

**Design and Pedagogy of the Introductory Programming Course**  
**Prof. Abhiram. G. Ranade**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology – Bombay**

**Lecture - 07**  
**Basic ideas in our Approach 2 More Examples**

Hello welcome back. We have been saying that we should tell the students that they have to pay attention to how they solve problems manually and we should help them in translating their descriptions of manual computation into a computer program. We have been doing this with several examples and we are going to continue with more examples.

**(Refer Slide Time: 00:46)**

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

Normally covered in Data Structures course,  $O(n)$  time algorithm.

- ▶ No response when the problem is posed in class.
- ▶ Example is written down on the board and students asked to tell how they know that these parentheses are balanced.
- ▶ Student response: "I repeatedly look for an open-close pair: "(" and erase it."  
What does erasing mean on the computer?  
How do you look for "(" after some erasures happen?
- ▶ Intro prog. students should be able to produce  $O(n^2)$  algorithm.



So the next example is deciding whether a given sequence of parentheses is balanced. Now this example is covered in the Data Structure course and  $O$  of  $n$  time algorithm given. We are not so worried about getting the best algorithm over here because the  $O$  of  $n$  time algorithm is clever. Our question that we want to answer over here is can the student solve the problem manually and if so can they translate how do they translate that solution to a computer program.

Can they do that; can we help them that do that. So when I posed the problem in class I usually do not get a quick response so I have done this and I did not get a quick response. So what did I do I went to the board and I said look I am going to give you something like this a sequence of parentheses maybe they could include also braces and brackets whatever it is and you have to write a program to tell me whether this is a balanced parentheses or not.

Well ideally we should explain what balanced parentheses is, but we do not really need to because students have been studying parentheses since standard 5 and they know what balancing parentheses means. So they do not give a response when the problem is posed first. So then you ask take an example and ask the students do you know can you tell me whether this example is balanced or not.

So obviously students will say yes I know how to solve problem I will tell you whether the parentheses are balanced or not and how do I do that. So here is possibly a typical response. Students might say I look for an open close pair and I erase it because I know that if something that contains an open, close pair adjacent open close pair is balanced and it is balanced if and only if what remains is also balanced or how do I generate a set of balanced parentheses.

Well I take one set of balanced parentheses and anywhere I want I can insert an adjacent pair of balanced parentheses. So students have this insight that is how they are actually solving the problem if you give it to them, what we are trying to get that is we are trying to help them to realize what is it that they already know. So that is really the key over here. They know how to do things, but they have not really thought about how they know.

And we want them to be aware and we want them to articulate how exactly we are doing things. Second step they should know that they are doing this and now we should tell them what does it mean to do it on a computer. So on a blackboard things are easy I find balanced parentheses. So this is balanced parentheses I am going to erase it. Well on a blackboard I can use a duster and just remove that chalk mark, but that is okay.

So what I will be left with is something like this with a space in between. Now things get a little complicated, but once you erase something what does it mean to say you look for adjacent open and close parentheses. So you have to amend your definition a little bit and you have to say that when I say I am looking for open close pair adjacent open close pair. I am allowing spaces to be in between the opening parentheses and the closed parentheses.

So now you have to think about and you have to tell the students what does that mean. I am going to use an index to first look for an array element containing open parentheses after that

I want to increment my index and look for closed parentheses, but if I get a space then I am going to ignore it. So we are really telling the students the equivalent of what they are thinking when they are working on the board.

And they are confident and they are very, very comfortable working on the board. So we want to transfer that confidence and that comfort level to what they do when they are programming. Now if you do these students will be able to write the program, it will take  $n^2$  square times because they will go over the same array elements several times because they may go back and forth to find adjacent open and close pairs, but that is okay.

I mean we should be celebrating at this point because students are able to solve something in the first programming course which most of us think that they can only solve in the second programming course which we teach them in the second programming course. So what we are doing is we are helping the student a lot. We are giving the student confidence; we are telling them how to translate their manual knowledge.

And we are persuading them that their manual knowledge is not useless and in fact their manual knowledge is the only thing that they have and that is what they have to work with and it is really great that is what we are persuading them. Now as I said earlier what is a sequence of balanced parentheses? Well it really needs a recursive definition. So this could be a motivation for teaching recursion in the course.

So we will come back to that a little bit later on.

**(Refer Slide Time: 07:26)**

## Example 5: 8 Queens problem

Discussed by N. Wirth in celebrated paper [Wir71] on "Stepwise refinement"

Stepwise refinement: "General" algorithm design strategy

- ▶ Principle 1: Decompose tasks into smaller tasks.
- ▶ Principle 2: Program = sequence of steps in which specifications are refined.

Specification: conditions that must be satisfied by the output.

Refinement: As program executes, more conditions are satisfied.

Problem considered by Wirth: "On a standard 8 x 8 chessboard, place 8 queens such that no queen captures another."

2 queens capture each other if one is exactly at one of the 8 compass directions with respect to the other, i.e. to North, North-East, East, South-East, South, ...

"Same row/column/diagonal"

Required: Representation for queen placement + algorithm for finding non-capturing placement.

My next example is the 8 Queens problem. So what is this problem? This problem has been discussed by Niklaus Wirth who is another luminary another forefathers so to say of computer or computer programming. He wrote a celebrated paper on stepwise refinement as long ago as 1971 addressing some of the problems that we are talking about how do you design program.

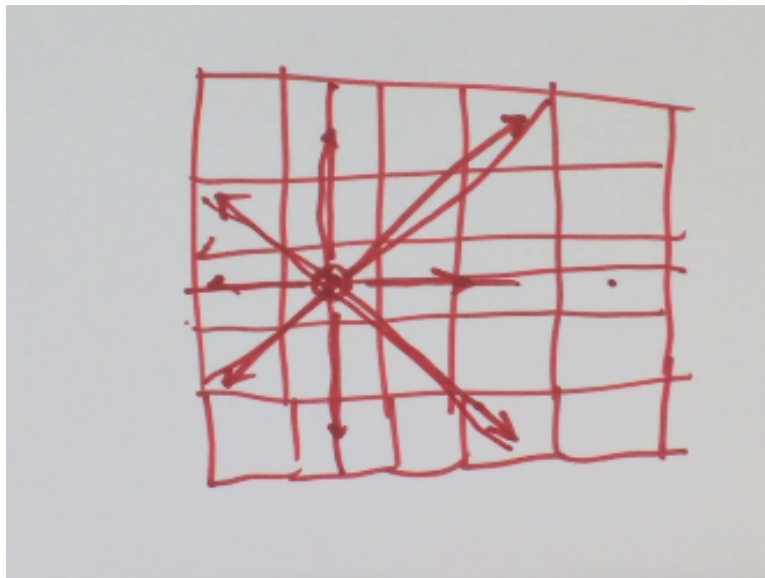
What is stepwise refinement? It is meant to be general algorithm design strategy and it says that if I want to write a program I need to decompose the big task of writing the program into smaller task. The second principle or the second way of designing programs is to think of the program as a sequence of steps in which the specifications get refined rather than thinking of the program as doing small tasks one at a time.

He says that you can think of the program, the program has a sequence of steps in which the specifications are getting refined and what is the specification? The conditions that must be specified by the output. So initially I might have said look that this is my output, but it does not satisfy all conditions. So I look at what conditions I need to satisfy and I try to get the output, I modify the output or I modify the variable which I am going to output.

And make that variable satisfy more and more conditions and the moment I satisfy all the conditions I can print out the answer. So I can think of algorithm design or problem solving as I have a big task to do and I break up into small tasks, but often I can think of my big task as satisfying a sequence of constraints and I am just going to get my program to satisfy more and more constraints.

So refinements mean as program executes more conditions are satisfied. So the problem that Niklaus Wirth considered was the 8 Queens problem which is that on a standard 8 by 8 chessboard and we want to place 8 Queens such that no Queens captures each other. So how does that work?

**(Refer Slide Time: 09:57)**



Well if you know what a chessboard is you probably know this problem already. I am going to draw a small chessboard and I am going to tell you what captures means just so that you understand the problem in case you do not know this, you do not know Queens and you do not know chess. So if I have a Queen placed over here then a queen is set to capture everything in its columns.

So everything that is north or south of it directly everything that is east or west of it directly everything that is say in the north east direction or the south west direction or the north west direction as well as the south west direction. So if there is anything over here that will be captured. So if I place a Queen over here I cannot place a queen over here or here or in these square or in these squares.

So two queens capture each other if one is exactly at one of the 8 compass directions with respect to the other. So we want 8 Queens to be placed on an 8 by 8 chessboard so that no Queens capture each other. Now why 8 well in any single row obviously there can be only one so therefore you cannot place more than 8. And therefore the idea is see if you can place all 8 is there a way to place 8.

So what is needed for solving this problem. Well we need a representation for placing Queens and we need an algorithm for a non capturing placement. So we are finding one more difference between real life problem solving or manual problem solving and computer problem solving. So real life problem may deal with problems which I have given to you on the board or it may deal with physical objects such as chessboards and Queens.

What we need to do is ask first what is the computer equivalent or we need to find a way of representing the chessboard and the Queen or more directly what is the queen placement, do we have a way of representing Queen's placement so that is the first part and then the second part is do we have an algorithm for finding all the non capturing placements.

**(Refer Slide Time: 12:40)**

### 8 Queens (contd.)

- ▶ Natural representation of solution: 64 bits Whether square has queen
- ▶ Better representation: 8 integers giving row position of queen in  $i$ th column.
- ▶ This is posed as an insight by Wirth.

Had students taken a physical board and tried placing actual queens, they would immediately discover that we do not need to consider the 64 bit solution.

- ▶ Wirth derives backtracking solution from scratch.
- ▶ Backtracking solution is quite hard to derive for introductory programming students.
- ▶ Much better: relate to prior training.
- ▶ Students know how to generate permutations from math.

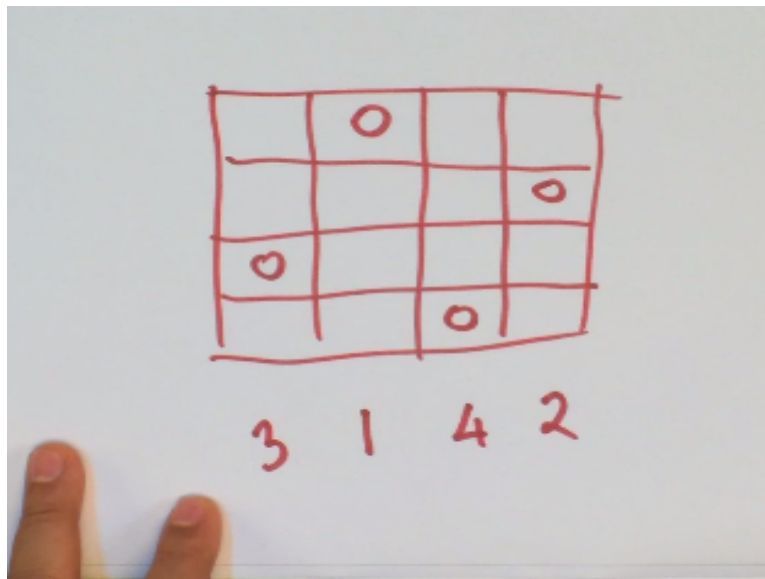
Point out that all possible ways of placing queens = all permutations, and so we proceed similarly.

Before we move to our recommendations let us go in the direction that Niklaus Wirth meant.

So Niklaus Wirth says that the natural representation of the solution is 64 bits why well there are 64 squares and we will use one bit to specify whether a given square contains a Queen or not. So if I give you a sequence of 64 bits you will know exactly which squares are to contain a Queen and which squares are not to contain a Queen. So that is the way how I can give you my solution to you.

Then he says that there is a better representation instead of me giving you 64 bits I can just give you 8 integers. So the  $i$ th integer gives the row position of the Queen in the  $i$ th column. I have one integer per column and that integer tells me where that Queen is.

(Refer Slide Time: 13:50)



So just to clarify this is what I might have. So I might have a Queen placed over here, a Queen placed over here, a Queen placed over here and a Queen placed over here. So to represent this Queen placement I might write down the string of numbers 3, 1, 4, 2 so this is where a 4/4 board so I will be giving you four integers. And from this you would be able to know that what I mean is that the queen should be placed over here, here and here.

So similarly we can do something for 8. Incidentally you might see that these four Queens are placed so that they do not capture each other. So this would be a solution to the 4/4 problem. Now the 8 integer representation is better because first of all it has fewer items to deal with. It is indeed better, but how do we get it.

How did we even think of 8 integers in the first place 64 bits is natural? The question says we want to place a Queen possibly in each square so think of 64 bits. So think of a bit per square so 64 bits but why 8 integers. This requires a little bit of thought and as Wirth says it is an insight, but if the students had taken a physical board and tried placing actual Queens they would immediately discover that we can never place 2 Queens in the same column.

And therefore we do not really need 8 bits per column we just had to say where the queen in the first column is going to be where the queen in the second column is going to be and therefore this better representation might have been discovered by our students just by manual work just because they are working with things manually. Another point to be noted in this example is the word derive the backtracking solution from scratch.

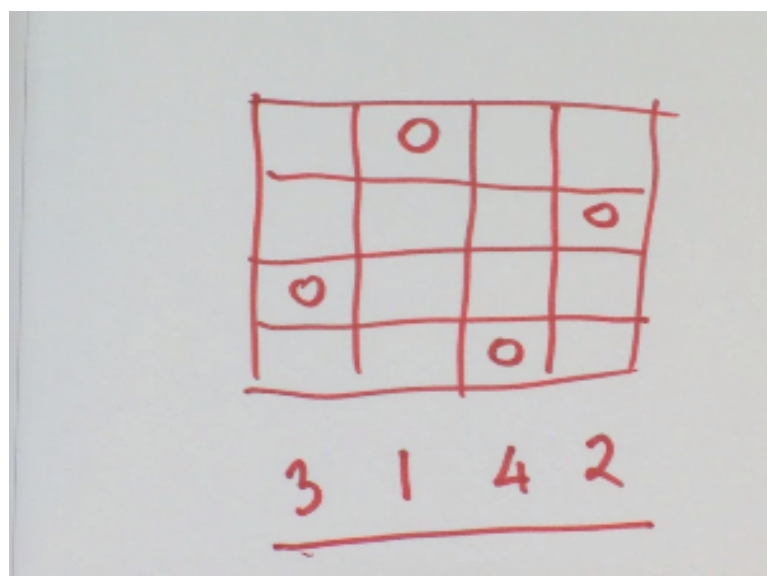
So what is the solution the final algorithm given by Wirth. Wirth says place the first Queen, place the second Queen check if it has been captured by the previous Queen. If not go to the next possible positions. So do it until you are unable to place a certain Queen. So if you are unable to place the fifth queen then go back and use the next position in which you can place the fourth Queen and you carry on in this manner.

Now this is the so called the backtracking solutions. People might actually be teaching this backtracking solution typically in the third year not really in the first year. So it is somewhat complicated. So the point is you certainly cannot expect students to derive or to invent this solution and because it is taught typically in an AI artificial intelligence course in some later semesters we can agree that it is not an easy thing.

So what should you do then. So backtracking solution is quite hard to derive, but it is not that difficult if you relate it to prior training. What do I mean by that? Well students have been taught permutations and combinations in Mathematics courses. So how do you generate permutations of 5 numbers or 5 people in 5 chairs. So that kind of thing people have been taught.

So what you should be saying over here is all possible ways of placing Queens is really all possible permutations of 8 integers or the integers one through 8 because if you come back to this solution what is this.

**(Refer Slide Time: 19:00)**

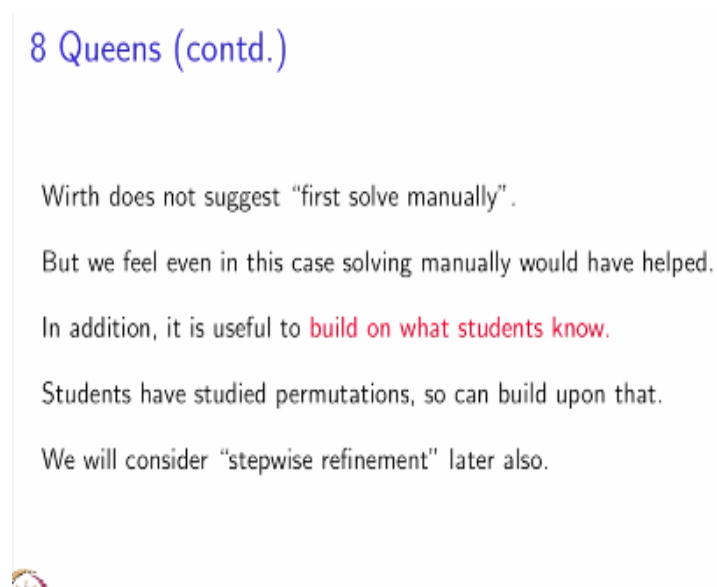




We know that in this sequence each number between 1 and 4 is going to appear exactly once and all possible numbers are potentially candidate positions for placing Queens. They may not give you non capturing Queens positions, but they will give you candidates which you need to consider. So in the 8 Queens problem you really want all possible permutations. So in 11th and 12th Mathematics you actually have done this problem of generating all possible permutations.

So go back to that and relate to what they already know or what they already know manually and from that if you want you can derive the solution to the 8 Queens problem.

**(Refer Slide Time: 19:50)**



8 Queens (contd.)

Wirth does not suggest "first solve manually".

But we feel even in this case solving manually would have helped.

In addition, it is useful to **build on what students know**.

Students have studied permutations, so can build upon that.

We will consider "stepwise refinement" later also.

So the points to be noted Wirth does not suggest that first solve manually. Wirth directly jumps into writing a computer program. So we were saying whether this strategy of first solving manually is obvious or not evidently it is not obvious. Wirth does not use that strategy, but we feel that even in this case solving manually would have helped a great deal and we want to point out that it is useful to build on what students already know.

So students have facility with working with physical board so use that facility. It is kind of what is it called kinesthetic some kind of thinking through your body. So it is useful to build on what students already know what they know how to do and also to refer to their Math knowledge of permutations which is immediately relevant over here. So students have study permutation so you can build upon that.

And stepwise refinement that Wirth points out is important and we are going to study it later

on, we are going to build upon it later on, but we feel that really coming back to our theme solving manually is a good idea and when we solve manually there are equivalents so what does it mean to go to the next position. So all those equivalence says between advancing Queens on a physical chessboard and doing the same thing on in an array can be discussed.

And should be discussed especially if you are finding that some of your students are struggling with it, but for struggling students you certainly should say look tell me how would you solve the problem manually because that is the domain where students are actually very comfortable.

**(Refer Slide Time: 21:56)**

### Summary: Manual computation and translation from it

- ▶ Get students to introspect and precisely state how they solve problems.
  - ▶ Encourage giving names to values they compute, e.g.  $s_i$  = sum in iteration  $i$ .
  - ▶ Encourage parameterization, e.g. if there are  $n$  rows, then ...
- ▶ Teach the incremental skill needed to translate the precise description of manual computation to computer program.
  - ▶ Explicitly teach how to design variables.
  - ▶ Discuss computer equivalents of manual operations such as "make a list"
  - ▶ Clarify to students that a computer program does not "see everything at a glance", but can only look at data through a small window.



So I am going to summarize the content of this lecture. So manual computation and translation from it what have we learned or what can we take away. The first point to be noted and I cannot emphasize this more. Get students to introspect and precisely state how they solve problems, encourage giving names to value they compute. For example, they might say  $SI$ = sum in iteration.

$SI$  is not a variable. It is just a name given to a value for ease of future reference. You want to encourage students to do parameterization. So if there are  $n$  rows that  $n$  should somehow come into your description then the second point is we should teach the incremental skill needed to translate the precise description of manual computation into a computer program. So what does it mean to make a list, what does it mean to find the next closing parentheses which is adjacent to an opening parentheses. What if there are spaces are in between?

Well in your computer program you will have code which will help you skip over the spaces. You will have to explicitly teach how to design variables, manual operations how to translate them then we have to clarify to students that a computer program does not see everything at a glance, but can look at the data only through a small window. So that ends the current lecture.

**(Refer Slide Time: 23:48)**

Should we teach students (manual) problem solving strategies?

In the next lecture we are going to address the question of should we teach students manual program solving strategies. So we will take that in the next lecture. Thank you.