11. The Other Queue Always Goes Faster (Queues and Stacks)

Waitin' for when the last shall be first and the first shall be last
Bruce Springsteen, The Ghost of Tom Joad,1995,
cf *The Bible*, St Mathew ch. 19, v. 26.

Queues and queuing are an integral part of our Society, especially for the English since if the stereotype is to be believed "*An Englishman, even if he is alone, forms an orderly queue of one*" (Mikes, 1946). There are many different ways to organise a queue however. Here we shall look at the three main types.

FIFO Queues

When I was at University, every year there was a draw held for accommodation. The higher up the draw you came, the greater the choice of accommodation you had for the following year and so the better room you got. Each week the next 20 names of the draw were announced. Even though each group of 20 names was drawn in order, that order was not used. Instead when the names were announced, each of the 20 had to fill in the form with their choice and get to the Accommodation Office as fast as possible. Whoever could run fastest (preferably while filling in the form at the same time) got first choice. The result was fairly chaotic.

Think of a group of people in a bank waiting for a cashier to come free. One way the people waiting could be organised is in a similarly random way – when a cashier comes free whoever gets there first is served next. Banks do not work like that, because fights would break out (as came close to happening outside the accommodation office). Instead, Banks organise the people. One way is to have a separate queue in front of each cashier. That is one data structure. However, it is very infuriating if you end up in the wrong queue, behind the child paying in 2000 5p pieces from their piggy bank that need counting! Banks do not want their customers angry, so they often use a slightly different organisation (i.e. data structure). They have a single queue, with the person at the front going to the next free position. That way it is fair for everyone.

Queues can thus be arranged in many different ways. The above queues have something in common, however. They are attempting to organise the people in a way that is fair in the sense that they are served in the order that they arrive. The first person to arrive is the first person to be served. This kind of queue is known to Computer Scientists as a **First-In-First-Out queue**, or **FIFO queue**. The queue at a bus stop is another typical example. In fact most things that we refer to as "queues" in every day life are FIFO queues. It is not the only form of queue however as far as computer scientists are concerned as we will see later.

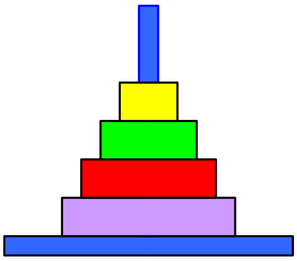
Chocolate, crisp and drink vending machines also use FIFO queues. They are filled from the back, with the next item being released from the front. They are not filled from the front because then some stock could remain in the machine for a long time, assuming the machine was rarely allowed to go empty. Items might then pass their sell-by-date. Supermarkets stock their shelves in the same way, adding new items with the longest sell-by-dates to the back, so the earlier sell-by-date items that were put there first are taken first from the front. Despite the slogan *Stack 'em High, Sell 'em Cheap*, in Computer Science terms, supermarkets use FIFO queues not stacks, which are something different.

Stacks

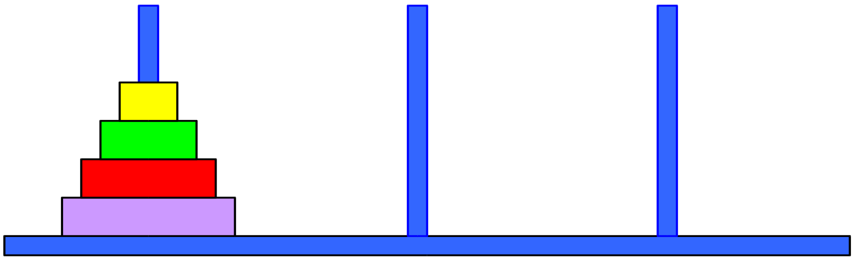
Think of a stack of chairs. They are also organised in a queue structure. However, it is a different kind of queue. It has the property that the last chair added to the stack is the first one to be removed. Try taking the first one that went on to the stack out first in a FIFO way. **Stacks** are **Last-In-First-Out Queues** or **LIFO queues**.

Stacks appear frequently in card games. For example, games like Gin Rummy have a discard pile. Players discard cards on to the top of the discard pile. A player can also pick up the top card from the discard pile and add it to their hand. They can not take any card that is not at the top. Similarly they cannot place a card into the middle of the discard pile. The discard pile is a LIFO Queue. The solo card game of Patience is also based around stacks. The cards are organised into a series of piles. Only the top card of the pile can be removed at any time and cards can only be placed on the top of each pile not in the middle. Each pile is thus a stack in Computer Science terms.

The way that stacks work is so important that most children are given a stack or two to play with by well-meaning relatives when they are toddlers! They obviously do so hoping the child will grow up to be a Computer Scientist! The toys that consist of a plastic pole on a base with a series of multicolour Polo-like disks are really just stacks. The disks can be threaded onto the pole, but can only be removed in the opposite order to the order in which they were placed. By playing with them, toddlers learn the rules for adding and removing elements from a stack.



A similar adult puzzle that relies on stacks is the Towers of Hanoi game. This game consists of three poles together with a series of disks of different sizes. To start the game the rings are placed on to the first pole in order of size with the biggest at the bottom. The aim of the game is to move all the disks to the third pole. However, only one disk can be moved at a time. Furthermore, at no time can a ring be placed on top of a smaller one. Again each pole is acting as a stack for disks though with extra restrictions.



One of the most unpleasant stacks is found in struggling businesses. The law requires that when people are made redundant, people are chosen for redundancy by some fair method. The most common "fair" method is called LIFO – last-in-first-out. The newest employees are the first to go. It is just a stack again.

Archaeology also works on a LIFO queue basis. As time passes, the layers of civilisation build upon the ruins and foundations of earlier generations. Centuries later Archaeologists start to remove the layers using fine tools and brushes. Thus the last layers added are the first layers removed. The most interesting things are likely to be the earliest, so the temptation must be to dig deep and start near the bottom in a FIFO manner, but this would risk damaging important finds, so a rigorous LIFO regime is followed.

A final place where stacks are evident is in the undo buttons in word processors and similar software. A stack of each operation performed is kept. Each time you type something a record of it is added to the stack. These actions can be undone, but only in the order that they were done. If you want to undo something you did 10 steps earlier, but leave everything else as it was, you cannot do it using undo. You would have to undo all 10 steps and start again from there. That is because only the top action can be removed from the stack at any time.

Priority Queues

A third kind of queue is known as a **Priority Queue**. This is a queue in which entries in the queue are given priorities and are dealt with in order of priority. This is the kind of queue often used at airline check-in desks. Airlines divide their customers into a series of classes such as "economy", "business" and "first" class. The higher the class, the more the customer pays. One of the things they are paying for is to be given priority so they do not have to queue for as long. Economy customers like me often have to queue for an hour or more to check in. First class customers expect not to have to queue at all. Business class passengers tolerate short queues but nothing more. If the airline gets it wrong they will lose their high paying customers to other airlines. They therefore do not use a FIFO queue, but use a priority queue. Each customer is given a priority corresponding to the class of their ticket. The check-in desks are then run in a way that ensures that passengers are served before those of a lower class. This is usually done using separate check-in desks for the different classes of passengers. The first class desk will only check-in business class customers if there are no first class passengers waiting. The business class desk will only process economy class customers when no business class passengers are waiting. You only have to wait if someone of equal or higher priority arrived before you. Thus the people of a single priority are arranged within a FIFO queue, as they are served on a first-come first served basis. However pushy he may be, a first class customer will not get served before another first class customer who arrived earlier.

Hospital waiting lists are also run on a priority queue basis. When someone is placed on a waiting list for an operation or to see a consultant, the urgency of the case is noted. If the complaint is minor, and the person in little pain, they will wait longer than someone who could die if their operation does not take place. All cities have a special high priority Casualty department, so that injuries that need immediate attention can get it. In recent years, there have also been suggestions that in Britain the patients of GPs from fund-holding practices have been able to jump to the front of the queue. Private patients have always been able to do this, as access to consultants is partly what they are paying for.

To-do lists can be organised as priority queues. This is especially easy if post-it notes are used. Most to-do lists contain urgent items that need to be done today, and less urgent items that do not need to be done for months, if ever. Ideally the most important or urgent things should be done first, irrespective of when they arrived. If your Boss phones to say he needs a new report by the end of the afternoon, you jump and ignore whatever you were doing. When a new task comes in, its post-it note should be placed in a position relative to its urgency or importance. The most important things are at the front of the list and they are done first. This is also a situation where the priorities of things in the priority queue may not be fixed. If your Boss phones and says the deadline for a report has changed from next week to first thing in the morning, its priority changes accordingly. The post-it note moves to a higher position in the list.

It is also possible to have a 2-dimensional priority queue, where each thing in the queue is given more than one priority assigned to it. Our to-do list could be organised like this. We spoke of two reasons for things to have high priority: urgency and importance. It is possible for something to not be too important, but if it is going to be done, it must be done by the end of the day. An example might be if I had an idea for a new exercise to give to a class. I already have other exercises, so it is not too important if I were not to turn it into a hand out, and perhaps I could do it on the board. However, if the class is tomorrow morning, if I am to do it I must do it today. It is urgent but not important. Writing exam questions on the other hand is important but not (right now at the start of term) urgent. I have to do it or I will be sacked, but not right now. If I give each task an urgency rating and an importance rating, I then can ensure I do not miss deadlines as well as ensuring that all critical things are done. A list is no longer the best way to organise my post-it notes. Instead I organise my priority queue as a table (i.e. array) with importance along one axis and urgency along the other. Post-it Notes are put in the boxes according to their urgency and priority. The thing I do next (the first thing out of the queue) is anything in the *very urgent – very important* box. If that is empty I then remove things diagonally *urgent – very important* and *very urgent – important* next and so on.

	very important	**important**	**not important**
very urgent			
urgent			
not urgent			

Summary

Queues are just lists where restrictions are placed on how elements can be added or removed.

A **FIFO queue** is one where entries can only be added at the end of the list and removed from the front. The first element to be added is thus always the first removed.

A **LIFO queue** or **stack** is a list in which elements are added and removed from the front: the last element added is always the first to be removed.

A **priority queue** is a list where the elements have priorities associated with them. The element with the highest priority is always removed first. If more than one element has the same priority, they are removed on a FIFO basis.