第一章作业.

1. 解: 执行次数 $f(n)=\sum_{i=1}^{n-1}\left((n-i+1)+(n-i+2)\right)+n$ $=(n+(n-1)+(n-2)+\cdots+2)+((n+1)+\cdots+3)+n$

$$=\frac{(n+2)(n-1)}{2}+\frac{(n+4)(n-1)}{2}+n=(n+3)(n-1)+n=n^2+3n-3.$$

2. 解: (1) $T(n)=3n^3+10n^2+2n=O(n^3)+O(n^2)+O(n)$

$$=O(n^3)$$

(2) $T(n)$ 的表达式为 $T(n)=2^n-1$. 下用数学归纳法

证明: $n=1$, $T(1)=1=2^1-1$ 合上式.

假设当 $n=k(k\geq 1)$ 时有 $T(k)=2^k-1$ 成立.

则当 $n=k+1$ 时 $T(k+1)=2T(k)+1=2\cdot 2^k-2+1$

$$=2^{k+1}-1 \text{ 合上式.}$$

综上, $\forall n\in N_+$, $T(n)=2^n-1$

$\therefore T(n)=O(2^n)+O(1)=O(2^n)$.

3. 证: (1) 依题知: $f(n), g(n)\geq 0$.

$\therefore \max\{f(n), g(n)\}\leq f(n)+g(n)$

$\max\{f(n), g(n)\}\geq \frac{f(n)+g(n)}{2}$

证明如下: 不妨设 $f(n)\geq g(n)$, $\max\{f(n), g(n)\}=f(n)$

$\Rightarrow \frac{1}{2}f(n)\geq \frac{1}{2}g(n) \Rightarrow f(n)\geq \frac{1}{2}[f(n)+g(n)]$.

$\parallel$

$\max\{f(n), g(n)\}$

当 $g(n)\geq f(n)$ 时同理.

$\therefore \frac{1}{2}[f(n)+g(n)]\leq \max\{f(n), g(n)\}\leq f(n)+g(n)$.

$\therefore$ 由 θ 表示法定义知: $\max\{f(n), g(n)\}=\theta(f(n)+g(n))$

(2) $\because a > b > 1$

$\therefore b^n \le a^n, \forall n \ge 1.$

$\therefore b^n = O(a^n)$

假设 $a^n = O(b^n)$ 即 $\exists N_0 \in N_+, c > 0.$ s.t. 当 $n > N_0$ 时有

$a^n \le c b^n$ 成立.

$\Rightarrow n > N_0, \left(\frac{a}{b}\right)^n \le c. \Rightarrow \left(\frac{a}{b}\right)^n$ 有界.

根据幂函数的性质 $\Rightarrow \frac{a}{b} \le 1 \Rightarrow a \le b.$

这与题设矛盾!

$\therefore a^n \neq O(b^n)$

综上, $b^n = O(a^n)$, $a^n \neq O(b^n)$.

证毕.

第二章作业.

1. 算法描述：

首先进行特判，若n=0或n=1则无需操作.

定义快慢指针，其中慢指针指向删除后新顺序表最后一个元素（起始时指向第一个元素）

快指针遍历顺序表，如果当前元素与慢指针当前指向的元素相同则快指针继续向前移动直至遇到不相同的元素，此时将慢指针后移至下一个元素，并将值赋给慢指针指向的对象. 如此循环直至快指针完成遍历.

伪代码：

```
void unique(int n){
    if(n==0||n==1)
        return;
    慢指针→顺序表第一个元素; 快指针→顺序表第二个元素;
    while(快指针没有遍历表){
        if(快指针指向元素!=慢指针指向元素){
            慢指针向后移动一个位置;
            慢指针指向元素=快指针指向元素;
        }
        快指针向后移动一个位置;
    }
    return;}
```

复杂度分析：空间上：算法只多开辟两个指针且数目与n无关，因此空间复杂度为O(1)；

时间上：算法通过一次遍历完成，故时间复杂度为O(n).

2. 算法描述：

用双指针实现单链表的翻转，后指针开始时指向空（假设链表的头结点即为第一个有效元素）前指针时刻指向后指针下一个元素，开始时先进行特判，若表为空则不用翻转，若表不为空，则前指针的 next 指针指向后指针，之后将前、后指针分别向后移动一个位置，继续进行该过程直至完成翻转（前指针为空）。

伪代码：

```
Listnode* reverse(*head){
    if(head==nullptr)
        return head;
    前指针 p=nullptr；后指针 q=头指针；
    while(q!=nullptr){
        指针 tmp=p->next;
        q->next=p;
        p=q;
        q=tmp;
    }
    return p; //最终 p 指向头结点。
}
```

复杂度分析：算法使用的附加单元有3个：头指针、尾指针以及用于维护节点的 tmp 指针，故空间复杂度为 O(1)；利用一次遍历完成翻转，故时间复杂度为 O(n)。

3. 算法描述：

利用快慢指针判断. 首先进行空表特判. 若表空，则没有环. 若表不为空，令快、慢指针同时指向表的起点. 快指针每次向后移动两个位置而慢指针每次向后移动一个位置. 若快指针追上了慢指针则说明链表中有环；若快指针走到了终点则说明没有环.

伪代码：

```
bool judge(){
    if(表为空)
        return false;
    慢指针 slow=头指针，快指针 fast=头指针;
    while(fast->next!=nullptr){
        fast=fast->next->next;
        slow=slow->next;
        if(fast==slow)
            return true;
    }
    return false;
}
```

复杂度分析：算法使用的附加单元为两个指针，故空间复杂度为O(1)；

而最多用一次遍历完成判断，故时间复杂度为O(n).