CM2035 Topic 7 Linked Lists, Stacks and Queues

Ian Sanders

October 2025 Session

UNIVERSITY OF LONDON

Recap

In the module videos we have seen

- The idea of a linked list as compared to an array,
- inserting into, searching and deleting from a linked list,
- stacks,
- inserting into (push), searching (peek) and deleting from (pop) a stack,
- queues, and
- inserting into, searching and deleting from queues of various types.

UNIVERSITY OF LONDON

Another ADT

In this webinar we are going to look at a slightly different type of queue.

A priority queue

We will do this for two reasons:

1. To reinforce the ideas behind dealing with linked lists, stacks and queues.
2. Priority queues are useful abstract data types in Computer Science – they are often used in graph algorithms.

UNIVERSITY OF LONDON

What is a priority queue?

> *In computer science, a priority queue is an abstract data type similar to a regular queue or stack abstract data type. Each element in a priority queue has an associated priority. In a priority queue, elements with high priority are served before elements with low priority.*

Wikipedia, 2025

UNIVERSITY OF LONDON

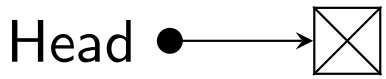
Example

Suppose now that we have an application where the smallest number from a collection of numbers is always the first to be processed.
We do not necessarily know all of the numbers at the beginning but we always use the smallest number that we have seen so far as we go along.
Suppose we start off with the numbers 2, 23, 5
Our queue would then grow as
2
2 23
2 5 23
If we then use the first number the queue becomes
5 23
If we add 7 then queue becomes
5 7 23
Note that the priority queue is sorted at all times.

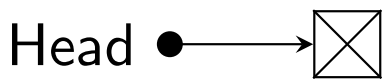
UNIVERSITY OF LONDON

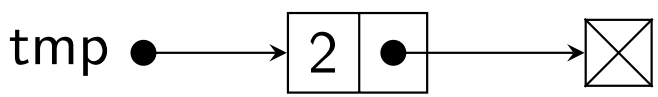
How do we use a priority queue to handle this?

Initially our queue is empty.

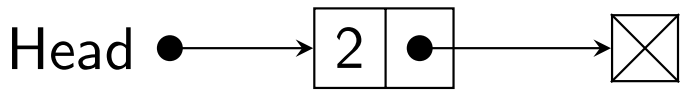


We need to insert 2 into the queue.
To do this we create a new node.





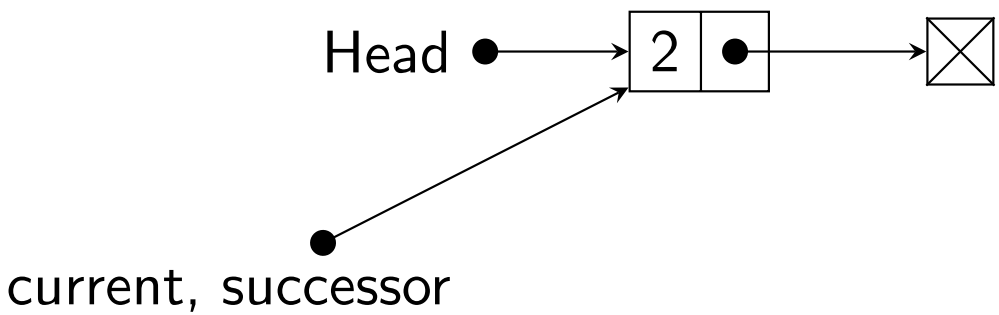
Here the queue is empty so we set `Head` to point to where `tmp` points.



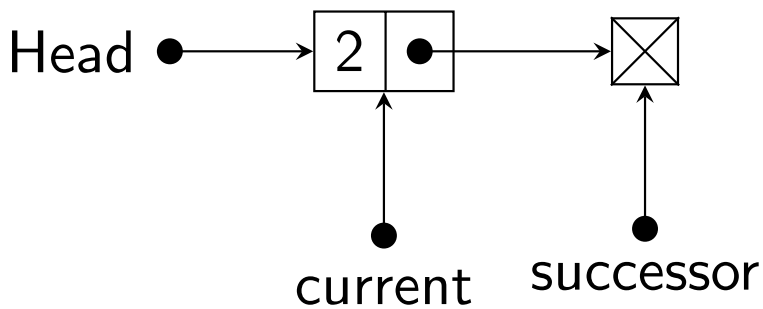
Now we need to add 23 to our queue.
The first thing to do is to search the queue to find where 23 should be placed.
Once we have found where to put it then we insert it there.

The process of doing so is shown below:

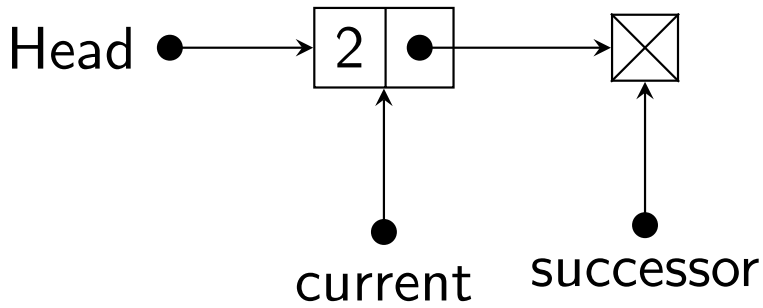
We create two new pointers `current` and `successor` to keep track of where we are in the queue.
We initialise both to point to the beginning of the queue.

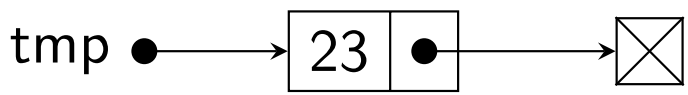


We start by comparing the value in the node pointed to by `current` with the number we are trying to insert.
Here the value we are trying to insert is bigger than the number pointed to by `current`.
We thus make `successor` point to where the pointer of `current` points.

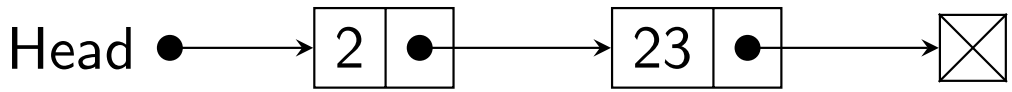


Here `successor` points to the end of the list.
This tells us the new node must be inserted between where `current` points and the end of the list (pointed to by `successor`).
We create a new node



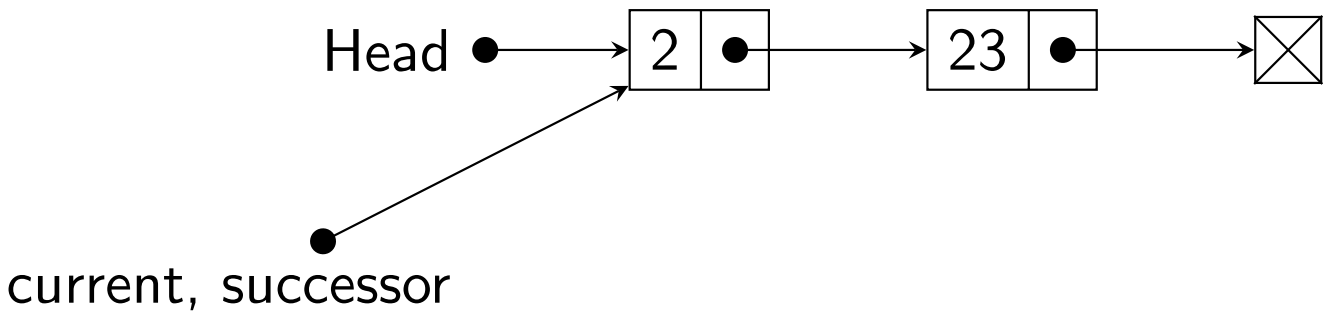


Now we set the pointer in the node pointed to by `current` to be `tmp`.

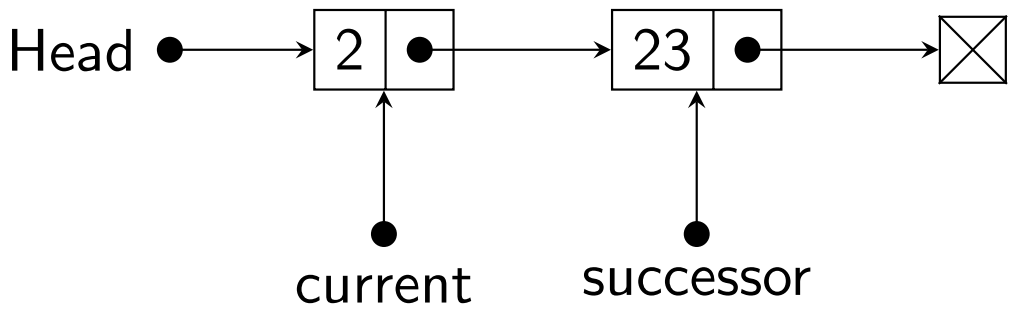


In this case we don't need to do anything else.

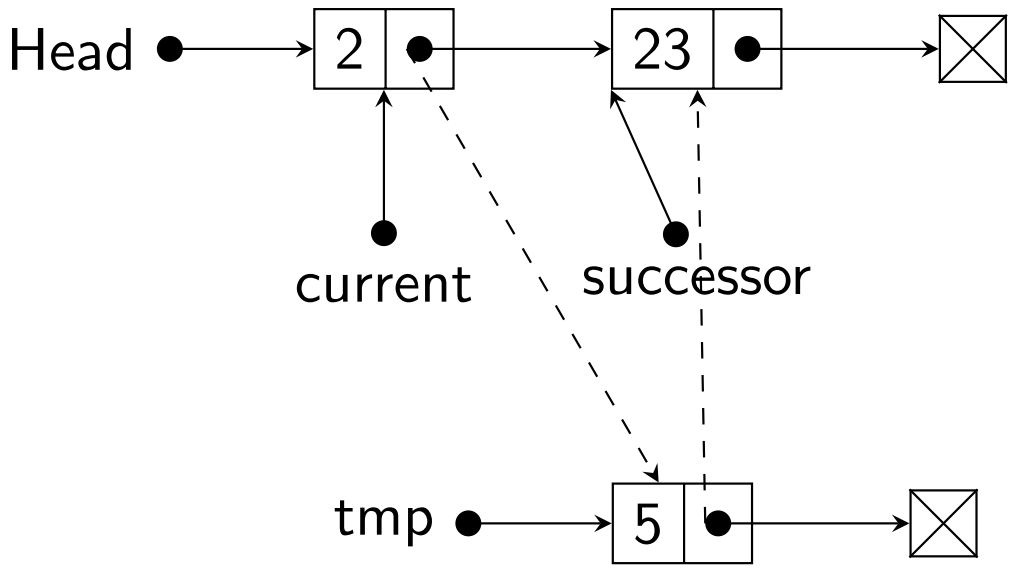
The next number to be inserted into the priority queue is 5.
We start the search process again.



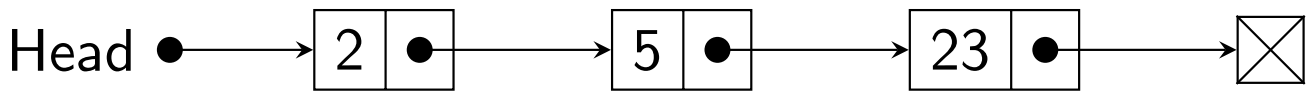
The value we are trying to insert is greater than that pointed at by `current`.
So we move `successor` on.



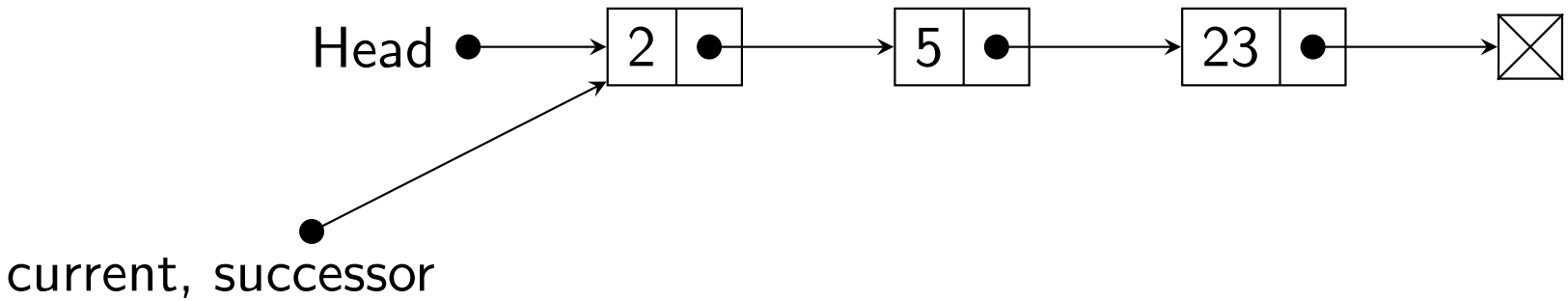
The value we are trying to insert is between where current and successor point.
So we create a new node and link it into the queue.

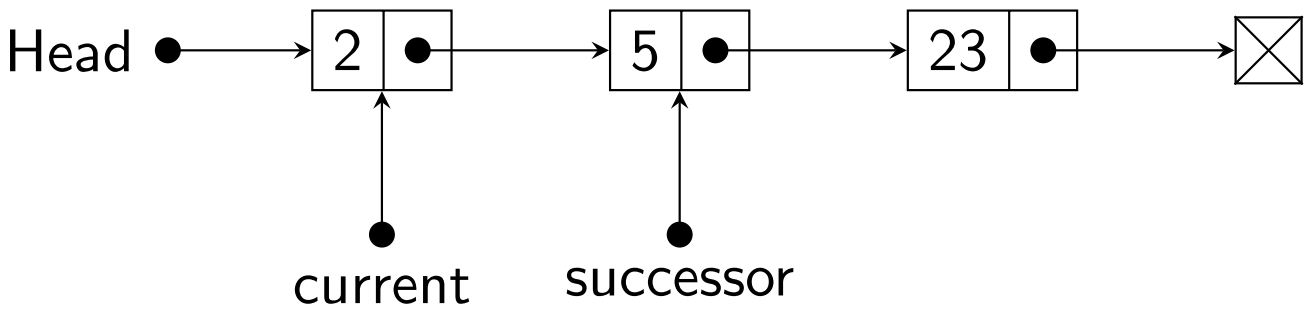


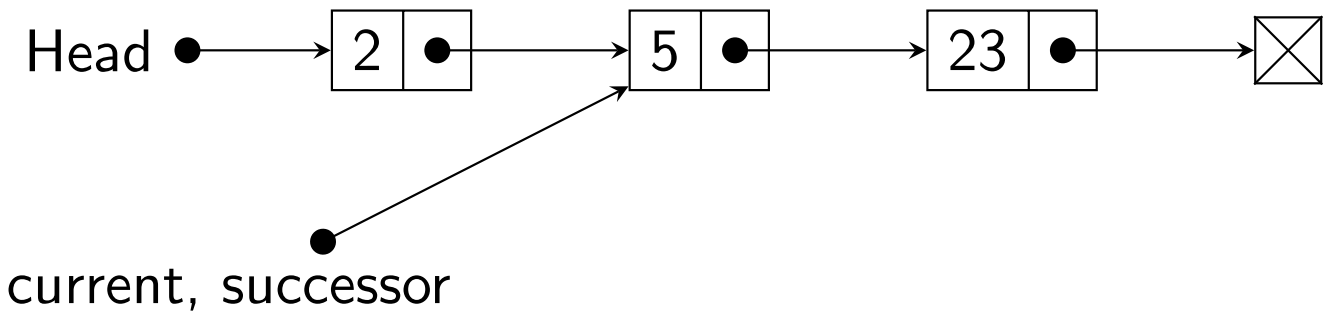
Our priority queue is now.

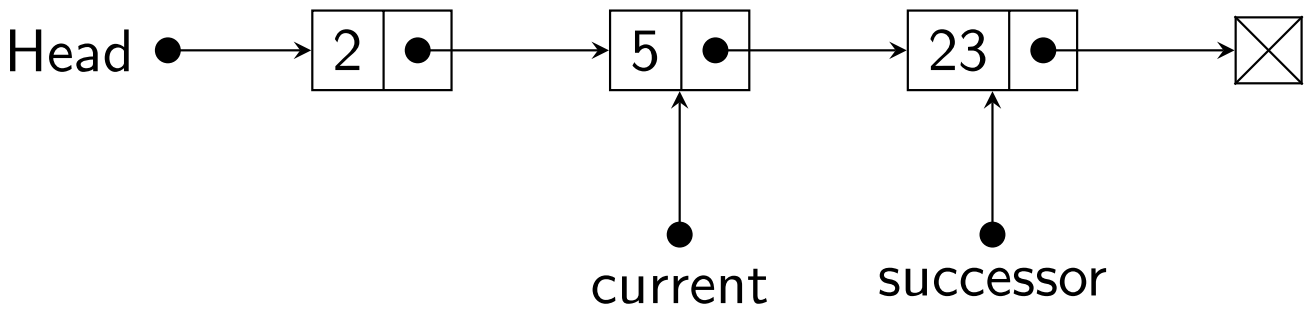


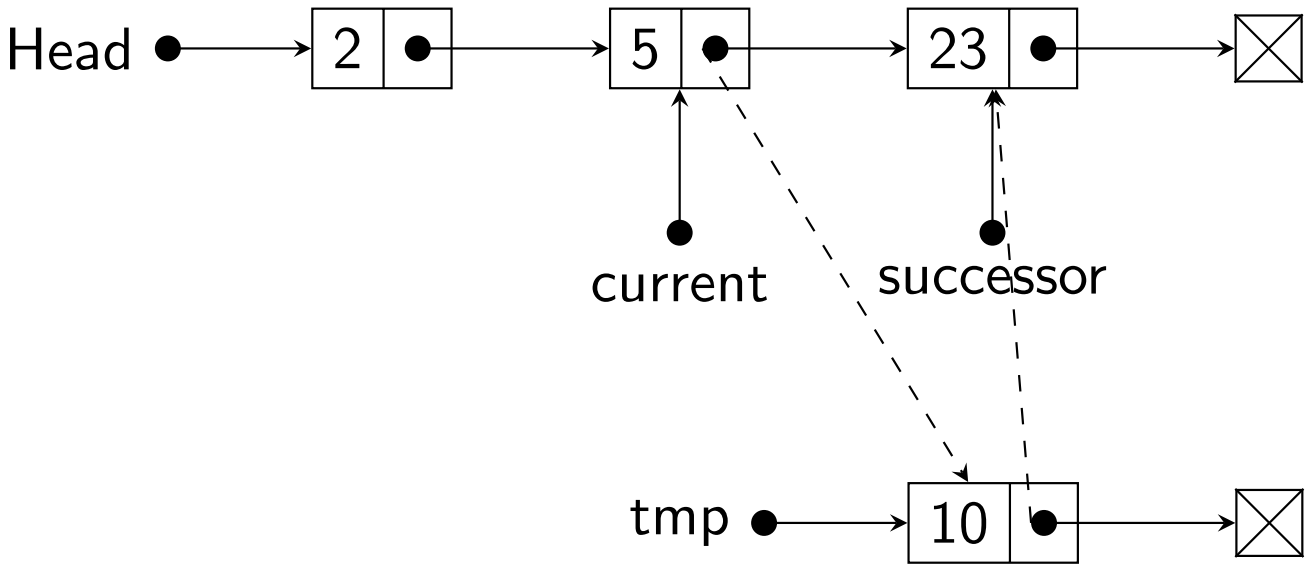
Suppose that we now need to insert the number 10 into the queue.
Once again we need to search for the correct position to do the insertion.
We start our search as before.



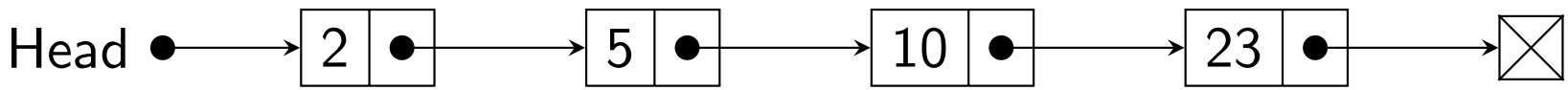




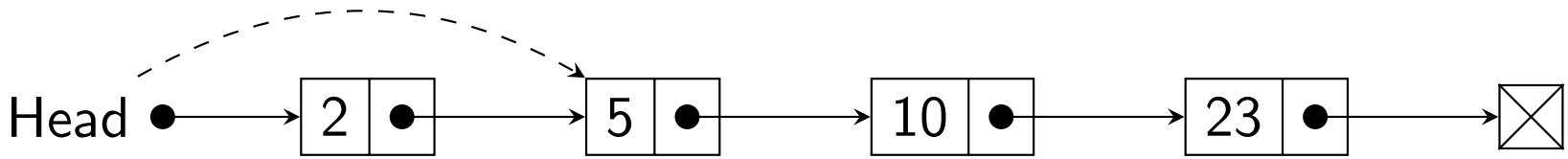




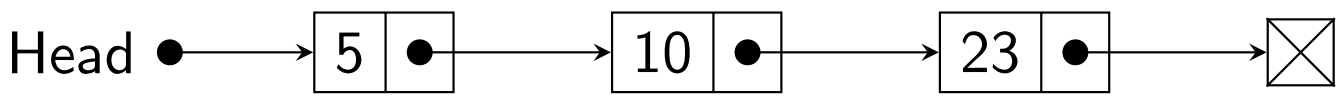
Our priority queue is now.



When we remove a number from our priority queue we always remove the first element in the list.
This is a simple change that involves making the head of the list point to where the pointer of the first element in the list points.



So the queue becomes



Analysis of priority queue operations

Insertion	Best case – insert at beginning of the list – $\Theta(1)$ Worst case – insert at end of the list – $\Theta(n)$ Requires searching and then constant time insert
Removing	$\Theta(1)$ – always remove first element in list

UNIVERSITY OF LONDON

Exercise

Write the pseudocode for inserting an element into and removing an element from a priority queue.

UNIVERSITY OF LONDON