

Re: Palindrome in Linked list

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2005-01/0764.html>

From: Roger Willcocks (roger_at_rops.org)

Date: 01/25/05

Date: Tue, 25 Jan 2005 02:00:30 -0000

"Willem" <willem@stack.nl> wrote in message
news:slrncva52t.1b7p.willem@toad.stack.nl...

> *CTips* wrote:

>) *Can you do it in $O(n)$ using $O(1)$ memory? I don't think you can do it*

>) *efficiently using recursion.*

>

> *Of course recursion is efficient, you're basically using the stack as $O(n)$*

> *extra memory. Note that the recursion is linear.*

>

> *And about $O(n)$ time and $O(1)$ memory: Are you allowed to alter the*

> *linked list? If not, I doubt it can be done within those restraints.*

> *(Assuming it's singly linked, of course. Doubly linked is easy.)*

>

Not forgetting, of course, that you can represent a doubly-linked list with a single link. See e.g. <http://c2.com/cgi/wiki?TwoPointersInOneWord> which leads to an $O(n)$ time $O(1)$ space implementation – rewrite the list as a bidirectional list; check for a palindrome; rewrite the list back to as it was originally.

But you don't of course need to reverse the entire list – just the first half – and you might just as well simply reverse the pointers as you go rather than doing any funky xor stuff...

```
/*  $O(n)$  time  $O(1)$  space check for palindromic singly-linked list */
```

```
/* reverse pointers until we get half way down the list, then start  
comparing
```

```
    outwards, fixing the pointers as we go */
```

```
int isPalindrome(list *head)
```

```
{
```

```
    list *tail, *walk = head;
```

```
    list *back = 0;
```

```
    int retval = 1;
```

```
    while (walk && walk->next) {
```

```
        tail = head->next;
```

comp.programming: Re: Palindrome in Linked list

```
walk = walk->next->next;
head->next = back;
back = head;
head = tail;
}
if (walk)
    head = head->next;
while (head) {
    if (head->c != back->c)
        retval = 0;
    walk = back;
    head = head->next;
    back = back->next;
    walk->next = tail;
    tail = walk;
}
return retval;
}

--
Roger
```