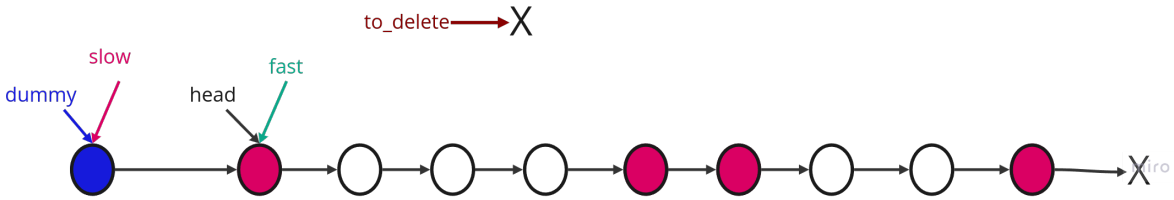
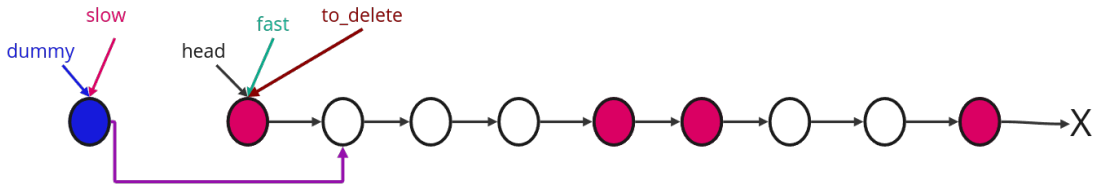
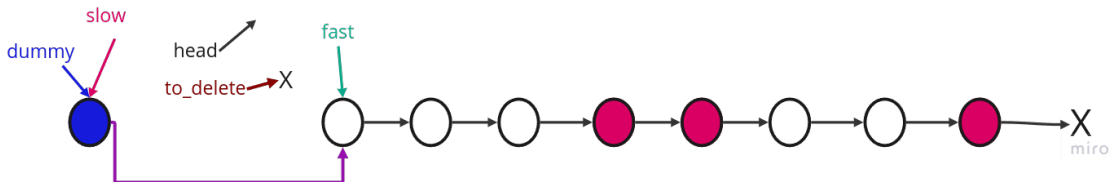
No memory leaks

```
/*
   Sorting+Binary search (No memory leaks)
   Use dummy node to make a unified pattern to all nodes

   Time complexity: O(nlogn)
   Space complexity: O(logn)
*/
typedef std::vector<int> vi;
class Solution {
   public:
      ListNode* modifiedList(vi& nums, ListNode* head){
         std::sort(nums.begin(),nums.end());

         ListNode* dummy=new ListNode(0,head);
         ListNode* fast=head;
         ListNode* slow=dummy;
         ListNode* to_delete=nullptr;

         while(fast){
            if(std::binary_search(nums.begin(),nums.end(),fast->val)){
               slow->next=fast->next;
               to_delete=fast;
            }
            else slow=fast;

            fast=fast->next;
            delete to_delete;
            to_delete=nullptr;
         }

         head=fast=slow=to_delete=nullptr;

         return dummy->next;
      }
};
```

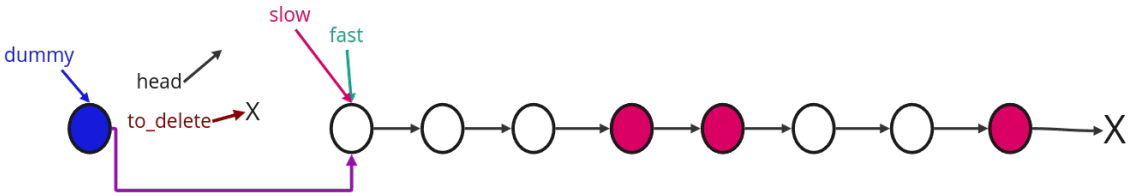


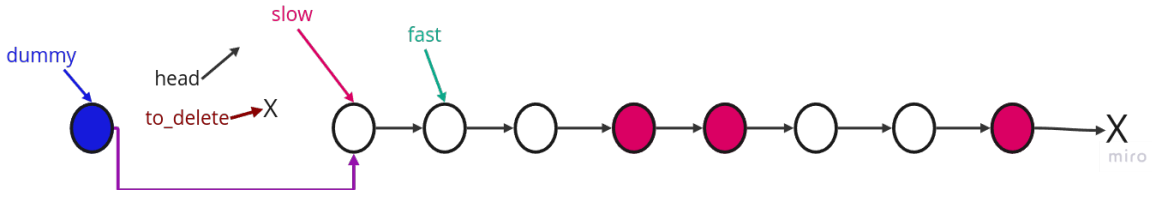
fast points to the node to delete:



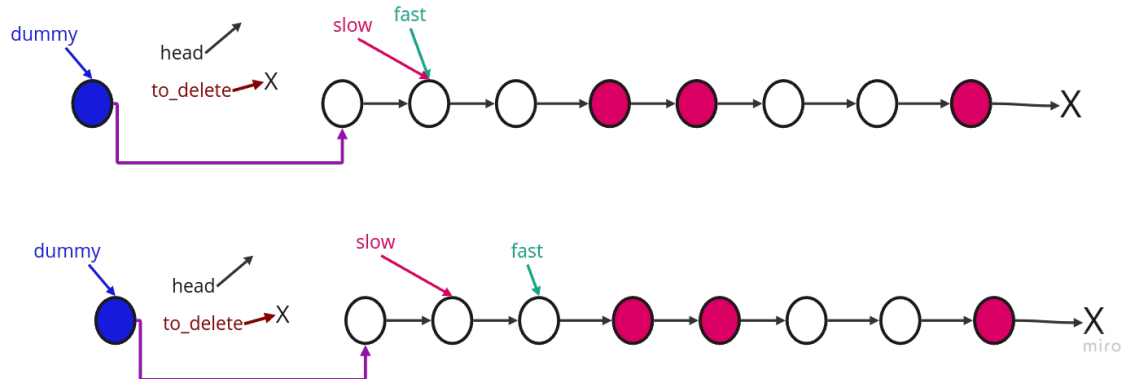


fast does not point to the node to delete:

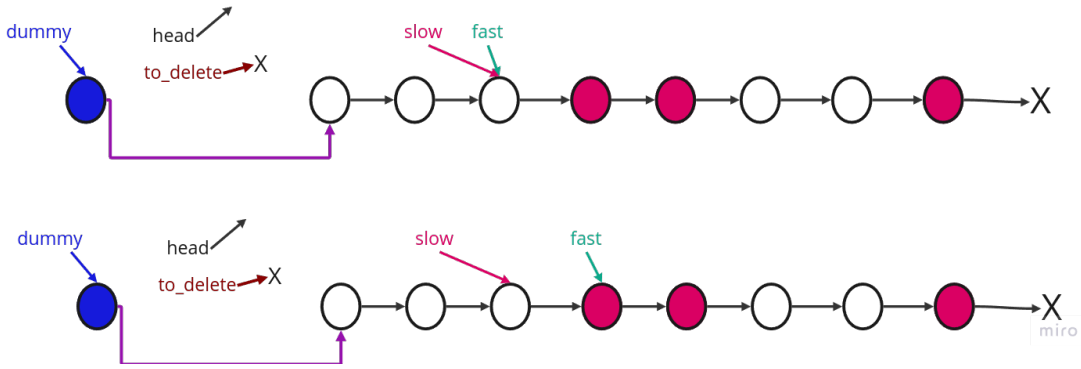




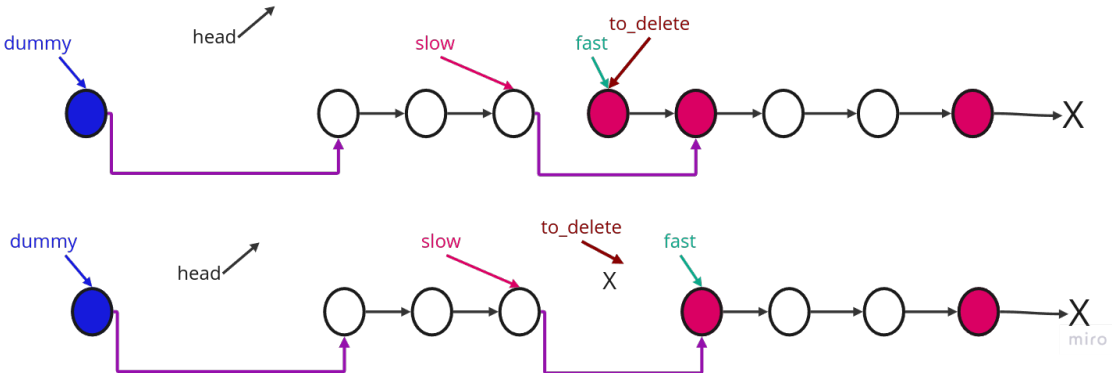
fast does not point to the node to delete:



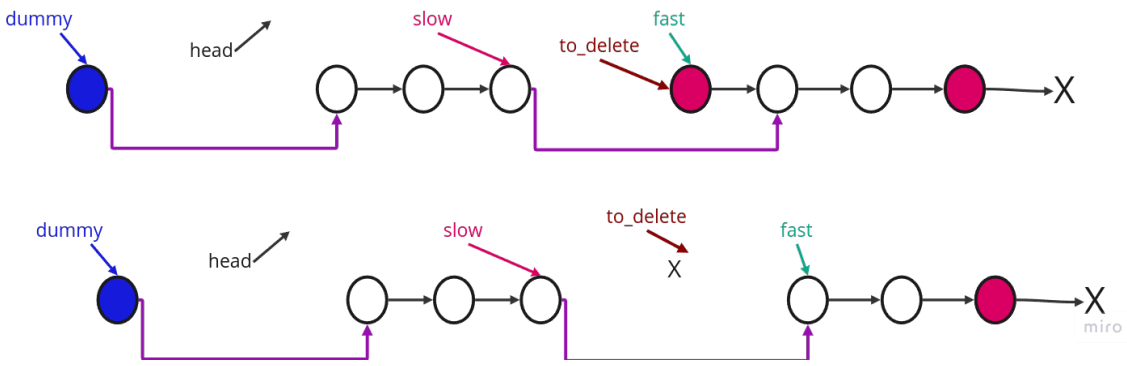
fast does not point to the node to delete:



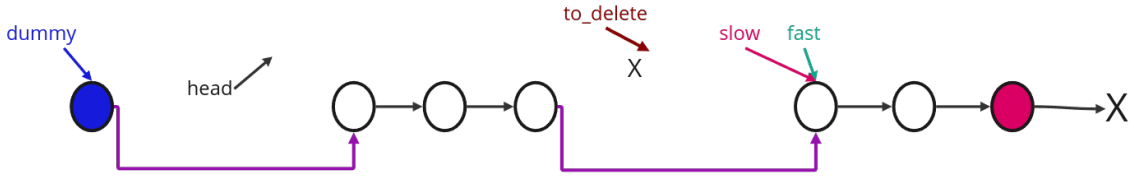
fast points to the node to delete:

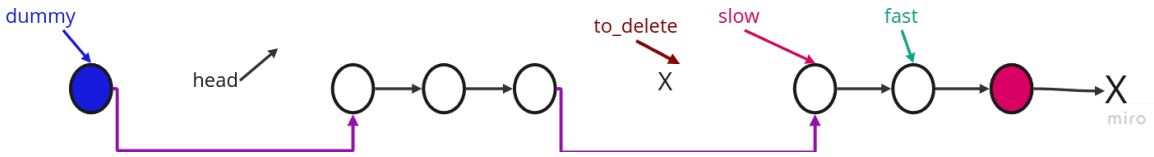


fast points to the node to delete:

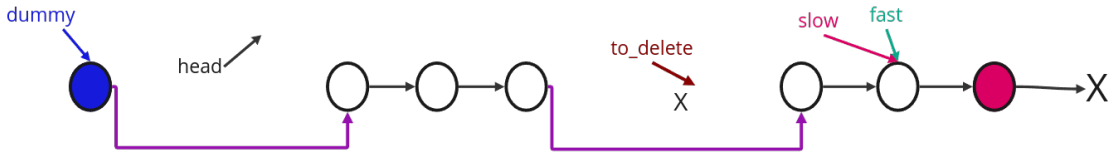


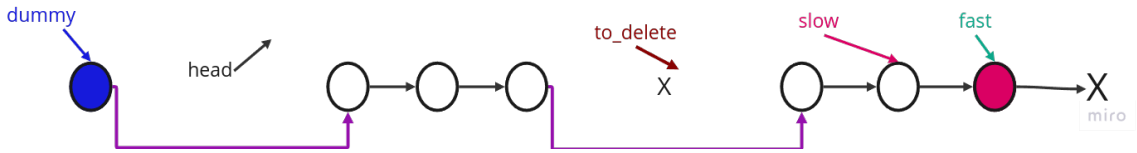
fast does not point to the node to delete:



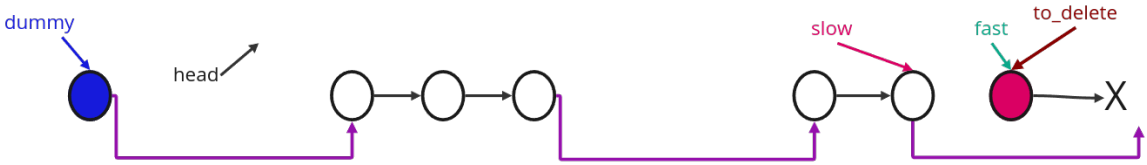


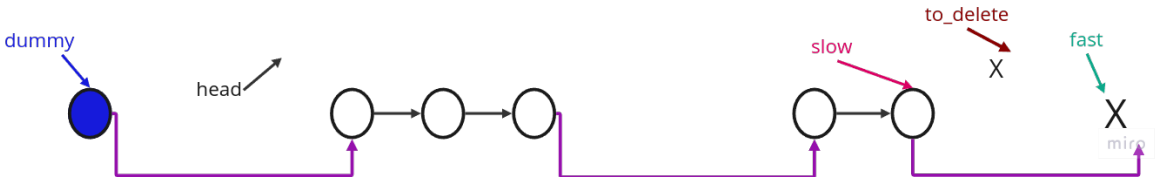
fast does not point to the node to delete:



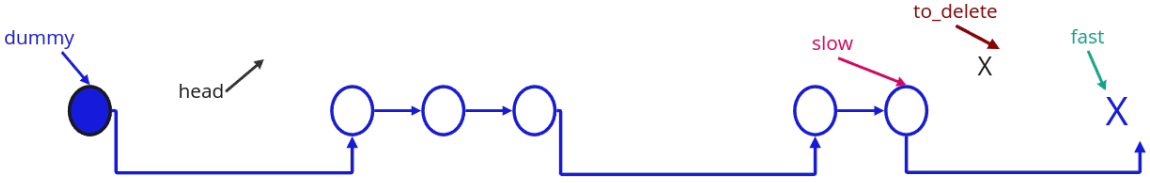


fast points to the node to delete:





At end the we get:



points all used pointers to null