1

* 1. There are two possible mappings for 1. Once 1 is fixed, there are 2 possible mappings for 2. Fixing 2 also fixes 3, 4 and 5. There are now two possible mappings for 6. Fixing 6[ fixes all remaining nodes. So we have 2\*2\*2 = 8 isomorphisms, including the identity.
  2. 1. T is a spanning tree for G iff it’s a connected acyclic subgraph of G such that all nodes(G) are included.
     2. To show: Adding a to T to create T’ forms a unique cycle.
        1. To show: Adding a to T creates a cycle.
           + T is a tree, by definition, there is a unique path between every pair of nodes.
           + In particular, there must be a unique path between x and y.
           + Now add a(x,y) to T to form T’.
           + There are now 2 paths from x to y.
           + Path 1: (x,y) using a
           + Path 2: (x,..n1, n2, ...y) where n1 and n2 are arbitrary intermediate nodes.
           + Therefore we have a cycle (concatenating P1 and P2 gives a path from x to x).
        2. To show: Adding a to T creates no more than one cycle.
           + Assume for a contradiction that adding a to T creates two cycles or more.
           + Let these 2 cycles be :

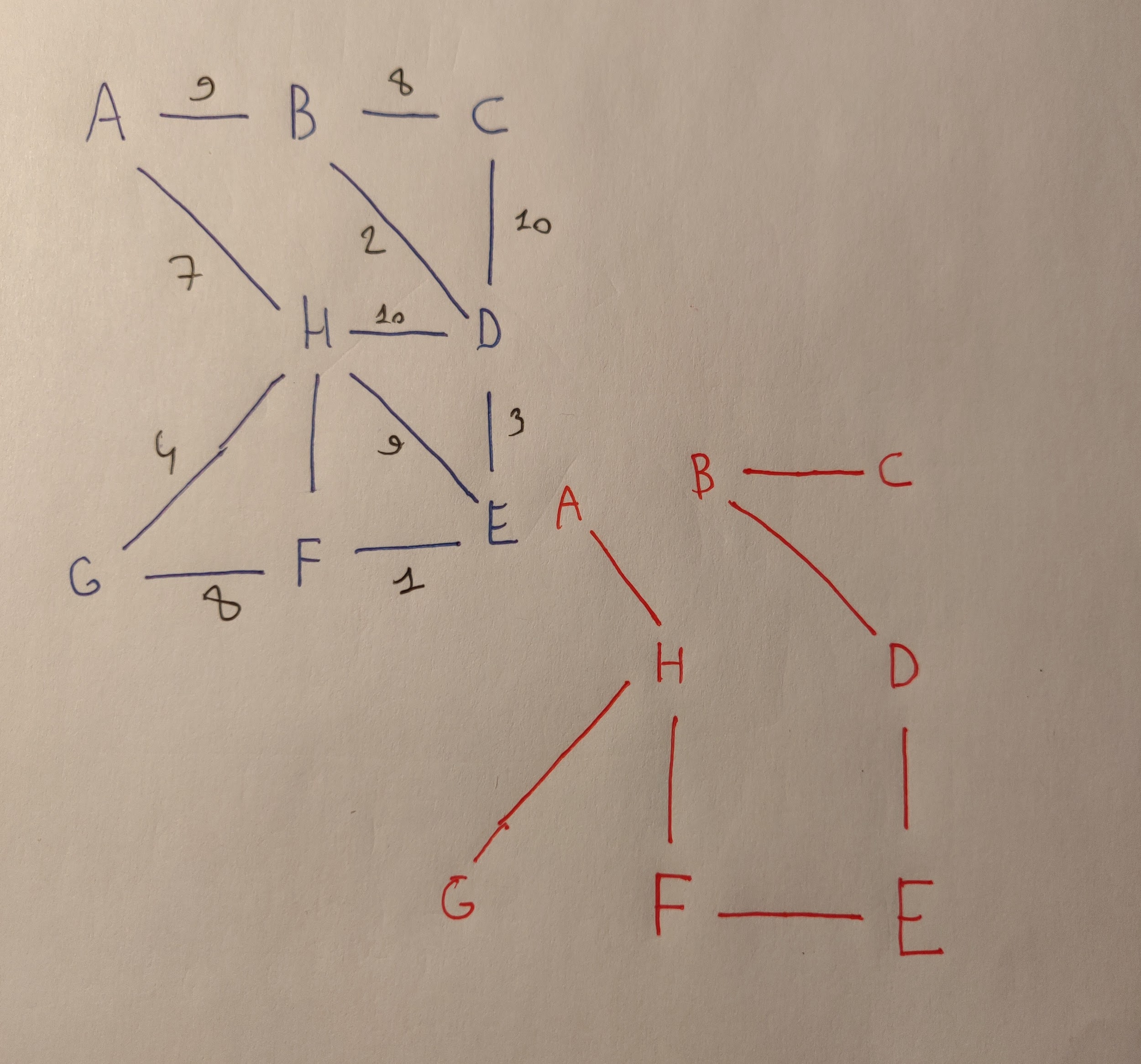
C1 = (x,y … x1, x2, … x)

C2 = (x,y… y1, y2, … x)

* + - * + Now remove a from T’, this must form T.
        + We still have 2 paths from y to x:

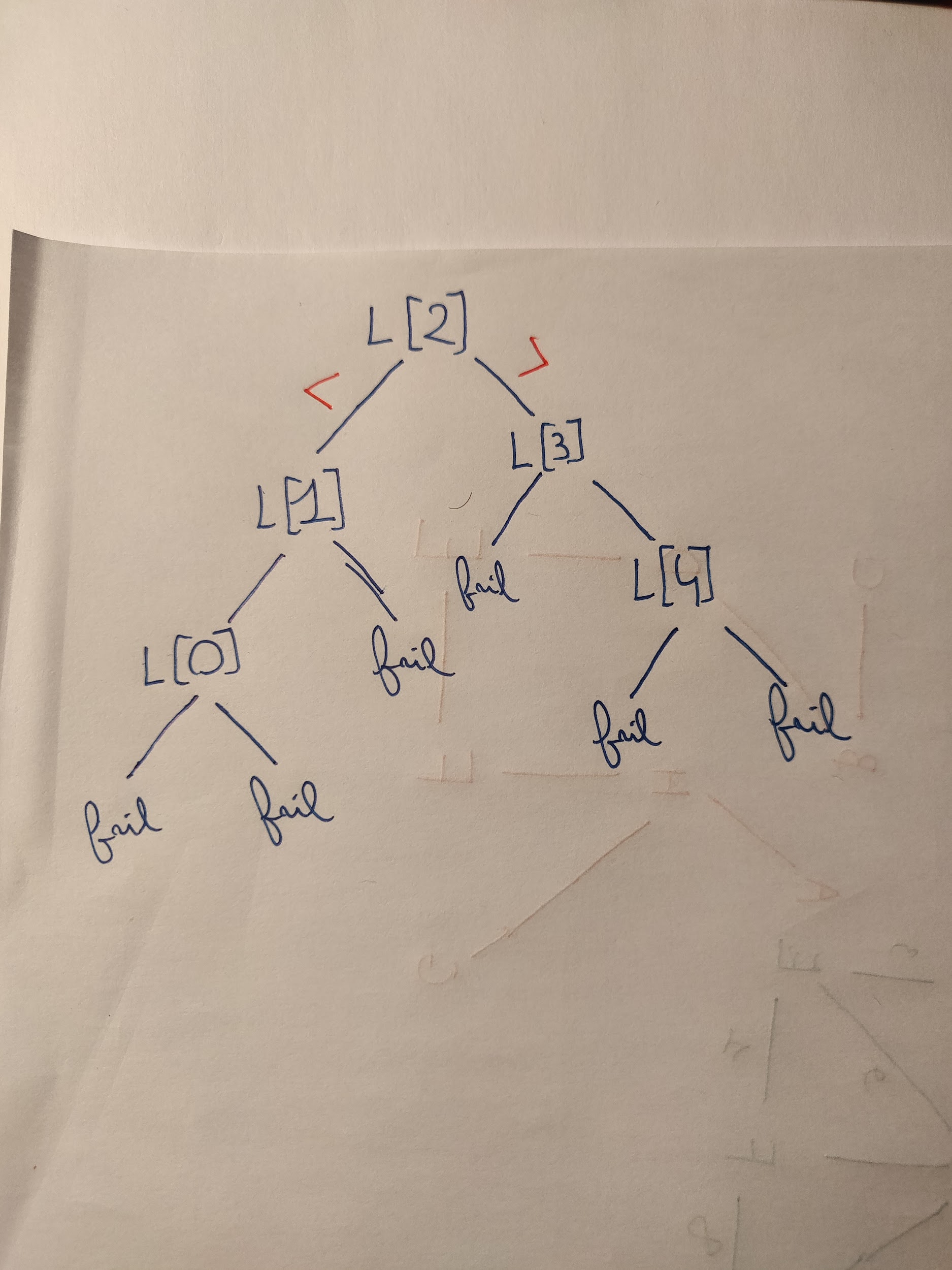
P1 = (y … x1, x2, … x)

P2 = (y… y1, y2, … x)

* + - * + Or T is a tree, by definition of tree, a path between a pair of node must be unique -> Contradiction.
    1. Order: FE -> BD -> DE -> GH -> HF -> AH -> BC 
    2. G has 8 nodes so a spanning tree of G must have 7 arcs. The 7 lowest weights in G are 1, 2, 3, 4, 5, 7, 8. There are two arcs with weight, 8 BC and GF, we’re currently using BC. If we were to swap to GF then we would have a cycle and T wouldn’t be a spanning tree anymore. Therefore, G has a unique MST.
  1. T has a unique MST. Assume for a contradiction that T isn’t an MST after applying the squaring weight function. Suppose that (G, W^2) admits an MST T’. If T’ isn’t the same as T then they must differ on at least one edge. The new edge in T’ must have a lower weight than the edge in T. That implies that there exist some weights W1 and W2 such that W1 < W2 and W1^2 >= W2^2. This is not possible -> Contradiction.

I was thinking instead of doing proof by contradiction, we can reason by

saying since there is a partial ordering of the arcs, squaring still preserves the partial ordering since the weights are natural numbers...?

1. 1. 1. 
      2. comparisons on average

**The question says “on the basis the element being searched for is in the list” - it should be (1\*1 + 4\*2)/5 = 9/5 = 1.8** (no extra comparison for what should be L[1] and L[4] as the element is guaranteed in the list.

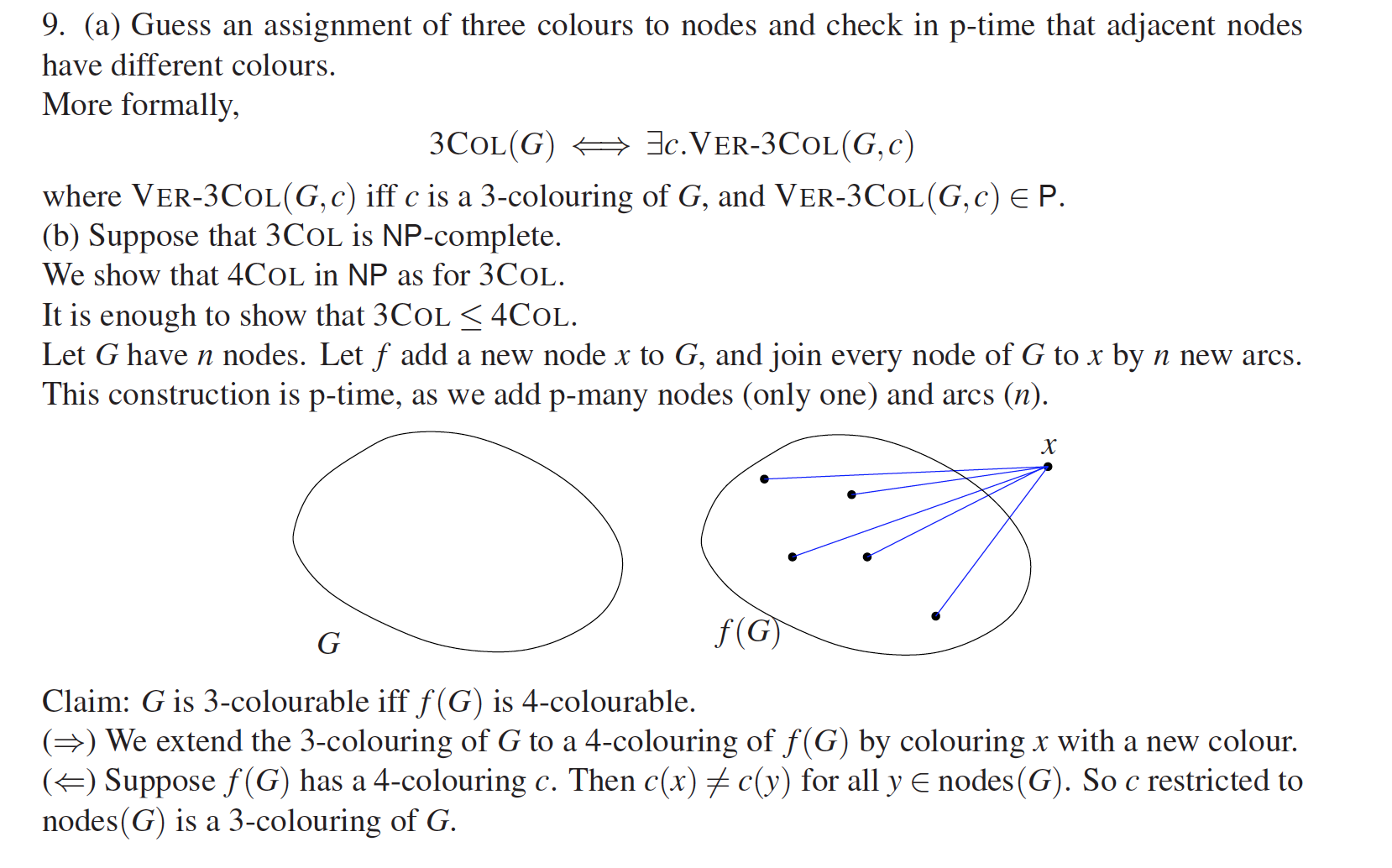
* + 1. W(1) = 0

W(n) = n - 1 + W(ceiling(n/2)) + W(floor(n/2))

* + 1. S(1)=0  
       S(n)=n/2 + 2S(n/2)My thinking was that MergeSort will correct the order of swapped elements on the level n=2 using 1 comparison and will then work with sorted lists only, which should require comparing all elements in the first merged list to the first element in the second merged list. But this might as well be wrong. (Adam's version)
  1. 1. It means that we can define a function f(x) that runs in polynomial time such that the inputs of the problem D are converted to equivalent inputs for problem D’. D(x) iff D’(f(x)).
     2. A graph is k-colorable if we can assign to each node a color, using at most k colors, without two adjacent nodes having the same color.
     3. 3Col is NP since we can check if an assignment of colors on a graph satisfies 3Col in P-time. The algorithm is trivial: for each arc of G, check that the endpoints have different colors.(and total number of colours does not exceed 3) This is clearly polynomially bounded on the input size since for a graph with n nodes we will have at most n(n-1)/2 arcs (this upper bound case arises with a complete graph).
     4. Since 3Col is NP-Complete, it is, in particular, NP-Hard, that means that for all problems D in NP, D <= 3Col holds. We also know for sure that 4Col is in NP since we can simply reuse the P-time algorithm described earlier for verification. That must mean that 4Col <= 3Col. On the other hand, we can show trivially that 3Col <= 4Col since we can reduce 3Col to 4Col simply by adding a new node (of the 4th color) and connecting it to every existing node in G. We then have 3Col ~ 4Col. Therefore, 4Col is NP-Complete.

**OFFICIAL ANSWER**

(a) is c.(iii) and (b) is c.(iv)



Answer from Graphs and Algorithms Exercises 9, Q9