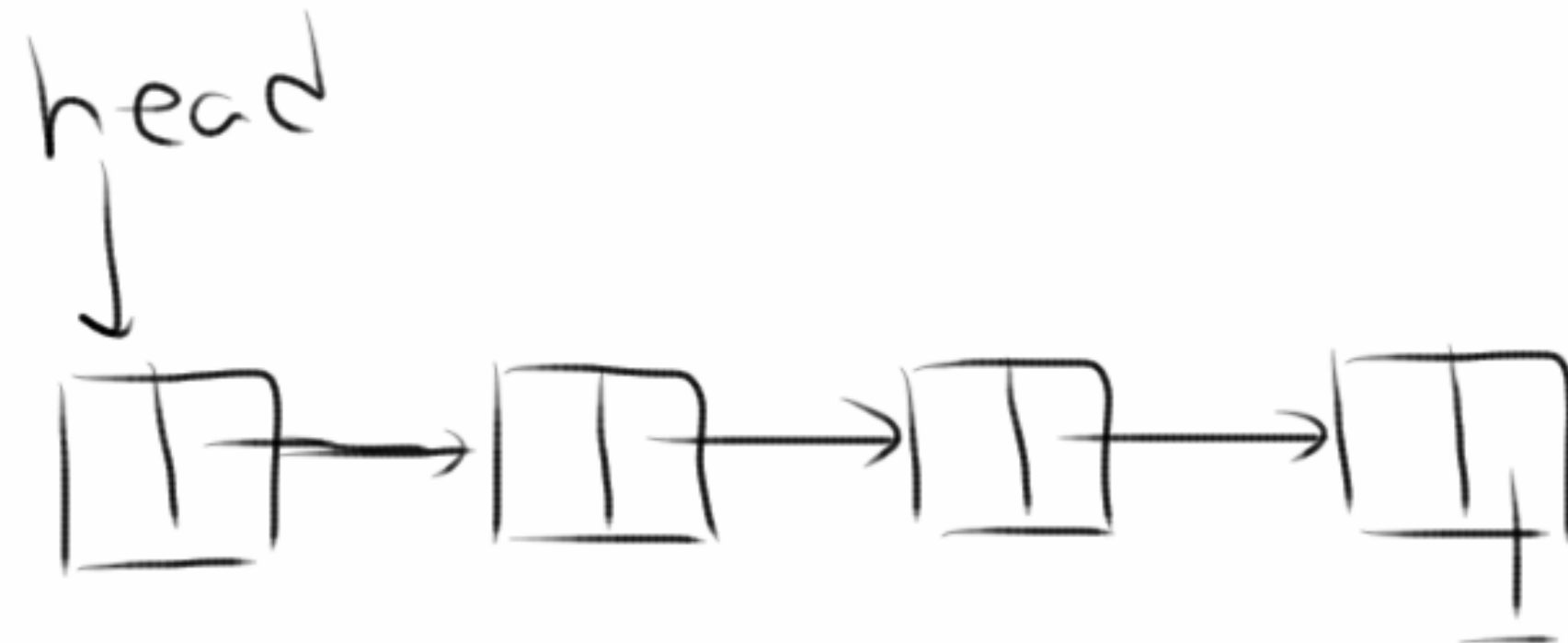


Linked-List

Basic Idea



- node -- 结点

data structure

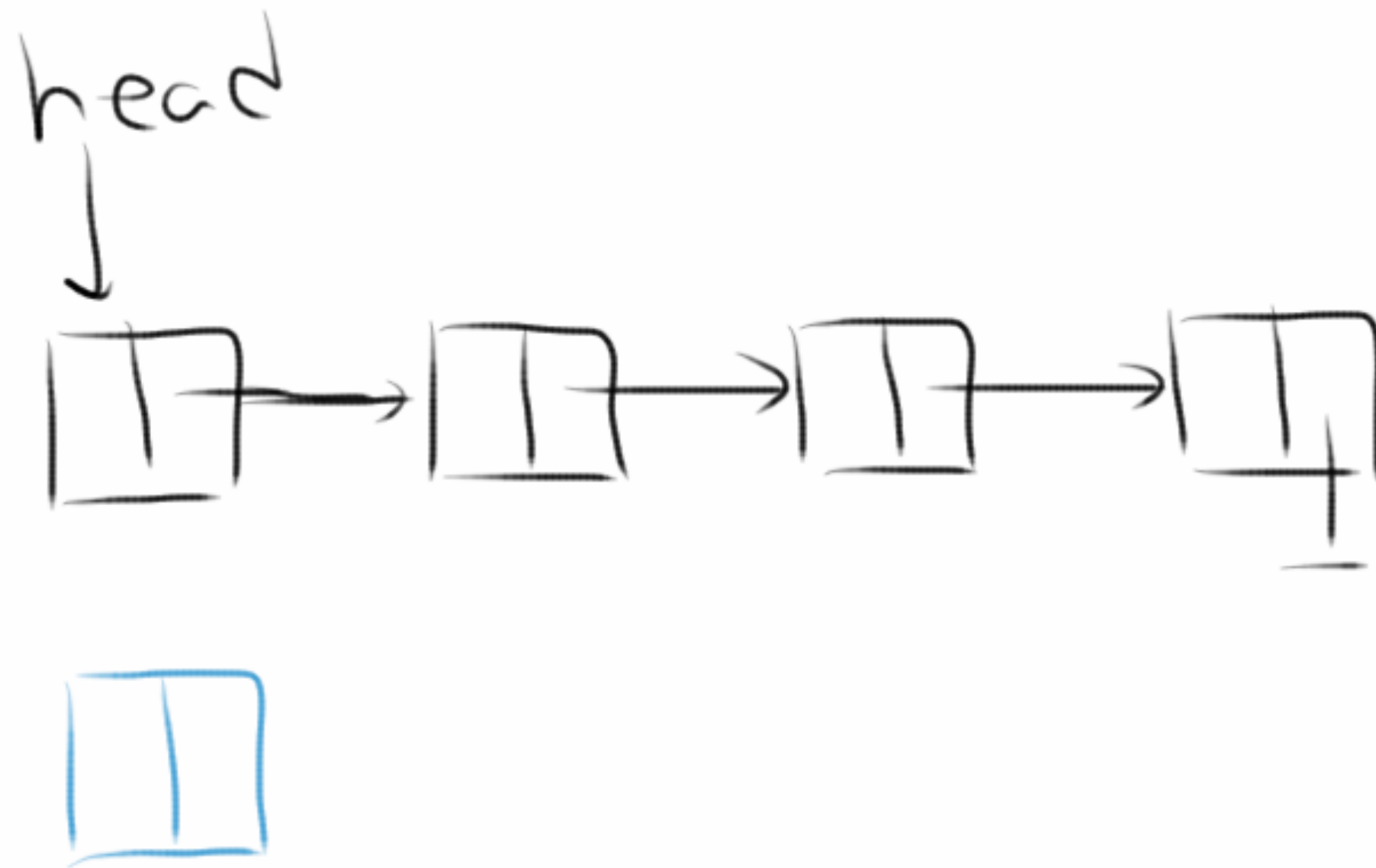
```
typedef struct _node {  
    int value;  
    struct _node* next;  
} Node;
```

basic operation

- insert head
- traversal/search
- append tail
- remove
- clear all

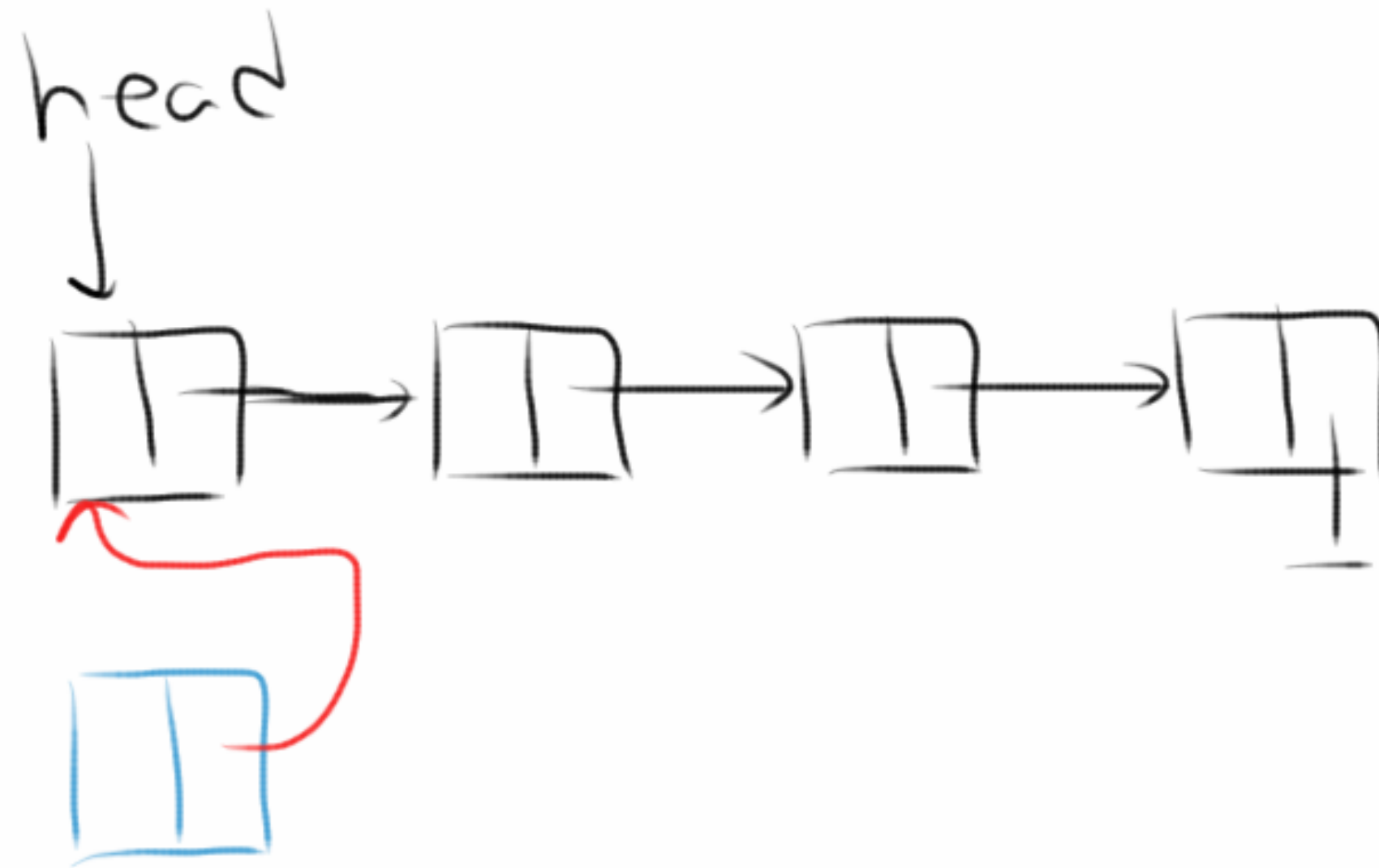
insert head 1

- Create a new node.



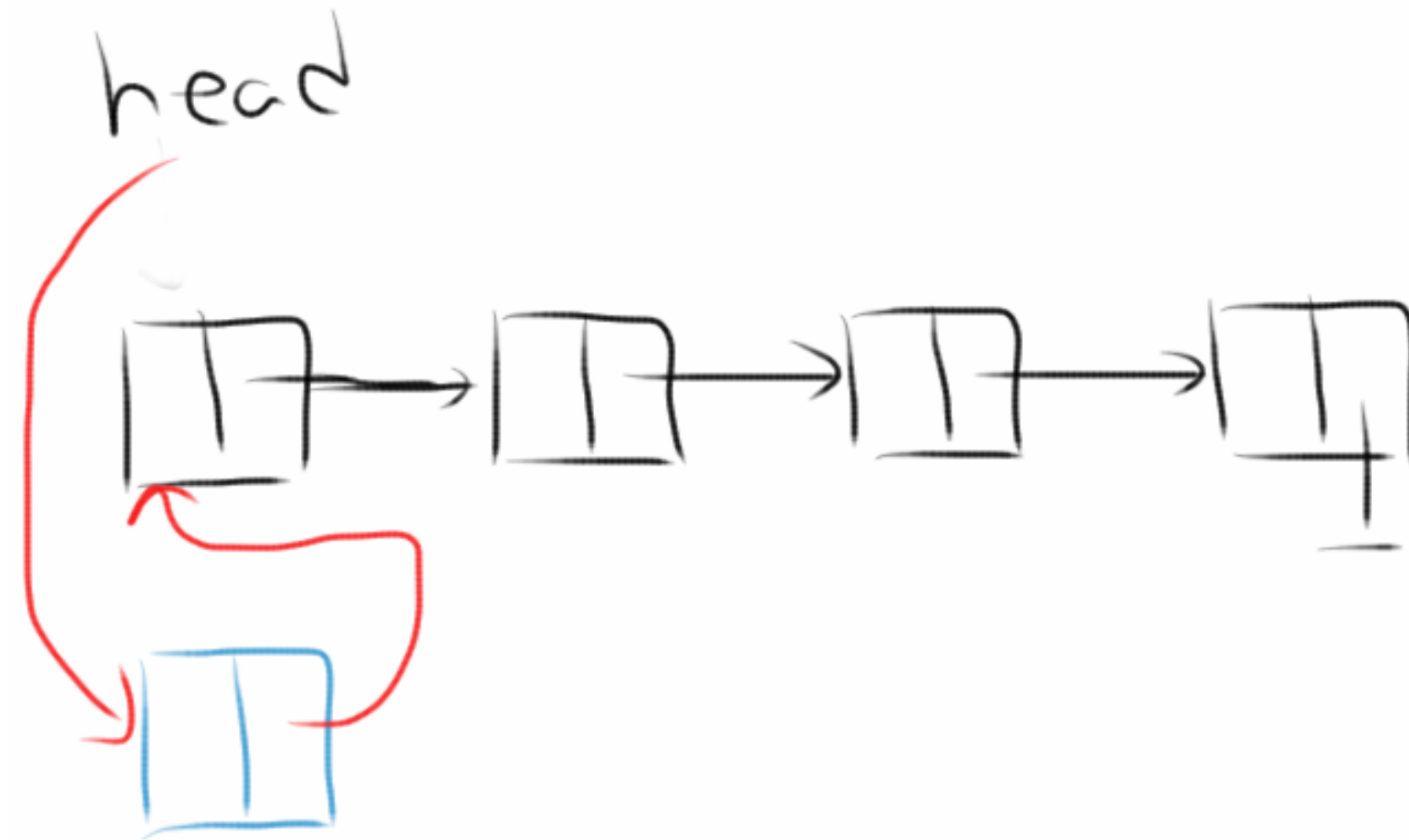
insert head 11

- Make the new next point to the head.



insert head III

- Point the head to the new node.



insert head IV

```
Node* n =  
(Node*)malloc(sizeof(Node));  
n->value = i;  
n->next = head;  
head = n;
```

- Any boundary condition?

make it a func?

- `void add_head(Node* head, int i);`
- `void add_head(Node** pHead, int i);`
- `Node* add_head(Node* head, int i);`
- ??

struct List

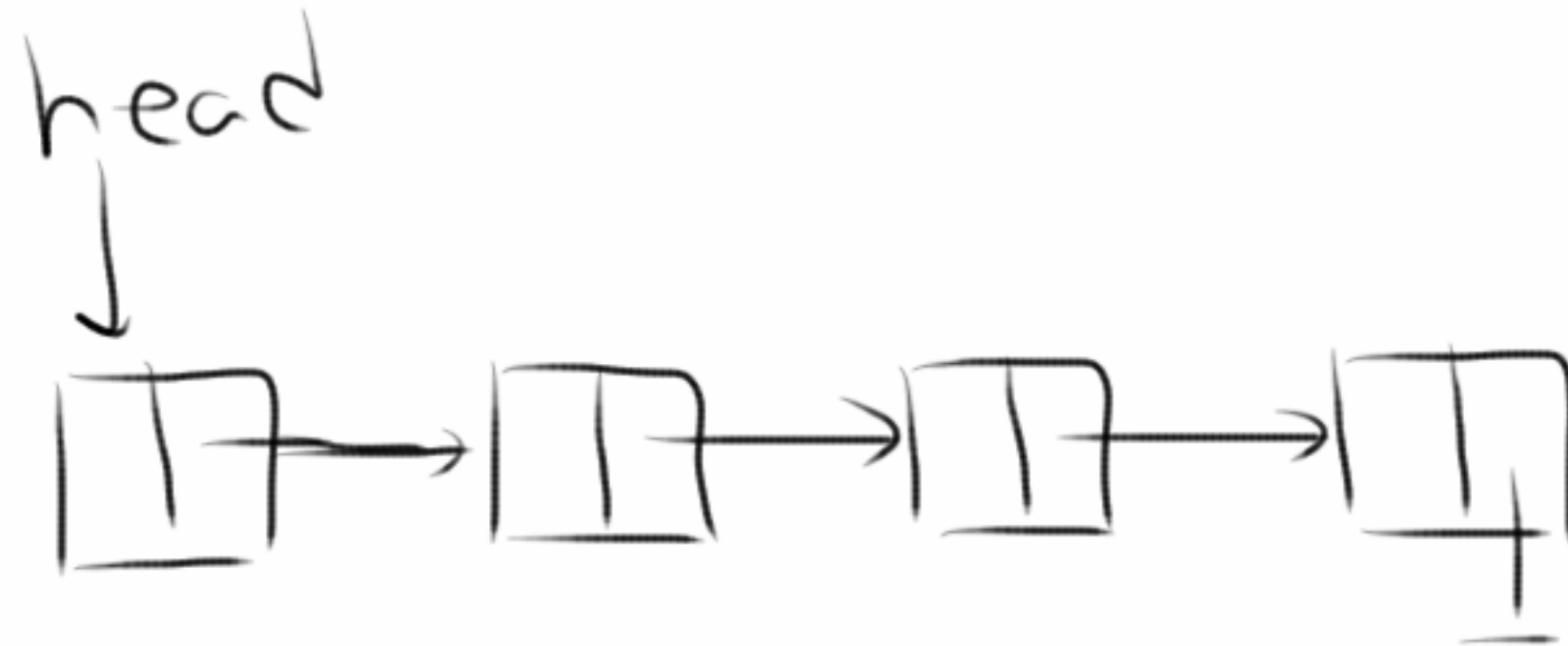
```
typedef struct {  
    Node* head;  
  
} List;
```

```
void add_head(List* list, int i);
```

traversal

- follow the next pointers

```
for ( p = head; p; p=p->next ) {  
}
```



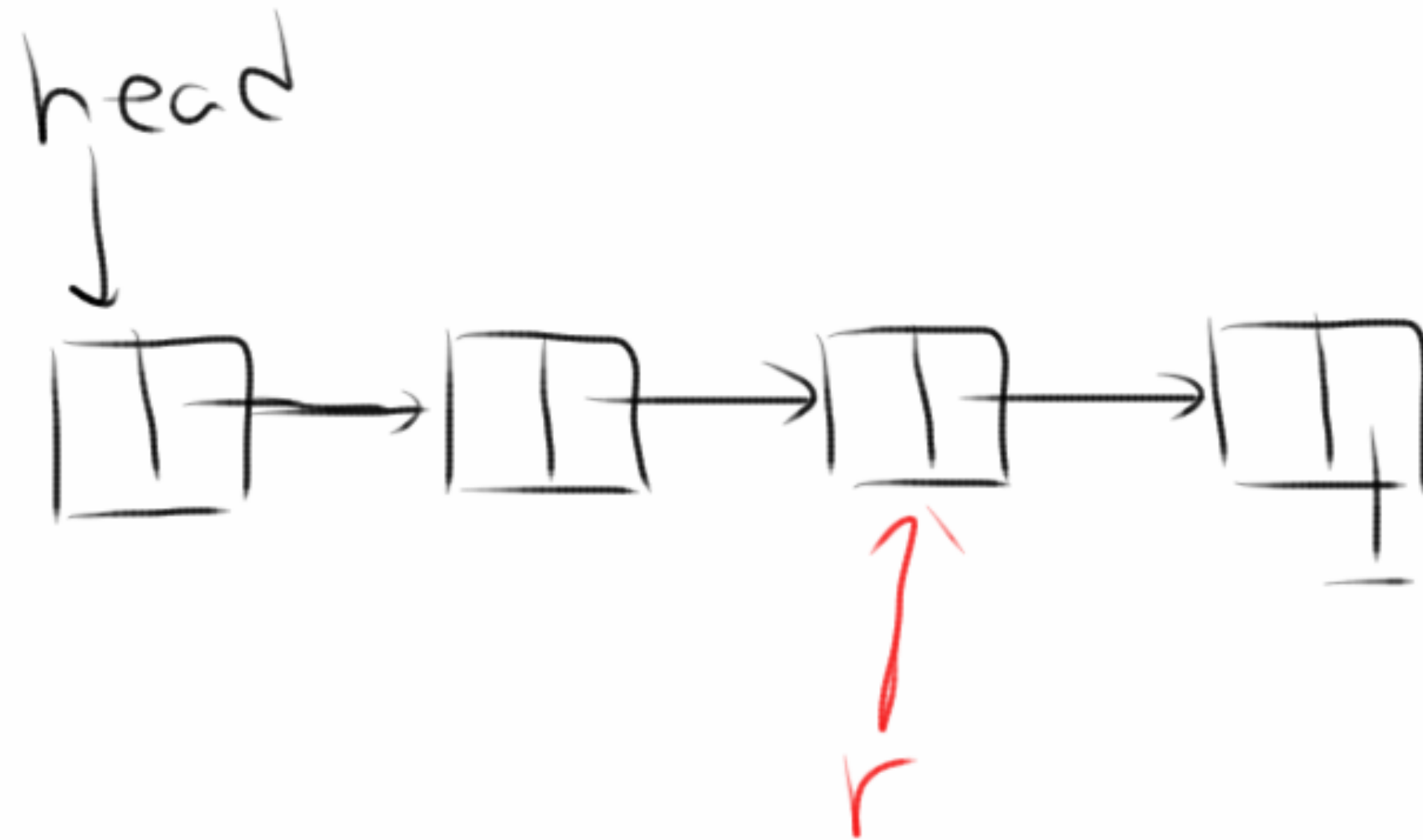
search

- find the value and return the pointer

```
ret = 0;
for ( p = head; p; p=p->next ) {
    if ( p->value == i ) {
        ret = p; break;
    }
}
```

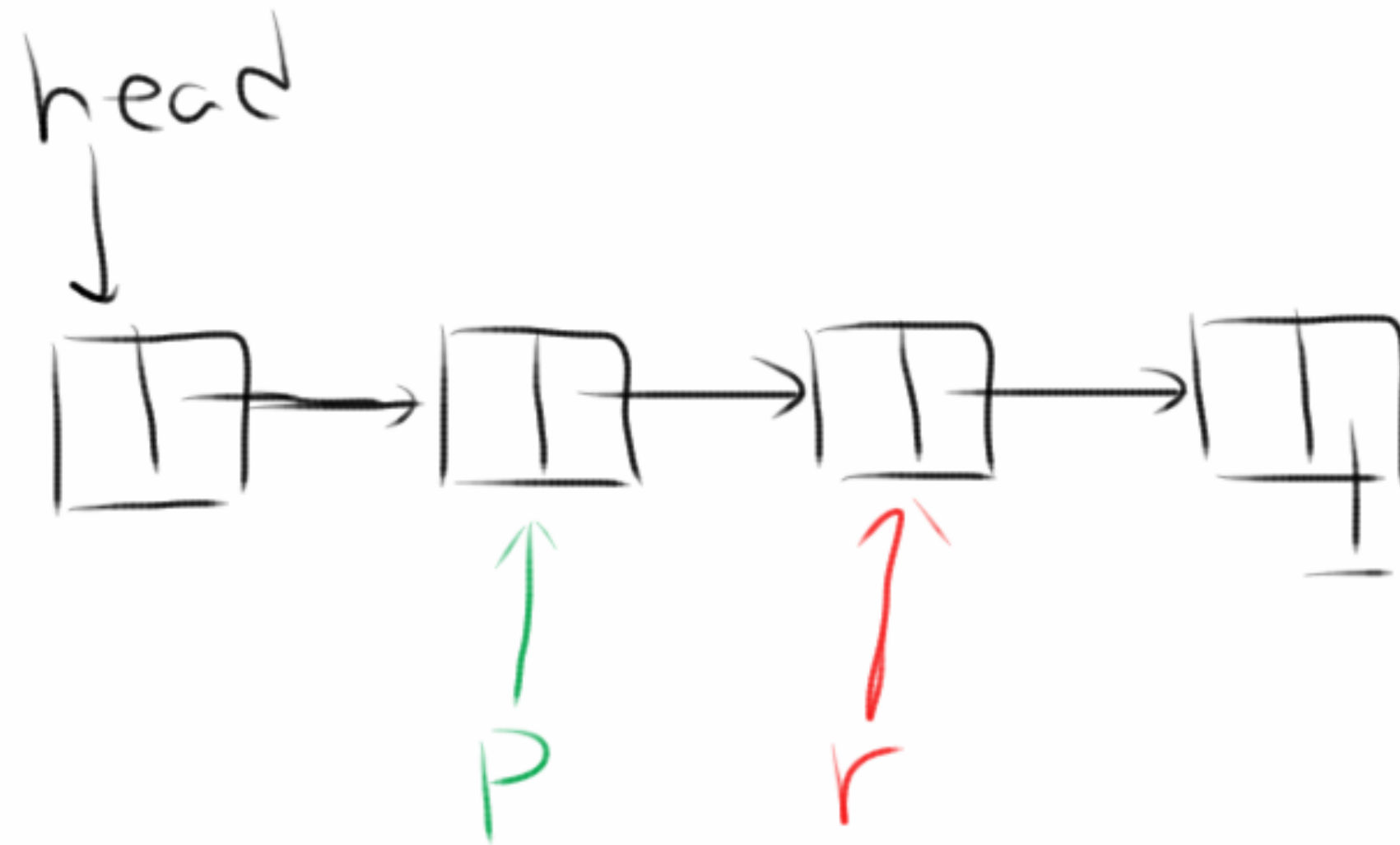
remove by a pointer l

- `remove(Node* r);`



remove by a pointer II

- find it's previous node p

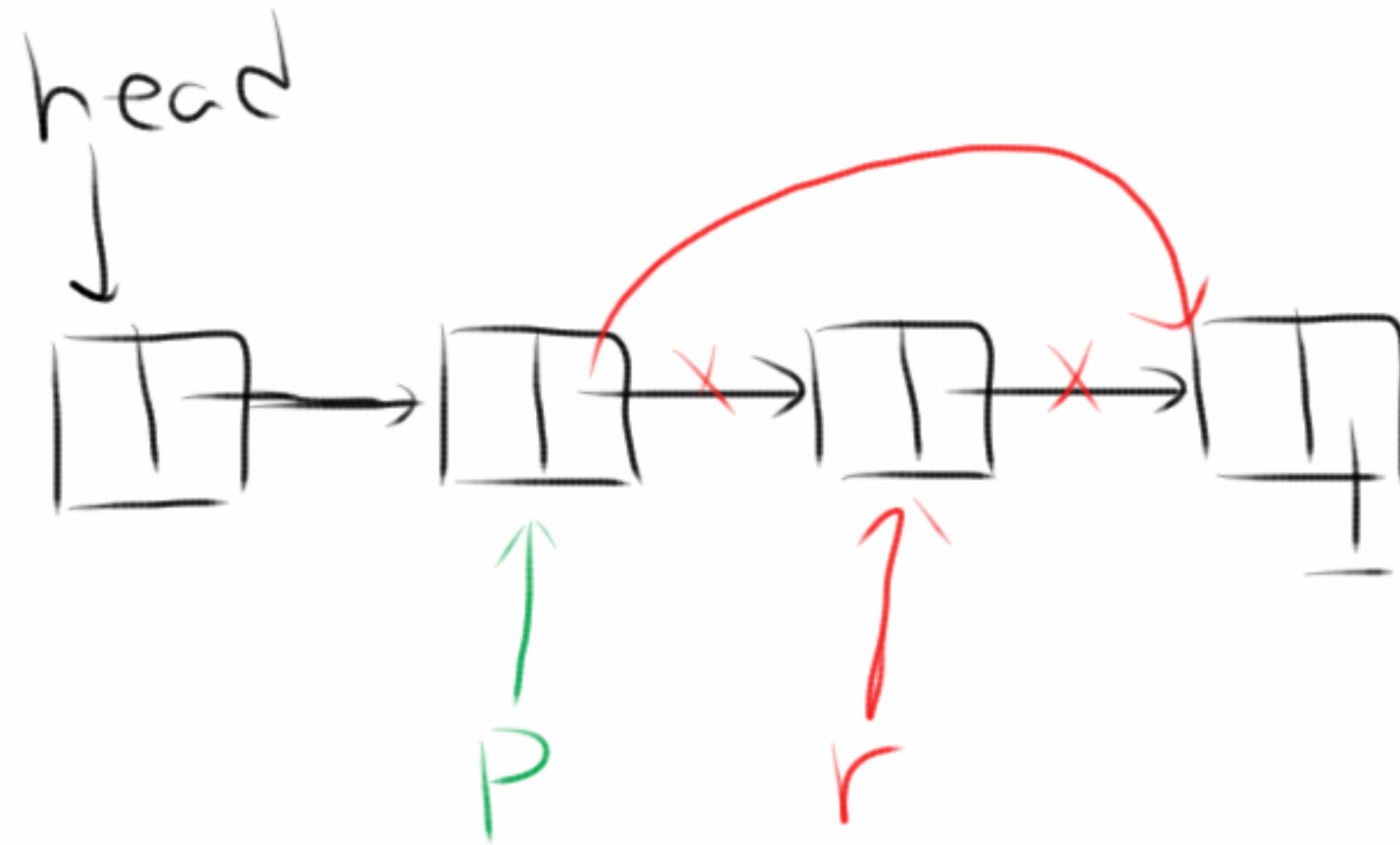


remove by a pointer III

```
for ( p = head; p; p=p->next ) {  
    if ( p->next == r ) {  
        }  
    }  
}
```

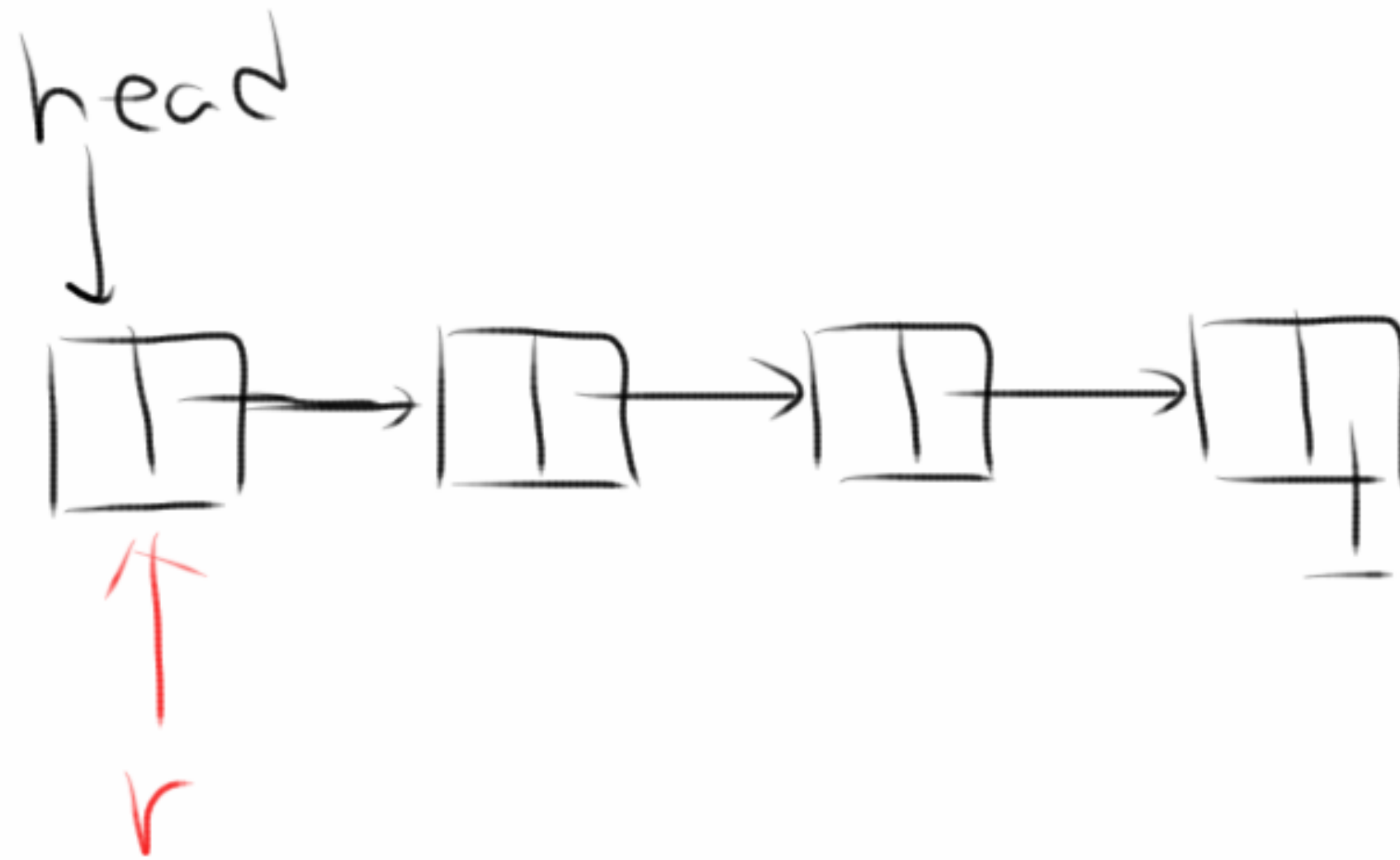
remove by a pointer IV

- `p->next = r->next;`



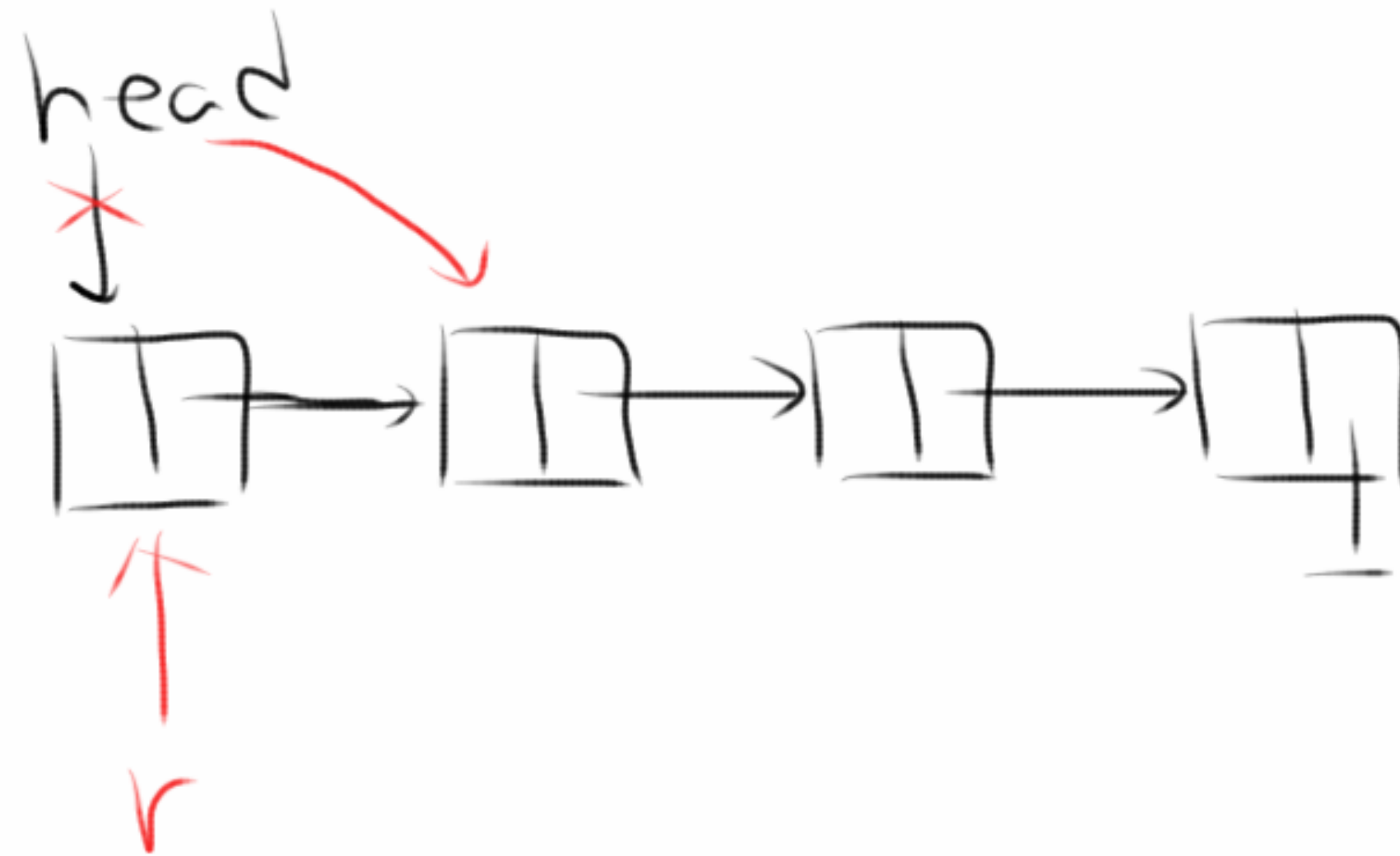
remove by a pointer V

- What if r is the head?



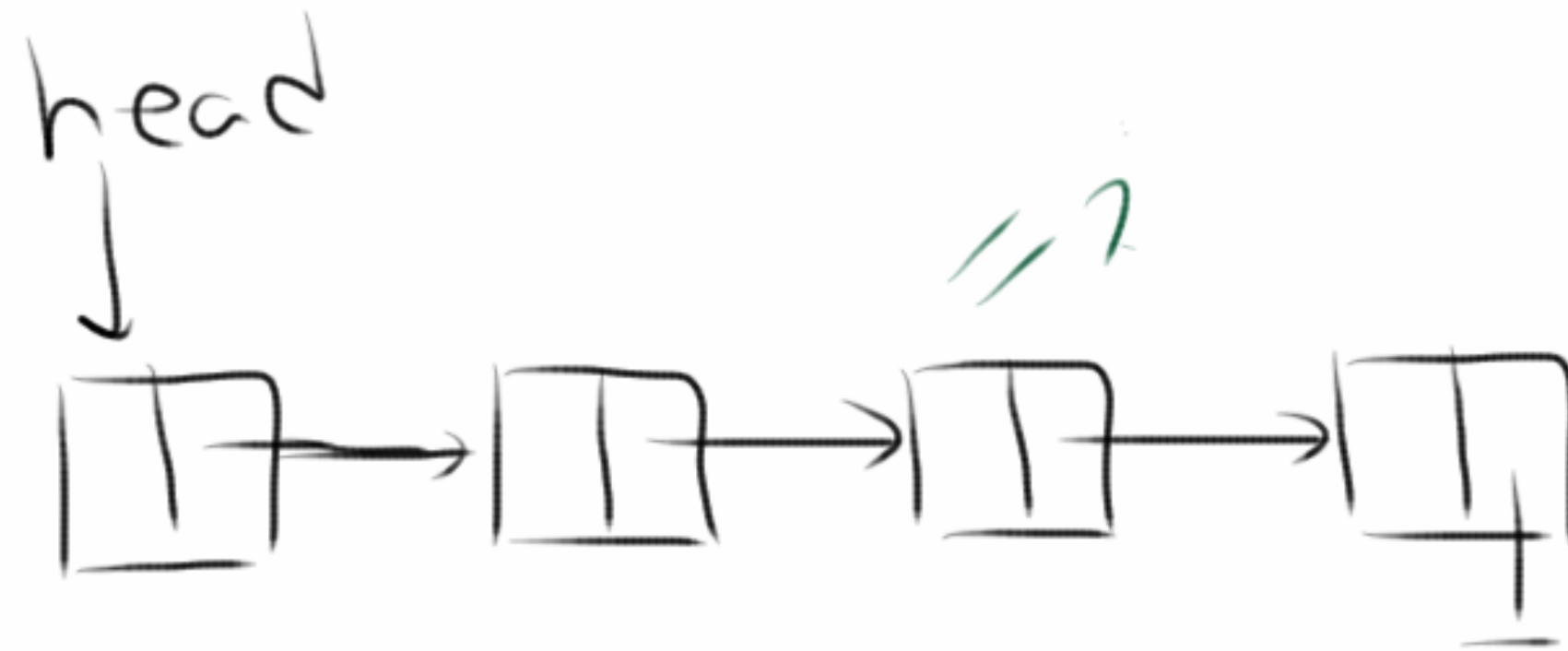
remove by a pointer VI

- new head = r->next



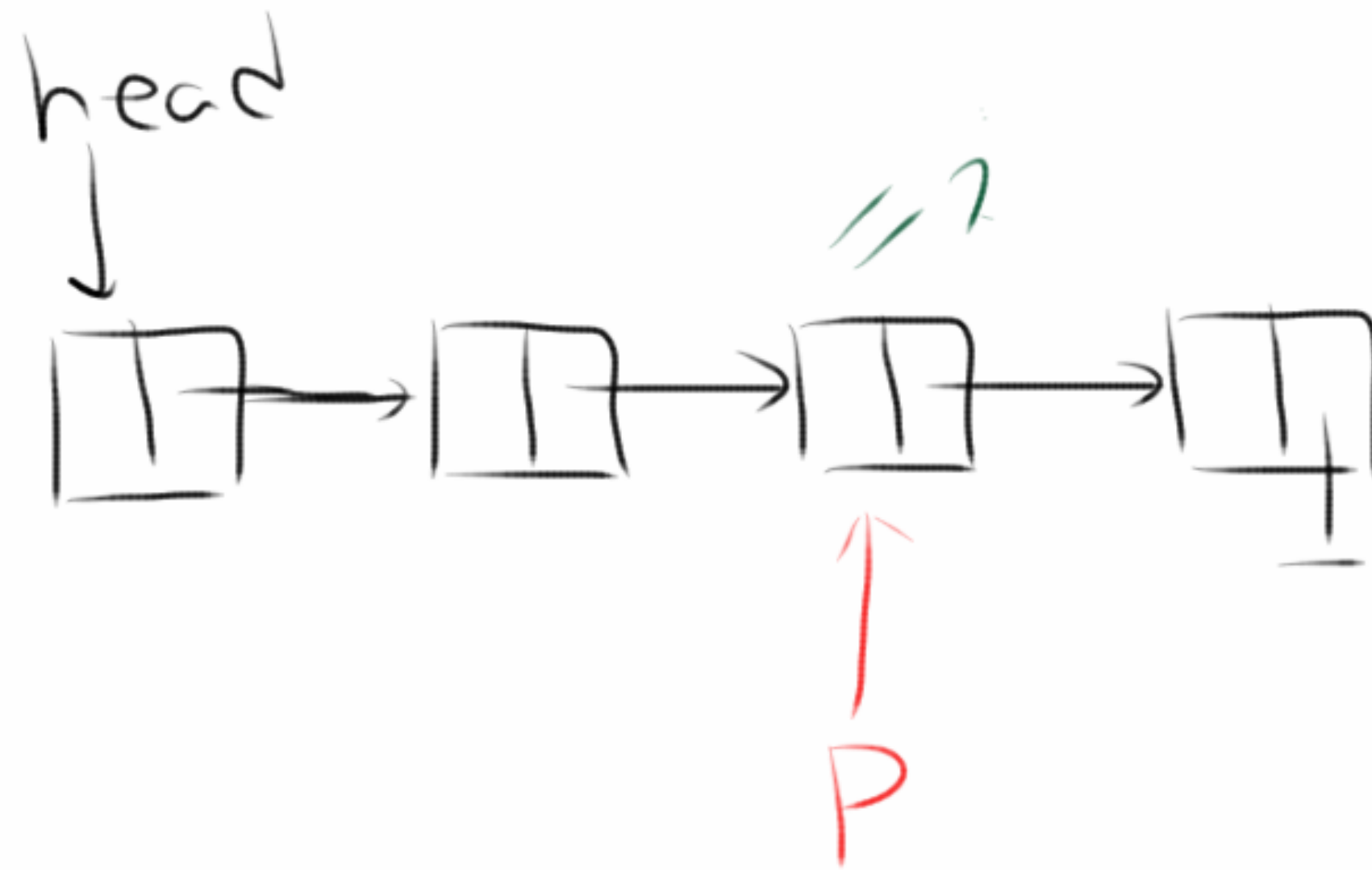
remove by a value I

- `remove(int i);`



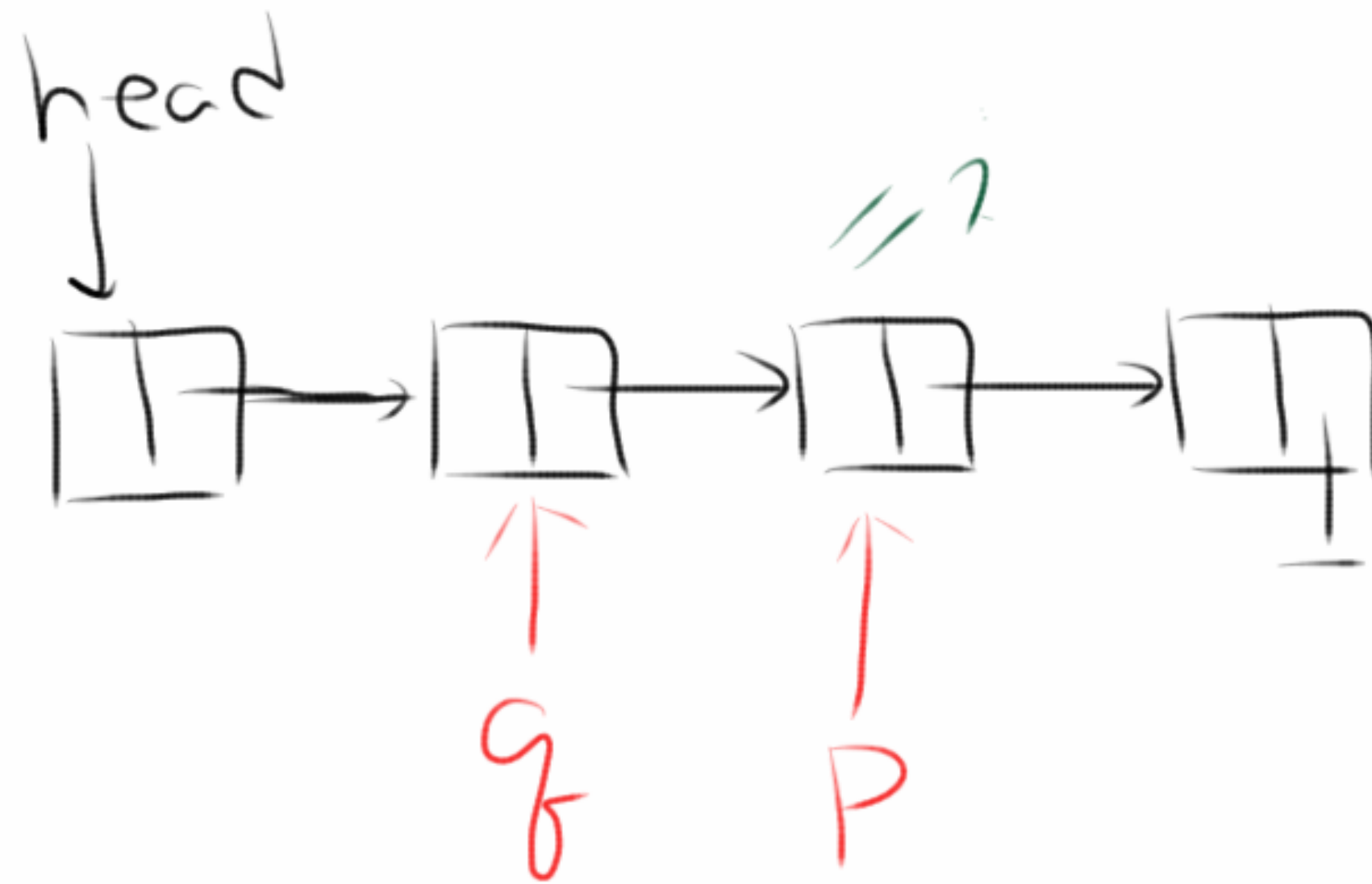
remove by a value II

- find $p \rightarrow \text{value} == i$



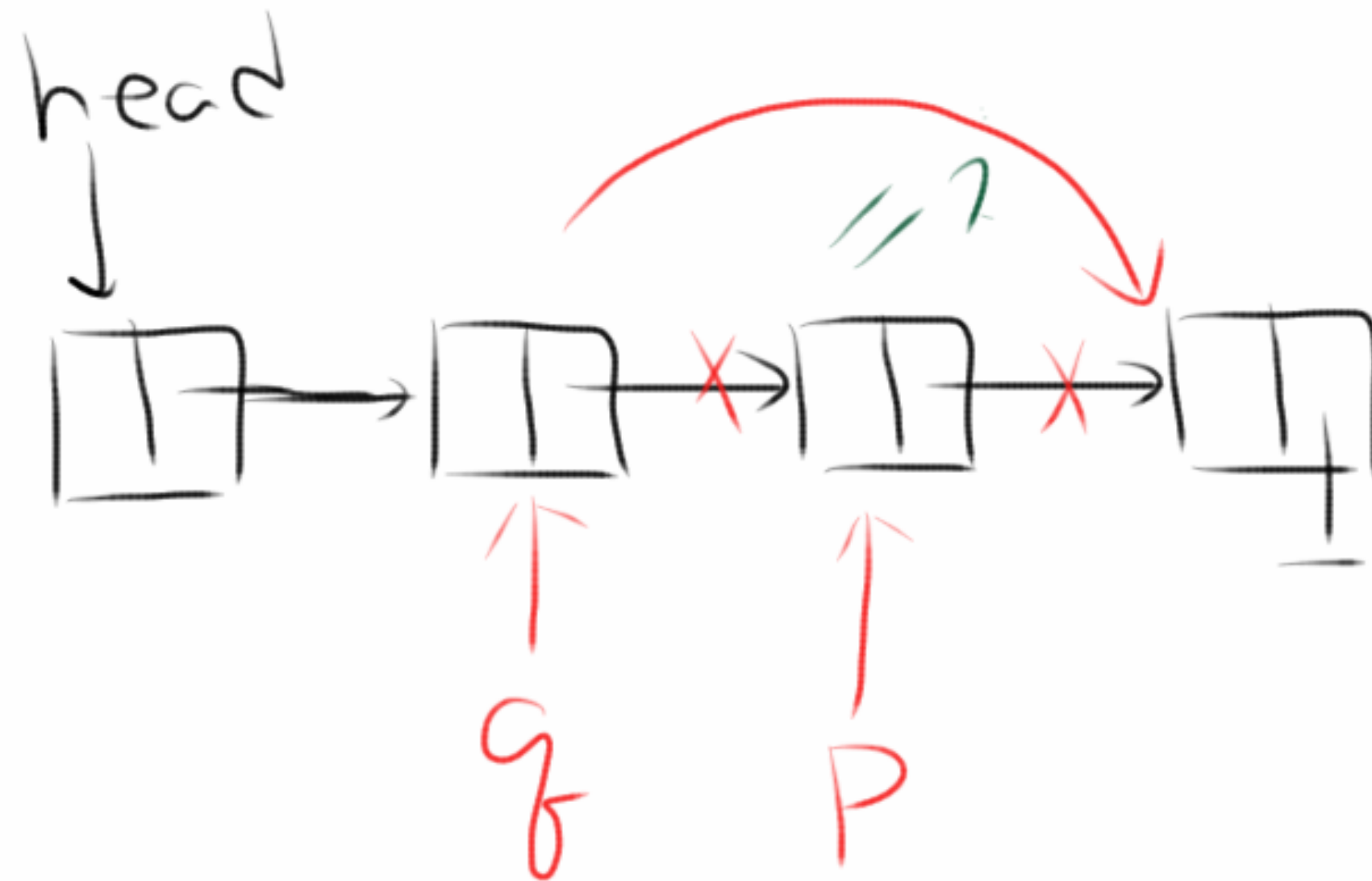
remove by a value III

- q for the previous node



remove by a value IV

- `q->next = p->next;`

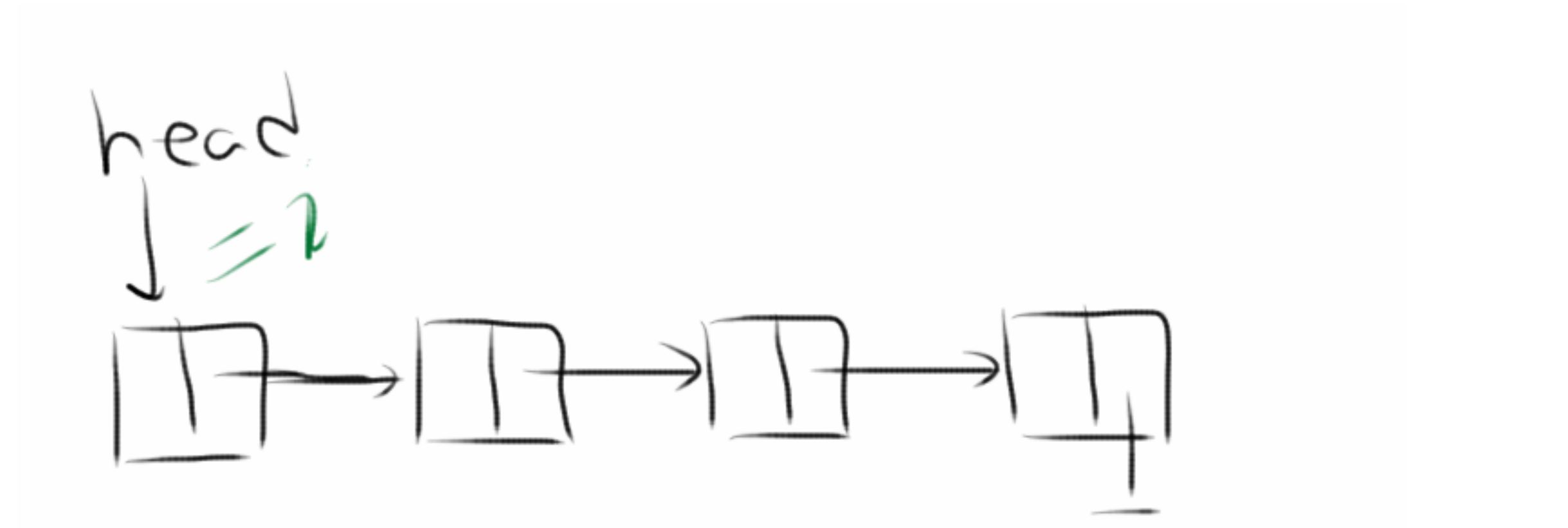


remove by a value V

```
for ( q=0, p = head; p; q=p, p=p-  
>next ) {  
    if ( p->value == i ) {  
        q->next = p->next;  
    }  
}
```

remove by a value VI

- what if `head->value == i`?

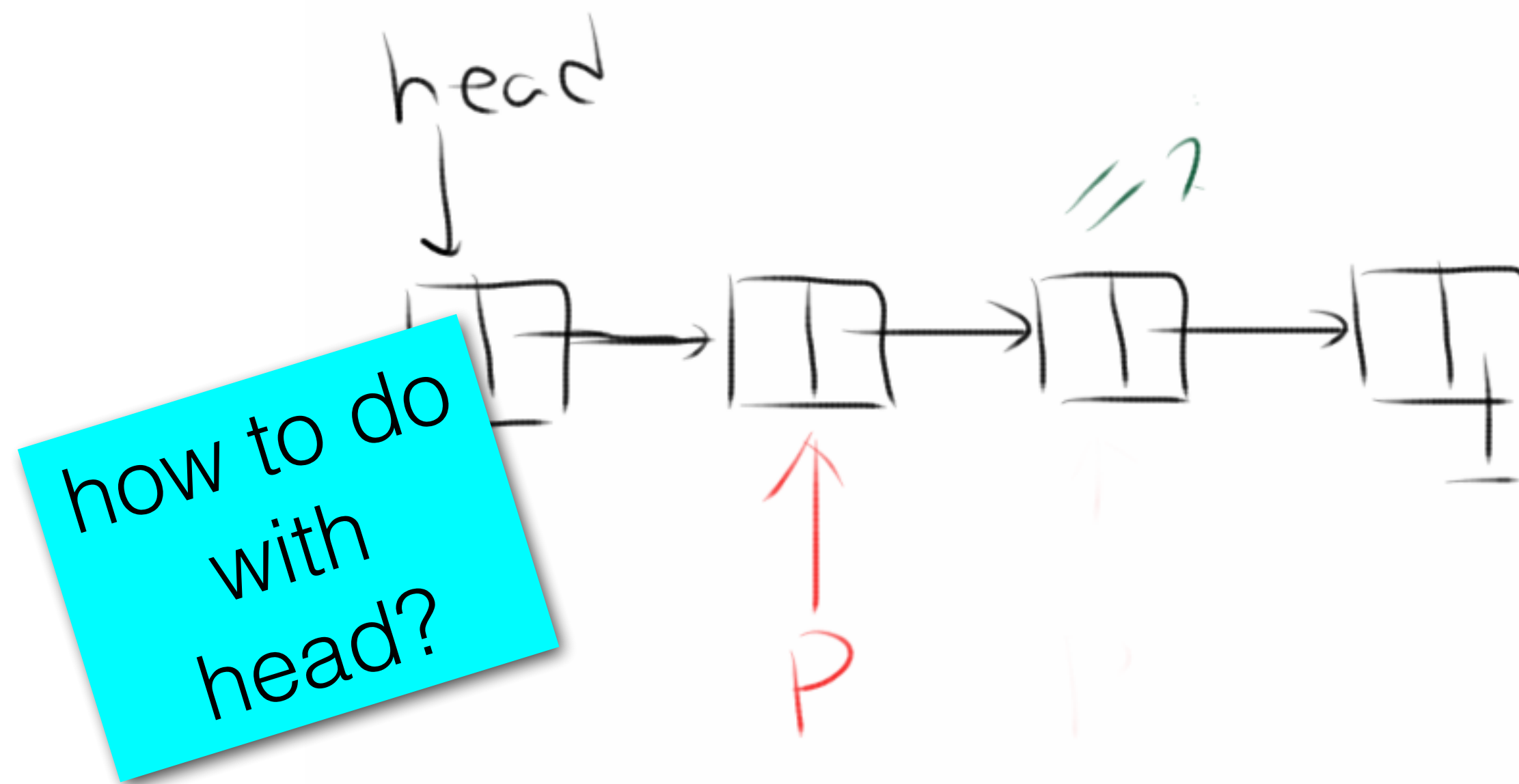


How do we find the boundary?

```
for ( q=0, p = head; p; q=p, p=p->next ) {  
    if ( p->value == i ) {  
        q->next = p->next;  
    }  
}
```

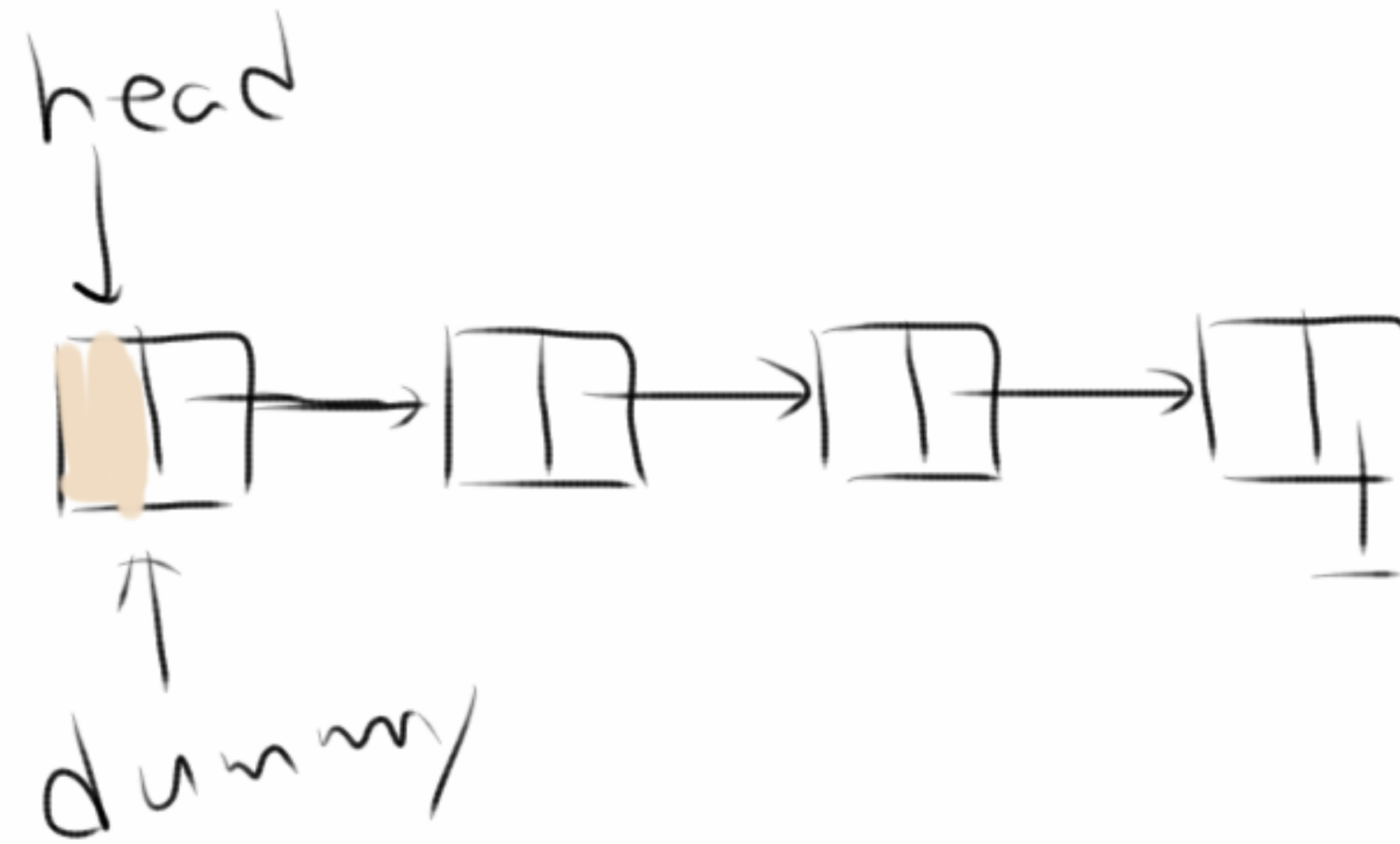
- Any pointer at the left of -> must be checked

remove by a value VII



- how about test `next->value`?
- `if (p->next && p->next->value == i)`

sentinel node



- a dummy head to make code smooth

all funcs with sentinel

- add_head
- append_tail
- traversal/search
- remove

any benefit?

clear the whole list I

```
void clear(Node *head)
{
    if ( head->next )
        clear(head->next);
    free(head);
}
```

clear the whole list II

```
for ( p = head; p; p=q ) {  
    q = p->next;  
    free(p);  
}
```

append tail

- find the tail
- `tail->next = n;`
- `n->next = 0;`
- what if empty list?

struct List II

```
typedef struct {  
    Node* head;  
  
    Node* tail;  
  
} List;
```

```
void add_head(List* list, int i);
```


all funcs with tail

- add_head
- append_tail
- traversal/search
- remove
- clear

any benefit?