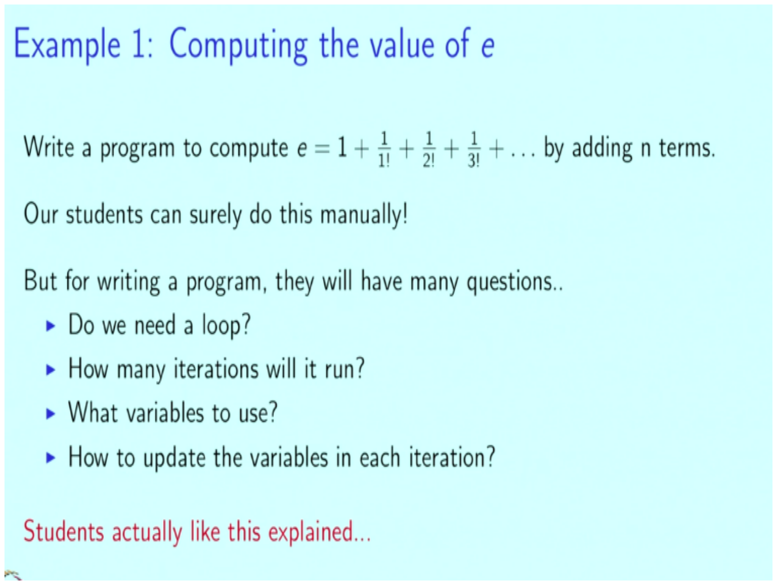
Design and Pedagogy of The Introductory Programming Course

Prof. Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology – Bombay

Lecture – 06
Basic Ideas in Our Approach. 1: Examples of translating manual algorithms to computer programs

Hello, welcome back. In this lecture we are going to see examples of translating manual algorithms to computer programs. So our first example is computing the value of e the base of a natural algorithm.

(Refer Slide Time: 00:38)



Example 1: Computing the value of e

Write a program to compute $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$ by adding n terms.

Our students can surely do this manually!

But for writing a program, they will have many questions..

- Do we need a loop?
- How many iterations will it run?
- What variables to use?
- How to update the variables in each iteration?

Students actually like this explained...

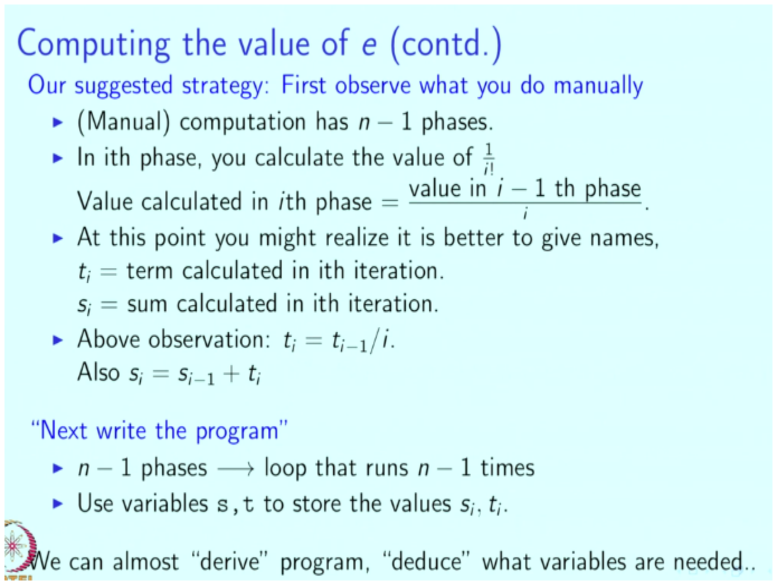
So the problem statement is write a program to compute e which is given by this infinite series 1 + 1/1! + 1/2! + 1/3! and so on and we want to add this series only to the first n terms. So we want to write a program to do this. Can our students do this manually? Of course they can. Now does it mean that they can write a program? Well they will have many questions if you ask them to write a program. For example, we need a loop. How many iterations will it run? What variable should he be choosing? How to update the variables in each iteration?

Now you and I may think that these questions are really simple, but they are actually not. I have talked to students and I have asked them I talk to you about this, I have written this in a book that I have written that I have written that is the book that has been mentioned along with this course.

Do you really like me to explain this? Do you really want me to explain this or do you think it is obvious? I was quite surprised that bright students as well as not so bright students came back and told me please, please sir keep this description.

We need this description. We find it very useful. It is very comforting for us. It gives us confidence in writing programs. So please put yourself, please imagine yourself as a novice student in the first programming course and approach the next slides from that point of view. So we want to compute the value of e.

(Refer Slide Time: 02:41)



Computing the value of e (contd.)

Our suggested strategy: First observe what you do manually

- (Manual) computation has $n - 1$ phases.
- In ith phase, you calculate the value of $\frac{1}{i!}$
  Value calculated in ith phase $= \frac{\text{value in } i - 1 \text{ th phase}}{i}$.
- At this point you might realize it is better to give names,
  t_i = term calculated in ith iteration.
  s_i = sum calculated in ith iteration.
- Above observation: $t_i = t_{i-1}/i$.
  Also $s_i = s_{i-1} + t_i$

"Next write the program"

- $n - 1$ phases $\longrightarrow$ loop that runs $n - 1$ times
- Use variables `s,t` to store the values s_i, t_i.

We can almost "derive" program, "deduce" what variables are needed..

What is our suggested strategy? What do we tell our students? Well, we will tell them first observe what you do manually. So for this they can say something right well. The manual computation does have n - 1 phases. In the ith phase, you calculate the value of 1/i! and you add it to the previous values; the previously calculated sum. Well what is it that you exactly calculate and how do you calculate it.

Well you want the value of 1/1!, but you know that in the i - 1 phase or in the previous phase, you already had calculated the value of 1/i - 1 !. To get the value of 1/i! You just have to divide by i. Do students observe this? Of course they will. Ask them to actually do the calculation. There we are going to do i! from scratch. They will immediately realize that oh I just calculate 1/5 ! to get 1/6 ! I just need to divide again by 6.

So they will make this observation and we want to alert them to make this observation. In fact if you ask students to write a program they will forget how they do this whole thing manually and they will compute whenever i ! from scratch every time in the loop and they will get confused, because there will be a nested loop and so on. And of course they will do extra computation. So although computational complexity is not a part cannot be a part of the first programming course this is an obvious economization.

And we should alert students and in fact the AICTE course and many other courses do say things like encourage students to minimize computation. So alert students to what they exactly do and this is what they really do. If you go along this path you will talk lot about the values and you may realize at this time that it is better to give names to values. So for example you could say that ti is the term calculated in the ith iteration.

Si is the sum calculated in the ith iteration. Remember that these are students who have already done in the 11th and 12th standards. So they are familiar with series. They do know giving names to terms of a series. So this is not going to be something that they are going to get confused about, but once they give these names, they can write their observations in a succinct manner. Instead of saying value calculated in the ith phase is value in i - 1 phase/I, we can just say ti which we want to calculate is ti -1/i.

So it becomes much more precise. It becomes much more compact and of course, do this slowly. So if somebody gets confused by writing ti tell them that this is what you mean and tell them why they are writing ti. We are writing ti because it is convenient for us and similarly is. So what we are doing in the ith step is doing the division and doing that addition. Next we want to write the program and we should write the program only after we have understood.

What it is that we are doing and preferably understood it in these very specific terms not generally that I do this, this, this, this in the next step this is what happens not in terms of examples, not that in the first step I am going to calculate 1/1! in the second step I am going to calculate 1/2 !, 1/3 !; no. Students can rattle that off, but tell them that you need to do a little bit more. You need to tell me what happens in general.

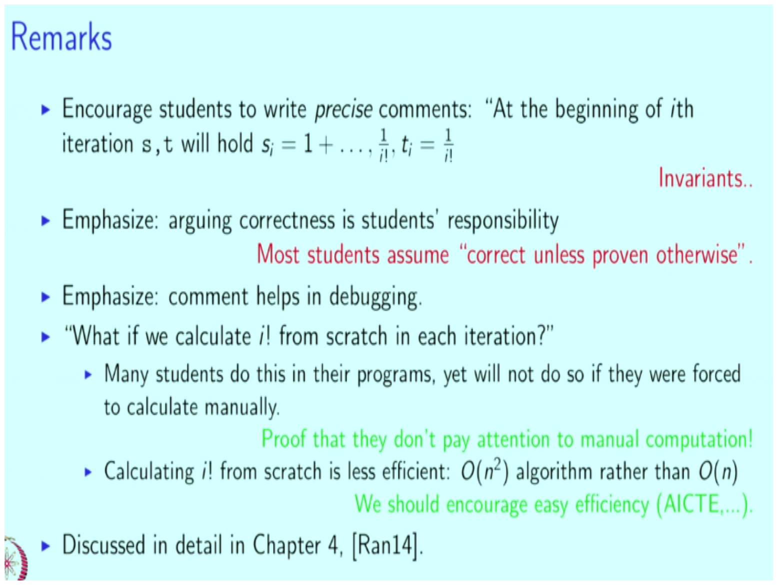
So this idea that students have to say what happens in general is important and therefore they need to give names and they need to write statements such as ti = ti - 1/i and they can say that this is what happens for all values of i. Once they have said this, only then are they ready to write the program. So now you can say something like if you have n - 1 phases if you are repeating the same computation n - 1 times, then that means you should have a loop that runs n - 1 times.

And in the loop you need to have variables. So it is natural to use variables s and t to store values si and ti. So you really need to go over this fairly slowly. You need to tell the students how you are going to update those si and ti, the order is important at the beginning si and ti will satisfy that ti = ti - 1/i and si = si - 1 + ti and when you make the changes this will of course not be satisfied, but you have to make the changes in such a manner that at the end s will again contain the desire si value.

And t will again contain the desired ti value at the beginning of a phase t should contain the value ti as written above. s should contain the value si as written above. And to make sure that this happens in the i + 1 phase we need to change the variables s and t in a certain manner. And the teacher should show by examples and with a little bit of patience exactly how those changes had to be made.

In other words, we can almost derive the entire program or deduce what variables are needed, what kind of statements we need to write inside the loop. In fact, even things like how the variables need to be initialized.

(Refer Slide Time: 10:03)



Remarks

- Encourage students to write *precise* comments: "At the beginning of ith iteration s,t will hold $s_i = 1 + \ldots, \frac{1}{i!}, t_i = \frac{1}{i!}$

  Invaluants..
- Emphasize: arguing correctness is students' responsibility

  Most students assume "correct unless proven otherwise".
- Emphasize: comment helps in debugging.
- "What if we calculate $i!$ from scratch in each iteration?"
  - Many students do this in their programs, yet will not do so if they were forced to calculate manually.

    Proof that they don't pay attention to manual computation!
  - Calculating $i!$ from scratch is less efficient: $O(n^2)$ algorithm rather than $O(n)$

    We should encourage easy efficiency (AICTE,...).
- Discussed in detail in Chapter 4, [Ran14].

So some remarks when writing a program we should encourage students to write precise comments and a very useful comment could be at the beginning of the ith iteration s and t will hold si = 1 + 1/1 ! + 1/2 ! all the way till 1/ i! or whatever it is that you want s to be and ti should be 1/ i! or again whatever it is that you want ti to be. Whatever value we want ti to be should be there in ti that is what your plan is and that is what you should state.

Notice that what you are encouraging the students to do are simply to write invariants and invariants are an important idea as you know. They are important in making sure that your program is correct and many times students also assume that once they write the program the teacher is going to correct it and the teacher is going to tell them whether the program is correct or not; No that is not at all how it works you have to tell your students.

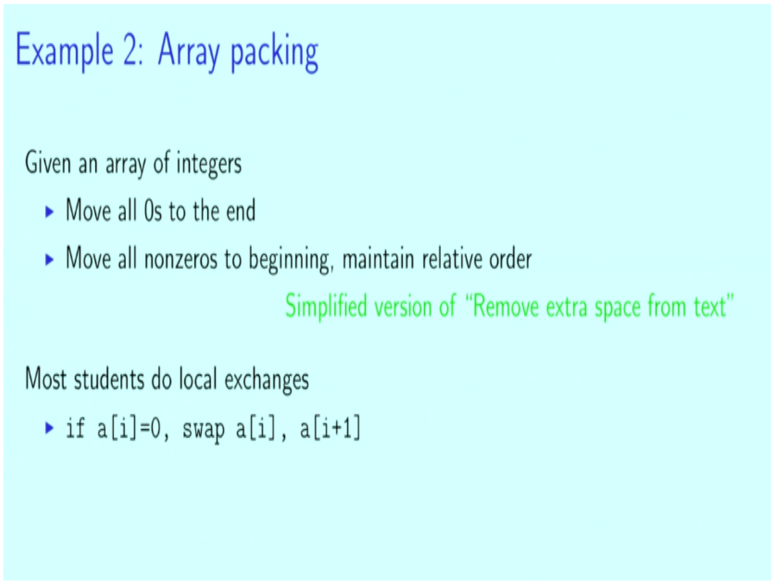
You have to tell your students that they are supposed to argue that their program is correct. So not that correct unless proven otherwise, but they are supposed to argue that their program is correct and in the first programming course they do not actually need to give a prove, but they actually need to say what is that they are expecting their program to do. So even if they are saying what s and t will hold that is a good solid step that is all that we expect, but we certainly expect that in the first programming course.

You can also tell them that if you write down these comments if you write down what you are planning are going to be contents of your variables that if you make a mistake you will know immediately what the problem is. You can test at the end of each iteration; Are my variables having the values that I expect? Some students as I said will calculate i ! from scratch and each iteration, but as you have said this is not quite how they would solve the problem annually.

And then some says see that this is a proof of the fact that students are not paying attention to manual computation and we are saying that manual computation is very important and of course as we said earlier calculating i ! from scratches less sufficient the program will have nested loops and it will be slightly more complicated and therefore in the spirit in which we are taking this example.

And in the spirit of efficiency it is useful if we encourage students to do it in this manner and write down the invariance or the plan for the program that they are going to write. This is discussed in great detail in chapter 4 of the book that has been mentioned with this course; An introduction to programming through c + + which you can see in the site or in the description.

(Refer Slide Time: 13:57)



Example 2: Array packing

Given an array of integers

- Move all 0s to the end
- Move all nonzeros to beginning, maintain relative order

Simplified version of "Remove extra space from text"

Most students do local exchanges

- if a[i]=0, swap a[i], a[i+1]

My second example is array packing. What is the problem over here? We are given an array of integers. We are required to move all the 0s in the array to the end. The non zeroes are to be moved to the beginning and in doing so we are supposed to maintain the relative order. A more realistic problem could have been that I am typing some text and in typing the text instead of putting only 1 space between consecutive words I am putting several.

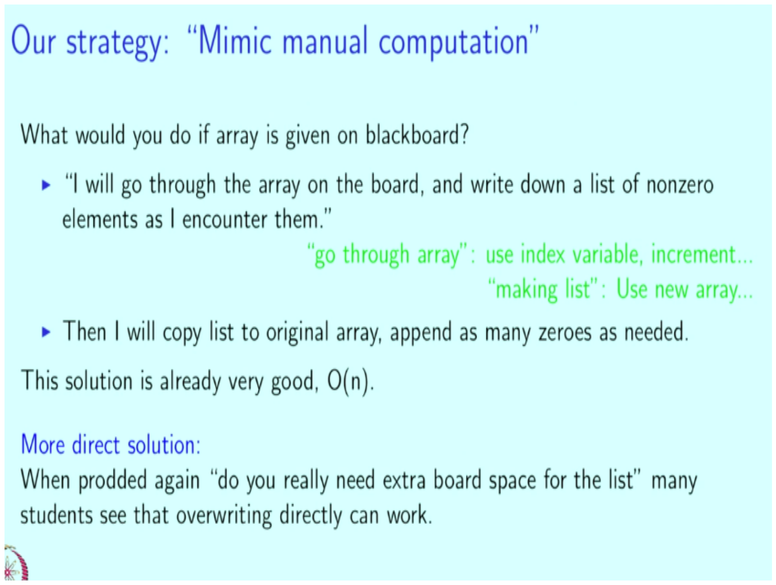
And you want to remove those extra spaces. So this is sort of a simplified version of this problem and for ease discussion we are going to take the simple version. Now how do students do this in absence of any instruction? So here is that what they will do. They will check if a of i = 0 if the array contains a 0 in a certain position. If there is a 0 then you know or you have been told that the 0 should be moved forward. So they swap.

So ai + 1 is brought forward and 0 is going to be moved to a i + 1th position. Now if you think about this carefully you will see that this step will have to be repeated n square times. Students do not always get that right and of course it is inefficient. You should be able to get and know algorithm for this but more importantly there is potential for making errors here. If there are consecutive 0s then if you are sloppy in your reasoning you may skip some 0s.

You may not move every 0 to the end. So this highlights 2 things you should not be sloppy and you should have a clear idea of what you are accomplishing as a result of each step and as you will see in a minute if you just think about how you solve the problem manually you will do this in a much easier manner. So what is our strategy? What strategy have we suggested? Our strategy is mimic manual computation. So the problem was stated as what do you with an array?

So now for manual computation we have to back translate it to a black board because we can manipulate a black board manually we can write or erase on the black board so a black board is much easier to deal with for manual computation. So what would you do if the array is given on a blackboard?

(Refer Slide Time: 16:56)



Our strategy: "Mimic manual computation"

What would you do if array is given on blackboard?

- "I will go through the array on the board, and write down a list of nonzero elements as I encounter them."

  "go through array": use index variable, increment...

  "making list": Use new array...
- Then I will copy list to original array, append as many zeroes as needed.

This solution is already very good, O(n).

More direct solution:
When prodded again "do you really need extra board space for the list" many students see that overwriting directly can work.

Many students will say I will go through the array on the board and write down a list of nonzero elements as I have encountered them. Notice that they are not going to say you are going to do local exchanges. They will realize that local exchanges are a lot messy why should I locally exchange. The moment they do a few local exchanges they will realize that I could have moved this to the end in 1 shot and how do I move that to the end in 1 shot?

What am I really doing there? So what we are telling the students is think about what are you really doing there really doing in all of this. Now the second step. What does it mean to go through the array? Well, when you want to translate it back to an array or go through the array on the black board, now go back to the computer how do you go through an array on a computer. Well for that you need to use an index variable;

You need to increment that variable and look at the entry which that index is pointing to. Then there is this step. We are going to make a list. How do you make a list? How do you make a list on a computer? On a black board making the list is easy. I just start writing on a different piece of the board. What do you do on a computer well as teachers we have to tell the students that you can also make a list on a computer? How do you do that? You use a new array.

You ask for a new array. You have another index for it and you insert element into an array. So you insert it at the current index value and increment the array and so on. So now you have copied the list of nonzero elements into a list. What do you do next? Now you can erase the old array on the blackboard and just copy this new list so now all the zeros are not there, but you wanted the zeros as well so the rest of the array you append with zeros.

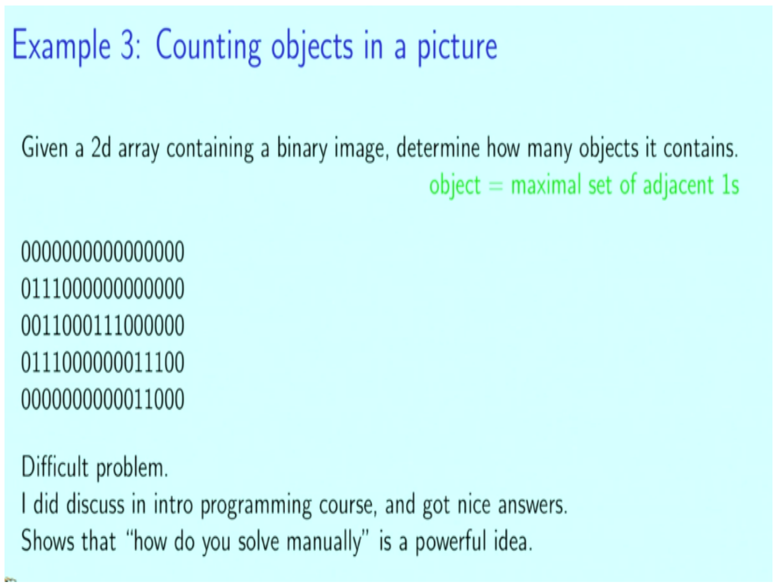
So this is reasonable blackboard algorithm and what you should do is you should work with the student in discovering the black board algorithm and tell the student what has to happen for translating the black board algorithm to a computer algorithm. What does go through array mean, what does making the list mean? Now the solution is actually very good; going through the array and copying the array does state O of n times.

So we have already made an improvement, but another improvement is possible. So there is a more direct solution in which we do not really need a separate list. So if you brought the students again you really need extra broad space for the list. Many students see that overwriting directly can work. You don't need an extra array. You can write at the beginning of the array because you are reading through a pointer which is further down.

So you do not really need to worry about those 2 lists overwriting one another. So the extra solution is important and student should get to it and they will; but on the other hand some students do not get to the extra solutions; that is okay. They already have a good solution. They already are learning that they should be introspective. And if they learn to make a correspondence between the manual algorithm and what they are writing on a computer.

And if they say that look what I write on a computer is correct because when manual algorithm is correct that that is there you have taught them what they need to learn.

(Refer Slide Time: 21:10)



Example 3: Counting objects in a picture

Given a 2d array containing a binary image, determine how many objects it contains.

object = maximal set of adjacent 1s

```
0000000000000000
0111000000000000
0011000111000000
0111000000011100
0000000000011000
```

Difficult problem.

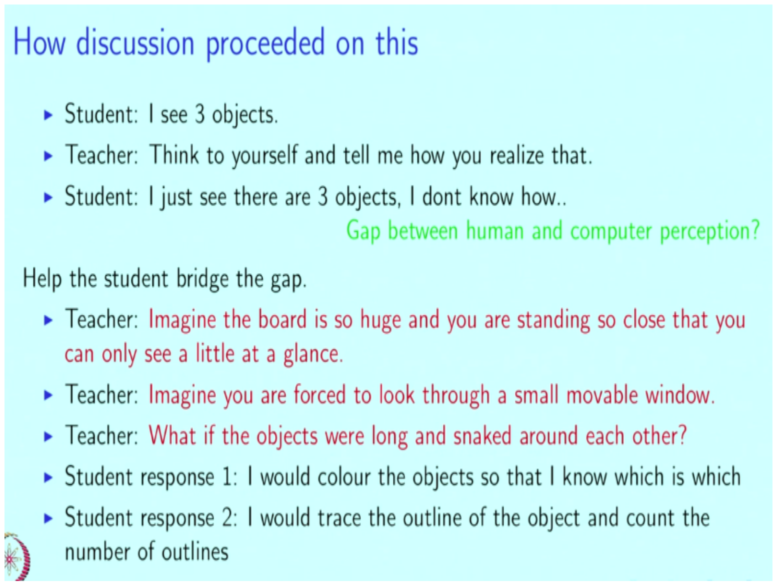
I did discuss in intro programming course, and got nice answers.

Shows that "how do you solve manually" is a powerful idea.

Third example, we are given a picture and we want to count the number of objects in it. So here is an example. So we are given a 2D array containing a binary image. What is a binary image? Well it contains 0s and 1s and an object is a maximal set of adjacent 1s. So in this picture, for example there are there is this region of 1s, this region of 1s, these regions of 1s is a contiguous region and that constitutes 1 object.

Similarly, there are other objects and those objects are what you are expecting to count. So there are 3 objects as you can see and we want to write a program which counts these 3 objects. Now let me mention that this is difficult problem. I did discuss this problem in the introductory programming course and I am reporting this to you because I got some really nice answers and those nice answers were obtained because I ask the students to think what would you do manually? So the answer will show that solving manually is a powerful idea.

(Refer Slide Time: 22:45)



How discussion proceeded on this

- Student: I see 3 objects.
- Teacher: Think to yourself and tell me how you realize that.
- Student: I just see there are 3 objects, I dont know how..

Gap between human and computer perception?

Help the student bridge the gap.

- Teacher: Imagine the board is so huge and you are standing so close that you can only see a little at a glance.
- Teacher: Imagine you are forced to look through a small movable window.
- Teacher: What if the objects were long and snaked around each other?
- Student response 1: I would colour the objects so that I know which is which
- Student response 2: I would trace the outline of the object and count the number of outlines

And also it shows you that it is not that immediate. Students don't get to that idea immediately, but when they get to the idea it is some kind of a revelation. So how does the discussion proceed? So I said okay. I told the student look what would you do manually. So the student says well I see 3 objects. Now you should ask at this point and tell me and tell the student well think to yourself and tell me how you realize that there are 3 objects.

At this point the student almost gets angry. The students have looked I see 3 objects. I do not know how I see 3 objects, but isn't it obvious. There are 3 objects. Do not you see 3 objects? What is there to tell? There is nothing like how do I realize it. I just see them. Now this is an important question. This is an important point because we have found a gap between human and computer perception. It seems that the student or human sort of look at the picture and immediately see what is it.

Whereas, the computer does not work that way. A computer has to look at 1 pixel at a time. One of those 0s are 1s at a time. So what we need is a strategy by which we can bridge this gap between humans and computer perception or check whether there actually is that gap. So what you should tell is that imagine that the board is so huge and you are standing so close that you can see a little at a glance. You can only see a little.

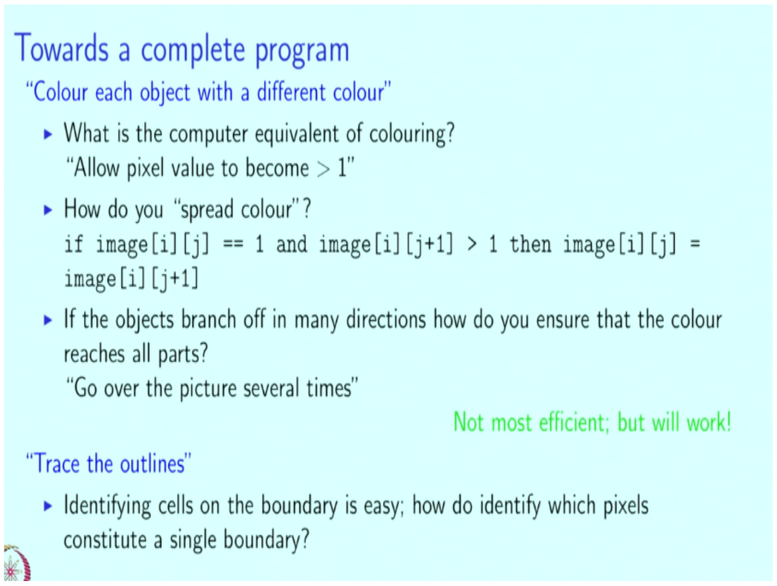
Then you would not surely see those 3 objects right away. What would you do then? So may be at this point the student will say so then I will go to the top of the board and I will sort of carefully walk around and make sure that I have seen the entire board and that way I might note 3 objects. Here is another thing that you can tell. You can say that look you are not allowed to look at the board entirely, but you are only allowed to look at the board through a small movable window.

If this is the condition, we will still be able to tell me how many objects there are. Now the students will have to think. The student will have to think about how that window is going to be moved. And now we have come to grips with the problem the way a computer needs to saw or even the way a student needs to solve it. Because and this is again something that you should remind the student what if the objects were long and complicated and a snaked around each other?

When you would not know that there are 3 objects? So you would have to do something else. So now the students really get to think and this is the crucial point. So 1 student might say, so I would colour objects so the moment I see that there is an object through my window I will colour it and I will colour every sort of spread the colour to its nearby cells, nearby excels and that way I know that this is an object that I have already coloured.

Another student might say that I would trace the outline of the object and I would count the number of outlines. When I trace the outline, maybe I will colour the outline to make sure that I am not going to trace it again.

(Refer Slide Time: 26:35)



Towards a complete program

"Colour each object with a different colour"

- What is the computer equivalent of colouring?
  "Allow pixel value to become > 1"
- How do you "spread colour"?
  `if image[i][j] == 1 and image[i][j+1] > 1 then image[i][j] = image[i][j+1]`
- If the objects branch off in many directions how do you ensure that the colour reaches all parts?
  "Go over the picture several times"

Not most efficient; but will work!

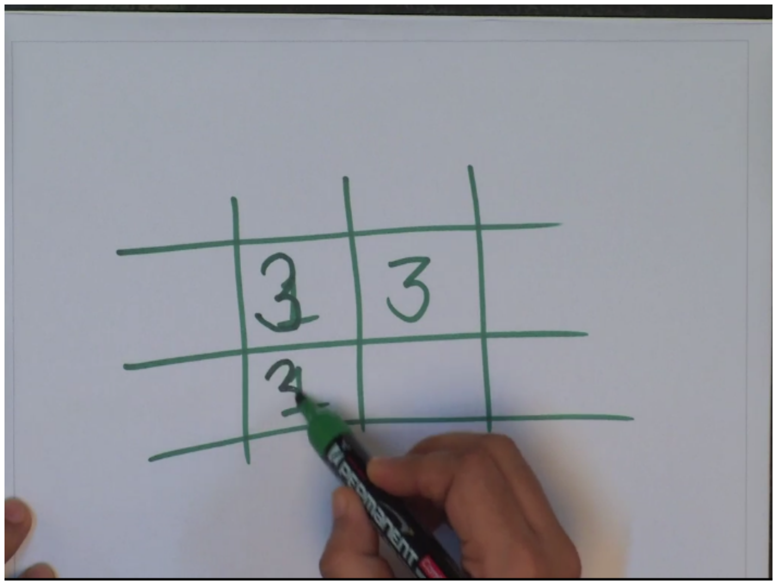
"Trace the outlines"

- Identifying cells on the boundary is easy; how do identify which pixels constitute a single boundary?

So these are great ideas of course there still need some work. What does it mean to colour each object with a different colour? What is the computer equivalent of colouring? So basically we have that array and the array contains pixel values which are 0s and 1. Now we are going to let those values become > 1.

So we are going to allow the program to write into that array or maybe we will make a copy of that array and write into it, but whatever it is that we will help to write into that array. How do you spread the colour? Well something like this if the ijth entry is 1 and the next entry is > 1, > 1 means we have coloured that entry. So then we will make that ijth entry equal to the next entries.

(Refer Slide Time: 27:28)



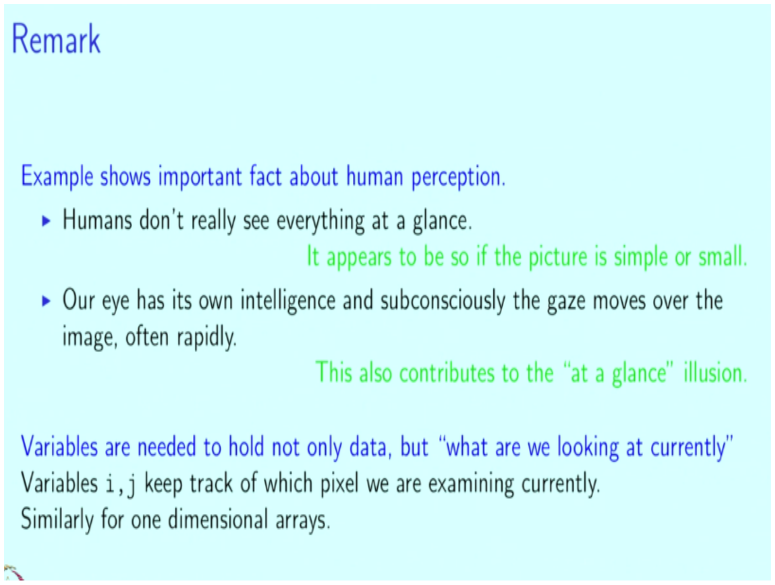
So to take a snap shot, these are my pixels. If this entry, I have made 3 and this entry is 1 then I know that this pixel is a part of this object. So I am going to change this pixel entry 2 or 3 saying that this is also a part of the third object. So in this way I am going to spread this 3 around. Now in the objects, branch off in many directions how do you ensure that the colour reaches all parts. This is actually a fairly nontrivial question. One answer is go over the picture several times.

You can make this work. It is not the most efficient. The efficient solution is a little tricky, but the non most efficient solution can be made to work if the student is patient. Just go over the array several times. The second idea was trace the outlines. How do you identify the cells on the boundary is easy so ij is a boundary? If (i - 1)j or ij - 1 and so on are 0 and ij is 1, but how do you identify the pixels which constitute a single boundary; that is a little hard.

So students will have to say so if I am looking in the north direction and the boundary is in the north direction I will need to go north east and check and then maybe I will go after north east I will check south and so on. So you will have to work that out a little bit, but the point is that the students are already coming up with nice ideas and they are doing so because they are thinking what do I do manually?

More complications if the objects have holes and then there are internal boundaries. So how do you deal with internal boundaries? So these are problems again all of these problems may not be solvable in the first year, but some variation of this problem can be given. I had given a variation of this problem in my first year course, a simplification of this problem and students were able to solve it.

(Refer Slide Time: 29:34)


Remark

Example shows important fact about human perception.

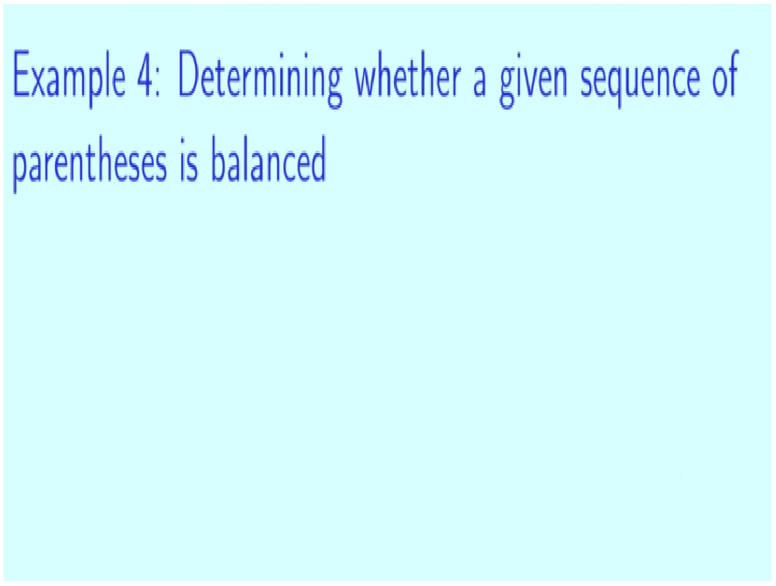
- Humans don't really see everything at a glance.
  It appears to be so if the picture is simple or small.
- Our eye has its own intelligence and subconsciously the gaze moves over the image, often rapidly.
  This also contributes to the "at a glance" illusion.

Variables are needed to hold not only data, but "what are we looking at currently"
Variables `i,j` keep track of which pixel we are examining currently.
Similarly for one dimensional arrays.

So some remarks. The example shows the important part about human perception. Humans don't really see everything at a glance. It appears to be so only if the picture is simple or small. Our eye has its own intelligence and subconsciously the gaze moves over the image often very rapidly. So this also contributes to the at a glance illusion. Then we have said that variables are needed to hold data, but here we are not only using variables to hold data, but we are using variables to note what we are looking at currently.

So variables i, j keep track of which pixel we are examining currently. So we are examining a (i,j). So pixel of ij that element of the array we are examining and ij are variables which are keeping track of what it is that we are examining and this is needed for 1 dimensional arrays as well.

(Refer Slide Time: 30:45)



Example 4: Determining whether a given sequence of parentheses is balanced

Alright so I am going to stop this lecture right now. We will take a break and we will come back and after that we will continue with the fourth example. Thank you.