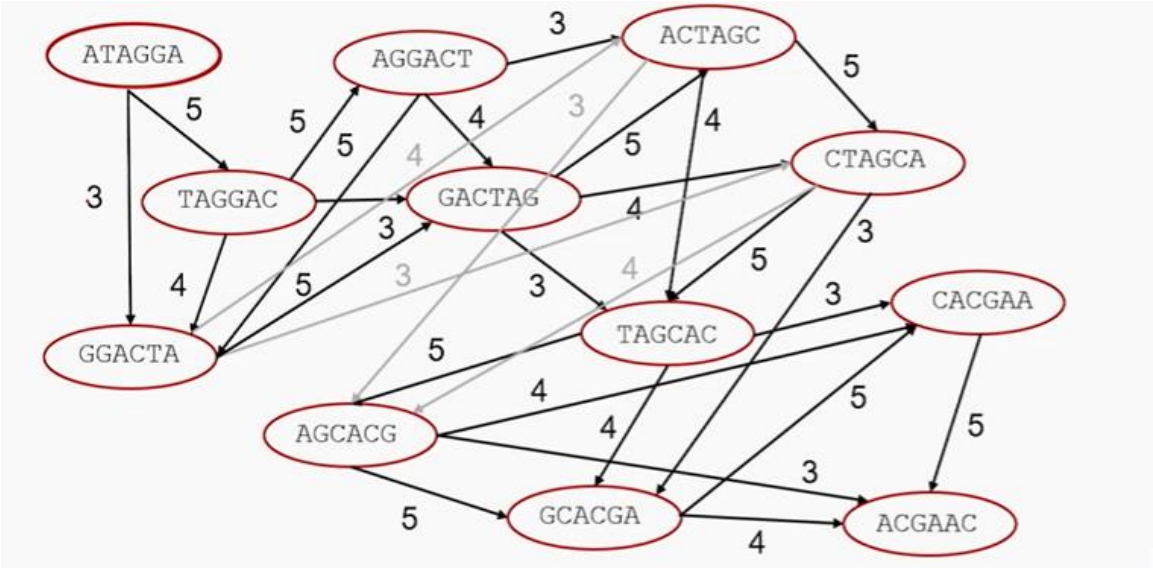
So, here we are removing these edges, right? So, these three edges we can remove, and what we are left with are these three edges here, ok? And as we traverse through this graph, if we start from here, we have a simple path, right? We do not again need to find the optimal path in this graph; we can simply traverse through this path, and this follows the Hamiltonian path, right? So, we are traversing every edge and every node exactly once, and we can derive the reference sequence, ok? So, as you can see, by just traversing through these nodes and edges, we can find the reference sequence in this for these 4 edges, ok.

So, now we can apply this idea to the full graph, ok? So, once we have understood the layout concept, we can now apply it to the full graph, ok? So, this is the full overlap graph that we built earlier. Now, we want to apply this layout method, ok? So, what we will see again if you look at this edge again, for example, is that this one can be inferred from these two connections. So, these two connections can say that these two nodes will share 4

overlap, a right overlap of 4 bases, ok.

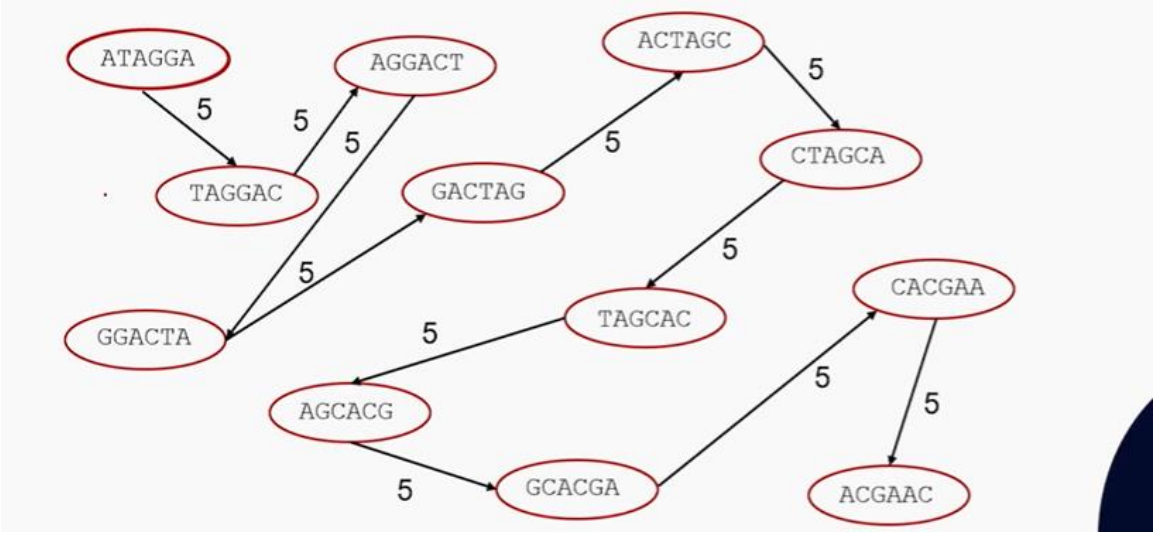
So, this means this is an inferred edge, ok? So, similarly, you can probably look for other inferred edges around the graph, ok? For example, this one here, ok, and this one again can be inferred from these two connections, ok. You have these two overlaps, each of length 5. So, this means these two nodes here will share an overlap of 4, ok? So, this edge can be inferred from these other two nodes and connections, ok?

So, this also becomes an inferred edge. Let us look at some edge where the weight is 3, or the overlap is 3 bases, ok? So, what about this one? Can this edge be inferred from the connections that we have in the graph? The answer is yes, right, because if you now let us say traverse from here to here, then to in this direction, and then in this direction, ok. So, you have three edges, right, which show five base overlaps each, and right, and they are in the same direction, and this edge can now be inferred, right? It is likely that these two nodes will show three base overlaps, ok. So, based on this logic, we can slowly remove these inferred edges, right?

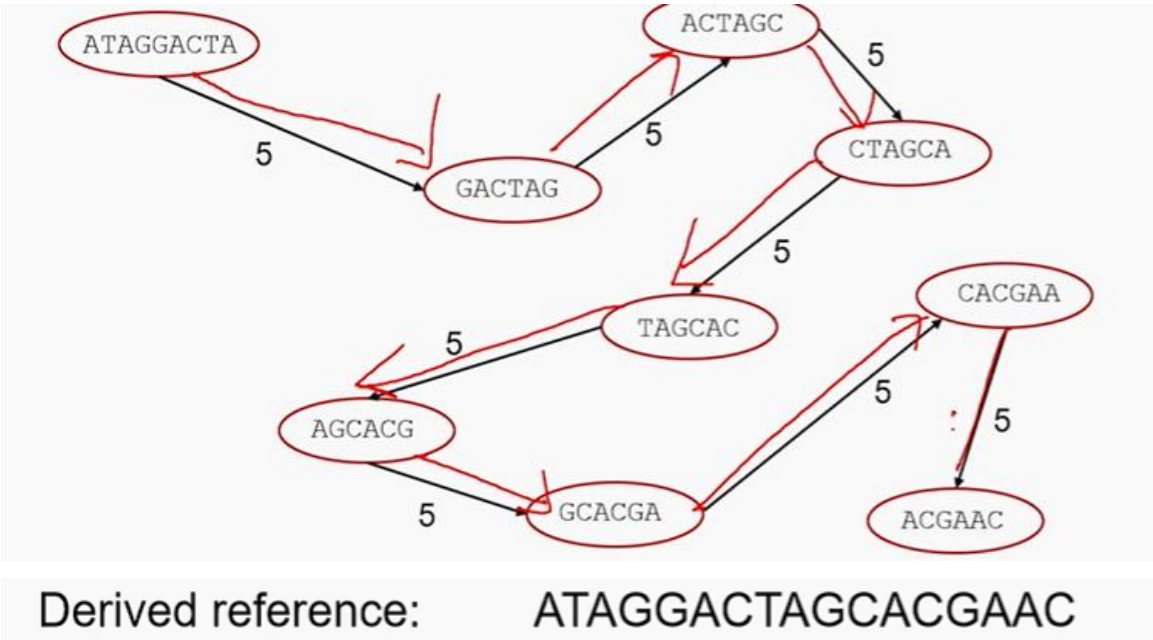


So, we can do this step by step. For example, in the first step, we can remove the ones, for example, the 4, right where we are kind of skipping one node. You can see this again in this example. For these two edges, this edge can be inferred, ok? So, this edge is skipping one node, right? As you can see from the nodes that show weight 3, they are skipping two nodes. You can see this through all the examples in this graph, and we can then slowly

remove, right, step by step, these edges that can be inferred, ok, and we will be left with only the edges or the direct connections, ok.



So, we are doing this process, right? So, we are skipping or removing these edges that skip nodes, ok, and I have removed them, and as you can see, we will be left with a very simple graph, ok. So, what I will do is urge you to actually do this with pen and paper so that you understand this concept better, ok? From the theory, you might be able to get it, but of course, if you do it yourself only, then you will realize how this works. So, now what we have is a very simple graph, and in a sense, we can simply traverse from here, right, through this graph, and we can get the reference sequence, right.



So, that is what we are going to do, right? We will traverse and we will merge, right, and this is the process, right? So, we are going to go through this, and there is only one path. We do not need to find another path. We simply need to traverse through these nodes and derive the reference sequence, and this is the derived reference as you go along these edges, right?

Also keep in mind the overlap length, right? So, that will tell us, right, how we should merge these nodes and how we can derive the reference sequence, ok? This is something that is also important, ok, and this is the reference, which is the correct reference sequence that we actually started with, ok, in this example, ok. So, I hope this is clear how the layout method actually works, ok? So, we will take another example of layout. Now, with a slightly more complex scenario, we again have taken this example before, ok?

So, here we have this very, very warm day, ok? So, why are you taking this example? Because, as you can see in this example, we have repeats, ok? So, we have these regions; this variation is repeated three times, ok? So, this will give us an idea of how this method will perform when you have repeats in the genome data, or when the different sequence contains a lot of repeats. This is again close to the real scenario, where mammalian genomes or human genomes contain a lot of repeats. So, again, we have defined the rich sequence, or we can say rich sequences of length 6, right?

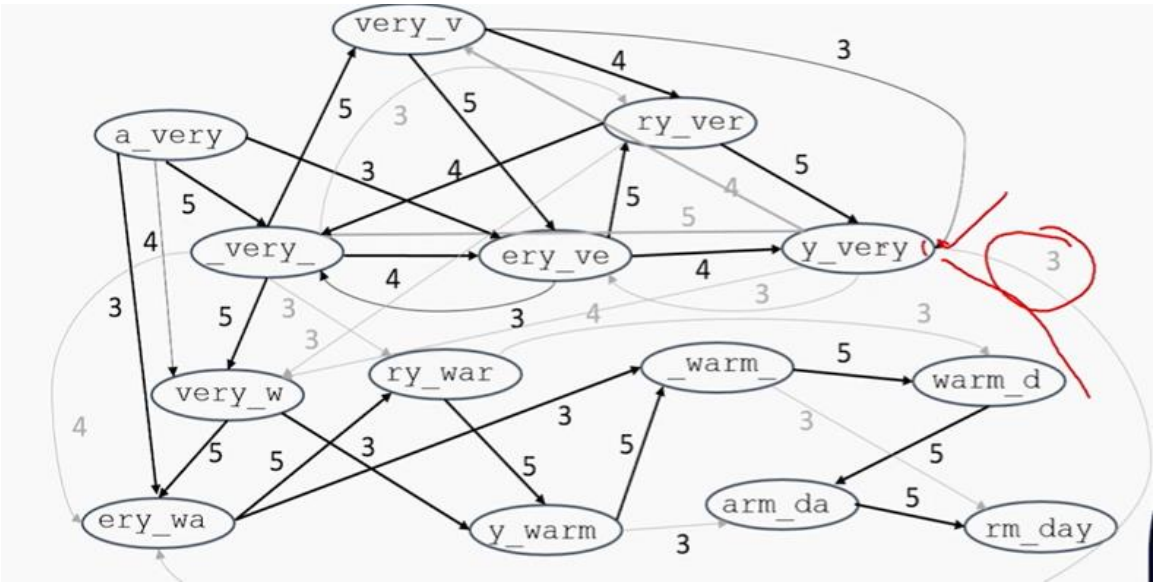
So, we have a length equal to 6 here, and we can derive all the reads, right? We are working in an ideal setting, right, where we get all possible reads in our data, which we can then assemble into the reference genome. So, here are the reads that we have, and I have built the overlap graph again. So, this will take a bit of time, right? You have to give time to understand all the connections or derive all the connections and add those edges and the weights, ok?

But again, it is a good exercise that you can do with a pen and paper, ok? So, once you have derived this, right, then we can apply this layout approach, ok, on this graph, and we can see which are the edges that we can mark as inferred edges or transitively inferred

edges, ok. So, as before, as you can see, right, if you have an edge like this one here, right, you can see this can be inferred from these two connections, ok, and this edge is skipping one node, this node, right. So, again, this is an inferred edge. So, we can remove this one, ok? So, in the first step, we can remove this one, and you can see we are starting to remove a few more as well because we know these are inferred edges, or these edges we can infer from the other connections that we have in the graph, ok?

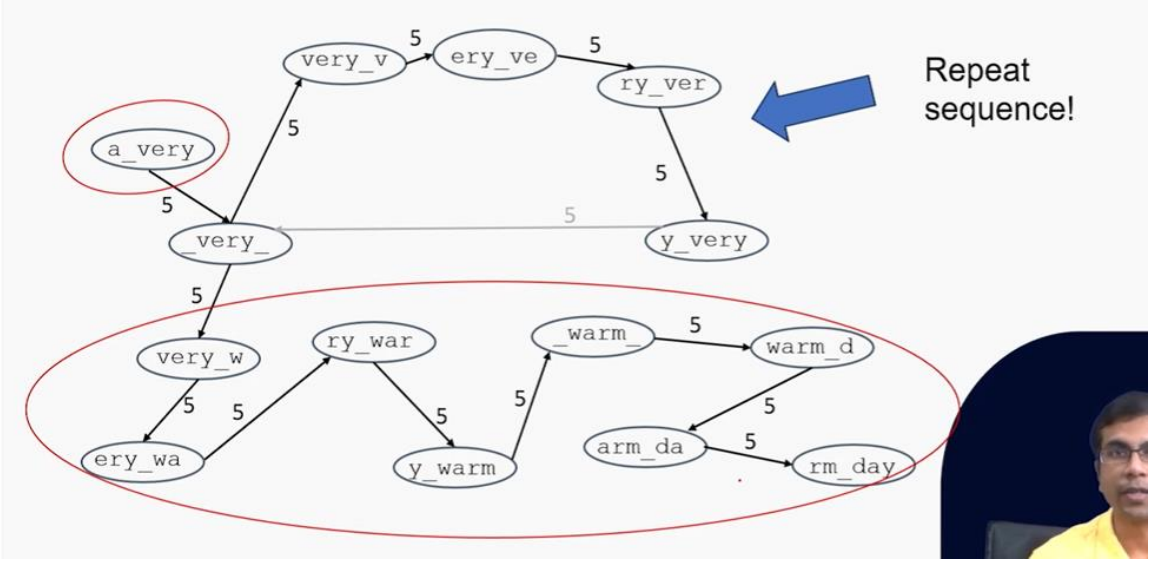
So, I can take this example for here, right, this one with 3, right, with 3 in the graph just for easy visualization, ok? So, can we infer this edge? So, this is between these two nodes here, right? So, this node and this node, ok? So, can we infer this one? So, do we have other connections through which we can infer this?

So, you have to look at this network a bit carefully, or maybe reorient this bit, right? You can infer this edge, ok? So, you have this connection here, you have this connection here, and you have this connection here, ok? You have these three direct connections, each with an overlap of 5.



So, through these three connections, you can infer this connection of weight 3. So, this connection can also be removed, right, because this is an inferred edge, ok? So, following this principle, we can now slowly remove these connections, and you can see that the graph starts to become very simple, ok? And I think we have reached a point where we cannot

remove any more connections, because these are all direct edges, ok? And what you see is interesting, right?



So, there are two parts in this graph that we can resolve very easily, right? We can traverse through this, right? So, this part is resolved on top. So, this part can be resolved, right? We know the sequence, right, and we can traverse through this and find the reference sequence, right, because there is a single path.

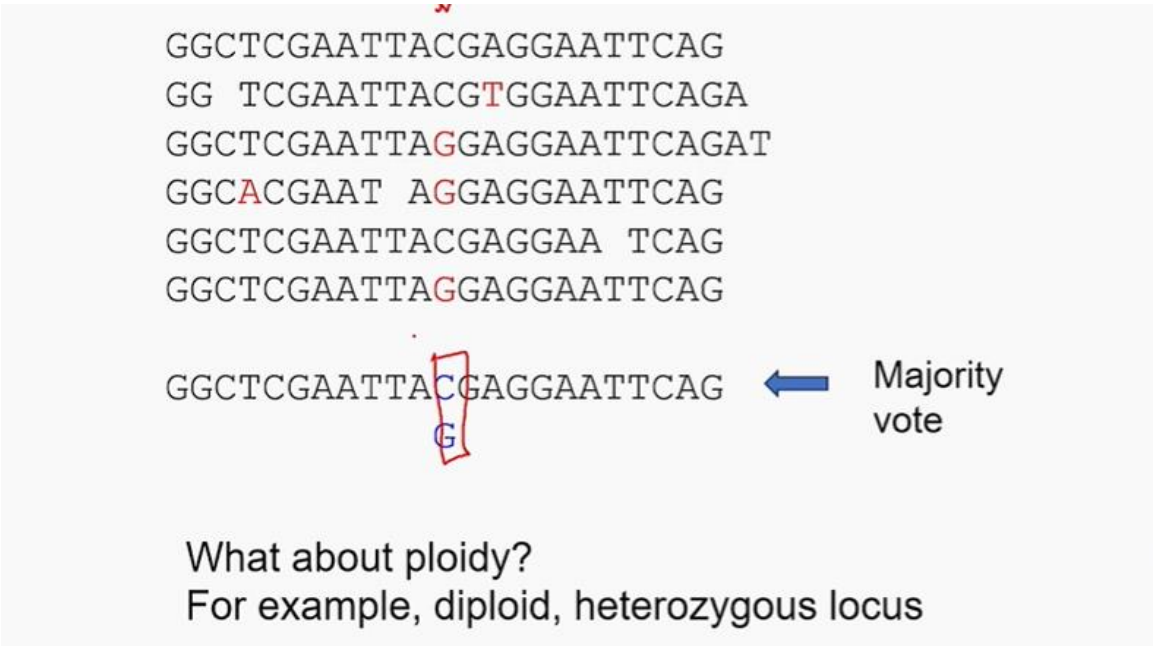
But there is another path, right, where we have this kind of cycle, ok? And this is because of the repeat sequence that we had in the data, ok? So, this cycle makes life complicated because we now have to think about it in terms of the Hamiltonian path problem. So, we have to traverse through this graph, and we have to touch every node exactly once. But with this cycle, right, how do you actually carry out this Hamiltonian path, ok, or how do you traverse this Hamiltonian path, ok?

And this is the problem with repeat sequences, ok? So, it kind of looks like a string, right? So, this part is straight, this part is straight, but here you have a loop or a node, ok, and that is causing a problem in the assembly, ok. So, as you can probably see, like SCS, you also have this problem with the Wells approach, right? So, if you have repeat sequences, they are going to hinder assembly, ok? So, these two parts are resolved, but the middle part, right, will contain something because this is a cycle and we cannot traverse through a Hamiltonian path.

The third step we have not talked about is the consensus step, right? So, overlap and layout—these two steps are done. So, the layout actually will give you the reference genome. The consensus step is actually to derive the most likely sequence for each contig. So, what we do when we reach this consensus is align sequences that make up a contig, right? So, we have determined that these are the reads that will determine this contig and then take a consensus approach, ok, and then determine sequence at each position by something called a majority port, ok.

So, this is what we do, right? So, if you can think of a specific position, right, a small position, you have these reads that are aligning, and you can see, right, some reads are showing some mismatches. For example, here you have this mismatch here, right, here, or here, and in some cases, you might also have these deletions, right, or insertions, and then what you would have to do is then determine what is the majority for each position in this read sequence, ok? So, for each position, we do the majority port, right, and for these positions where you have these variants, as you can see, right, we take the majority port and say, ok, the majority actually determines the reference genome sequence, ok. So, this is how we can get the reference genome sequence from the data.

Now, one complication that we will have is that you know about ploidy, right? So, for example, if you are working with the human genome, we are deployed, right? So, if you have ploidy and if you are working with a heterozygous locus, you have two copies, right? So, one copy is coming from the mother, and another copy is coming from the father, right? So, and there are variations you can see, right? Maybe in this position here, ok, we have two nucleotides, right?



So, one is C, another is G, and they are at equal frequency, ok? So, 50-50, or close to 50-50 in some cases, right? In that case, how do you decide the reference sequence? So, in that case, you have to actually consider that position as heterozygous, ok? And so you will have these two bases that are possible for that position, ok?

So, this is the consensus step, right? And the major drawbacks of this OLC method we have already mentioned are that building the overlap graph is time-consuming; it takes a lot of time because we are doing this pair-wise comparison across a large number of reads, and these graphs are quite big. So, when I say big, right, we have billions of reads in there, right, where each of these reads will be a node in the graph, and we would have to store this structure in the memory, right? When you are running through a computer or a server, you would have to remember this structure, and the program will generate this structure and keep it in memory for deriving the reference sequence. So, this means that in the real world, it will take a lot of memory to construct these overlap graphs, ok?

So, here are the references for this part, right? So, to summarize, the OLC method starts with the overlap graph, as we have seen. The novelty is the layout approach. So, instead of searching for the most suitable path like SCS does, the OLC method actually removes paths that can be inferred. So, it removes edges from the overlap graph that can be inferred from

other connections, and in that way, it actually simplifies the process of finding the reference sequences. So, if you have repeats, the OLC method will at least resolve certain parts in the form of contigs, and it will indicate where the repeats might be present.

So, in our simple example, we have seen that we are forming something like a loop or not in the structure of the overlap graph, right? But in the case of other types of sequences, you might see a much more complicated structure. So, what will happen at the end? You might have two contigs that can be inferred, right, or maybe more, and then in some regions you will not have any reference inferred, right? So, these will be the gaps in the reference sequence, ok? And what we have now seen is also a repeat in the assembly.

We have seen this earlier, and this is something that we will see repeatedly through our approach. Thank you very much.