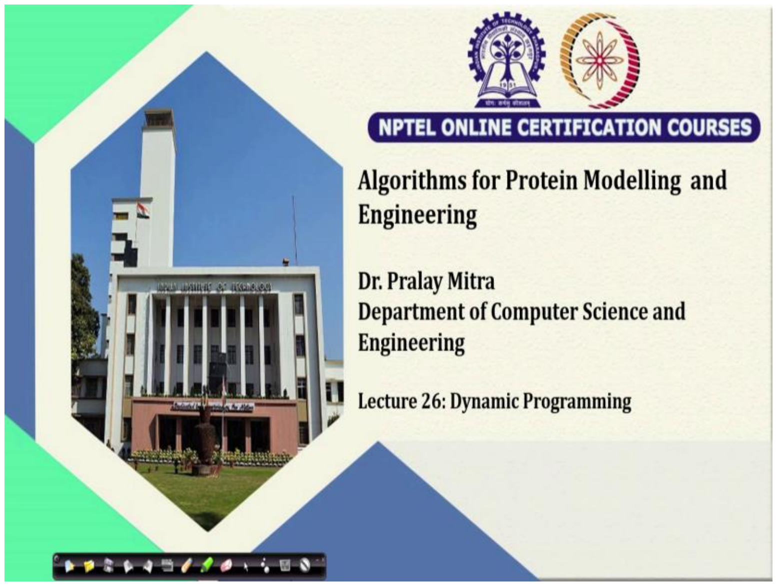
Algorithms for Protein Modelling and Engineering
Professor Pralay Mitra
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 26
Dynamic Programing

(Refer Slide Time: 00:19)



NPTEL ONLINE CERTIFICATION COURSES

Algorithms for Protein Modelling and Engineering

Dr. Pralay Mitra
Department of Computer Science and Engineering

Lecture 26: Dynamic Programming

Welcome back, so on the last week we have discussed some measure to check that whether two protein structures are aligned properly or not, so for that the simplest one in terms of the implementation and which has some impact also, so that we have discussed on the last class that is root mean square deviation.

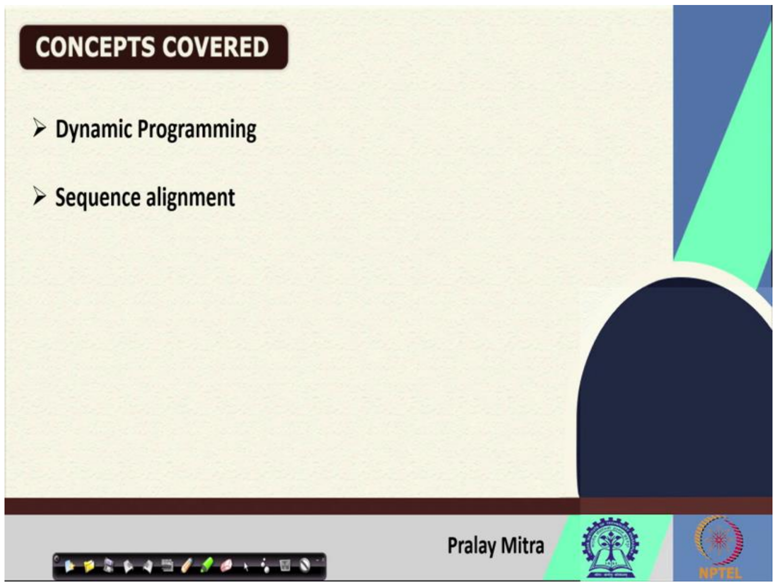
And when we are computing root mean square deviation, then backbone level, backbone trace level or all atom level computation has been done, but what I mentioned on that class is that without aligning the structure, so we cannot able to compute that RMSD, because if two structures are separately placed say like this, then if I compute RMSD that is not correct. So, we need to align those two structures.

So, that is one criteria that we deferred the discussion and I promise that that I will discuss next, another is that given two protein sequences and also when the structure is known and you are trying to compute basically the structural alignment like say RMSD then you need to know the correspondence, because between two which two atoms you will compute the RMSD that you should know.

So, specifically when it is two protein structures whose sequences are dissimilar. So, if there are two structures corresponding to one protein sequence, say let us assume that I am say designing protein folding algorithm where the input sequence is same and during the simulation process a number of say structures has been generated, some are decoy structures, which means that they are not correct but they looks like correct.

So, in case of those structures, so, correspondence is known, because it is the same sequence, but if two sequences are different and irrespective of whether those two structures are going to be the same or they are different, so we need to establish the correspondence. So, we will do that one in this week. So, let us first start with the dynamic programming, because this dynamic programming will be useful to set up the correspondence between two amino acid sequences, or two protein sequences, if they are not same.

(Refer Slide Time: 02:33)

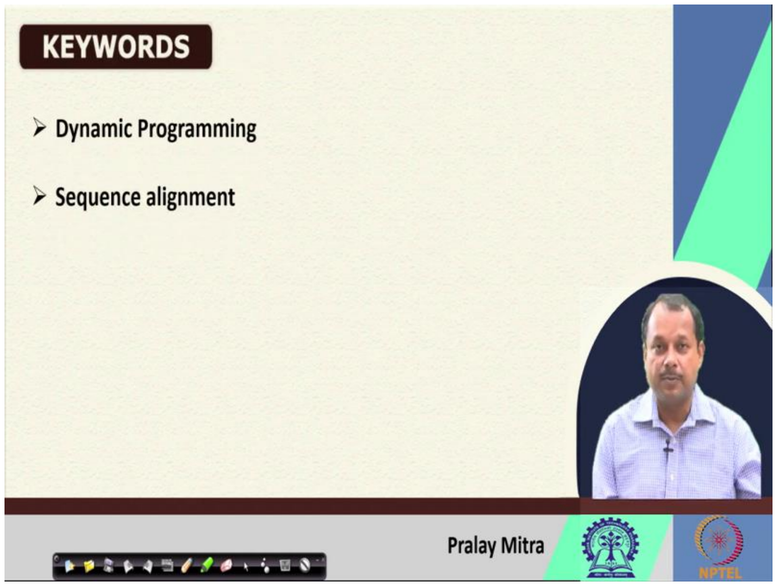


CONCEPTS COVERED

- Dynamic Programming
- Sequence alignment

Pralay Mitra

(Refer Slide Time: 02:55)



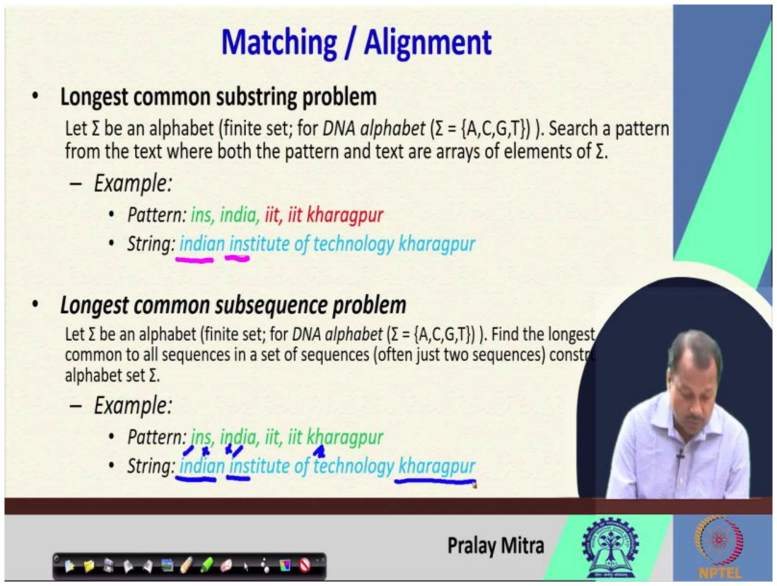
KEYWORDS

- Dynamic Programming
- Sequence alignment

Pralay Mitra

So, the content we are planning to cover is that dynamic programming and then sequence alignment. So, regarding the sequence alignment, so first we need to know that whether what is the best alignment for them and after that one, the aligned sequences from there we will identify the correspondence. So, regarding this key word I have picked the dynamic programming and sequence alignment, so both are same.

(Refer Slide Time: 03:01)



Matching / Alignment

- **Longest common substring problem**
  Let Σ be an alphabet (finite set; for *DNA alphabet* (Σ = {A,C,G,T})). Search a pattern from the text where both the pattern and text are arrays of elements of Σ.
  - *Example:*
    - *Pattern: ins, india, iit, iit kharagpur*
    - *String: indian institute of technology kharagpur*
- ***Longest common subsequence problem***
  Let Σ be an alphabet (finite set; for *DNA alphabet* (Σ = {A,C,G,T})). Find the longest common to all sequences in a set of sequences (often just two sequences) constr alphabet set Σ.
  - *Example:*
    - *Pattern: ins, india, iit, iit kharagpur*
    - *String: indian institute of technology kharagpur*

Pralay Mitra

Now, before I start actually for the sequence alignment, so I need to tell you the matching and the alignment. So, there are two terms in this context, so one is called as the longest common substring problem, another is called as the longest common subsequence problem. So, let us start with the substring problem. So, substring indicates that let sigma be an alphabet, a finite set for DNA the alphabet list is A, C, G, T, such a pattern from the text where both the pattern and text are arrays of elements of sigma.

So, that is the definition. Now, if I look at some of the examples, so the pattern is say ins, India, IIT, IIT Kharagpur and if the string which is given to you is Indian Institute of Technology Kharagpur, then you can identify them. So, inside this Indian Institute of Technology Kharagpur, now if you look then ins is present here, ins is present here, then India is present here, then IIT is present where? IIT is not present, IIT Kharagpur that is also not present.

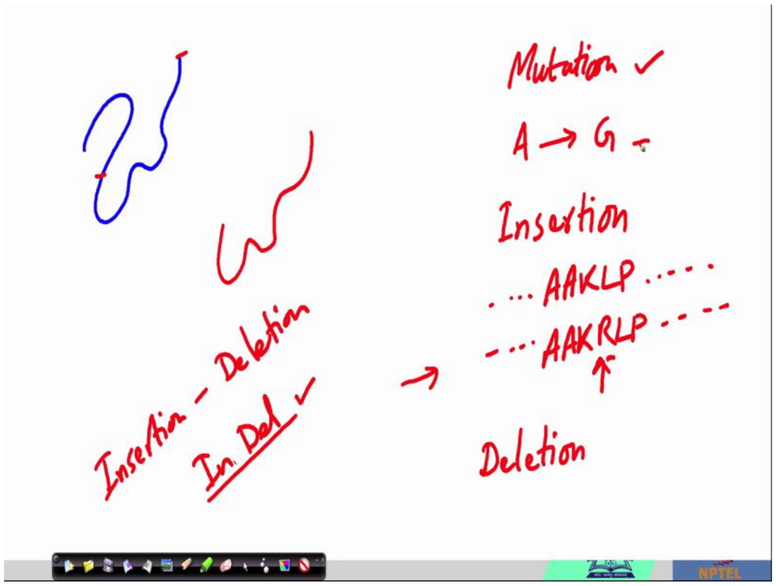
So, when I say it is a substring which means the consecutive occurrence of it, so now for ins India I see the consecutive occurrences, but for IIT or IIT Kharagpur for this string Indian Institute of Technology Kharagpur I did not find any substring. Although part of it say IIT Kharagpur, Kharagpur is matching to Kharagpur from the pattern and the string but that is the part matching, not the whole matching so, that is not correct.

So, green color in the pattern indicates that that particular substring is present, whereas the red color indicates that particular substring is not present. Now, what is the common subsequence problem? So, the longest common subsequence says that let sigma be an alphabet, again finite set for DNA alphabet it will be A, C, G, T. Find the longest common to all sequences in a set of sequences, often just two sequences and construct that on the alphabet set sigma.

Now, if I take the example, same pattern and same string then I can say that all are present as a subsequence in the string. So, how is it possible? So, let me pick a separate color, so ins, so ins is present here like the previous one, then India, India is present here like the previous one. Now if I consider the IIT, then IIT I can get from several places, so I can get from 1, 2, 3, that is one IIT, another I can get say this one, then this one, then this one, that is my IIT, so in several ways actually I can get the IIT.

And when I say that IIT Kharagpur, then this IIT will remain after that one this Kharagpur will be added. So, that is the common subsequence problem. Now, I believe that you understand the difference between these two that is one is the substring problem, where the pattern occurs in a constitutive way. Another is the subsequence problem, so the order is same, but it may not be the consecutive in nature. Now, among these two, so this substring is very much relevant for us that we understand, but the subsequence will also be relevant in case of protein.

(Refer Slide Time: 07:14)



So, why it will be relevant? I am going to detail you here. So, if you look at say one protein sequence and corresponding to that let us assume one structure is there, something like this. Now, let us assume that there is another structure and that structure is something like say like this. Now, if I take these two-structure blue color and the red color, now between these two, blue and red, if I go for the substring search, then perhaps I will find some substring by visual inspection also I can see that perhaps from this part to this part, there will be a substring.

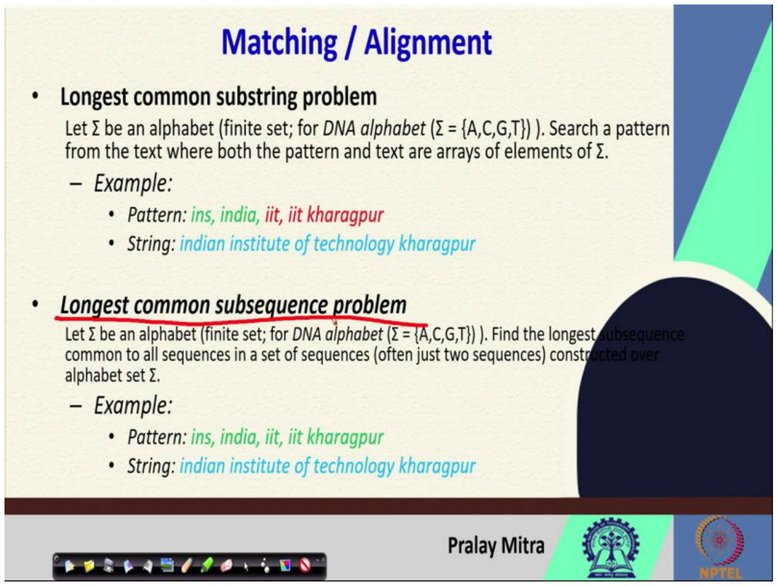
And when there will be a substring? So, for this part actually, I can able to establish a correspondence between the blue structure and the red structure, but if you consider the existence or the evolution and during the evolution process, some mutation will happen, mutation means that one particular amino acid corresponding to one protein structure will be changed. Another possibility is that there may be some insertion of some amino acid during the evolution, or there may be some deletion during the evolution process.

So, considering those different cases, so what I may insert mutation it means the change of one amino acid to another amino acid, say from A, I mean alanine to say glycine, something like this, another is a insertion which means initially the sequence was say AAKLP like this, after evolution process, so it looks like say AAKRLP like this, so this R is an insertion of the amino acid.

Now, another situation may be deletion. Now, in case of deletion it will be just opposite to the insertion which means initially this AAKRLP, so it is not a small one so on both the sides actually they are extending that is why I am putting some dot on the left-hand side and the right-hand side, and after the evolution process say this R has been removed, or R has been deleted, and because of that deletion, now I am getting AAKLP. So that is why sometimes this insertion and deletion are combined and this insertion and deletion are called as the In Del operation, sorry this will be In Del, In Del operation, this is considered as one single word.

So, this In Del operation or because of the mutation, it may happen that there is no one to one correspondence, and because of that one, so the order may be same, but because of this say insertion or deletion or some mutation say at that, at some position in between the one-to-one correspondence is not present.

(Refer Slide Time: 10:38)



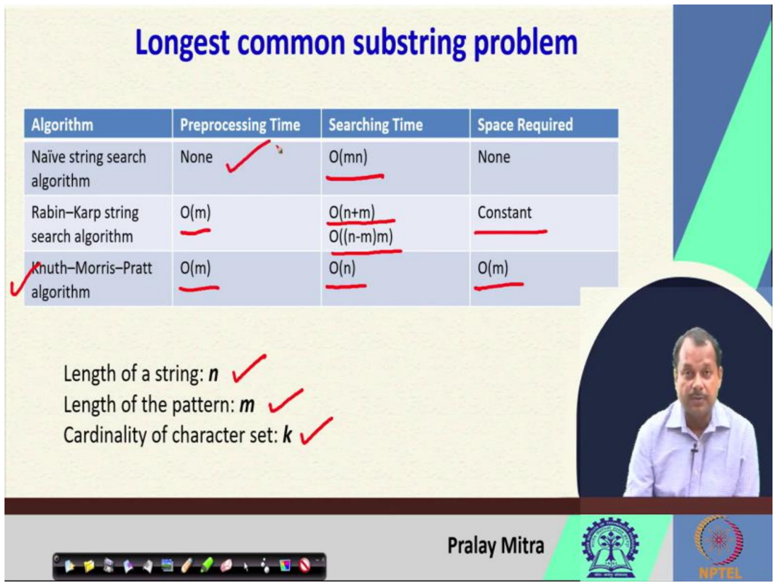
Matching / Alignment

- **Longest common substring problem**
  Let Σ be an alphabet (finite set; for *DNA alphabet* (Σ = {A,C,G,T})). Search a pattern from the text where both the pattern and text are arrays of elements of Σ.
  - *Example:*
    - *Pattern: ins, india, iit, iit kharagpur*
    - *String: indian institute of technology kharagpur*
- ***Longest common subsequence problem***
  Let Σ be an alphabet (finite set; for *DNA alphabet* (Σ = {A,C,G,T})). Find the longest subsequence common to all sequences in a set of sequences (often just two sequences) constructed over alphabet set Σ.
  - *Example:*
    - *Pattern: ins, india, iit, iit kharagpur*
    - *String: indian institute of technology kharagpur*

Pralay Mitra

So, considering these facts actually, we will be more interested about a longest common subsequence problem in the context of this protein sequence alignment. Now, this longest common subsequence problem we wish to solve using dynamic programming problem.

(Refer Slide Time: 10:55)



Longest common substring problem

Algorithm	Preprocessing Time	Searching Time	Space Required
Naïve string search algorithm	None	O(mn)	None

Rabin–Karp string search algorithm	O(m)	O(n+m)
		O((n-m)m)
			Constant

Knuth–Morris–Pratt algorithm	O(m)	O(n)	O(m)

Length of a string: ***n*** ✓
Length of the pattern: ***m*** ✓
Cardinality of character set: ***k*** ✓

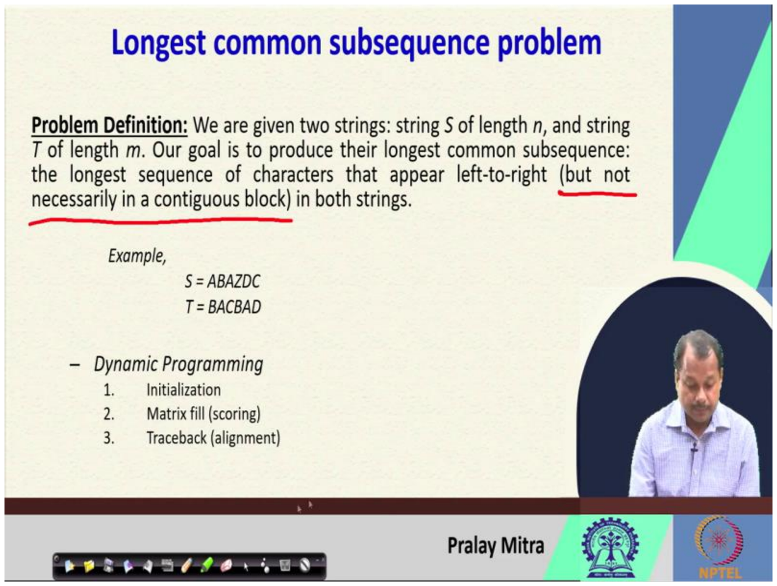
Pralay Mitra

But for the time being for the actual completion of the discussion, I would like to mention you that longest common substring problem there are several algorithms, so Naive bass string, so Naive string search algorithm which is very simple, so corresponding to each character in the string, in the pattern you will search inside the string there will be a nested for loop concept.

And if the length of the string is n and the length of the pattern is m, then actually the searching time requirement is order of mn, this will be order of mn. Now, there is RK or Rabin Karp string search algorithm rule for which pre-processing time is order of m and then space requirement is constant and searching time maybe order of n plus m, or order of n minus m multiplied with m. Again, length of the string is n, length of the pattern is m, and cardinality of the cardiac character set we are assuming k.

Now for KMP or Knuth Morris Pratt algorithm, which is the fast data and fastest among these three, pre-processing time is order of m and searching time is order of n and space required is order of m. So, you can clearly see that the last one is going to be the fastest one, but from an implementation point of view, the first one will be the simplest one. But these are for the completion of the discussion and we are not interested about this longest common substring problem in the context of this protein structure.

(Refer Slide Time: 12:39)



Longest common subsequence problem

Problem Definition: We are given two strings: string *S* of length *n*, and string *T* of length *m*. Our goal is to produce their longest common subsequence: the longest sequence of characters that appear left-to-right (but not necessarily in a contiguous block) in both strings.

Example,

S = ABAZDC

T = BACBAD

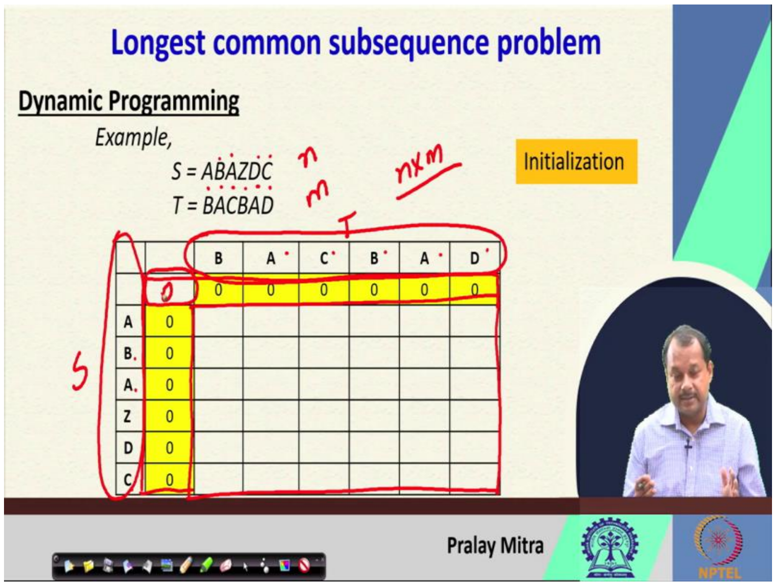
- *Dynamic Programming*
  1. Initialization
  2. Matrix fill (scoring)
  3. Traceback (alignment)

So, longest common subsequence problem. So, the definition is that we are given two strings, string S of length n, and string T of length m, our goal is to produce their longest common subsequence, the longest sequence of characters that appear left to right but not necessarily in a contiguous block, in both strings. So, please note this. So, but not necessarily in a contiguous block, that is the basic difference between the substring and the subsequence.

But, the order matters, the order in which way actually they will, the order in which way they will appear that matters. Now, if I give you some example here, the same example, I am not using I am giving some different example, and if S is one such string and T is another such string, where the string contents ABAZDC for S, and for string T it is BACBAD. Now, I need to go for the longest common subsequence search.

The process or the method I will be developing based upon the dynamic programming problem, so where clearly there are three steps in dynamic programming, one is called as the initialization step, second is called as a matrix fill or the scoring phase, and third one is called as a traceback or alignment. Now, this we will elaborate in the next slide.

(Refer Slide Time: 14:34)



Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC n

T = BACBAD m

nxm

Initialization

T

S

		B	A	C	B	A	D
	0	0	0	0	0	0	0
A	0						
B	0						
A	0						
Z	0						
D	0						
C	0						

Pralay Mitra

So, the same two strings S and T we are now considering and we mentioned that there are three steps in dynamic programming, initialization, scoring or matrix fill and the traceback. So, first step is my initialization. So, what I am doing corresponding to two strings, I am first defining or say declaring one matrix whose size is n cross m, where I am assuming that the cardinality of S is n, and say cardinality of T is m.

Now, in this example actually they are same, but they may not be necessarily same. So, I think that thing is clear to you, and for the generalization purpose we are assuming that the cardinality I mean the number of characters in the string S is n and the number of characters in string T is m. So, first what we will do that we will declare one matrix, and that matrix is of dimension n cross m.

So, this is my matrix, in this matrix along the row what I have done you see that the S string is placed. So, ABAZDC taking from ABAZDC along the rows, and along the columns T strings are represented that is BACBAD, BACBAD. Now actual matrix is here, this is my matrix. So, this ABAZDC or BACBAD although I represented in a tabular form, assuming that it will be convenient for you to look at it and then understand, but that is not part of the matrix, the matrix is going to be an integer matrix.

Assuming the score function that I am going to define is an integer value. So, only this red rectangle is actually my matrix. Now, in this matrix at the initialization phase, what I need to do, I need to add one extra row, another extra column, so that way my matrix will now be augmented and we will take a same, something like this. So, now the dimension of by matrix in order to align two strings of length n and m respectively will be n plus 1 cross m plus 1.

Now, regarding this insertion of the first row and the first column, in some algorithm or implementation you can see that their index is considered as minus 1 row and minus 1th column, or you can consider 0th row and 0th column, if you start with minus 1th row and minus 1th column then as per the C implementation you have to go up to less than m after up to less than n, or m minus 1 and n minus 1.

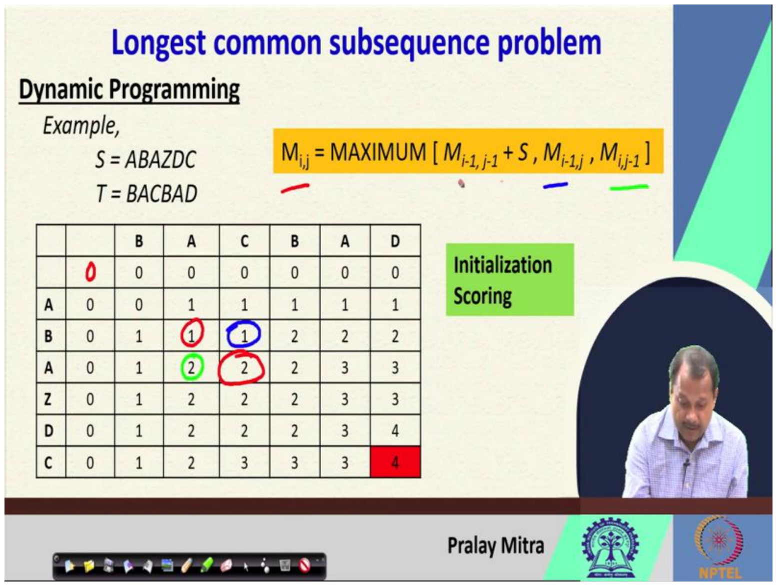
But, if you consider that you are starting from the 0, I mean after the augmentations, 0th row and 0th column is the augmented row and column, then you have to go up to m and n. But in both the cases the dimension of augmented matrix is now going to be m plus 1 cross n plus 1. Now, at the initialization phase, what I did?

I declare one matrix after looking at the cardinality or the number of characters in the two strings, then I augmented one row and one column which means if the length of my one string is n and length of another string is m, then I declare one matrix of size n plus 1 cross m minus 1.

After declaring that one also I am assuming that in each row there are some characters from one string and each column there are characters from another string, that you can maintain actually in one dimensional matrix, so this part you can maintain in one dimensional matrix, it non-necessarily be part of your matrix.

Additionally, what I have done in this initialization phase is that, the augmented row and column that I have inserted is having the value 0, all filled with 0, that is my evolo background, row and column that you understand. Now in this case, this particular cell position will not be of much use, but still if you want you can put 0 here, it will be used at the beginning but prefer a better to use 0 there. So, that is in summary my initialization phase. Next is my scoring.

(Refer Slide Time: 19:50)



Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

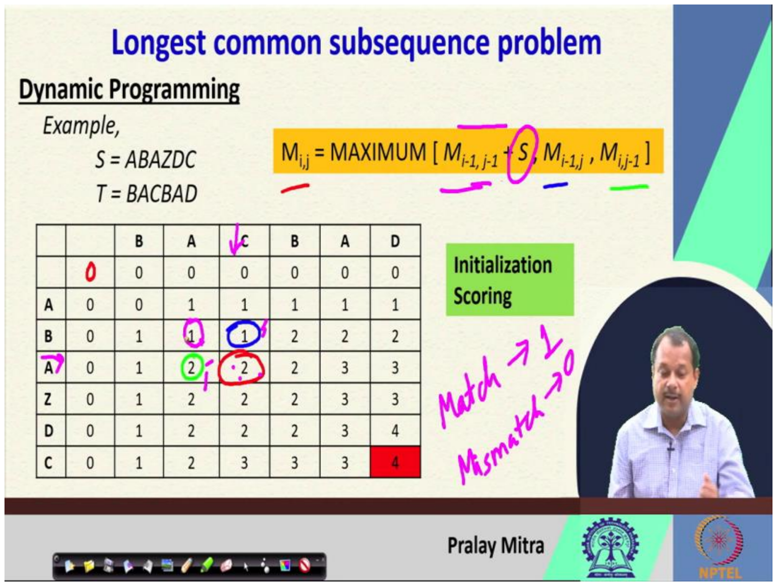
T = BACBAD

$M_{i,j}$ = 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	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	4

Initialization
Scoring

Pralay Mitra



Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

$M_{i,j}$ = 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	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	4

Initialization
Scoring

Pralay Mitra

Match $\rightarrow$ 1
Mismatch $\rightarrow$ 0

So, in scoring, so in scoring, after the initialization, in scoring first I define one score function that is M i,j equals to maximum of M i minus 1, j minus 1, plus S, M i minus 1, j, M i, j minus 1. So, if I draw the matrix, so, do not look at the content first, so that I will explain later. So, if you look at the content, so the matrix I have defined and then in this matrix initial or the first row and first column is augmented and they are filled with a value 0 that I have done, now I wish to fill the cell position of the matrix.

If I assume that the name of my matrix I mean the variable name of the matrix is capital M, then M i,j, which means this one, will indicate ith row and jth column that means one cell position it is indicating M i,j, now at any instance if I consider one such cell position like this without any loss of generality, we are assuming say we are in the process of scoring, actually I did not do that one I will start from the beginning, but just to explain the score function first I am assuming that we are in some middle of the filling the matrix, and M i,j indicates this particular cell position.

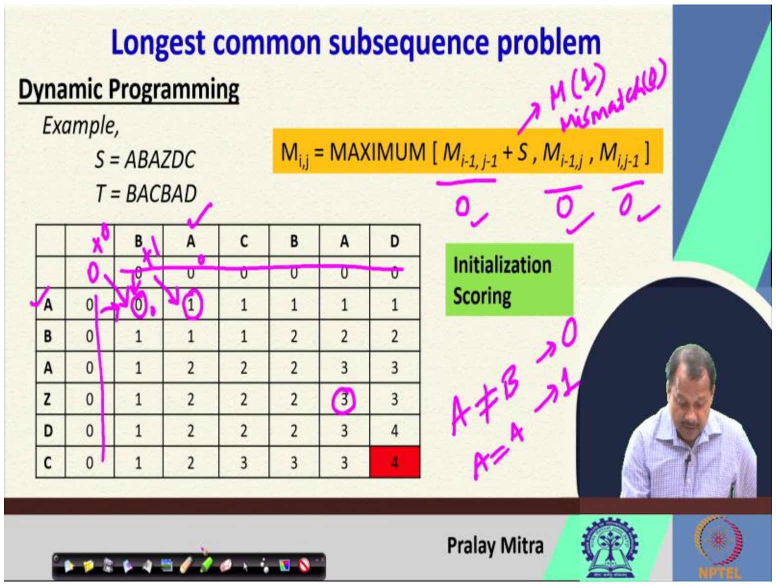
If M i,j is considering this cell position, then how the value will be filled? In order to fill that value, I have taken three positions, what is that? M i minus 1, j minus 1, I mean this position, then what I have taken that M i minus 1 j, so from here I am here i minus 1 and j, so i minus 1 is here row, and then j the same column, so this one, then I have considered this green M i,j minus 1, so M i,j minus 1, for one column left. So, these three actually I considered.

Now, if I say for clear understanding, if I give another color here, say pink, then this pink is basically this 1. So, this pink actually is M i minus 1, j minus 1, but along with this one, one extra term will come that is my S, so this S I have to define. After defining S then what I will do with M i minus 1, j minus 1 or I mean with this 1 I will add S then that will be one value, then I will have this one as one value, this one as another value, among those three, which is the maximum I will pick that one and put in the red circle I mean that ij position.

So, that is the scoring function and that is also the logic or algorithm behind filling up this matrix. Now, regarding this S, so at this position from this position when I am coming to this position, then two new characters has been explored. So, one will come from here and another will come from here. So, these two characters, if those two characters are same, I mean if there is a match then I will have some score function let us assume 1, if they will not match, I mean mismatch then I will have 0.

So, that is my S for the timing, we will see lot of variations in S, and if we change the variation if we change the content of S then accordingly the alignment will also change that we will see one after another. Now, for the time being the score function I believe is clear to you, so at any position I have to check, it is up, left and diagonal up, these three positions but along with the diagonal up, one match or mismatch score S will be added. After the addition among those three which is the maximum, that value will come and will be placed here. Now, let us start the filling the matrix.

(Refer Slide Time: 25:15)



Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

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

→ M(1)
mismatch(0)

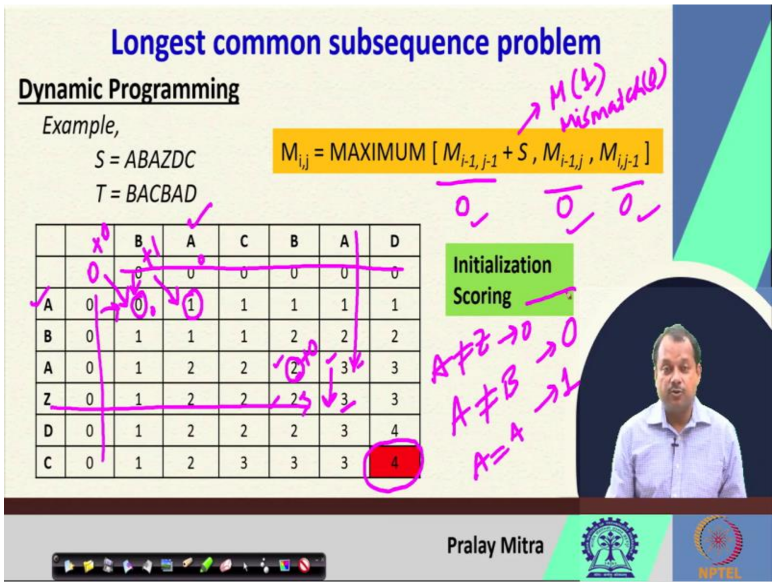
0 0 0

		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	🟥 4

Initialization
Scoring

A ≠ B → 0
A = A → 1

Pralay Mitra



Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

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

→ M(1)
mismatch(0)

0 0 0

		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	🟥 4

Initialization
Scoring

A ≠ Z → 0
A ≠ B → 0
A = A → 1

Pralay Mitra

So, when say I am at this position that is my first position because this and this has filled during the initialization phase and it is also 0. Now, when I am at this position what I can see and for the S I mentioned match is 1 and mismatch is 0. Now, if match is 1 and mismatch is 0 then at this position character is A and B, since A and B which means A not equals to B, that means it is a mismatch so 0. So, if it is 0 then with this 0, 0 will be added, so this value is my 0.

Next M i minus 1, j, from here it is 0, from here it is 0, all three are 0. If all three are 0's, then I can pick one of them arbitrarily. So, I can take either this or this or this, and I will place that value here, while I am placing the value here, then one good thing for future trace backing will be that among these three which value is basically placed here, I will keep a track of that.

Now, you understand right now, that when these three 0's are there, so arbitrary I have chosen 1, so if it is arbitrary, which means there are three options and anyone is correct right now, so which means either it can come this way, or this way, or this way. So, among these three, I will keep only one as the information that from where the value has come actually. After this one, what I will do?

So, for this one, this is A and this is A which means there is a match, A equals to A which means match. So, for this position along with this 0, 1 will be added, along with this 0 nothing will be added, nothing will be added and 1 will come here, so arrow is like this. Now, this way, I have to fill the total matrix.

Now, if I come at some position say I am considering this 3, or say not considering these 3, so what I am doing that I need to compute this content 3, it is already filled, but how it is filled? I will check that one, how? For this one 3 so the neighboring cell position 1, 2, 3 I have to consider, so I have one value 2, another value 3, another 2 plus match or mismatch because for these diagonal I have to check match or mismatch.

Now, for this position 3 position, this is A and this is Z, so A is not matching with Z, which means it is 0. So, with 2, 0 will be added, so which means 2 here, 2 here and 3 here, among these three, which is maximum? 3 is maximum, so 3 is coming down here, I mean the arrow will be like this and this value will be 3.

That way I will fill the total matrix, and when I will fill the total matrix then I will have at the m nth position assuming say I have started from say 0, 0 row and 0 column after the augmentation, so at m n position actually I will have the value which is the total alignment score after aligning S and T. So, that is my scoring stage, the next stage will be trace backing from this scoring, that I will discuss in the next lecture. Thank you very much.