**Feel free to criticise and add on comments**

3ai. A graph G is connected if for all x, y in nodes(G), there exists a path from x to y.

3aii. A cycle is a path of non-repeated arcs that starts and ends at the same node.

3aiii. A degree of a node is the number of incidents made with any arc within the graph with that node.

3bi. A graph has a Euler circuit iff all its nodes are of even degree.

Consider a graph G with at least 2 odd nodes (can’t have 1 as we can’t have an odd total degree). If we start from either an odd or even node, then we have to end at an(other) odd node as there are an odd number of arcs incident with each odd node. When passing through a node, we use two of its arcs each time, but this reduces to a situation when the node will only have 1 arc of which can be entered from to get to the node but we can’t find a path outward from the node without reusing of one of the older arcs.

3bii. A graph has a MDC if it is connected, apart from nodes of degree 0.

Consider a graph G. We can ignore the nodes of degree 0 as we only need to use each arc twice, not reach every node. For each arc in arcs(G), we add a parallel arc to each one, transforming the graph G to G’ of which each node now has double their previous degree. This means that each node in G’ is of even degree, so by (3bi) there exists a Euler circuit in G’, which translates to a MDC in G.

3ci.



Articulation point is top right corner.

3cii.

Assume that a connected graph G with n nodes does not have a node of a degree greater than 1, so that every node within G has just degree 1 (if they have 0, G is not connected). Then G will have a total degree of n. This means that G will only have n / 2 arcs. If n is >= 3, this contradicts a connected graph G with n nodes having at least (n - 1) arcs.

Hence, we must have a node of at least degree 2.

3ciii.

Assume that a graph G with n nodes is 2-connected, meaning that G is connected and has no articulation points. We know that as G is connected, it must have at least (n - 1) arcs. We know by (cii) that there exists a node of at least degree 2. If we remove it and the 2 or more arcs connected to it to form the graph G’, we know that G’ is still connected as G has no articulation points. As G’ is connected, with (n - 1) nodes, it must have at least (n - 2) arcs. Then G must have at least (n - 2) + 2 (or more arcs which we removed) = n arcs.

3di.

Say that we have a connected tree T, with no nodes of degree 1. Then this means that every node is of degree 2 or more (can’t have 0 or T is not connected).

As T is connected, this also means that there exists a path from any node to any other node. For a node x, this means that we can find a path to one of the other nodes in nodes(T), but as x has a degree of 2 or more, we know one of the nodes in nodes(T) \ {x} must connect back to x. This is a cycle. This is a contradiction. T, a tree, must be acyclic.

Hence, T must have a node of degree 1.

3dii.

Show by induction that a tree with n nodes has exactly (n - 1) arcs.

Base case: n = 1. Trivial.

Inductive step: n = k is true, that a tree with k nodes has (k - 1) arcs.

Consider a tree T with (k + 1) nodes. For T to be connected, we know that it must have at least k arcs. By (3di), we know that T must have a node of degree 1. If we remove that (and the arc adjoined to it) to make the graph T’, then we know that T’ is acyclic (removal does not introduce cycles) and T’ is still connected as the removed node does not affect the paths between other nodes as it was only a node of degree 1 that wouldn’t be contained in these paths.

As T’ is a tree with k nodes, it must have exactly (k - 1) arcs. Then T has (k - 1) + 1 (the arc we removed) = k arcs.

4ai.

Weigh 1 and 2 together. If they’re the same weight, then we are done with the groupings being [1,2] and [3,4].

If not, weigh 1 and 3 together. If they’re the same weight, then we know the groupings are [1,3] and [2,4]. Else, then [1,4] and [2,3].

Draw above as a decision tree.

4aii.

Our worst case is that we require 2 weightings. With just 1 weighting, we are not able to differentiate each coin into its respective group, so at least 2 is needed.

4aiii.

Consider 6 coins. Pick any one of these coins, such as coin 1 (without loss of generality). We can define an equivalence relation ~ such that 1 ~ 2 if coin 1 has the same weight as coin 2. Hence, there are two equivalence classes we can have, those that relate to coin 1, and those that don’t. We won’t be able to tell if one class is the genuine coins or if it is the fake, but we know that both sets have the same amount of elements.

We can then compare the first 4 coins of the other coins with the one we chosen, and place them into their respective equivalence classes of either relating to 1 or not. One will be of size 2 and the other of size 3. We don’t need to compare the last coin, as we know that both sets must be of the same size.

Hence, the number of comparisons is 4.

4bi. W(1) = 0, W(n) = (n - 1) + W(n - 1).

The worst case for Quicksort is when the list is already sorted.

For each element, Quicksort will find (and confirm) the position of an element e by comparing it to each other element. This is (n - 1) comparisons for a list of size n. We will need to repeat the process with the remaining (n - 1) elements which are already sorted, but we do not need to consider e as we know that it is in the correct position. This boils down to doing quicksort on a sorted list of size (n - 1).

4bii. This is a linear arithmetic series, by expansion.

W(n) = (n - 1) + (n - 2) + (n - 3) + (n - 4) + … + 0 = n (n - 1) / 2.

4biii. In the average case, our recurrence relation is

We still have the (n - 1) comparisons that we have to make with each element to confirm a pivot elements position in the list, but we now consider the average case that we perform a quicksort on the sublists each side of the pivot element. The position of the pivot element will determine the size of each sublist, but we can apply a probability of 1 / n to generalise this.

4biv.