12. Seeing the Wood for the Trees (Dynamic Data Structures)

It is often better to be in chains than to be free.
Franz Kafka. *The Trial* (1925) Ch. 8.

Linked Lists
Queues and stacks are variations on lists where entries can only be added and removed in certain ways. Another variation on a list is known as a **linked list**. A linked list is just a list whose entries are spread around rather than being placed together. Each entry contains something indicating where the next one is so that the list entries can still be followed in turn.

Suppose you were organising a summer fete. As one of the events, you decide to hold a quiz about your local village high street (these days perhaps it should be about your supermarket) where each question is on a particular location. You could just give out to entrants a piece of paper with the questions on. This would be a list (or perhaps an array if the number of questions were fixed) in the sense of our previous discussion. An alternative however, would be to turn the quiz into a treasure hunt. You would then initially give out just a single question rather than them all. For example,

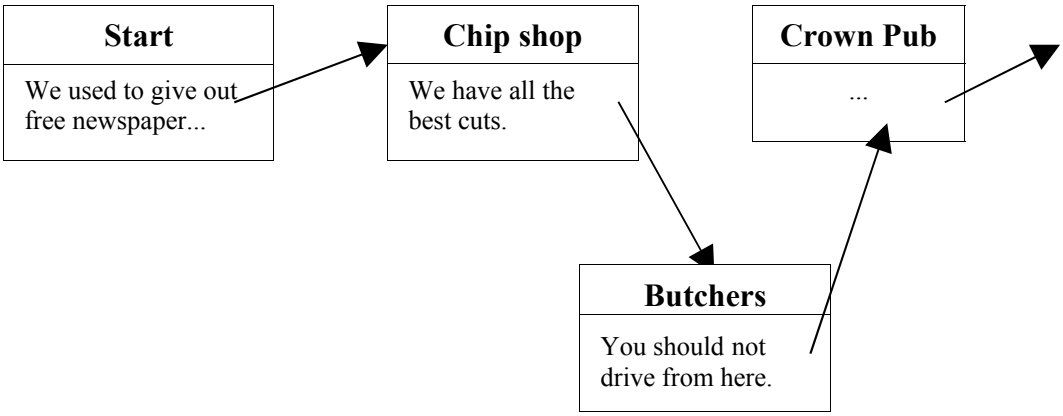
1. *We used to give out free newspapers but no one ever read them.*

The answer to this would be the fish and chip shop, since fish and chips used to be wrapped in newspaper. To get the next question, you would need to correctly work out the answer to this question, and then actually go to the fish and chip shop. At the Fish and Chip Shop you would find the next question. Its answer might lead you to the pub, and so on. Each question is pointing to the next location. The different buildings are forming a linked list.

By representing the information in a linked list, the quiz has changed. Now you can not just ignore questions. The only way to know what Question 3 is, is to go to the location pointed to by Question 2. To get to the end of the treasure hunt (or even know where the end is), you must go to every node in the list in order.

Suppose having designed the treasure hunt, and put out all the questions round the village – ie set up the linked list structure – you fall ill. The local butcher takes over the organisation from you. He wants to boost his trade so he decides to add an extra question about his own shop. Not wanting to be too obvious, he makes it the answer to the second question. However, since the treasure hunt was already set up, you only gave him the first question not the other locations. How does he add his question?

He must follow the chain to the chip shop, remove the question that is there, pointing to the pub and replace it with a new question pointing to his Butchers shop. He then follows the link to the Butchers and puts the question about the Pub he removed from the Chip Shop at the Butchers.



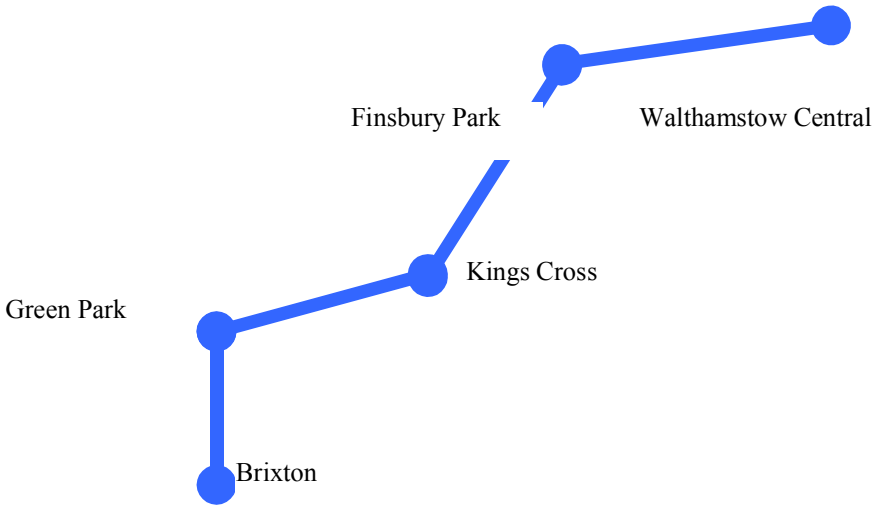
Adding a new entry in any linked list is done in the same way. We must follow the links to the node before the right position, and put pointer to the new position there. We then put the pointer that was there in the new location to re-establish the broken chain.

Orienteering courses also use the linked list structure: a series of places linked together in a chain. Orienteering is the sport where you have to map read you way round a course. It is like cross-country running except that you do not have to stick to paths and are not told the route to take. You are just told the series of points you must visit and the order you must visit them. You can either try to go in straight lines (through streams and thickets, down steep valleys only to go straight up the other side), or you can navigate round obstacles (going further to a bridge or contouring round the valleys). You are given a map at the start, with all the points you have to visit marked on it. The fastest person to complete the course wins. This is not necessarily the fastest runner, but the best combined runner and navigator. Orienteering is thus the sport of following a linked list through forest and moors as fast as possible. This is made more obvious in the string courses. These are special orienteering courses put on for toddlers at big events. A large cut-out figure of a cartoon character, or similar, is placed at each site that must be visited. A long piece of string is then run all the way round the course, linking the cartoon characters together. Each toddler is shown the end of the string at the start, and their task is to follow it to each of the characters. The characters are the nodes of the linked list, the string the links.

Doubly-linked Lists

The Victoria Underground line in London is also a linked list. It starts at Walthamstow Central, then chains together a series of stations in a single line, finishing in Brixton. The stations are the nodes of the list and the track the links. In fact it is a **doubly-linked list**. In all the previous examples of linked lists, the links are traversed in one direction only. A person travelling on the Underground can travel in either direction. Several times, I have been so engrossed in the book I was reading that

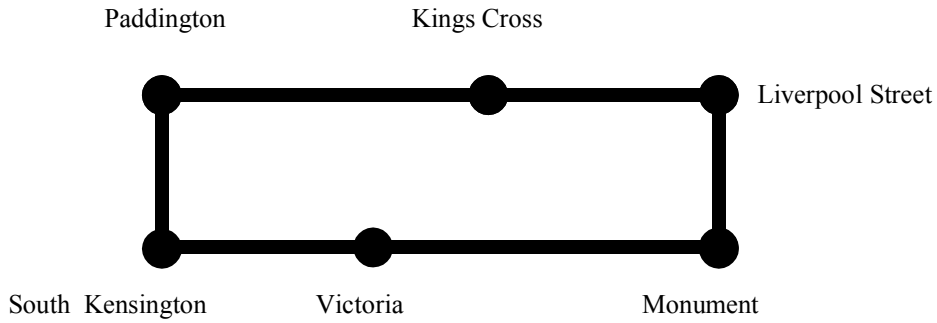
I have missed my stop. This is not a problem as I can just hop platforms and travel back in the opposite direction along the same link.



Circular Linked Lists

Linked-lists can be never-ending just like normal lists. An example of a **circular linked list** from childhood is the Daisy chain. A series of Daisy flowers are linked together into a necklace by piercing a hole in the stalk with your finger nail, and threading the next one through. Think of the flower heads as the nodes and the stalks as the links. Whilst being made they form a normal linked list. Once finished and the last daisy is threaded through the first, you have a circular linked list.

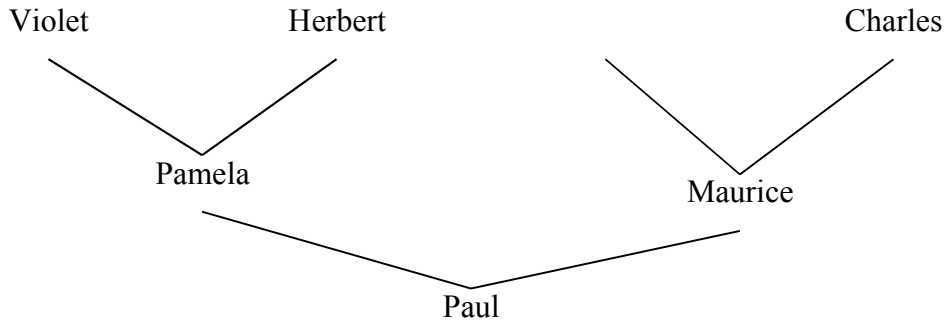
Back on the London Underground, the Circle Line is a **circular doubly-linked list**. Trains travel in either direction round a never-ending loop. Break the loop and you get a normal doubly-linked list again. Thus if a part of the Circle line needed to be closed for repairs, to get between what used to by adjacent stations, you would instead have to travel the full length of the line unless you switched to a different line.



Trees

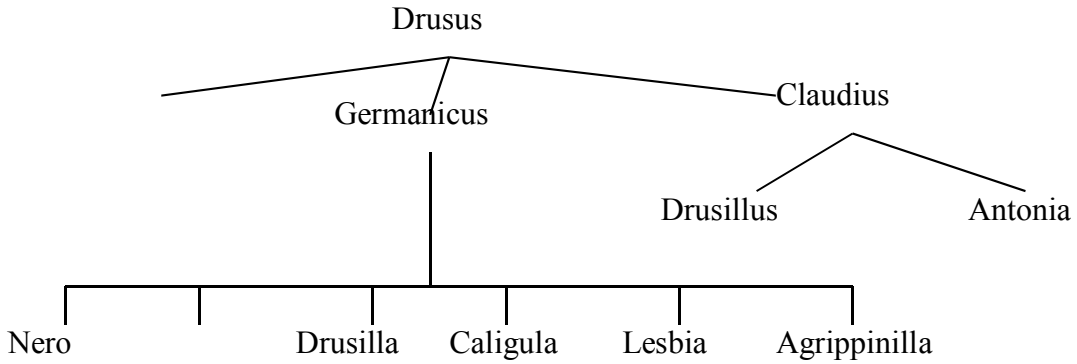
Whilst list and array like structures are common there are other non-linear ways of organising information that can make the structure easier to use. When someone tracks down their ancestry, the way they represent the information collected about their grandparents, great grandparents and so on is not as a list but as a family **tree**. At the bottom of the page you write your name, placing your mother and father above you, each connected by a line to you to show the relationship. Their parents are each

placed further up the page, and so on. In this way you get a tree like structure, with information (here the names of people) stored at the joints. The bottom-most joint (with your name in it) is known as the **root** of the tree. Each junction is called a **node** and the lines joining nodes are called **edges**.



This tree is a **binary tree** since each person only has *two* (genetic at least) parents so each node has two edges coming out of it.

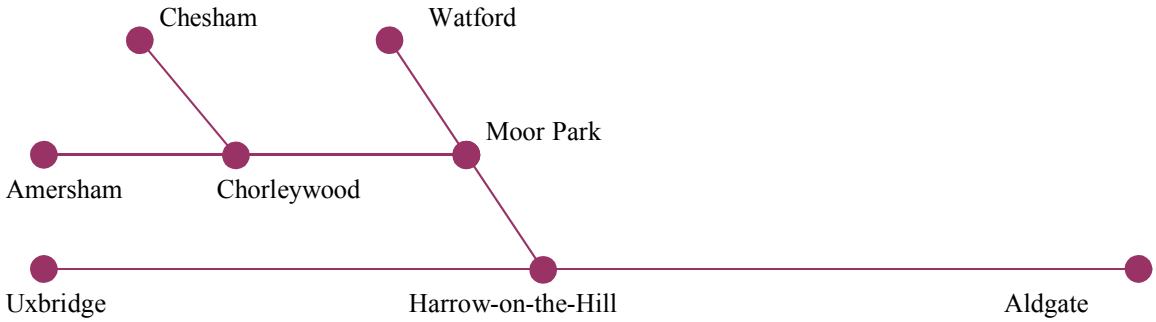
The similar problem of representing the descendants of a famous person from history would also normally be given as a tree. This time however it is not binary, since now we link a person with their children, and there is no fixed restriction on the number of children a person can theoretically have.



The Descendants of Drusus:
A Portion of the Family Tree of the Roman Emperors
Adapted from (Graves, 1941)

Taxonomists also use tree data structures to classify living things. Each level of the tree is given a name. The first level gives the *Kingdom* (plant or animal), the next the *division*, and so down to the individual *species* (Human, Herring Gull, Oak, etc). This tree structure was not the first way that living things were classified. Early attempts including listing things by properties such as their colour or sexual characteristics. These other ways of organising the information were not very successful, however. The reason that a tree structure works best is because it mirrors the way living things evolved. Species within the same family had a common ancestor, later than those of different species. It is thus another form of family tree.

Rail networks are sometimes constructed as trees. This allows trains to travel from end to end, but with outlying areas having fewer trains visit. The London Metropolitan Underground line is an example of this. It starts at Aldgate, in the city (its root) and through most of its length is a single chain of stations. However, at Harrow-on-the Hill it branches, with one branch continuing to Uxbridge. The other branch splits again at Moor Park, to Watford, and again at Chorleywood to Amersham and Chesham.



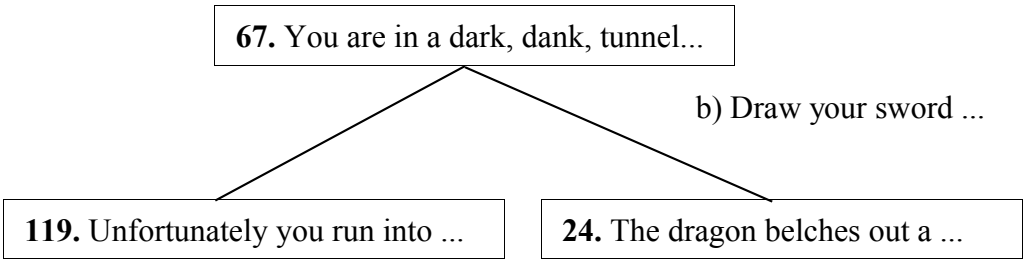
Each of the stations in between (not shown above) are also nodes in the tree. They are just nodes with a single edge in and a single edge out. A linked list is thus a really simple tree in which all nodes have only one link out.

A tree is a good organisation of the line, because it is intended to carry commuters into the city. On the whole people wish to travel between the leaves and the root, rather than between two leaves (Uxbridge and Amersham say). The trains will get fuller as they get closer to the City as more people are picked up at each station. Assuming all trains run between Aldgate and one of the other terminal stations, this also means trains in the busy centre (between Harrow-on-the-Hill and Aldgate) are very frequent where they are most needed, whereas trains in the suburbs are less frequent. Thus by organising the line in this way its efficiency is increased. However, if the situation changed and Uxbridge say turned into a city centre in its own right, this would be a poor organisation.

It is possible to buy adventure game books, where you play the main character and make decisions about what you do and so change the story. They consist of numbered paragraphs with questions at the end of each. Depending on what your answer is, you read a different numbered paragraph next. So, for example, a paragraph might be something like:

> ***67.*** *You are in a dark, dank tunnel. Icy water drips onto your arms and things scuttle over your feet in the dark. A faint light appears ahead. You round a corner to be faced by a fire-breathing dragon.*
>
> *Do you*
>
> *a) Turn around and run back down the corridor, screaming for help at the top of your voice.*
>
> *Go to Paragraph* ***119.***
>
> *b) Draw your sword and scream "Prepare to die".*
>
> *Go to Paragraph* ***24.***

Assuming it is not possible to get back to a paragraph you have already read, or end up at the same paragraph by two different routes, such a book is also a tree data structure. The very first paragraph, where the story starts is the root. Each of the questions represents an edge in the tree, and the nodes are the paragraphs themselves. The **leaves** of the tree are the final paragraphs (e.g. where you die). Reading the book once involves traversing the tree from the root, out to a **leaf**, and there are as many different story lines as there are leaves. The above paragraph is actually a tree fragment of the form:



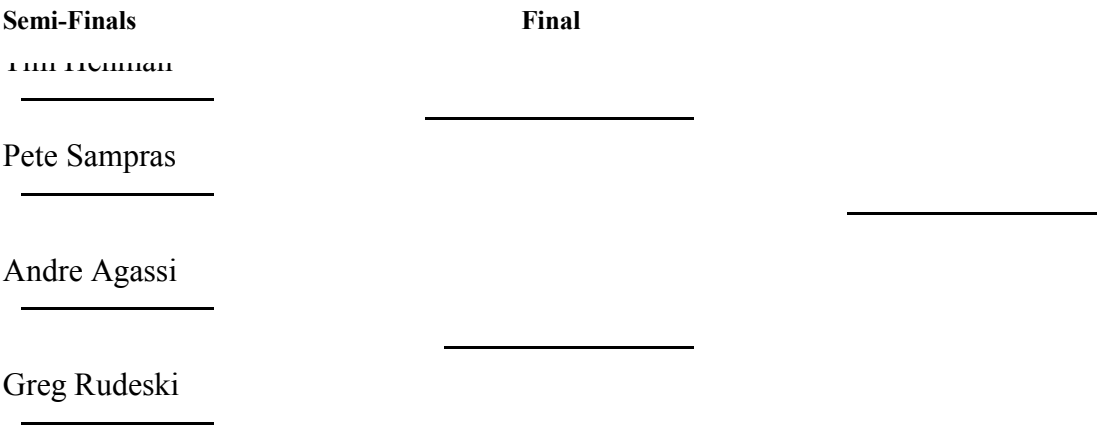
In this kind of tree, we not only place information in the nodes, but also on the edges – the question you are answering. We similarly could have labelled the edges of the family tree with an indication of whether the parent was the mother or father.

A similar idea to the treasure hunt is sometimes used in "fun" orienteering events. Instead of being given a map with all the points on at the start, competitors are just given a map with the first point to visit on. At that site you pick up a map with the next point. Usually the map will have three points on it, and be accompanied by a multiple-choice question. Each answer corresponds to a circle drawn on the map. Only the circle corresponding to the correct answer will hold the next question. This is just a variation on the treasure hunt, but with the pointers given by circles on a map, and their being several possible destinations from each point. It is thus a tree with one long trunk running from start to finish, but with one or more additional single edge branches from each node.

Trees are also the normal way of organising people at work. Schools have one Headmaster, several Deputy Heads, a whole series of Year Heads, and then the masses of teachers. The Army has a similar tree-like hierarchy of Majors, Colonels, Sergeants and so on. Most firms have a single Managing Director, several Department Managers, and so on. One of the first pieces of paper I was given when I first started working at Middlesex contained a tree-like diagram showing me the hierarchy used there (with my position firmly at one of the leaves). Over the next couple of weeks I was given 3 more copies of this by different people – it was obviously considered very important that I understood the tree that I was now a part. Why are Organisations organised as trees? One reason is that it makes communication and so management easier. Everyone has a single Boss above them in the tree to whom they report. No one has the problem of having two people ordering them to do different things. That person also reports upwards so that information can be filtered so that the more important people at the top are not overwhelmed with trivial information. Similarly, information can be passed down efficiently either to everyone, or just to those parts of the Organisation that it concerns. Each manager also has a small manageable number of people that they are directly responsible for. Recently many companies have "de-layered". They have scrapped layers of the tree – the middle management. They have

not scrapped the tree however. They have just made it shallower, so that information does not need to travel so far.

We saw earlier how an adjacency matrix is used to keep track of football results. It is a table indicating the scores between each pair of teams, or the date of the fixture if they have not played. Is this the structure chosen for all sporting competitions? It works best for leagues. Knockout competitions use a different structure for keeping track of fixtures: the tree. Every year as Wimbledon starts all the newspapers have pull-out specials, the centrepiece of which is a tree for you to fill in as the competition progresses. Each level of the tree is a round. The leaves are the starting competitors and the root will (eventually) name the winner. The same structure is used for the FA Cup, the World Snooker Championships and in fact every other knockout competition whether international or just an internal club competition. The tree structure works best because knockout competitions are in effect themselves trees. They are competitions the aim of which is to find the root of a tree.



Why is a tree structure the basis of so many competitions? It is because it is an efficient way of narrowing down a group to a single person by pair-wise games in that the number of actual games played is small. The tree structure ensures that the person who wins has proven themselves to be better than everyone else. Also the players in each round have played an equal number of games up to that point so there is fairness in the number of games played.

Trees are so useful that evolution has selected them in the construction of our bodies. For example, blood is carried through our bodies in arteries in a tree structure:

> "*One large artery, the aorta, leaves the heart and gives off smaller branch arteries to the various parts; these branch again and again to form finally very fine arteries known as arterioles*" (Armstrong, 1932)

Our veins, which carry the used blood back to the heart, use a similar structure. Our lungs also have a branching tree structure to get the maximum amount of oxygen into the blood. Why are trees so useful for this? The tree structure gives a way of covering a very large area. A tree with only a small number of branches has a very large number of leaves. It also reduces the distance to the leaf, so the blood or oxygen travels the smallest distance possible whilst reaching the maximum number of locations.

Graphs

The first railway lines constructed were single chains of towns and cities. To get between any two, you had to pass through all stations in between. As the network grew, this single chain could have been continued until all the towns and cities in the country formed a single chain, zig-zagging around the country. That is the equivalent of putting information into a list. It could alternatively have developed in to a tree like the Metropolitan line described earlier. This would be appropriate if most people wished to travel between their home city and London say (to some extent the British network was built like this by the London-centric planners). In general, however, large numbers of people wish to travel between other cities too. For example, the parents of a student at Manchester University, might live in Sheffield. It would be very inconvenient if they had to travel South towards London, changing in Peterborough, say, every time they wished to take some washing home, rather than by going directly over the Pennines. For reasons like this the rail network is not a tree but a **graph** data structure.

A tree connects nodes by edges so that there is a single root, and such that each node has a single edge in to it, though it may have any number leaving. With a tree, each node is on one and only one path from the root node. Graphs are like trees in that they connect nodes with edges. However, they are not restricted to the tree-like shape. Any node can be connected to any other node.

The edges of graphs can have directions associated with them, indicating which way along them you can travel. Consider a map showing the roads of a city. Most cities these days have one way systems, so if the roads were represented just by lines, it would not be much help in driving around. Instead a **directed graph** is used to represent the streets. You are told the direction along which you can travel for each edge. One way of doing this would be to use arrows to give the direction (with double-headed arrows to indicate a normal road). A-Z maps use a more subtle method but with the same intention of turning the graph into a directed rather than undirected one. Two-way streets have bold lines down both lines bounding the road. If it is a 1-way street, it has a bold line only down one side, with the side this is done indicating the direction of travel allowed.

Tree data structures are just graphs with the special "tree" shape. Suppose the author of the adventure game book liked happy endings. She had thought one up that she was especially proud of; so much so in fact that she wanted every one who played the book to have that ending. This was except of course for those who were such sad individuals as to have had been toasted by dragons, had their juices sucked out by a hoard of spiders or had their skin peeled off by a Pog Troggler. Thus all the stories in the book that did not end abruptly would have as the second last step: *Go to paragraph **42***, where paragraph 42 holds the happy ending:

> ***42:*** *You triumphantly arrive at the palace your quest completed. The hand of the Prince is yours. You march into the thrown room, to find it deserted. Blood is splashed around the walls. An uneasy feeling descends upon you. The whole palace seems devoid of life. Eventually you find a single serving boy, cowering in a flour bin in the pantry. George (as that is his name) explains that while you were away a whole army of Pog Trogglers had ravaged the city and slaughtered everyone. You thank your good luck that you were away on*

the quest at the time, as otherwise you too would have died a very unpleasant death. You ride off into the sunset with George and live happily ever after.

THE END

The book is no longer a tree as several branches converge on this node. However, it is still a graph. Similarly, if any of the choices take you back to a paragraph (node) you have visited before then whilst the book is still a graph, it is not a tree. Note that the book is also a directed graph. You are told how to get from one paragraph to the next, but once you get there, there are only instructions of how to continue on, not of how to get to the paragraph you came from. All the links have a direction through the story associated with them

More interesting variations on the simple list-based board games involve turning the list into a directed graph. Snakes and Ladders is an example. Instead of each square only being linked to the next, some squares contain the base of ladders, ie edges connecting squares to later squares. Other squares contain snakes – edges going back to earlier squares.

Random Access or Serial Access Data Structures

Lookup tables work on the principle that given some label or index you can go straight to an object. Music CDs have this property. I bought a CD "Californication" of the Red Hot Chili Peppers because of the brilliant track "Otherside". Once I know that "Otherside" is track 4, I can use that information to go straight to that track at any time. It takes no (noticable) difference in time between my making the selection and the track starting to play than if it had been track 15. I can go straight to the entry. That is a basic property of an array structure: arrays allow **random access** in the sense that you can get to any entry, given its position just as quickly as any other.

Contrast this with the situation had I bought a tape of "Californication" instead of a CD. Now to get to track 4 I would need to play or fast forward past the first three tracks keeping count in some way of which track I was up to. Accessing track 15 would now take much longer than accessing track 4. In fact the later the track appears on the album, the longer it would take to get to it. A tape allows only **serial access**: the tracks have to be accessed in turn starting from one end (or the current position). Linked list structures are like this, as you cannot go directly to any node but the next one.

Tapes are fine if we want to play an album from start to end (such as when in a car). CDs are much better if you frequently want to play single tracks. Indexes in books allow random access to an essentially serial structure (books are written to be read from start to finish). Text books usually have an index as random access is often needed: you dip into them looking for information on a specific thing. Novels never have an index as they are intended to be read from cover to cover. The last thing the author of a detective novel wants to give you is a way of looking up whodunnit before you get to the end.

Summary

By connecting elements by links rather than requiring them to be placed one after another, gives a data structure more flexibility.

A **linked list** is just a list where each element or **node** is connected by a link or **edge**. To move from one element to the next we follow the link.

Doubly-linked lists have links in both directions between connected nodes. This makes it possible to move from a node to both the next node in the list and to the previous node.

Circular linked lists have links between the first and last element of the list.

Lists are a single sequence of things. **Trees** have a branching structure, with each node linked to multiple next nodes. Each node has only one previous node, however. The first node in a tree is called the **root** of the tree.

Graphs are the most general linked node structures. Any node can be attached to any other. Linked lists of the various kinds and trees are just examples of graphs with restrictions on how the graph can be constructed. Graphs can be **directed** if the edges have directions associated or **undirected** if there are no directions.