Out for Shopping-Understanding Linear Data Structures English

[MUSIC PLAYING]

[MUSIC PLAYING]

TANZEELA ALI: Hi, it's Tanzeela Ali. I'm a software engineer, and also a teacher at Superior University, which is in Pakistan, in the city of Lahore.

Now, Lahore is known as a city of lively Heart people. We like to keep ourselves engaged in activities such as food, traveling, shopping. And in today's lesson, we're going to do the same thing. We're going to learn some computer science while we are on shopping. So are you excited about what this journey brings for you? Let's get started.

SYED WASIF RAZA: Hello. My name is Wasif Raza. I'm a graduate from Forman Christian College. But today, I'm a student again, just like you guys. And the class is going to begin soon enough, so let me make sure I have all the stuff that I need. So you know, I wouldn't want to start my class without my pen and notebook.

OK, let's see. Oh, no. It looks like I've left all my stuff at home. Well, I better go down to the bookstore and get all my stuff before my class starts. So why don't you come join me, guys?

Now, let me try and think of all the stuff that I need. I'll need a notebook, a pen, a file cover, a calculator, and a highlighter. Let's try to memorize all these things. A notebook, a pen, a file cover, a calculator, and a highlighter.

You guys see what I did? For each of the items that I need to bring from the store, I have allocated a finger of my hand. Now, this means that for the five items that need to get, I have pre-allocated slots in my memory and I have defined them for these items. This sort of an implementation is called an array based implementation.

Now, in this particular case, I'll be going to the store, and I have stored all my data in an array. Now, do you think this is a good implementation for this case? Because if I'm on the way to the store and I suddenly remember that I need to get something else, would I be able to accommodate that new item?

Why don't you guys give this a thought? Discuss with your friend and teachers, and I'll get back to you real soon.

MEHAK ALI: Hi, my name is Mehak Ali. I'm a student of Punjab University College of information technology. And right now, I'm here for my class of data structures.

But wait a second. Where is Wasif? I think he is late again. He's really a forgetful man. Seriously. Well, in the meantime, let's just learn something.

As we have seen in the previous example, arrays can contain limited amount of data, whether it's static or dynamic. If the array is full, we have to create another array of larger size and copy the data off the previous array, which is quite time consuming.

Well, a better approach is to use a link structure. A linked list, as the name depicts, consists of a linked structure. It starts from a node called head node. Representing the start of a list.

A node of a list may be considered as a boy having two hands. One is carrying the data, and the other one is carrying the address of the next node. And same is the case with other nodes, and so on, but the last node pointing to nothing.

So what do you think? What difference does it make? Have a discussion with your fellows, and we'll see you soon.

TANZEELA ALI: Hi, welcome back after the break. Let's sum up the findings.

If we need to insert a new data element, then the previous node will simply break its link to its next node and will start pointing to the newly added one. So in this way, data can be inserted with two simple steps without disturbing the other elements of array.

Same is the case with the size of array. Each element is added without having a need to be stored consecutively, as the previous node is storing the address of next element. So a linked list can store its nodes anywhere in the memory, and you can add as much node as the memory allows.

But with some advantages, linked list also comes with some trade off. One of them is you can only go further in a list, not backward, because a pointer to the previous node is not stored anywhere. Another tradeoff we see is the size of list. As you have to store the address of next node as well, so the size increase by double.

SYED WASIF RAZA: Hello, everyone. Welcome back. Now you know the main difference between an array and a linked list. Now, an array only lets you store data in pre-defined slots in your memory, which means you cannot add anything else once those slots have been filled, while the linked list is more dynamic. It lets you add and remove elements at your will.

Now, if I'm going to bookstore and I suddenly remember that I don't need the calculator that much, all I need to do is remove it from my list and I'll be good to go. So now I'll be on my way.

ALI: Hey, Wasif.

SYED WASIF RAZA: Hey, Ali. How are you, man?

ALI: How are you?

SYED WASIF RAZA: I'm fine, as well. I'm just running to the bookstore.

ALI: Oh, great. I need some sheets for my test.

SYED WASIF RAZA: You need some sheets for your test. I'll get some sheets for your test, buddy.

ALI: Thanks a lot, man.

SYED WASIF RAZA: OK. I'll see you right here when I come back, OK?

So see how easy it was? All I needed to do was added element into my list that was new, and I'm just ready to go. Now, let's just continue on--

AHMED: Hey, Wasif.

SYED WASIF RAZA: Hey, Ahmed. How are you?

AHMED: Good. Are you going to the bookstore?

SYED WASIF RAZA: Yeah, I'm going to the bookstore.

AHMED: Oh, can you help me out please?

SYED WASIF RAZA: Yeah, what do you need, man?

AHMED: I just need a marker.

SYED WASIF RAZA: You need a marker. OK, I'll get it for you and I'll meet you right here again, OK?

AHMED: I'll be waiting.

SYED WASIF RAZA: So now you can see in most situations, you almost definitely have to add more elements into your list. And now-- hey, Bakr, how are you?

BAKR: Fine. What about you?

SYED WASIF RAZA: I'm fine.

BAKR: Are you going to the bookstore?

SYED WASIF RAZA: Yes, I am going to the bookstore.

BAKR: I need a notebook. Can you bring it for me?

SYED WASIF RAZA: Yeah, sure. I can get a notebook for you, buddy.

BAKR: Thanks, man.

SYED WASIF RAZA: Thank you. OK.

So we've got our work cut out for us. But I just realized one thing. I'm going to the bookstore right now, but when I'll be coming back I'll be meeting all my friends again, but in a different pattern. Why don't you guys take a bit of time, discuss with your friends and teachers, and I'll get back to you.

Hey, guys. Welcome back. So did you figure it out?

Right now I am going to the bookstore, but when I'll be coming back, I'll be backtracking. And the friends that I met first will be the friends that I meet

now last, meaning that the friend who asked me at the end to get an item for them will now get their item first.

This sort of an implementation is called last in, first out. And there's a data structure in computer science that follows this exact principle. It's called a stack.

Just like a stack in real life, when you place a couple of books on top of each other, the book that you place on the top will always be the first one that you remove from the stack. The computer science stack works in the exact same way. You just keep placing your data on top of each other, and when you pick something up, the thing that you place in the end will be the thing that you place up first.

Now, I'm going to go on my way to the bookstore while you guys learn a little more something about stack. I'll see you soon.

TANZEELA ALI: To keep it simple, stack is a data structure that follows the rule of last in, first out. In stack, data can be inserted and retrieved from a single access point. If the data needs to be inserted, it will be added to the top of stack, also called pushed to the stack. While retrieving the data that was inserted the last time will be the first one to come out, also called pop.

Now that you know what a stack actually is, go for the examples in real life as well as in computer science. Discuss with your fellows and teachers, and we are waiting for you here.

MEHAK ALI: Hi, welcome back. I was having a snack break. Do you want one? You know what, what I'm thinking? I'm just trying to get the last chip before eating the other one, but I just can't. Do you know why?

Take this pack as an example of stack. At the time of manufacturing, the chip which was inserted at first is now at the bottom of the pack. Same, the chip which was inserted at last is on its top, so I cannot remove the other entries before taking out the topmost entry, and so on.

Well, we have many more examples of stack in our real life. Now, take this wrist as an example of a stack. Now, I'm going to put these bangles one by one on my wrist, or should I say I'm going to push those elements one by one in this stack.

So first of all, now, the yellow one, then the red one. Then blue one. As you have seen, I have pushed three elements on this wrist, or should I say, on the stack. Now, what if I want to get the yellow bangle before the other one? But it's impossible.

So when I want to remove them, first of all, I will remove this bangle, blue one, which was inserted at last. Then the red one. And after that, the yellow one, which was inserted at first.

You can find many examples of stack in computer science, as well. Undo and redo in many word processing software uses stack. Whenever to do something, undo, the most recent thing you have done is undone. Same as redo thing works.

Navigation inside a browser also uses stack. By clicking the back button, you can reach the most recent page.

Function call hierarchy also uses stack. Now, we have two functions, F1 and F2. In this case, our code snippet is calling function F1, which will be pushed on function called stack. But before reaching the return statement of F1, we have called another function, F2. So F2 is pushed on stack.

Now, F2 will be executed line by line. And after executing, its return statement, F2, will be popped back from stack. And then F1 will be popped back after its execution.

So we have seen many examples of stack in our real life and computer science, as well. Now, let's see where Wasif is, because our class is about to start, and it's really getting late.

SYED WASIF RAZA: Welcome back, guys. Wow, you're learning so much. Now you know what is last in and first out, the principle on which the stack data structure works. Now I have another situation for you guys.

Here you can see a bunch of customers waiting for the shopkeeper to entertain them. Now, think about this for a moment. If the shopkeeper chooses the last in, first out implementation to serve his customers, will it be a good approach?

Now, I need you to play a game. You will play the shopkeeper, and five of your friends will play as customers. Now, each customer will come to you

after a time span of 10 seconds, meaning a friend will come to you. He will ask you to get them something, and you will take 15 seconds to complete that request.

Now, each friend will come to you after a timespan 10 seconds and you will take 15 seconds to complete their request. Now, why don't you do this and tell me what is the disadvantage of using last in first out in a situation like this, and I'll see you soon.

TANZEELA ALI: Welcome back. I hope you have figured out the drawback of using stack to store the customers on a bookshop.

We see that if the customers are stored in a stack, then imagine what if the customers keep coming and add their selves in a stack? The shopkeeper may serve them using the last in first out rule. But what if he's not fast enough to serve all the customers, or the wait prolongs, or if the customers keep coming? And what will happen to the poor fellow who came before all of them? He may get angry, or even leave the store.

For this purpose, a data structure is needed which follows the rule of first in, first out. A queue is that. A queue serve the purpose. A queue is a structure of having two ends, one for inserting the data, and the other one to retrieve the data. So the data that was inserted first will be the first one to come out.

So now, if you use a queue instead of stack to store the customers in a bookshop, the person who came before all of them will be the first one to be served. Isn't it fair now?

Now that you know what a queue is, let's look for the examples of queue in computer science.

MEHAK ALI: Queue can be observed commonly in our daily life, as well. As we have seen, on cash counter, when you are checking out items, people who are standing in front of you will check out their item first. And if someone come after you, their turn will become after your turn.

Same is with some computer science applications. In some processing algorithms, operating system uses queue to schedule processing, just like CPU is working on some application and doing processing, other application request has arrived. That request will be pushed into queue until the previous one is not processed.

See? First come, first serve, or should I say first come, first process. Same is the case with the event handler of a web server. When you are using a shared printer with your computer, printer does this job by using a queue when many printing requests are made.

So today we have learned many new data structures-- arrays, linked list, stack, and queue. So what do you think? Which data structure is best among them? Discuss with your fellows, and we'll see you soon.

SYED WASIF RAZA: Hey, guys. Welcome after your long and detailed discussion about which data structure is the best. Now, some may think that a list is better because it helps you insert and remove an element from anywhere on the list. Some may think that a stack is better because it helps you in many practical examples, such as undoing a function or maintaining a hierarchy of functions while you're coding. Some may think that a queue is better, because it helps you maintain your data in a first in, first out fashion.

Well, are you ready to find out which one is the best? The answer is none of these data structures are better than the other. It completely depends on the problem you're trying to solve.

Now, we saw that each data structure has some specific characteristics that they exhibit. These characteristics map upon the problem's attributes.

So in order to find the correct solution to a problem, we need the correct data structure. If a problem needs to be solved using a stack and you use a queue instead, you might be able to solve that problem, but it will not be efficient, nor will it be correct.

To sum it all up, there is no data structure that is better than the other. It completely depends on the situation you are trying to apply that data structure to.

Now, since I've got all my stuff, I'll be going to my class.

Hey, guys. The data structures that we studied today aren't the only ones out there. There are many other complex problems that need to be solved with more complex data structures. Now, today's lesson was just to give you a basic understanding about the most common data structures that used in the industry today, but there are many other problems that cannot be sold very

effectively with these data structures that we studied today. Isn't that right, Tanzeela?

TANZEELA ALI: Exactly. The knowledge we gained today is just a basic understanding of the data structures. There is so much more to learn about in this context. For example, the data structures we learned today are linear, which means they grow in a linear fashion.

There are data structures which we can call non-linear. For example, we can take the example of a 2D queue. Consider the example of a cash counter where two cashiers are available to serve you. Both of them have different queues, and they can entertain different customers. One more example is the priority queue, in which data is stored in a first in first out manner, but it is also stored on the basis on priority, and retrieved in the same fashion, as well.

MEHAK ALI: Graph is one of the various data structures which consist of nodes that are called vertices, and they are connected with each other through edges. It's just like the example of cities that are connected with different kind of roads.

SYED WASIF RAZA: This was also just a basic introduction of the advanced data structures. There is still so much out there for you guys to learn. The purpose of this lesson was just to introduce to you the basic data structures and give you a taste of the advanced data structures, as well. We wish you best of luck with future learnings.

TANZEELA ALI: Thank you so much for giving your time to our module. goodbye.

Hi, thanks for choosing our module. This module focuses on the data structures, which are the different techniques to store data in computer science and how this data can be manipulated effectively.

Now in this lesson, we have tried to relate the real life stuff with the storage of data in computer science, how you can store the items of lists while you are on shopping, store the map of your friends in a stack, or you can store the customers in a queue to manage them effectively.

And in today's lesson, we have divided the module in four equal parts. In the first part, we will see how a linked list is better than an array. In the second

part, we will see what is a stack, and what is its significance over the list. In the third part, we will see what is queue, and how it is effective in storing the data while a queue is needed. In the fourth part, you have to organize a poll in your class and ask your students which data structure do you feel is better than all of them.

In the last segment, you have to conduct a poll in your class. You have to ask the students which data structure is better than all of them. They may vote for their favorite data structure. And in the end, you have to make them realize that no data structure is better than anyone.

We have divided this lesson into different segments. And in each segment break, you can perform some activities. The detail of all the activities is provided in the document attached with the module.

This lesson provides the initial understanding of data storage and data structure in computer science. There is so much more to learn about it in the future. Encourage your students to ask the question as much as they can, as it could be a fruitful thing for them.

At the end, thank you very much for giving your precious time to our modules. Goodbye.

[MUSIC PLAYING]