1) Sorting Large Numbers

(a) **Show that Counting-Sort is not an option by analyzing the runtime of Counting-Sort on this data**
For the sake of contradiction let us assume that for range of n integers in $n^4 - 1$ that with Counting Sort it can be sorted in a runtime of O(n).
Recall from the class notes and the CLRS textbook that the overall run-time of Counting Sort is $O(k + n)$ (In the textbook it writes down in Θ runtime but from theorem 3.1's iff proof if Θ holds then big O must too) where k is the maximum value of integer and n is the number of integers.
Then let k = $n^4 - 1$ then $O(n^4 - 1 + n) = O(n^4)$ since from a mathematical standpoint $n^4 > n$. Hence we have a contradiction.
Therefore, Counting Sort is not an option on this data.

(b) **Show that unmodified Radix-Sort is not an option by analyzing the runtime of Counting-Sort on this data.**
From Lemma 8.3 of the CLRS book for n d-digit numbers, k possible values, Radix Sort can sort for $\Theta(d(n + k))$.
For a base 2, k = 1, and the value of d = $\lg n^4 = 4\lg n$. Then using Lemma 8.3 then it follows that

$$\mathrm{O}(d(n+k)) = \mathrm{O}(4\lg(n)(n+1)) = \mathrm{O}(4n\lg n + 4\lg(n)) = \mathrm{O}(n\lg n)$$

Therefore, an unmodified Radix-Sort is not an option.

(c) **CLRS states that "we have some flexibility in how to break each key into digits", and prove a relevant Lemma 8.4.**
From Lemma 8.4 of CLRS

For a n b-bit numbers and any $r \in \mathbb{Z}^+$ s.t. $r \leq b$, Radix Sorts these numbers in
$$\Theta((\frac{b}{r}(n + 2^r))$$

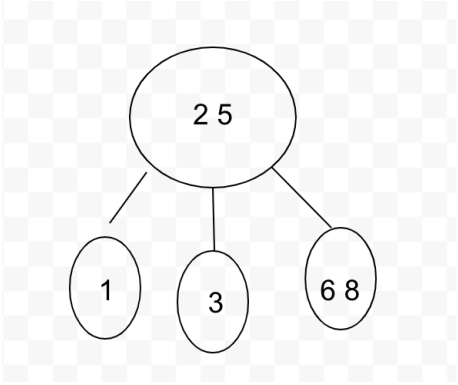
The numbers we are dealing with has $4\log(n)$ bits in other words then $b = 4\log(n)$.
Suppose that we let r = $\log(n)$ then the range of the number is 0 to $k = n - 1$. Then Radix Sosrt sorts these numbers

$$\Theta(\frac{4\log(n)}{\log(n)}(n + 2^{\log(n)}) = \Theta(4(n + n) = \Theta(n)$$

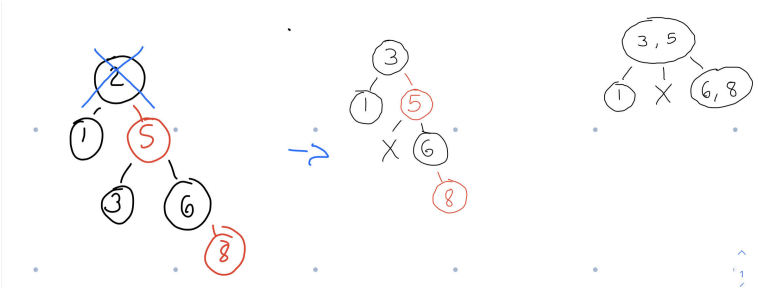
Recall that for a stable Counting Sort that the sorting time is also $\Theta(n)$.
Hence proved.

2) Red-Black Trees and (2,4) Tree Deletion

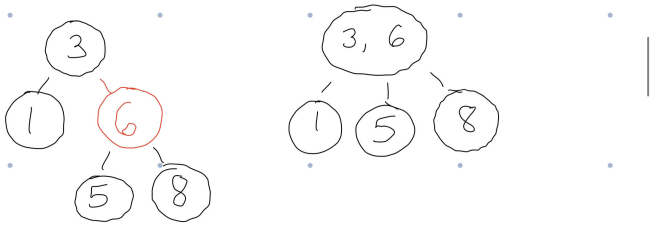
(a) **Draw the 2-4 tree that corresponds to the RBT shown below**



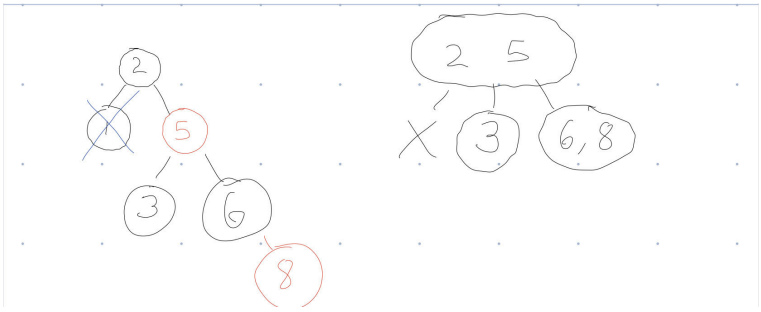
(b) **Delete key 2 from the red/black tree shown above, and show the deletion in the (2,4) representation**
Delete 2 from the Red and Black Tree:



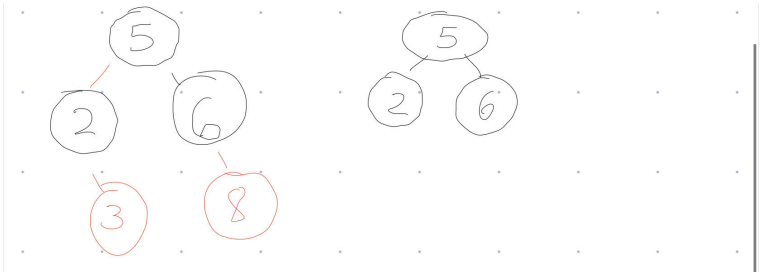
Then we have a case where x is black and has a red child hence we perform a rotation and then restructure the 2-4 tree, therefore, from CRLS case 4 and G and T case 1.



(c) **Delete key 1 from the initial red/black tree shown above**
Delete 1 from the original Red Black Tree
Then we have Case 3 from G and T.



Since x is red then adjust so another case applies then after adjusting it we get a case where black sibling and children are both black: re-color, which is the G and T case 2



3) Red-Black Tree Height

(a) **What is the largest possible number of internal nodes (those with keys) in a red-black tree with black height k? What is the height of the corresponding 2-4 tree?**.
The largest possible nodes with a red-black tree with black height k is $2^{2k} - 1 = 4^k - 1$ since the RB-Tree is full with layers of red and black nodes alternating.
The height of the corresponding 2-4 Tree is k - 1 since the 2 - 4 trees are defined by its black nodes.

(b) **What is the smallest possible number of internal nodes (those with keys) in a red-black tree with black height k? What is the height of the corresponding 2-4 tree?**
The smallest amount of internal nodes in a red black tree with black height k is $2^k - 1$ from the CLRS book's Lemma 13.1 proof, it was shown that the subtree rooted at any node x contains at least $2^k - 1$. The height of the corresponding 2-4 tree is k - 1.

4) Red Nodes in Red-Black Tree

Consider a red-black tree formed by inserting n nodes with RB-Insert. Prove that if $n > 1$, the tree has at least one red node.
Suppose that we insert n nodes such that n is a value $n > 1$. Then any case of insertion that deals with n = 1 will be ignored.
When the first node is inserted it will be black since the root of a Red-Black trees are black from the red and black properties.
Next, line 16 of the pseudocodes given in the CLRS book, the second node inserted will be red. Let us break this into different cases:

- Case 1: Most cases for 1 in the CLRS book is when n = 1 so we ignore it
- Case 2: The inserted node is left as the color red then there is at least one red node
- Case 3: In line 12-13 of the pseudocodes from the CLRS book for RB-INSERT-FIXUP(T,z) the parent is colored black and the leaves one of the nodes the color red and then exits the while loop. This implies that there is at least one red color.

5) O(n) sort of variable length integers

Suppose we have a way of representing positive integers with variable numbers of digits, for example "3", "61" and "317" may be included. Assume that there are no leading 0s, for example, "317" not "00317". You are given an array of positive integers under this representation where the total number of digits over all the integers in the array is n. Show how to sort the array in O(n) time.

To begin with sort the integers by their length using bucket sort since they make a bucket of the possible digits. From the CRLS book we know that bucket sort takes a runtime of O(n). Next we sort them using the radix sort and from Lemma 8.3 radix sort takes $O(d(n+k))$ for a constant d and k is the possible values. From the given all the integers in the array is n then, $O(d(n+n)) = O(2dn) = O(n)$.
Finally, we should concantate the sorted list.