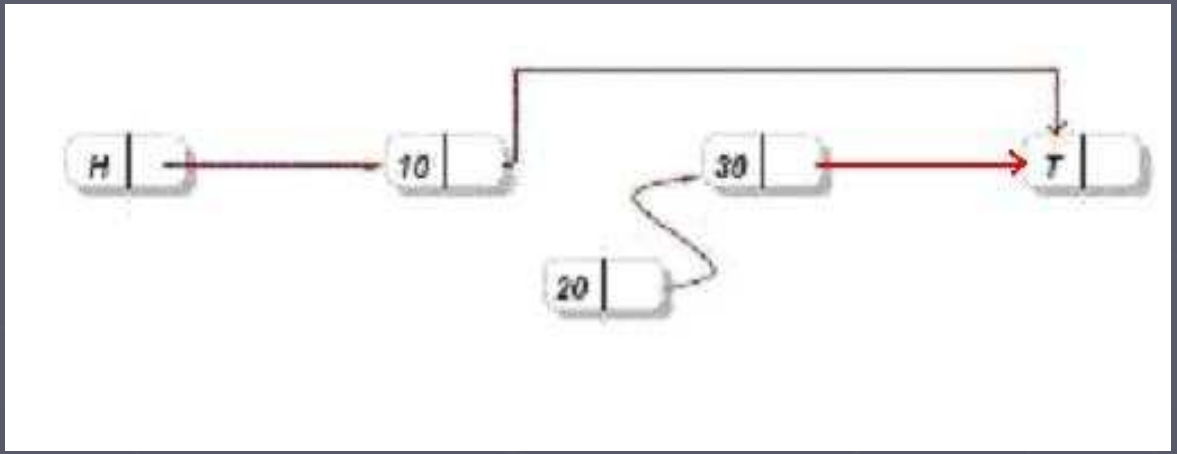
Shared Data Structures

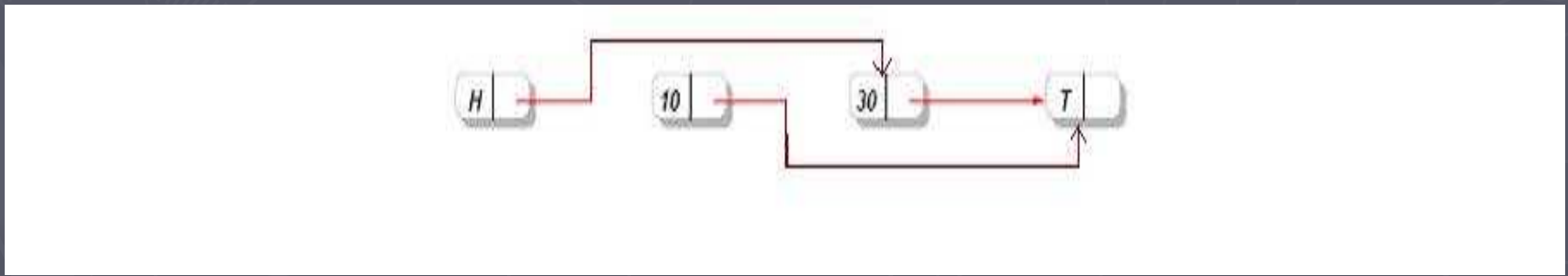
Shared Lists

Μια απλά συνδεδεμένη ταξινομημένη λίστα

Synchronization Problems when Accessing Shared Lists

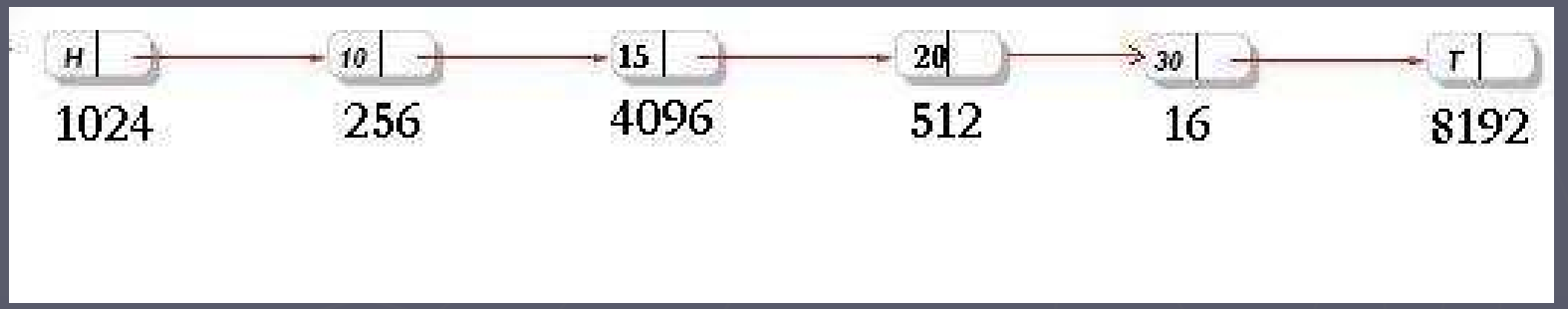


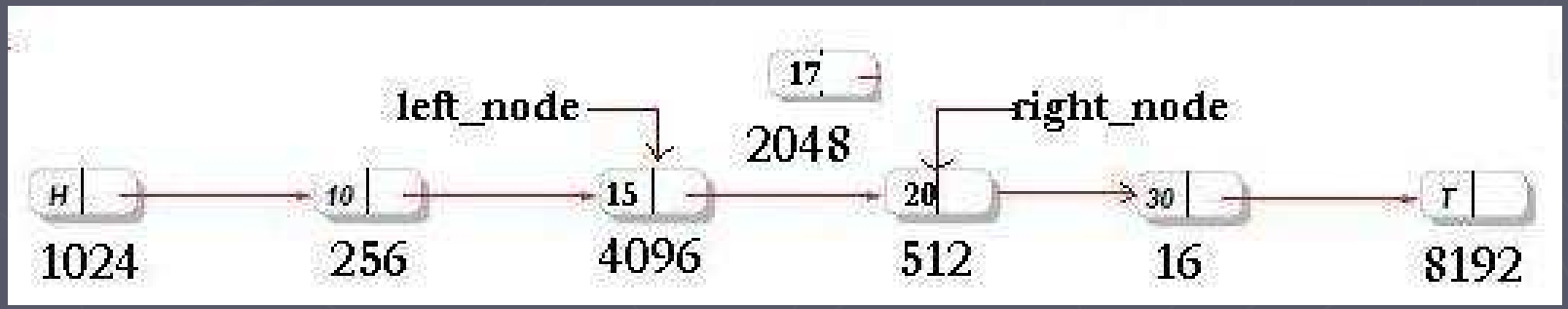
Εισαγωγή του κόμβου 20 και ταυτόχρονη διαγραφή του κόμβου 30 σε μια λίστα

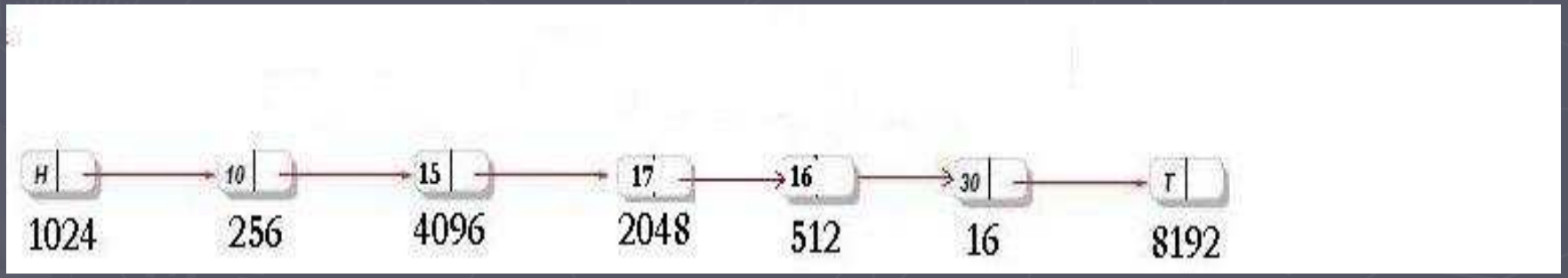


Ταυτόχρονη διαγραφή του 10 και του 30 από την λίστα

The ABA Problem







Ο πιο εύκολος τρόπος επίλυσης του ABA προβλήματος είναι να αποθηκευθεί μαζί με κάθε δείκτη next, μια ετικέτας ακεραίου η οποία θα αυξάνει κάθε φορά που ο δείκτης αλλάζει τιμή.

```
Class Node<KeyType> {
        KeyType key;
        Node *next;
        Node (KeyType key) {
        this.key = key;
        }
}
```

```
Class List<KeyType> {
      Node<KeyType> *head;
      Node<KeyType> *tail;
       List() {
            head=newNode<KeyType>();
           tail=newNode<KeyType> ();
       }
}
```

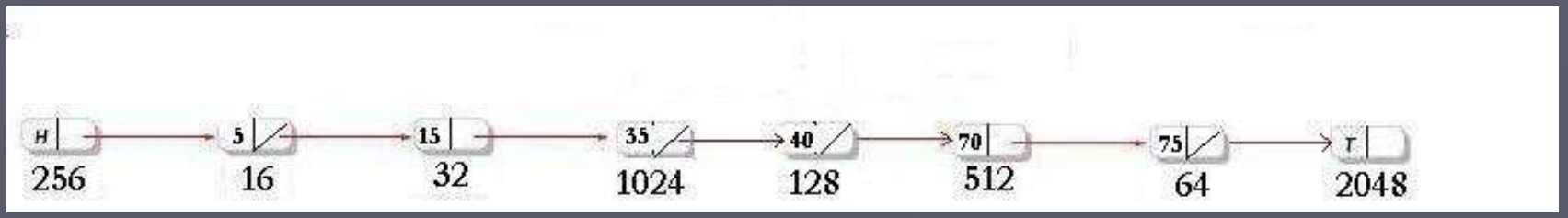
Structs που χρησιμοποιεί ο αλγόριθμος

Harris' Linked List Implementation

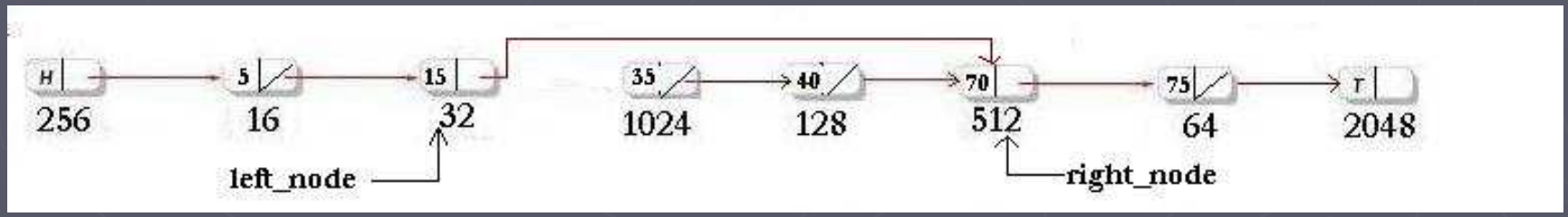
```
private Node *List::search (KeyType search_key, Node **left_node) {
Node *left_node_next, *right_node;
search_again:                                          // Η μέθοδος search με δοθέν κλειδί
        do {
(1)         Node *t = head;
(2)         Node *t_next = head.next;
(3)         do {
(4)             if (!is_marked_reference(t_next))  {
(5)                 (*left_node) = t;
(6)                 left_node_next = t_next;
                }
(7)             t = get_unmarked_reference(t_next);
(8)             if (t == tail) break;
(9)             t_next = t.next;
(10)        } while (is_marked_reference(t_next) || (t.key<search_key)); /*B1*/

(11)        right_node = t;
(12)        if (left_node_next == right_node)
(13)            if ((right_node != tail) && is_marked_reference(right_node.next))
(14)                goto search_again;
(15)            else return right_node;
(17)        if (CAS (&(left_node.next), left_node_next, right_node))
(18)            if ((right_node !=tail) && is_marked_reference(right_node.next))
(19)                goto search_again;
(20)            else return right_node;
        } while (true);
}
```

Harris' Linked List Implementation



Λίστα με μαρκαρισμένους τους κόμβους 5, 35, 40, 75



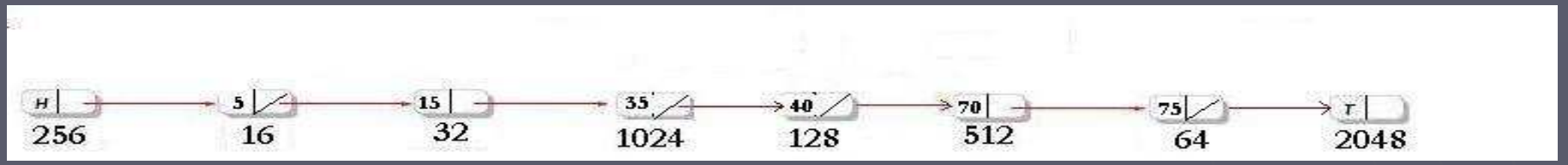
Η προηγούμενη λίστα μετά την εκτέλεση της search(70,head).

Harris' Linked List Implementation

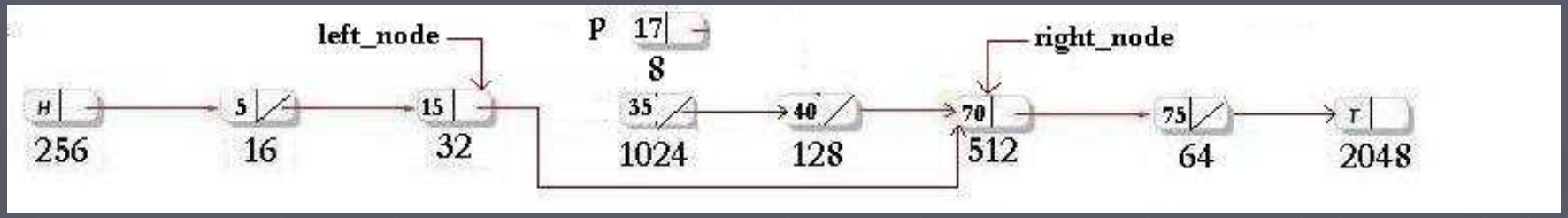
```
public boolean List::insert (KeyType key)
{
(1)       Node *new_node = new Node(key);
(2)       Node *right_node, *left_node;
          do {
(3)                 right_node = search (key, &left_node);
(4)                 if ((right_node != tail) && (right_node.key == key))
(5)                       return false;
(6)                 new_node.next = right_node;
(7)                 if (CAS (&(left_node.next), right_node, new_node))
(8)                       return true;
(9)       } while (true);
}
```

Η μέθοδος εισαγωγής κόμβου insert με δοθέν κλειδί

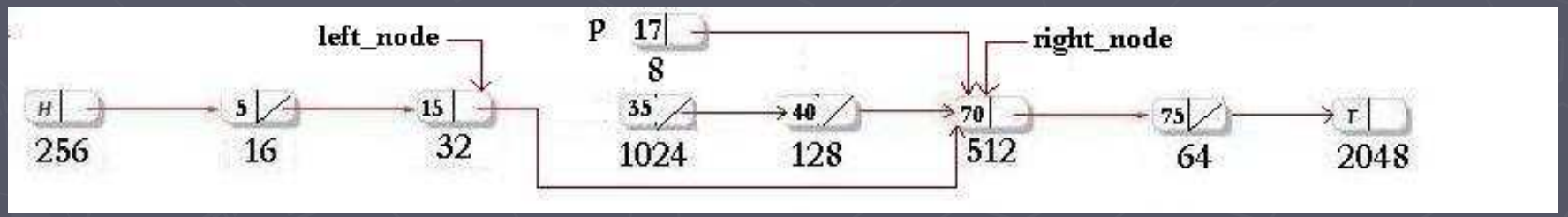
Harris' Linked List Implementation



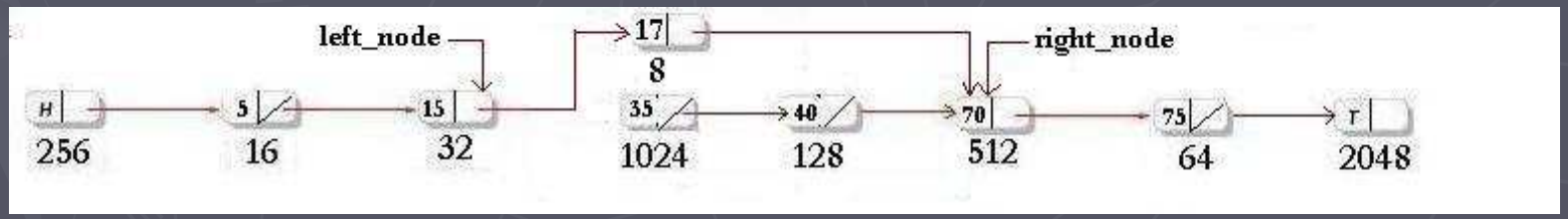
Λίστα με μαρκαρισμένους τους κόμβους με κλειδιά 5, 35, 40, 75



Η λίστα μετά την εκτέλεση της search(γραμμή 3).



Η λίστα μετά την εκτέλεση της γραμμής 6 της insert.



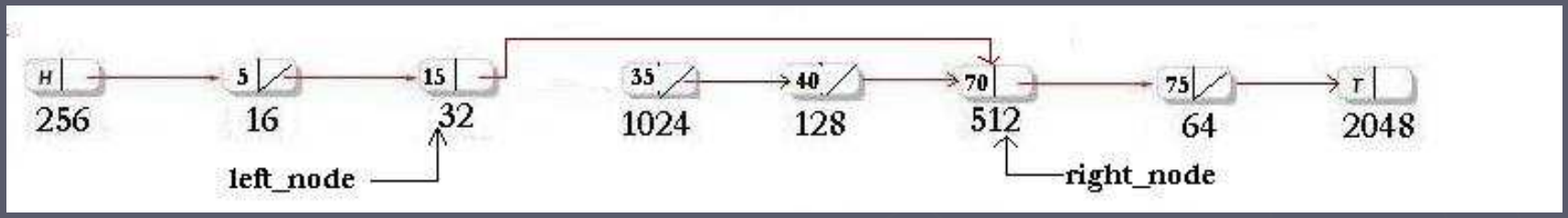
Η λίστα μετά την ολοκλήρωση της κλήσης insert(7).

Harris' Linked List Implementation

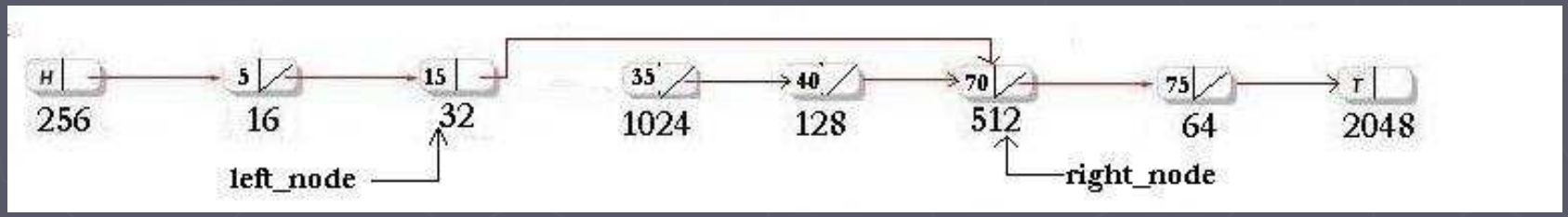
```
public boolean List::delete (KeyType search_key)
 {
(1)      Node *right_node, *right_node_next, *left_node;
          do {
(2)                  right_node = search (search_key, &left_node);
(3)                  if ((right_node == tail) || (right_node.key != search_key))
(4)                           return false;
(5)                  right_node_next = right_node.next;
(6)                  if (!is_marked_reference(right_node_next))
(7)                        if(CAS(&(right_node.next),right_node_next,
                                    get_marked_reference(right_node_next)))
(8)                           break;
(9)      } while (true);
(10)     if (!CAS (&(left_node.next), right_node, right_node_next))
(11)         right_node = search (right_node.key, &left_node);
(12)     return true;
}
```

Η μέθοδος διαγραφής κόμβου delete με δοθέν κλειδί

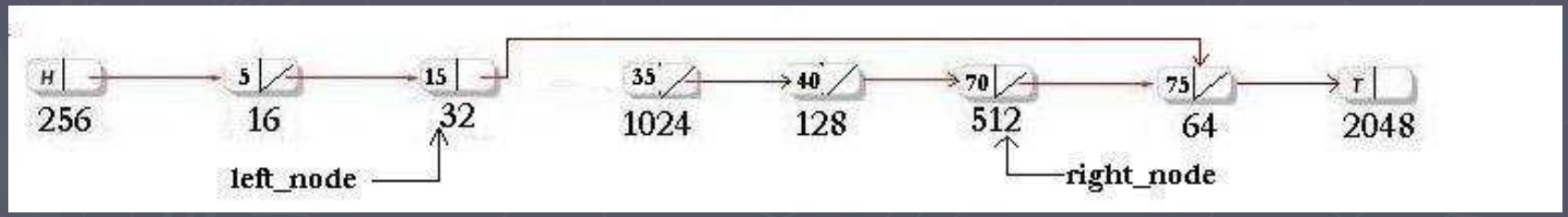
Harris' Linked List Implementation



Η λίστα μετά την εκτέλεση της γραμμής 2 της delete



Η λίστα μετά το μαρκάρισμα του κόμβου



Η λίστα μετά τη φυσική διαγραφή του κόμβου