union_and_intersection_explanation

February 21, 2022

0.1 Union and Intersection explanation

0.1.1 Union

Design Consideration The main objective for the union function, is to output a new linked list containin all elements present in two inputed linked lists. To achieve this I introduce a set type data structure `intermediate_set`. This set helps to hold all values in both lists for itereating while appending them to the output linked list `union_list`. I choose a set() datastructure to also ensure no element is repeated.

Time and Space complexity analysis For the time analysis we have two for loops which are dependent on the largest linked list $N1$ and set size $N2$. A reasonable assumption is that $N2 <= N1$. Hence the time complexity can be expressed as $O(N1) + O(N2)$ Which indicated a linear complexity dependent on the sie of largest input linked list. i.e. $O(n)$.

Space complexity can be analysed to have used a set of size $N2$ and output linked list of same size (disegarding input lists sizes) hence expressed as $O(N2) + O(N2)$. Therefore space complexity is also linear i.e $O(n)$.

0.1.2 Intersection

Design Considerations This function is supposed to return a linked list containing elements present in both the first and second linked list that serve as inputs. Since there shouldn't be repetition I choose to use sets as the intermediate data structures. I first create a set to store the unique elements in the larger sized input. Then, I iterate through the smaller sized input and check if its elements are present in the newly created set. If present I add them to the `intermediate_set` data structure.

Time and Space complexity The time complexity as in union ends up being linear since $O(N1) + O(N1) + O(N2) = O(n)$. Where $N1$ represents size of larger input list and $N2$ represents size of `intermediate_set`. Space complexity is also linear cinsisting of first set, intermediate set and returned output list i.e. $O(N1) + O(N1) + O(N2) = O(N1)$ which is essentially $O(N)$.