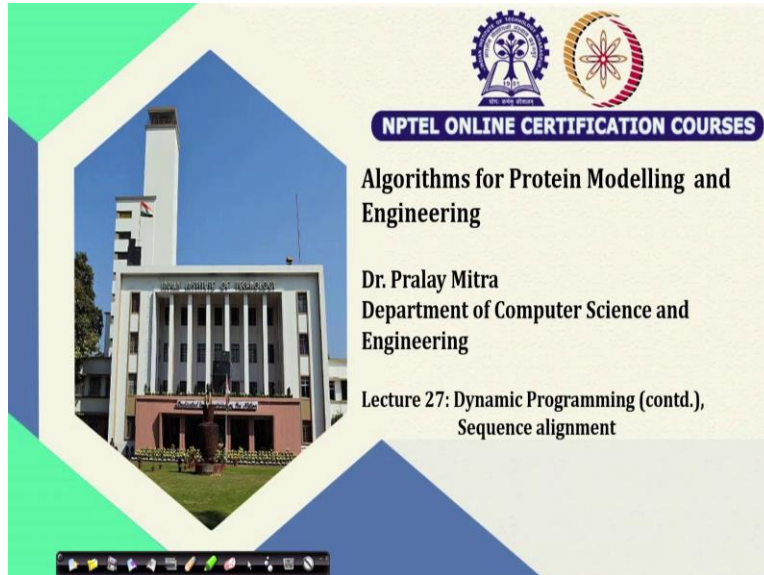


**Algorithm for Protein Modelling and Engineering**  
**Professor Pralay Mitra**  
**Department of Computers Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture 27**

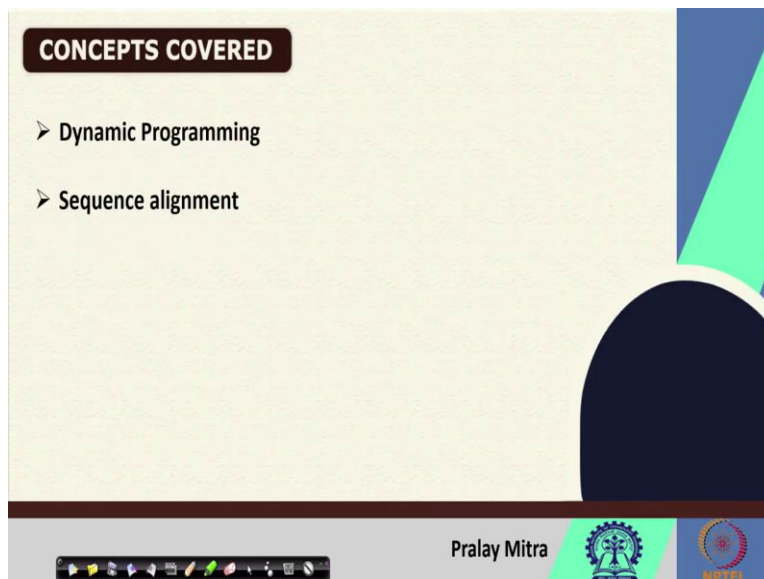
**Dynamic Programing (Continued), Sequence Alignment**

(Refer Slide Time: 00:15)



The slide features a central image of a building framed by a green and blue geometric border. At the top right, there are two circular logos: the Indian Institute of Technology (IIT) logo and the NPTEL logo. Below these logos is a blue banner with the text "NPTEL ONLINE CERTIFICATION COURSES". The main text on the slide reads: "Algorithms for Protein Modelling and Engineering", "Dr. Pralay Mitra", "Department of Computer Science and Engineering", and "Lecture 27: Dynamic Programming (contd.), Sequence alignment". A navigation bar is visible at the bottom of the slide.

(Refer Slide Time: 00:19)




The slide has a light beige background with a dark blue and green geometric design on the right side. A dark blue rounded rectangle at the top left contains the text "CONCEPTS COVERED". Below this, there are two bullet points: "➤ Dynamic Programming" and "➤ Sequence alignment". At the bottom, there is a navigation bar with the name "Pralay Mitra" and the IIT and NPTEL logos.

(Refer Slide Time: 00:25)

**KEYWORDS**

- Dynamic Programming
- Sequence alignment

Pralay Mitra



Welcome back, so let us continue with the dynamic programming and then the sequence alignment part, we will go next. So, the concept that I will be covering here the same one the dynamic programming and the sequence alignment. So, we started with the longest common subsequence problem were given two protein sequences, basically we need to find a common sub sequences among them.

(Refer Slide Time: 00:32)

**Longest common subsequence problem**

**Dynamic Programming**


Example,  
S = ABAZDC  
T = BACBAD

$$M_{i,j} = \text{MAXIMUM} [ M_{i-1,j-1} + S, M_{i-1,j}, M_{i,j-1} ]$$

Initialization  
Scoring  
Backtracking

	B	A	C	B	A	D
	0	0	0	0	0	0
A	0	0	1	1	1	1
B	0	1	1	1	2	2
A	0	1	2	2	2	3
Z	0	1	2	2	3	3
D	0	1	2	2	3	4
C	0	1	2	3	3	4

Pralay Mitra



### Longest common subsequence problem

**Dynamic Programming**

Example,  
 $S = \text{ABAZDC}$   
 $T = \text{BACBAD}$

$$M_{i,j} = \text{MAXIMUM} [ M_{i-1,j-1} + S, M_{i-1,j}, M_{i,j-1} ]$$

		B	A	C	B	A	D
		0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	4
C	0	1	2	3	3	3	3

Initialization  
 Scoring  
 Backtracking

Pralay Mitra

### Longest common subsequence problem

**Dynamic Programming**

Example,  
 $S = \text{ABAZDC}$   
 $T = \text{BACBAD}$

$$M_{i,j} = \text{MAXIMUM} [ M_{i-1,j-1} + S, M_{i-1,j}, M_{i,j-1} ]$$

		B	A	C	B	A	D
		0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	4
C	0	1	2	3	3	3	3

Initialization  
 Scoring  
 Backtracking

← S

← T

A  
B A

Pralay Mitra

So, we discussed about this matrix filling, so in the dynamic programming process, so there are three steps I mentioned, one is the initialization where we augmented, we declared the matrix and then we augmented the matrix with the size  $m + 1, n + 1$  cross  $m + 1$  where  $n$  and  $m$  are the number of characters in two sequences or two strings  $S$  and  $T$  that I mentioned.

After the initialization phase the scoring is also done, and during the scoring the matrix we have used is mentioned here it is  $M_{i,j}$  equals to maximum of  $M_{i-1,j-1}$  which means diagonal position plus  $S$ , which means the match or mismatch score. So, for the timing our score  $S$  either 1 or 0, 1 indicates it is a match, 0 indicates it is a mismatch.

Then another we are considering from the top that is  $i-1, j$  another we are constrained from the left that is  $i, j-1$  in the same row one column left. So, among these three whichever is the maximum with that value, I am filling the  $i$ th and  $j$ th cell position, while filling that one it may possible again, I am mentioning that one out of those three numbers from which I am actually computing the maximum there may be more than one numbers with the same value and that value is going to be the maximum.

Maybe all three with the same value, maybe two with the maximum values. So, in those cases I will pick any one of them arbitrarily. I also mentioned during the scoring phase to perform one additional task that is keep a track of the information that from which cell actually I am coming to this cell, I mean out of three maximum values which value has been picked, if I keep an information from there, I mention the trace backing will be easy for you.

Finally, this way if I go after the matrix filling, I will reach to the last cell position that is my cell  $m, n$ , assuming I have started from  $i$ th row and sorry  $0$ th row and  $0$ th column. So, this is my  $m, n$  position, now from here or say I can say better right  $n, m$ , because that is the way I am mentioning this  $n, m$ , position. Now, if you remember that while filling this matrix, I keep one information from where it is coming.

So, again if I take the example of this one on the last slide on the last lecture, I took the same one. So, here you can see that one value will come from here, another will come from here plus  $A$  and  $Z$ , there is a mismatch, so it is with  $0$ , another will come from say sorry there will be no plus here it will be directly  $2$ . So, out of these three, clearly these  $3$  is maximum and I am putting it here.

So, on the last lecture I mentioned that you keep an information of this arrow I mean from where it is coming. Now, if you keep this information, the advantage in your trace back will be you just reverse this arrow, so you erase this part and you reverse it and your trace backing is done. So, that way you will keep on trace backing. Now again, when you are trace backing, then there may be possibility that three or two different possibilities are there, you can pick any one of them out of this.

(Refer Slide Time: 04:49)

**Longest common subsequence problem**  
**Dynamic Programming**  
 Example,  $S = \text{ABAZDC}$   
 $T = \text{BACBAD}$

$M_{i,j} = \text{MAXIMUM} [M_{i,j-1}, M_{i-1,j}, M_{i-1,j-1}]$

Initialization  
 Scoring  
 Backtracking

		B	A	C	B	A	D
	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	4
C	0	1	2	3	3	3	3

Handwritten annotations on the slide include:  $S = \text{ABAZDC}$ ,  $T = \text{BACBAD}$ ,  $M_{i,j} = \text{MAXIMUM} [M_{i,j-1}, M_{i-1,j}, M_{i-1,j-1}]$ , and a green box containing "Initialization", "Scoring", and "Backtracking". Blue arrows on the table trace the path from (4,8) to (3,7) to (2,6) to (1,5) to (1,4) to (1,3) to (1,2) to (0,1). To the right, "A B" is written, and "←S" and "←T" with arrows indicate backtracking directions.

(Refer Slide Time: 07:11)

$S = \text{ABAZDC}$   
 $T = \text{BACBAD}$

Handwritten annotations include:  $S = \text{ABAZDC}$ ,  $T = \text{BACBAD}$ , "A B", "B A C", "A B", "←S", and "←T". A diagram shows a grid of circles representing the DP table, with blue arrows tracing the path from the bottom-right corner towards the top-left corner, corresponding to the backtracking path in the slide above.

# Longest common subsequence problem

## Dynamic Programming

Example,  
 S = ABAZDC  
 T = BACBAD

$$M_{i,j} = \text{MAXIMUM} [ M_{i-1,j-1} + 1, M_{i,j-1}, M_{i-1,j} ]$$

Initialization  
 Scoring  
 Backtracking

	B	A	C	B	A	D
A	0	0	1	1	1	1
B	0	1	1	1	2	2
A	0	1	2	2	2	3
Z	0	1	2	2	2	3
D	0	1	2	2	2	3
C	0	1	2	3	3	3

← S  
 ← T



Pralay Mitra



# Longest common subsequence problem

## Dynamic Programming

Example,  
 S = ABAZDC  
 T = BACBAD

$$M_{i,j} = \text{MAXIMUM} [ M_{i-1,j-1} + 1, M_{i,j-1}, M_{i-1,j} ]$$

Initialization  
 Scoring  
 Backtracking

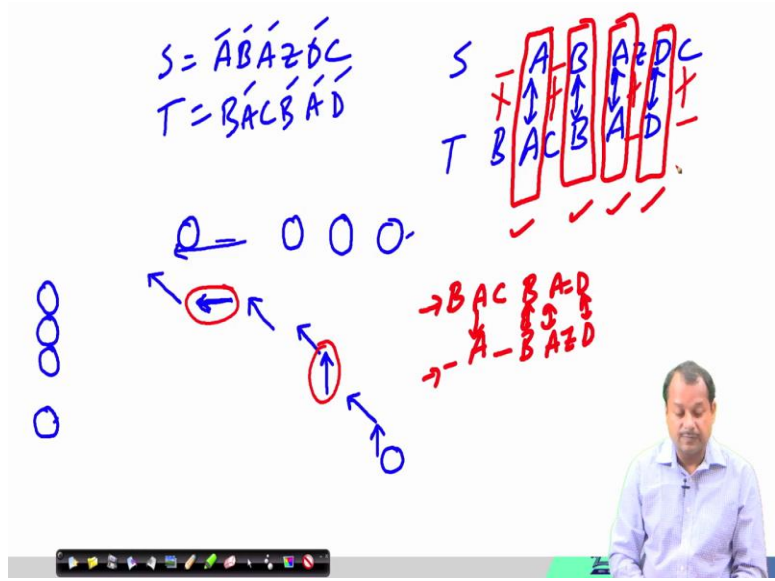
	B	A	C	B	A	D
A	0	0	1	1	1	1
B	0	1	1	1	2	2
A	0	1	2	2	2	3
Z	0	1	2	2	2	3
D	0	1	2	2	2	3
C	0	1	2	3	3	3

→ B A C B A Z D  
 → A B A Z D



Pralay Mitra





That way if I start a fresh for this for backtracking or trace backing, then this is my starting position that is my 4, and clearly, I can see among these three this is with the highest value, so it definitely has come from here, so this is my trace back. Now, this 4 is from where this, this or this now D and D was basically D and D was basically match because of this one, three was added with one that means it has come from here, so this is my trace back.

Now out of these three now this A and Z, so this is 3, so it is coming from here now 2, 2, 2 and 3 definitely it is coming from here, then 1, 1, 1 but it is 2 it is coming from here, then 1, 1 sorry 1, 0, 0, so among these three this is coming from here, now from this 1 all are 0, so because of the match that is from here. So, this is my trace backing, and if I follow this trace backing then I will have that alignment too.

So, what will be my alignment? Now, you see that this B after that one the from this matching A and A it has come, so this BACBD and ABAZDC, so A then B was there, so here A is matching BA after that one, so whenever I am basically you see that whenever I am going to left which means my column is changing but row is not changing, which means the character in the column will be changed and if the column actually is represented by T, so the character in T, so if I say this is my T this will be changed because of this movement, but for this S it will not be changed.

Whereas, if I go for this upward movement, then the rows will change, but column will be fixed which means this S which is actually represented here, that will change, but the column will be fixed, that way for this ABAZDC. So, S equals to ABAZDC and T equals to BACBAD,

BACBAD. So, what I can have is say this A is matching with this A, BA, then this is my B this is my, this is my C, this is my C, then 4 is matching.

So, you can see that who else will be matching here, so matching positions, so this A is matching with this A, this is diagonal movement, this is another diagonal movement this B is matching with this B, then this is another diagonal movement this A is matching with this A and then this is another diagonal movement, which means these D is matching with this T. So, here ABA and D is matching.

So, this ABA and D is matching with whom? So, ABAD, ABAD that means that if I, if I look for that alignment, then what I can see is that so there are two strings say S and T. Now A B A, so here B is not matching, A is matching, then C is not matching, then B is matching, then A is matching, then D is matching, but Z is not matching, then D is matching, then C is matching.

So, this A is matching, B is matching, A is matching, D is matching. And who is not matching? So, because of that I am putting one blank here. So, this B is not matching, this C is not matching, this Z is not matching, this C is not matching, so red color is not matching, and for that I am putting one hyphen or underscore indicating it is not matching.

Now, if you go back then we can see that how many diagonal movements, non-diagonal movements are there? Here is one, here is one, these two non-diagonal movements are there, now because of that following that alignment, so this trace backing indicates that what is my alignment? These trace backing indicates that my alignment is, so what I did was one possible alignment that is there.

Now, another possible alignment is say B A then this A is matching, so A, but C is not matching, so B has gone outside, so C is not matching, so there will be a blank, then B A D and here B A, but here there will be AZ and D. So, this is matching, this is matching, this is matching, this is matching, here is a blank here is a blank, and here there is a blank. So, this is that trace backing and the alignment.

And after that alignment, what you will get this kind of alignment, now if you go back, you see that because of that alignment, when now come back to our structural alignment and for the structural alignment I mentioned two things are important, one is that correspondence and after that correspondence, you have to align those two structures.



Now in the correspondence part, now I have the correspondence, given these two strings, I have identified their longest common subsequence and I will say that, so this is one correspondence, this is one correspondence, this is one correspondence, this is another correspondence. Now regarding others, so there are some blanks in another cell and there is no correspondence, it is fine, so there is no correspondence.

Now, if you assume that S and T represents two protein structures having two different protein sequences then, after the structural alignment, again structural alignment we did not discuss I will come shortly to that one, after the structural alignment in order to check that what is the correspondence between which two say amino acids, I will compute the root mean square deviation, then this alignment will give you that information.

So, this is one of the applications also of the dynamic programming in case of structural alignment, but this is not all actually dynamic programming has a lot of other applications also specifically in the sequence alignment part, we discussed these multiple possibilities here. So, solution is that we have to check what is best. Now, I believe that based upon the discussion that we have done, you can go ahead and implement that one for any language.

(Refer Slide Time: 13:15)

**Longest common subsequence problem**  
**Dynamic Programming**

Example,  
 $S = \text{ABAZDC}$   
 $T = \text{BACBAD}$

Space complexity:  $O(N^2)$   
Time complexity:  $O(N^2)$

	B	A	C	B	A	D
	0	0	0	0	0	0
A	0	0	1	1	1	1
B	0	1	1	1	2	2
A	0	1	2	2	2	3
Z	0	1	2	2	2	3
D	0	1	2	2	2	3
C	0	1	2	3	3	3

*(m+1)(n+1)*  
*m=n*

Pralay Mitra

Now, we are talking about this complexity, as you can see, in order to fill the matrix, I need to define the matrix which means, the space requirement is there and space requirement is  $m + 1$  multiplied with  $n + 1$  which means  $m \times n$  or in general, so  $m + 1$  it is basically sorry, so it is

basically size of the matrix and the size of the matrix is nothing but  $m + 1$  multiplied with  $n + 1$ .

Now assuming  $m$  equals to  $n$ , so you can consider that this is nothing but  $n$  square that is here, that is my space requirement. And in order to process that matrix during the scoring and during that trace backing, then the time complexity is also order of  $n$  square, that is also order of  $n$  square. So, in both the cases it is order of  $n$  square. So, space complexity as well as the time complexity.

(Refer Slide Time: 14:33)

The slide is titled "Sequence Alignment" in blue text at the top center. It contains two main bullet points: "Pairwise" and "Multiple". Under "Pairwise", there are three sub-bullets: "DOT matrix", "Dynamic programming" (with a red checkmark), and "Word method (efficient heuristic method; e.g., BLAST)". Under "Multiple", there are four sub-bullets: "Dynamic programming", "Progressive method (e.g., CLUSTAL, T-Coffee)", "Iterative", and "Motif finding". A small video inset of a man in a light blue shirt is visible on the right side of the slide. At the bottom, there is a navigation bar with the name "Pralay Mitra" and two logos.

- Pairwise
  - DOT matrix
  - Dynamic programming ✓
  - Word method (efficient heuristic method; e.g., BLAST)
- Multiple
  - Dynamic programming
  - Progressive method (e.g., CLUSTAL, T-Coffee)
  - Iterative
  - Motif finding

Now, this sequence alignment can be done in several ways. So, we discussed only one that is dynamic programming, but it can be done using dot matrix, it can be done by word method, so these dynamic programming we have discussed for pairwise, I mean given  $S$  and  $T$ , there is another level of sequence alignment when I am actually aligning multiple sequences, there also I can exploit dynamic programming, some progressive methods like CLUSTAL, T-Coffee, some iterative or motif finding methods are also there, we are not going to discuss all those techniques.

(Refer Slide Time: 15:20)

**Applications**

- Alpha hemoglobin and beta hemoglobin are subunits that make up a protein called hemoglobin in red blood cells. Notice the similarities between the two sequences, which probably signify functional similarity.
- Many distantly related proteins have domains that are similar to each other, such as the DNA binding domain or cation binding domain. To find regions of high similarity within multiple sequences of proteins, local alignment must be performed. The local alignment of sequences may provide information of similar functional domains present among distantly related proteins.

Pralay Mitra

NPTEL

Now, there are several applications of these dynamic programming. So, these are say sequence alignment problem in general. In sequence alignment pairwise or in multiple sequence alignment. So, say one such application is like alpha hemoglobin and beta hemoglobin are subunits that make up a protein called hemoglobin in red blood cells.

Notice the similarities between the two sequences which probably signify functional similarities, many distantly related proteins have domains that are similar to each other such as the DNA binding domain or cation binding domain. To find regions of high similarity within multiple sequences of proteins, local alignment must be performed. The local alignment of sequences may provide information of similar functional domains present among distantly related proteins.

So, basic idea here is that when say I am trying to understand the functionality of two (prop) of two protein and for me one function is known then it might be good idea to go for pairwise sequence alignment and if I see that there is significant match in their alignment, I mean the score value is very high, then I can also infer that probably two sequences are same. So, if I know the function of one protein sequence, then I can infer that okay the second protein sequence may also have the same function. So, that is the basic idea.

(Refer Slide Time: 16:54)

**Dynamic Programming**

S = BACBAD  
T = ABAZDC

Initialization

Matrix Fill

Score function  
 $M_{ij} = \text{Max} [ M_{i-1,j-1} + S_{ij}, M_{i-1,j}, M_{i,j-1} ]$

Traceback  
 $Tb_{ij} = \{ \swarrow, \uparrow, \leftarrow \}$

Gap Penalty:

Opening
Extension
Combined

Handwritten notes: 20 AA, 20x22, 20x20, 2, A → G, A = A 1, A → C 0, A → W 0, A → K 0

Pralay Mitra

Now, the dynamic programming that we have used was the simplest one, because the match and mismatch score was only considered which means that if there is a match say AA then I say 1, if it is AB then I say 0, that is one thing. Another thing is that when say I am moving along the row or along the column, then I mentioned that there is no such penalty, it is fine, so there is no match.

That way, sometimes it may be misinterpreting the protein alignment, the part, the reason specifically is that say when I say that A equals to A, A equals to A, absolutely no problem, so it can have a score function say 1. But when I say A equals to so, A is trying to align with B not equals to, then I am telling 0, and when I say that say instead of B, let me consider one amino acid actually say C, C is cysteine.

So, A is aligning and C is cysteine then I am telling 0, and when I say A is trying to align with say W, W tryptophan, which is the bulkiest one, then I am telling 0, when I it is trying to say align A with say K, I mean aligning with the lysine then also I say 0. Now, you see that alignment of alanine with the cysteine, alignment of alanine with the lysine, alignment of alanine with the tryptophan, so from biochemical point of view, they are completely different.

But what I am doing? So, I am putting 0 for everybody. So as if I am considering A C W K all are with equal probability it is occurring, but actually not, during the evolution process in the biology or in the biochemistry show there occurrences will vary, it is not same. And that way I

cannot say that when say A is trying to align with C to be 0, A is trying to align with W to be zero C, say A is trying to align with K that is all are 0's, I cannot say.

So, there must be some variations to incorporate the information that, so there is a very good chance that say A will be say aligned with Z, alanine and glycine because there is only small difference at the side chain that is say CH<sub>3</sub>. Now, the chances for A and W is very less, because one is the smallest one and other is the bulkiest one.

So, in order to differentiate between these different alignment, so what we need, we need to have some varied score function and ideally, so since there are 20 different amino acid so probably so 20 cross 20, or 20 cross 20 divided by 2 number of possibilities will be there. So, this 20 cross 20 is required, if I assume that during the evolution process the probability of changing A to G is different compared to G to A, then it will be 20 cross 20.

So, it is not symmetric property, but if it is a symmetric property, I mean probability of during the evolution probability of converting A to G is same as G to A in that case it is 20 cross 20 divided by 2. But those many number of different score functions are required. And that requirement will be placed as the score function here, that is why now I modified my score function  $M_{i,j}$  where the first modification I did was  $S_{i,j}$  instead of S. Last time it was S, it was assuming binary value either 0 or 1, 0 means mismatch, one means match, now, I am telling  $S_{i,j}$ .

Now, this  $S_{i,j}$  will signify that what is the character at the *i*th position, what is the character at the *j*th position in the respective string and from one score function say 20 cross 20 or 20 cross 20 by 2 for protein sequence alignment, you pick that probability value and you put that score value here. So, that is the first modification I did in my dynamic programming in order to make it more practical for the biological use rather than it is just a theoretical algorithm.

The next one is that gap penalty. Now, what is gap? Gap indicates when I say move either this way or this way, so vertically up which means row is changing, but column is fixed, which means in one string I am changing, but another string there is no change means I am introducing hyphen, hyphen, hyphen, hyphen in my alignment previous lecture, or when I am moving say left then also the column is changing, changing, changing, row is not changing, row is fixed.

Which means the string at the row th position is getting hyphen, hyphen, hyphen, hyphen and in the column position it is changing, it is fine. But you know that in some positions of a protein it

is a good idea that some gap may be introduced I mean that is allowed vertically up or left but, in some position, it may not be, that is one thing.

(Refer Slide Time: 23:04)

Handwritten notes on a whiteboard showing sequence alignment and calculations. The top part shows two sequences: AACCTAG and AAGGGAG AAT. A red box highlights a gap in the first sequence. Below, a sequence A-A-C-C-T-A is aligned with A G A C C A C T A A. To the right, there are calculations: 20 AA, 20x22, 20x20, and 2. At the bottom, there are handwritten notes: A -> G, A = A, A -> C, and A -> W.

Another is that in protein sequence alignment, when say you try to introduce some gap, I mean vertically up or down once it will be introduced one gap then let it be compared to frequently you are changing. What I am trying to say is say I am doing one alignment say AA say CCTAG and then and in another sequence, it is something say AACCTAG then here say AAT and now here AAGGGAG.

So, this kind of alignment is allowed, but, if I say then another one is say AGACCACTTAA, so this has a problem, this has a problem, the problem as you see that frequently it is breaking. Now, in this case, this break may mean something to you and that meaning is for this break this mean this can have some meaning to you and the meaning maybe during some process either experimental or evolution or some process probably that region has deleted or that has missed that may be one inference.

But what is the justification for having so many small, small gaps between two characters or two amino acids as such that there is no such justification. So, that is why you understand that this I can prefer even there are 10 gaps, I mean 10 hyphens, but these I cannot prefer where even there are say only 5, 6, gaps.

(Refer Slide Time: 25:08)

**Dynamic Programming**

S = BACBAD  
T = ABAZDC

Initialization

Matrix Fill

Score function  
 $M_{ij} = \text{Max} [M_{i-1,j-1} + S_{ij}, M_{i-1,j}, M_{i,j-1}]$

Traceback  
 $Tb_{ij} = \{\nwarrow; \uparrow; \leftarrow\}$

Gap Penalty:

- Opening
- Extension
- Combined

Handwritten notes: AACCTT, AAA, opening, extension, -10-4 = -14, -10-1 = -11, 20x20/2 + 20 = 220.

So, to deal with that situation this gap penalty has been divided into three parts, one is opening, another extension, another combined. So, in some algorithm you may have that combined means that there is only one penalty and whenever say there is a gap you just put it, another is that whenever there is a gap and it is just opened, I mean that you we did that alignment, so, AACCTT I am not putting the next one and another possibility was say like this.

Now, here opening indicates these penalties, so when the gap is first started and extension indicates this is my opening, after that one it is extended, in this case it is opening, it is opening, it is opening, it is opening. Now, clearly you understand if you wish to avoid too much gap opening then what you can give for this opening, you can put a very high value say minus 10 and for this extension what do you can give say minus 1 only.

So, two opening in this case will put a penalty of minus 20, whereas in this case one opening is minus 10, 1, 2, 3, 4, so minus 10 minus 4 which means minus 14 in this case, in this case it is 20 which means only two gaps that is not allowed because both are opening. So, that way gap penalty may have three different meaning. So, opening extension and combined.

Specifically, in combined it can have one actually value for this opening and extended both and when say you are having opening and extended sometimes this opening also indicates that at least one gap is introduced, so this may incorporate one extension along with it. So, when I introduced one opening which means extension extra addition is not required.

So, that way you may consider that actually opening is minus 9 and because of one extension since I opened and then it is extended, so, one more minus 1 is added, so in total it is minus 10, or you can consider that with this 1 minus 1 again and then it is minus 11. So, in that way it will be minus 11, but here it will be 14.

So, 1, 2, 3, 4, so 11, 12, 13, 14 this is 14 and here it is sorry, it will be minus 22, yes minus 11 minus 11, minus 22. So, these two modifications we have done on our basic dynamic programming algorithm, one is the introduction of a score function, which indicates in protein sequence alignment, the position two different amino acids position may not have the same probability of the score function that is one thing.

Another thing is that when I am introducing some gap, which means I am moving from row wise or moving column wise during my trace back then some underscore or gap has been introduced, so that underscore or gap for that what I am doing is I am introducing some penalty, I am not allowing that one definitely best thing is that if there is nothing on and if nothing occurs like that, but it may also possible something may occur, if occurs I will try to minimize that one by giving some penalty.

And following the biological rule the score function I have designed, following the biological rule what I will design this opening extension penalty also. Biology says that, so for two different amino acids their probability will be different, accordingly score function will be different. Since 20 different amino acids are there, so theoretically either 20 cross 20 such score values will be there or assuming that the operation is symmetric in nature during the evolution process 20 cross 20 divided by 2, so that will be, but diagonal position will be there.

So, I mean it will always actually 20 cross 20 divided by 2 along with that 20 more will be added that diagonal position, so that I believe you can understand basically we have to take the lower triangular or upper triangular matrix, I will show you that shortly. So, that is one thing and this regarding the opening an extension, opening is allowed, but once it is opened allow as much extension as it want, but do not go for frequent opening.



(Refer Slide Time: 30:10)

**Alignment Score Function**

$$W(k) = c_{\text{open}} + c_{\text{length}} * k$$

$$M_{i,j} = \text{MAXIMUM} [$$

$$M_{i-1,j-1} + S_{i,j},$$

$$M_{i-1,j-k} + w(k) \text{ (k = 1, ..., j-1),}$$

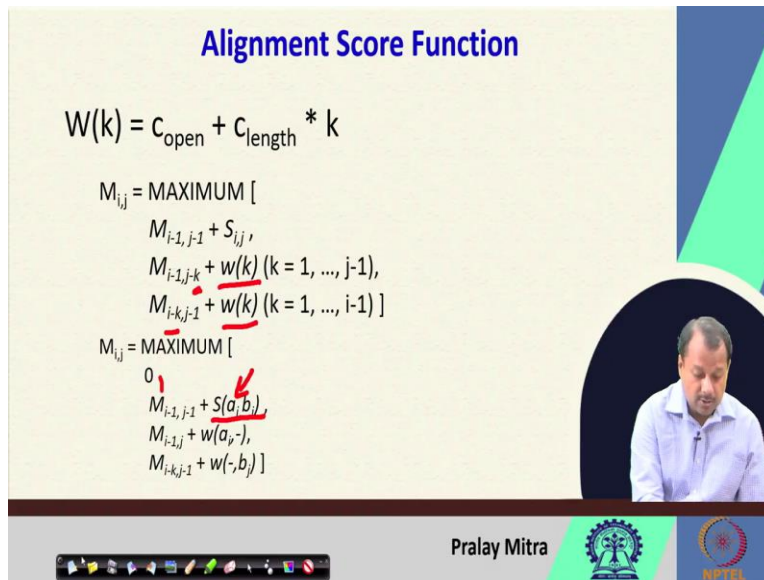
$$M_{i-k,j-1} + w(k) \text{ (k = 1, ..., i-1) ]}$$

$$M_{i,j} = \text{MAXIMUM} [$$

$$0,$$

$$M_{i-1,j-1} + S(a_i, b_j),$$

$$M_{i-1,j} + w(a_i, -),$$

$$M_{i-k,j-1} + w(-, b_j) ]$$


Now, if I vary that alignment score and if I write  $W(k)$  equals to  $C_{\text{open}}$  plus  $C_{\text{length}}$  multiplied with  $k$ , where  $k$  is basically the length of my gap and if I modify the score my matrix filling equation that maximum of  $M_{i-1, j-1} + S_{i,j}$ ,  $M_{i-1, j-k} + W(k)$ , so you note it down this  $k$  here and  $W(k)$ .

So, when  $k$  number of gaps has been introduced or  $i - k$  and  $W(k)$  when  $k$  number of gaps are introduced along the row and column, then I will have one such alignment score value and with this score value, I will have one alignment, if I along with this if I make another change to the maximum of where  $0$  has been incorporated after this  $0$  sorry there will be a comma.

So, one comma is introduced, so  $S(a_i, b_j)$  this is the score function, then  $i - 1, j - 1$  then hyphen,  $M_{i-1, j-1} + S(a_i, b_j)$ , if it introduced this one then you see that we are utilizing the score function  $S(a_i, b_j)$  or  $S(a_i, -)$  and then we are incorporating the wait for this gap penalty combining these two information we are having two different score functions, they will signify two different kinds of alignment, what is that? I am going to that.

(Refer Slide Time: 31:46)

### Global Pairwise Sequence Alignment


- Needleman-Wunch

$$W(k) = c_{\text{open}} + c_{\text{length}} * k$$


*NW*

$$M(i,j) = \text{MAXIMUM} [$$

- $M_{i-1,j-1} + S_{ij}$  (substitution matrix),
- $M_{i-1,j-k} + w(k)$  ( $k = 1, \dots, j-1$ ),
- $M_{i-k,j-1} + w(k)$  ( $k = 1, \dots, i-1$ ) ]



Pralay Mitra



So, for this the first one it is global pairwise sequence alignment. So, which will try to align two sequences which are optimized globally. And it is called as that Needleman Wunch algorithm, you probably heard the name, sometimes it is also called as the NW algorithm and at the core it is basically this maximum of this.

(Refer Slide Time: 32:20)

### Local Pairwise Sequence Alignment


- Smith-Waterman

$$W(k) = c_{\text{open}} + c_{\text{length}} * k$$


*SW*

$$M(i,j) = \text{MAXIMUM} [$$

- 0,
- $M_{i-1,j-1} + S(a_i, b_j)$  (match/mismatch),
- $M_{i-1,j} + w(a_i, -)$ ,
- $M_{i,k,j-1} + w(-, b_j)$  ]



Pralay Mitra



Next, what is about the second one, it is the local pairwise sequence alignment locally it tries to optimize that one you remember that there was one 0 incorporated here, so there is there will be a one comma here and  $S(a_i, b_j)$  for the match and mismatch that corresponding to that there will be

one score function and it is called as the Smith Waterman in short SW or local pairwise sequence alignment.

So, these are the two different alignment techniques locally and globally and you understand starting from the dynamic programming algorithm how we arrived at this position where the biological information like evolution and their say probability of opening closing, what is the gap opening penalty, closing penalty, extension penalty and how it will vary those things we have discussed, only one thing is remaining that is how we will compute the score matrix and that we are going to discuss in our next lecture. Thank you very much.