**单链表的遍历**

1. **实验目的和要求**

已知一个带有表头结点的单链表，结点结构为

Data Next

假设该链表只给出了头指针list。在不改变链表的前提下，请设计一个尽可能高效的算法，查找链表中倒数第k个位置上的结点(k为正整数)。若查找成功，算法输出该结点的data值，并返回1:否则，只返回0。要求：

1. 描述算法的基本设计思想
2. 根据设计思想实现该函数

**二、实验内容**

1. 实验准备
   1. 理论知识介绍
      1. 时间复杂度计算: 通常采用下属办法来估计算法在给定输入下的计算量：
         1. 根据问题的特点合理地选择一种或几种操作作为“标准操作”，将标准操作作为一个抽象的运算单位。
         2. 确定每个算法在给定输入下共执行了多少次标准操作，并将它作为算法的计算量。
         3. 利用求和定理和求积定理相结合可以计算出最终的时间复杂度
      2. 单链表
         1. 在单链表中，每个结点由一个数据元素和一个后继指针组成。后继指针指向储存该元素的直接后继的结点。
         2. 终端结点的后继指针为空指针，表示它没有后继结点。一个单链表可以由一个指向起始结点的指针表示。
         3. 由于单链表的数据结构的特点，单链表只能从头向后遍历，而且遍历时间复杂度为O(n)，相比顺序表慢多了，但是单链表的插入和删除的时间复杂度只有O(1)，比顺序表快。
2. 实验项目
   1. 分析
      1. 题目中所给的数据结构是单链表，要求我们找到倒数第k个元素，但是单链表没有办法从尾部向前遍历，所以我们不能直接找到第k个元素。
      2. 比较直接的想法就是先从头向尾遍历一次整个单链表，得到整个单链表的总的长度n。然后再从头到尾搜索n-k次，得到的结点就是我们想要的结点
      3. 上述的方法比较直接，但是平均需要将整个单链表遍历1.5次，效率并不高，我们可以想办法来只对单链表进行一次遍历
   2. 方案
      1. 为了只对单链表进行一次遍历，我们可以利用两个指针同时遍历的方法。当第一个指针开始对单链表进行遍历的时候，第二个指针先不动。当第一个指针对链表搜索了k-1次之后，第二个指针开始对链表进行遍历。
      2. 之后，当第一个指针搜索到单链表最后一个结点的时候，第二个指针正好指向倒数第k个结点。
   3. 源代码

int FindLastNumber(int k, SinglyLinkedList<int> sll, Node \*list) {

Node \*p1, \*p2 = list;

for (int i = 0; i < k; i++) {

if (p1->next) {

p1 = p1->next;

}

else {

return 0;

}

}

while (p1) {

p1 = p1->next;

p2 = p2->next;

}

cout << p1->data << endl;

return 1;

}

**三、实验小结**

1. 重点

对单链表的结构的了解以及单链表是如何进行遍历的。

1. 难点

如何设计一个时间复杂度较小的遍历单链表倒数第k个元素的算法。

1. 解决方法

利用两个指针同时对单链表进行搜索的方法来对单链表进行遍历，只要我们将这两个指针相互之间的距离保持在k，那么当第一个指针到达最后一个结点并成为空指针时，第二个指针恰好指向倒数第k个元素。

1. 收获、体会

时间复杂度对于程序来说是一个很重要的部分，通常情况下最直接的方法的时间复杂度比较大，所以这种办法并不是最优的，这个题目中最直接的方法就是先将单链表遍历一遍获得单链表的长度再通过单链表的长度决定第二次指针搜索次数，但是这种方法时间复杂度较大。我们通过两个指针的方法减少了遍历单链表的次数，从而提高了程序的效率。这启发我们在完成程序的时候首先注重的是程序的算法的设计，而不是程序的具体实现，优秀的算法带给我们的效率的提升是显著的。