时间、空间复杂度分析

> problem1.pdf 对该程序的时间复杂度和空间复杂度进行分析。

```
//move函数时间复杂度O(n)
//空间复杂度O(1)
template <class elemType>
typename sLinkList<elemType>::node * sLinkList<elemType>::move(int i)
const
{
    node *p = head;
    //while循环时间复杂度最高，为O(n)
    while(i-- >= 0) p = p->next;
    return p;
}
```

```
//insert函数时间复杂度O(n)
//空间复杂度O(1)
template <class elemType>
void sLinkList<elemType>::insert(int i, const elemType &x)
{
    node *pos;
    //move函数时间复杂度最高，为O(n)
    pos = move(i - 1);
    pos->next = new node(x,pos->next);
    ++currentLength;
}
```

```
//remove函数时间复杂度O(n)
//空间复杂度O(1)
template <class elemType>
void sLinkList<elemType>::remove(int i)
{
    node *pos, *delp;
    //move函数时间复杂度最高，为O(n)
    pos = move(i - 1);
    delp = pos->next;
    pos->next = delp->next;// 绕过delp
    delete delp;
    --currentLength;
}
```

```
//erase函数时间复杂度O(n3)
//空间复杂度O(1)
template <class elemType>
int sLinkList<elemType>::erase(const elemType &x, const elemType &y)
{
    //for循环时间复杂度为O(n3)
    for(int number = x; number <= y; ++number)
        //while循环时间复杂度为O(n2)
        while(search(number) >= 0)
            //remove函数时间复杂度O(n)
            remove(search(number));
    return length();
}
```

```
//traverse函数时间复杂度O(n)
//空间复杂度O(1)
template <class elemType>
void sLinkList<elemType>::traverse() const
{
    node *p = head->next;
    //while循环时间复杂度为O(n)
    while (p != NULL) {
        cout << p->data << " ";
        p=p->next;
    }
    cout << endl;
}
```

```
//main函数时间复杂度为O(n3)
//空间复杂度O(n)
int main()
{
    //while循环时间复杂度为O(n2)
    //空间复杂度O(n)
    while (flag and cin >> iniList)
    {
        //insert函数时间复杂度最高，为O(n)
        //空间复杂度为O(1)
        myList.insert(myList.length(), iniList);
        flag = cin.get()!='\n';
    }
    flag = true;

    //while循环时间复杂度为O(1)
    //空间复杂度O(1)
    while (flag and cin >> iniList)
```

```
    {
        range[i] = iniList;
        ++i;
        flag = cin.get()!='\n';
    }
    //erase函数时间复杂度O(n3)
    //空间复杂度O(1)
    cout << myList.erase(range[0], range[1]) << endl;

    //traverse函数时间复杂度O(n)
    //空间复杂度O(1)
    myList.traverse();

    return 0;
}
```