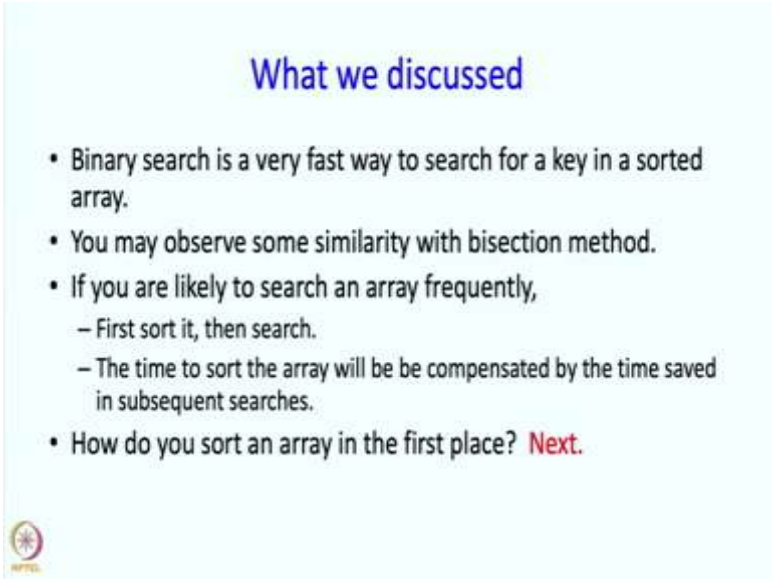
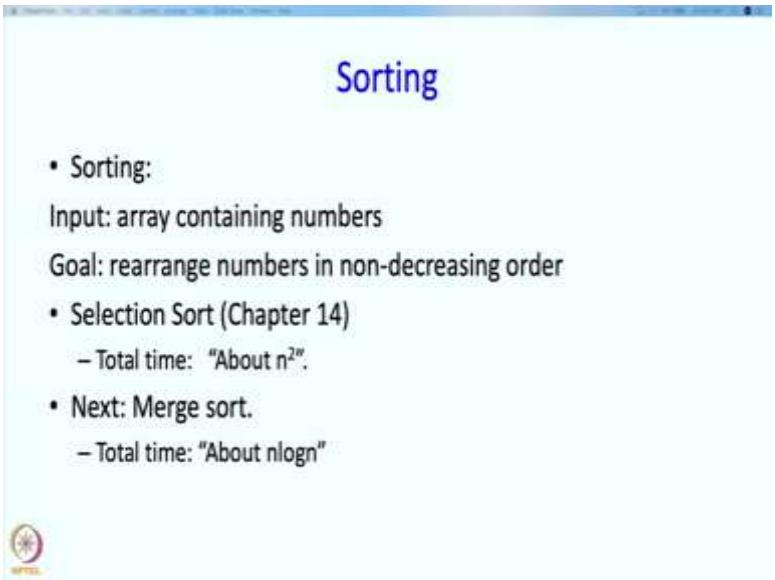
An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 18 Part- 3
Arrays and recursion
Mergesort overview

Welcome back. In the previous segment we saw binary search.

(Refer Slide Time: 0:20)

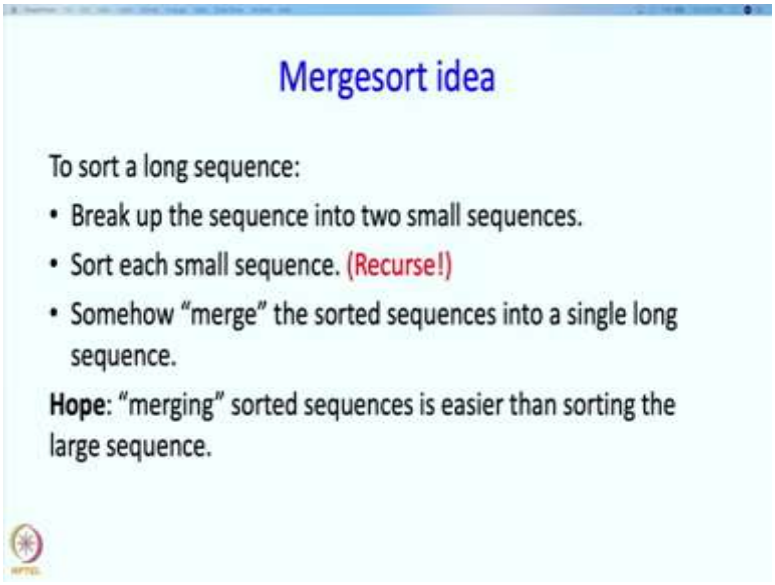




In this segment we will see a nice sorting algorithm. So what is the sorting problem? Well the input is an array containing numbers and the goal is to rearrange numbers in non-decreasing order. We have already seen one algorithm for it, the so the so called selection sort algorithm. And we said there that the time taken is going to be about n^2 or rather proportional to n^2. Now

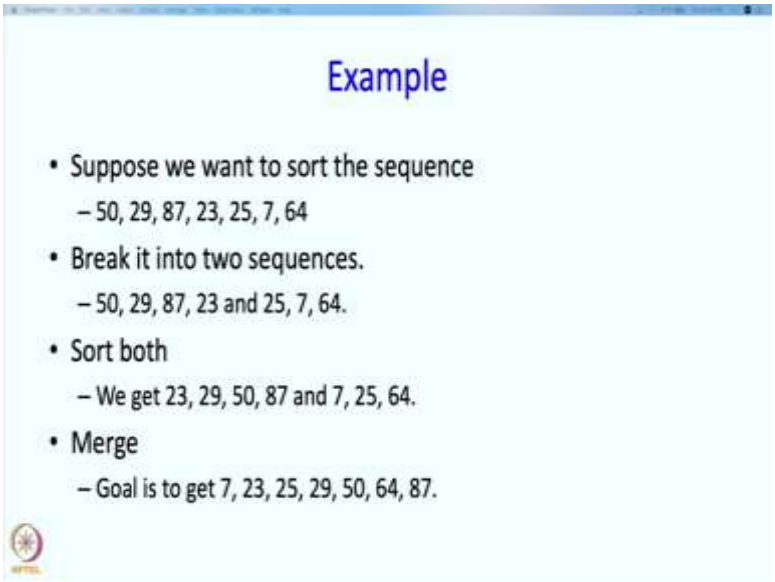
we are going to see an algorithm called merge sort whose time is going to be proportional to n log n so this is going to be considerably faster than selection sort.

(Refer Slide Time: 1:04)



So let me first describe the main idea of merge sort. So suppose we are given a long sequence or an array a long array which holds that long sequence. So we are going to take that sequence and break it into small sequences, then we are going to sort each small sequence independently separately and this is going to happen using recursion. Then we are somehow going to merge the sorted sequences into a single long sequence. The hope is that merging the sorted sequences is easier than sorting the large sequence in the first place. And yes this works out as we will see soon. So let us take an example just to make sure that this structure is understood.

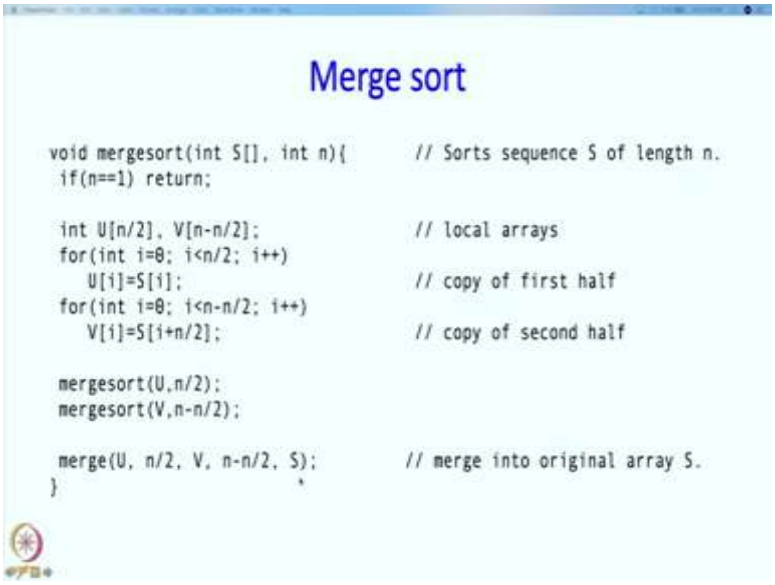
(Refer Slide Time: 2:05)



So say we want to sort the sequence. So the sequence has seven elements and let us say the sequence is sitting in an array of size 7. But really let us think about it as sequences rather than arrays for the minute. So if you want to sort it, our first step was to break it into two sequences. So let us say the first four elements form the first sequence and the last three elements form the second sequence. So we are going to sort both, so when we sort both we are going to get this sequence over here.

So this is the sorted version of this and this is the sorted version of this, so these are the two sequences we now have, after we have recursively sorted the smaller sequences, then we are going to merge them. So in the merge, from this input these sequence and this sequence given to us we want to produce this output. So we want that we want to have the same numbers but they should be in increasing order or non-decreasing order really. So that is sort of the overall overall structure of what we are attempting to do. So that overall structure can already be written into a program and so let us do that.

(Refer Slide Time: 3:22)



So merge sort is our main function and this is going to sort sequences a sequence S of length n, so that sequence is an argument. So there is a natural base case if n=1 then there is nothing to do, we just have to return, the S itself is already sorted. If n is bigger than 1 then we said that we are going to break it up into two small sequences so let us call those two sequences U and V. So you want we are going to break it nearly into halves, so this is going to be n/2 and this is going to be n-n/2.

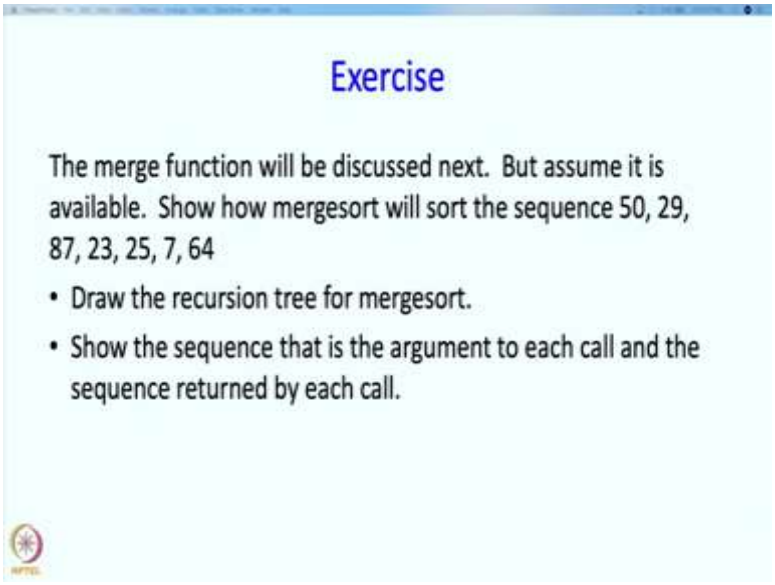
Notice that I am not writing n/2 over here that is just to guard against the possibility of n being odd and things like that. If n is or then that will not work, so what I am doing is this plus this and just this is making sure that this plus this is equal to n. So that that way I know that all the elements will either come over here or come over here. Then in this step I am going to make a copy of the first half into U and in this step I am going to make a copy of the second half into into V.

So this starts off where this ended and so I am going to get a copy of the second half. So I have created now my two small sequences, what do I do next? I have to sort them. So how do I do that? Well this is where recursion comes to my rescue and so I am going to just issue this call. Sort sequence U whose length is n by 2 and similarly I am going to sort sequence V whose length is n minus n by 2.

At this point I have those two sequences sorted okay in U, so they went into merge sort in U and they are coming back in U. And now I am going to merge these two sequences. So U and V are the two sequences that I had and I am going to get them merged into S. So that is it

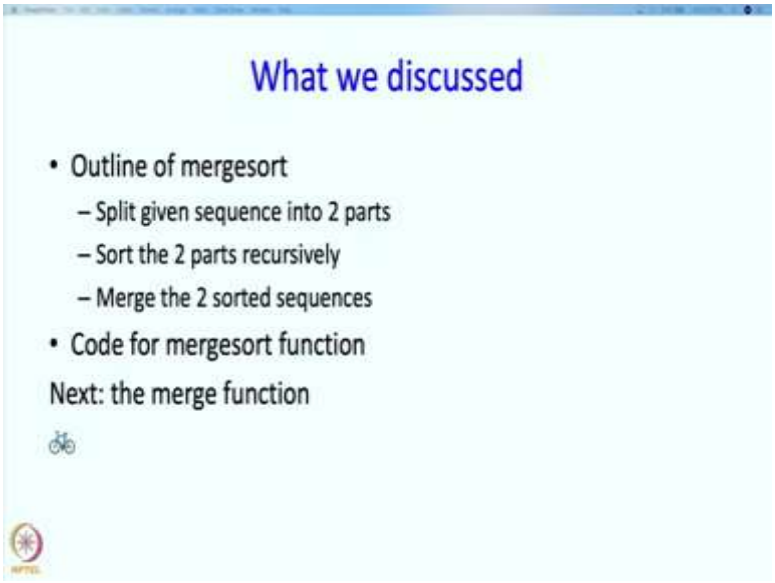
because S will have the final sorted sequence which is exactly what we wanted over here. The result had to be in S itself, so that is what we are going to have.

(Refer Slide Time: 5:55)



Now I want to leave you with an exercise before I conclude this segment which is that we will discuss the merge function next, but suppose it is somehow available, I would like you to show me how merge sort will sort the sequence 50, 20 the given sequence. Basically I want you to draw the recursion tree which we have learnt earlier. So in the recursion tree, show the sequence that is the argument to each call and the sequence returned by each call. So this will just this will just be a test of whether you understand what the algorithm is doing at the high level.

(Refer Slide Time: 6:37)



So what have we discussed? We have discussed the outline of merge sort and this consists of split the given sequence into two parts, sort the two parts recursively and merge the two sorted sequence sequences. Then we also discuss the code for the mergesort function and what remains to discuss is the merge function. So that we are going to discuss in the next segment, so we will take a quick break.