hw1

1.

假设对于常数 a 我们有 $f(n) = O(n \log_a n)$，那么根据定义我们可知

$$\exists\, c > 0,\ \ s.t.\ \ f(n) < c \cdot n \log_a n \quad \forall n >> 2$$

接着考虑另一个常数 b

$$\log_a n = \log_a b\ \log_b n$$

因此我们可以把第一个式子表示为

$$f(n) < c \cdot n \log_a b\ \log_b n$$

令 $c_1 = c \cdot \log_a b$，则可以得出 $f(n) = O(n \log_b n)$，即常数 a 取任意大于1的值均等价。

2.

```
int sum(int n) {//计算n=1+1×2+1×2×3+......+1×2×......×n
    int s = 0; //初始化最终结果，O(1)
    for(int i = 1; i ≤ n; ++i) {//全部n个元素循环，O(n)
        int p = 1;//初始化累乘器，O(1)
        for(int j = 1; j ≤ i; ++j)//循环，O(i)
            p *= j;//累乘O(1)
        s += p;//累加，O(1)
    }
    return s;//返回结果，O(1)
}
```

时间复杂度 $\mathcal{O}(1) + \mathcal{O}(n)\,(\mathcal{O}(1) + \mathcal{O}(i)\mathcal{O}(1) + \mathcal{O}(1)) + \mathcal{O}(1) = \mathcal{O}(n^2)$

(2)

```
int fac(int n) {//计算n=1+1×2+1×2×3+......+1×2×......×n
    int p = 1, s = 0;//初始化累乘器和累加器，O(1)
    for(int i = 1; i ≤ n; ++i) {//n次循环，O(n)
        p *= i;//累乘，O(1)
        s += p;//累加得到的累乘结果，O(1)
    }
    return s;//返回累加结果，O(1)
}
```

时间复杂度 $\mathcal{O}(1) + \mathcal{O}(n)(\mathcal{O}(1) + \mathcal{O}(1)) + \mathcal{O}(1) = \mathcal{O}(n)$

3.

(1)

```
int f(int n) {//计算斐波那契数列的第n个数
    if (n<=1) return 1;//判断是否到达递归基，是则返回1，O(1)
    else return f(n-1)+f(n-2);//否则递归计算前两项和
}
```

首先把 $\mathrm{f(n)}$ 花费的时间记作 $\mathrm{T(n)}$

$$\mathrm{T(n)}=\begin{cases}1, & \mathrm{n}\le 1\\ \mathrm{T(n-1)+T(n-2)+1}, & \text{other}\end{cases}$$

令 $\mathrm{S(n)=(T(n)+1)/2}$，则

$$\mathrm{S(n)}=\begin{cases}1, & \mathrm{n}\le 1\\ \mathrm{S(n-1)+S(n-2)}, & \text{other}\end{cases}$$

此时可以看到 $\mathrm{S(n)}$ 的递归式与斐波那契数列一致，初始值也一样。根据斐波那契数列的计算公式可知(这里计算公式要根据n=0和n=1时的值进行调整)

$$\mathrm{S(n)}=\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right)$$

$$\mathrm{T(n)=2S(n)-1}=\mathcal{O}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}\right)=\mathcal{O}(2^n)$$

(2)

```
int f(int n) { //计算斐波那契数列第n个数的值
    int a = 1;//初始化第一个数，O(1)
    int b = 1;//初始化第二个数，O(1)
    for (int i = 2; i<=n; ++i) {//从第三个数开始循环，O(n)
        int c = a + b;//计算新的斐波那契数列元素，O(1)
        a = b;//更新a为上一个斐波那契数列元素，O(1)
        b = c;//更新b为当前斐波那契数列元素，O(1)
    }
    return b;//返回第n个斐波那契数列元素，O(1)
}
```

时间复杂度为 $\mathcal{O}(1)+\mathcal{O}(1)+\mathcal{O}(n)(\mathcal{O}(1)+\mathcal{O}(1)+\mathcal{O}(1))+\mathcal{O}(1)=\mathcal{O}(n)$

(3)

假设这里延续上面两题的题设，认为 $\mathrm{fib}(1)=1$，$\mathrm{fib}(0)=1$

```
float power(double x, int l) {//计算x的l次方
    if (l==0) //如果l是0
```

```
        return 1;//直接返回1
    else {
        double temp = power(x, l / 2);//递归计算x的l/2次方
        temp = temp * temp; //temp平方
        if (l % 2 == 1) temp *= x;//如果l是奇数，则再乘上一个x
        return temp;//返回temp的值
    }
}

int f(int n) {//计算斐波那契数列第n项
    n += 1;//把n加1
    const double a = sqrt(5);//是斐波那契数列计算公式的根号5
    double temp1 = power((1+a)/2, n);//计算公式中的一项
    double temp2 = power((1-a)/2, n);//另一项
    double answer = (temp1-temp2) / a;//计算第n项
    return int(answer + 0.5);//四舍五入到最近的整数返回
}
```

$\mathrm{power(x,l)}$ 函数中，总的时间复杂度是 $\mathcal{O}(\log l)$

函数 $\mathrm{f(n)}$ 中，最主要的就是两次调用 $\mathrm{power(x,l)}$ 函数，其他都是 $\mathcal{O}(1)$ 的时间复杂度，因此 f 的时间复杂度是 $\mathcal{O}(\log n)$

4.

master theorem 大部分情况可以表示为：将原问题划分为 a 个规模为 n/b 的子任务，任务的划分、解的合并耗时 $\mathrm{f(n)}$，那么 $T(n) = a \cdot T(n/b) + \mathcal{O}(f(n))$

a. $T(n) = 4T(n/2) + n$

可以看出$a = 4$，$b = 2$，$f(n) = n$，满足 $\mathrm{f(n)} = \mathcal{O}(n^{log_b a - \epsilon})$，因此 $T(n) = \Theta(n^2)$

b. $T(n) = 2T(n/4) + \sqrt{(n)}$

$\mathrm{f(n)} = \Theta(n^{\log_b a} \log^k n)$，其中 $k = 0$，因此 $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(\sqrt{n} \log n)$

c. $T(n) = 3T(n/4) + n \lg n$

$\mathrm{f(n)} = \Omega(n^{\log_b a + \epsilon}) = \Omega(n^{\log_4 3})$，因此 $T(n) = \Theta(n \lg n)$

d. $T(n) = 9T(n/2) + O(n^3)$

$\mathrm{f(n)} = \mathcal{O}(n^{\log_b a - \epsilon}) = \mathcal{O}(n^{\log_2 9 - \epsilon})$，因此 $T(n) = \Theta(n^{\log_2 9})$

5.

证明：插入在位置 $r=1$ 的概率是 $1/n$，在位置 $r=1$ 插入需要移动 n 个元素，因此时间复杂度是 $\mathcal{O}(n)$；

插入在位置 $r=2$ 的概率是 $1/n$，在位置 $r=2$ 插入需要移动 $n-1$ 个元素，因此时间复杂度是 $\mathcal{O}(n-1)$

以此类推，插入在位置 $r=i$ 的概率是 $1/n$，在位置 $r=i$ 插入需要移动 $n-i+1$ 个元素，因此时间复杂度是 $\mathcal{O}(n-i+1)$

插入在位置 $r=n$ 的概率是 $1/n$，在位置 $r=n$ 插入时间复杂度是 $\mathcal{O}(1)$

那么平均时间复杂度为：

$$
\begin{aligned}
E(T(n)) &= \frac{1}{n}(\mathcal{O}(n)+\mathcal{O}(n-1)+\ldots\ldots+\mathcal{O}(1)) \\
&= \frac{1}{n}\mathcal{O}\left(\frac{n(n+1)}{2}\right) \\
&= \mathcal{O}\left(\frac{n+1}{2}\right) \\
&= \mathcal{O}(n)
\end{aligned}
$$