PAT题目的单链表题目

　　PAT的单链表基本都是静态链表。可以当成常规的单链表来做，会比较符合出题人的意图。但也有一些别的办法来完成静态链表的题目。

1097 Deduplication on a Linked List (25分)

Given a singly linked list *L* with integer keys, you are supposed to remove the nodes with duplicated absolute values of the keys. That is, for each value *K*, only the first node of which the value or absolute value of its key equals *K* will be kept. At the mean time, all the removed nodes must be kept in a separate list. For example, given *L* being 21→-15→-15→-7→15, you must output 21→-15→-7, and the removed list -15→15.

Input Specification:

Each input file contains one test case. For each case, the first line contains the address of the first node, and a positive *N* (≤105) which is the total number of nodes. The address of a node is a 5-digit nonnegative integer, and NULL is represented by −1.

Then *N* lines follow, each describes a node in the format:

```
Address Key Next
```

where `Address` is the position of the node, `Key` is an integer of which absolute value is no more than 104, and `Next` is the position of the next node.

Output Specification:

For each case, output the resulting linked list first, then the removed list. Each node occupies a line, and is printed in the same format as in the input.

Sample Input:

```
00100 5
99999 -7 87654
23854 -15 00000
87654 15 -1
00000 -15 99999
00100 21 23854
```

Sample Output:

```
00100 21 23854
23854 -15 99999
99999 -7 -1
00000 -15 87654
87654 15 -1
```

解法一：常规单链表破解

1. 用先历遍一下，用hashTable记录结点的绝对值。
2. 历遍的时候碰到一个绝对值相同的结点时，删除原来的结点，放在另外一个单链表上。
3. 然后历遍输出
4. 需要注意的时，删除了结点之后，p的地址不用后移。否则当出现连续的相同数据时，会只删掉一半的结点。下面提供一份测试数据。

   ```
   00100 5
   99999 1 87654
   23854 1 00000
   87654 1 -1
   00000 1 99999
   00100 1 23854
   ```

   正确的输出应为

   ```
   00100 1 -1
   23854 1 00000
   00000 1 99999
   99999 1 87654
   87654 1 -1
   ```

代码参考：

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;
const int inf = 100100;
vector<int> hashTable(inf/10);
class node
{
public:
    int value, next;
    node()
    {
        value = 0, next = -1;
    }
};
vector<node> list(inf);
int head = 0, num = 0;
void init()
{
    cin >> head >> num;
    for(int i=0;i<num;i++)
```

```
    {
        int parent = 0;
        cin >> parent;
        cin >> list[parent].value >> list[parent].next;
    }
}
int main()
{
    init();
    int p = head, q = -1, spe = -1;
    hashTable[abs(list[p].value)] = 1;
    while(p!=-1&&list[p].next!=-1)
    {
        if (hashTable[abs(list[list[p].next].value)] == 0) {
            hashTable[abs(list[list[p].next].value)] = 1;
            p = list[p].next;
        }
        else if( hashTable[abs(list[list[p].next].value)]==1)
        {
            int temp = list[p].next;
            list[p].next = list[temp].next;
            list[temp].next = -1;
            if(spe==-1) q = spe = temp;
            else
            {
                list[q].next = temp;
                q = list[q].next;
            }
        }

    }
    p = head, q = spe;
    while (p!=-1)
    {
        if (list[p].next != -1)
            printf("%05d %d %05d\n", p, list[p].value, list[p].next);
        else
            printf("%05d %d %d\n", p, list[p].value, list[p].next);
        p = list[p].next;
    }
    while (q!=-1)
    {
        if (list[q].next != -1)
            printf("%05d %d %05d\n", q, list[q].value, list[q].next);
        else
            printf("%05d %d %d\n", q, list[q].value, list[q].next);
        q = list[q].next;
    }
    return 0;
}
```

解法二：sort大法

1. 给结构体设立一个标识，按要求排序直接得结果

代码参考：

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;
const int inf = 100100;
struct node
{
    int address, value, next, flag = -2 * inf;
};
vector<node> list(inf);
vector<int> hashTable(inf / 10);
int head = 0, num = 0, mainCnt = inf, othCnt = -inf, cnt1 = 0, cnt2 = 0;
void init()
{
    cin >> head >> num;
    for(int i=0;i<num;i++)
    {
        int temp;
        scanf("%d", &temp);
        list[temp].address = temp;
        scanf("%d %d", &list[temp].value, &list[temp].next);
    }
}
bool cmp(node a,node b)
{
    return a.flag > b.flag;
}
int main()
{
    init();
    int p = head;
    while (p!=-1)
    {
        if (hashTable[abs(list[p].value)] == 0) {
            list[p].flag = --mainCnt; cnt1++;
            hashTable[abs(list[p].value)] = 1;
        }
        else {
            list[p].flag = --othCnt; cnt2++;
        }
        p = list[p].next;
    }
    sort(list.begin(), list.end(), cmp);
    num = cnt1 + cnt2;
    for(int i=0;i<num;i++)
    {
        printf("%05d %d ", list[i].address, list[i].value);
        if (i == cnt1 - 1 || i == num - 1)
            printf("-1\n");
        else
            printf("%05d\n", list[i + 1].address);
    }
    return 0;
}
```