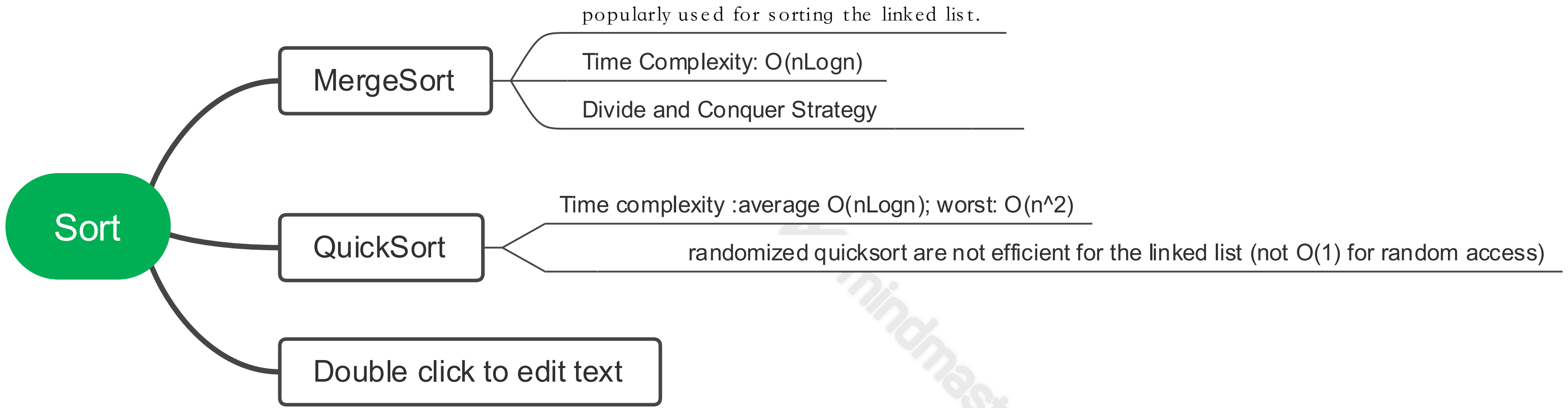
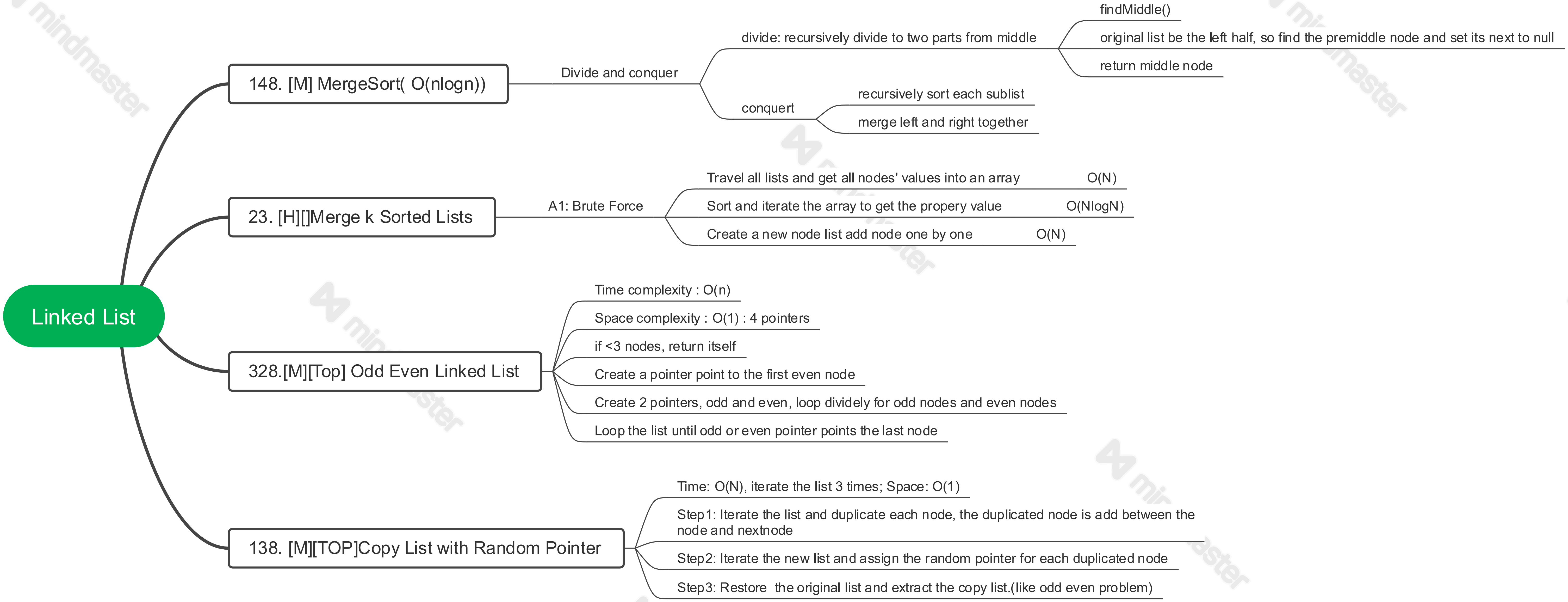


Sort

- MergeSort
  - popularly used for sorting the linked list.
  - Time Complexity: O(nLogn)
  - Divide and Conquer Strategy
- QuickSort
  - Time complexity :average O(nLogn); worst: O(n^2)
  - randomized quicksort are not efficient for the linked list (not O(1) for random access)
- Double click to edit text



- Linked List
  - 148. [M] MergeSort(O(nlogn))
    - Divide and conquer
      - divide: recursively divide to two parts from middle
        - findMiddle()
        - original list be the left half, so find the premiddle node and set its next to null
        - return middle node
      - conquert
        - recursively sort each sublist
        - merge left and right together
  - 23. [H][]Merge k Sorted Lists
    - A1: Brute Force
      - Travel all lists and get all nodes' values into an array O(N)
      - Sort and iterate the array to get the propery value O(NlogN)
      - Create a new node list add node one by one O(N)
  - 328.[M][Top] Odd Even Linked List
    - Time complexity : O(n)
    - Space complexity : O(1) : 4 pointers
    - if <3 nodes, return itself
    - Create a pointer point to the first even node
    - Create 2 pointers, odd and even, loop dividely for odd nodes and even nodes
    - Loop the list until odd or even pointer points the last node
  - 138. [M][TOP]Copy List with Random Pointer
    - Time: O(N), iterate the list 3 times; Space: O(1)
    - Step1: Iterate the list and duplicate each node, the duplicated node is add between the node and nextnode
    - Step2: Iterate the new list and assign the random pointer for each duplicated node
    - Step3: Restore the original list and extract the copy list.(like odd even problem)