**第一课**

(1). 用伪代码写出求整数最大公因子的欧几里得算法。

递归法

Euclid

Input: 整数

Output: 的最大公因子

|  |  |
| --- | --- |
|  | if  then |
|  | return |
|  | if  then |
|  | return |
|  | return Euclid() |

循环法

Euclid

Input: 整数

Output: 的最大公因子

|  |  |
| --- | --- |
|  | while  do |
|  |  |
|  |  |
|  |  |
|  | return |

(2). 考虑如下伪代码

F(n)

1 if n=1

2 then return 1

3 else return n\* F(n-1)

请说明F(n)的功能

当为正整数时，计算

(3). 时间复杂度是否是衡量一个算法性能的主要标准，为什么？

(4). 包含死循环的程序是不是算法，为什么？

包含死循环的程序不是算法。

算法需要满足有穷性，即在有限步内停止。包含死循环的程序可能导致程序无法正常停止，不满足有穷性，因此不是算法。

(5). 是否可以通过提高算法的时间复杂性来降低其空间复杂性？

可以。

为了使计算机存储更少的数据，算法可以进行更多的计算来获取需要的数据，代价就是计算过程中多次访问有限已知数据的时间开销。例如归并排序和冒泡排序都是稳定排序，归并排序时间复杂度，空间复杂度，使用中间数组存储排序过程中的数据，时间复杂度较低；而冒泡排序时间复杂度，空间复杂度，没有使用中间数组，而是使用多次遍历的方式，空间复杂度较低但时间复杂度较高。

**第二课**

1. 证明或否证：*f* (*n*)+o(*f*(*n*))=Θ(*f*(*n*))

证明：





令

则



综上，*f* (*n*)+o(*f*(*n*))=Θ(*f*(*n*))

1. 证明：Θ(*f*(*x*)) + O(*g*(*x*)) = O( max(*f*(*x*), *g*(*x*))).

证明：







令

则





综上，

1. 证明或给出反例：Θ(*f*(*n*))∩o(*f*(*n*))=∅

证明：



假设

则

这与矛盾

因此

即

1. 证明：设*k*是任意常数正整数，则*logkn* = o(*n*)

证明：

|  |  |
| --- | --- |
|  | = |
|  | = |
|  | … |
|  | = |
|  | =0 |

所以

1. 证明：*logn*!=Θ(*nlogn*)

证明：

先证



再证





当时，



综上，

(6) 用迭代法解方程 T(n)＝T(9n/10)+n

T(n) = T(9n/10)+n

= T(92n/102)+n+n

= T(93n/103)+n+n+n

= …

= T(9kn/10k)+kn n=(10/9)k

= T(1)+kn

= T(1)+nlog(10/9)n

=

(7) 解方程 T(n)=6T(n/3)+logn

令n=2m，则

T(2m)=6T(2m/3)+m

令T(2m)=S(m)，则

S(m)=6S(m/3)+m

S(m)=

T(n)=

(8) 解方程 T(n) =3T(n/3 +5) + n/2

令n=m+15/2，则

T(m+15/2)=3T(m/3+15/2)+m/2+15/4

令T(m+15/2)=S(m)，则

S(m)=3S(m/3)+ m/2+15/4

a=3, b=3,

f(m)=m/2+15/4=

S(m)=

T(n)=

(9) 解方程 T(n)= T ( ⎡n/2⎤ )+1

T(n) = T ( ⎡n/2⎤ )+1

= T(⎡n/4⎤)+1+1

= T(⎡n/8⎤)+1+1+1

= …

= T(⎡n/2i⎤)+i

令 n/2i=1，得 i=logn

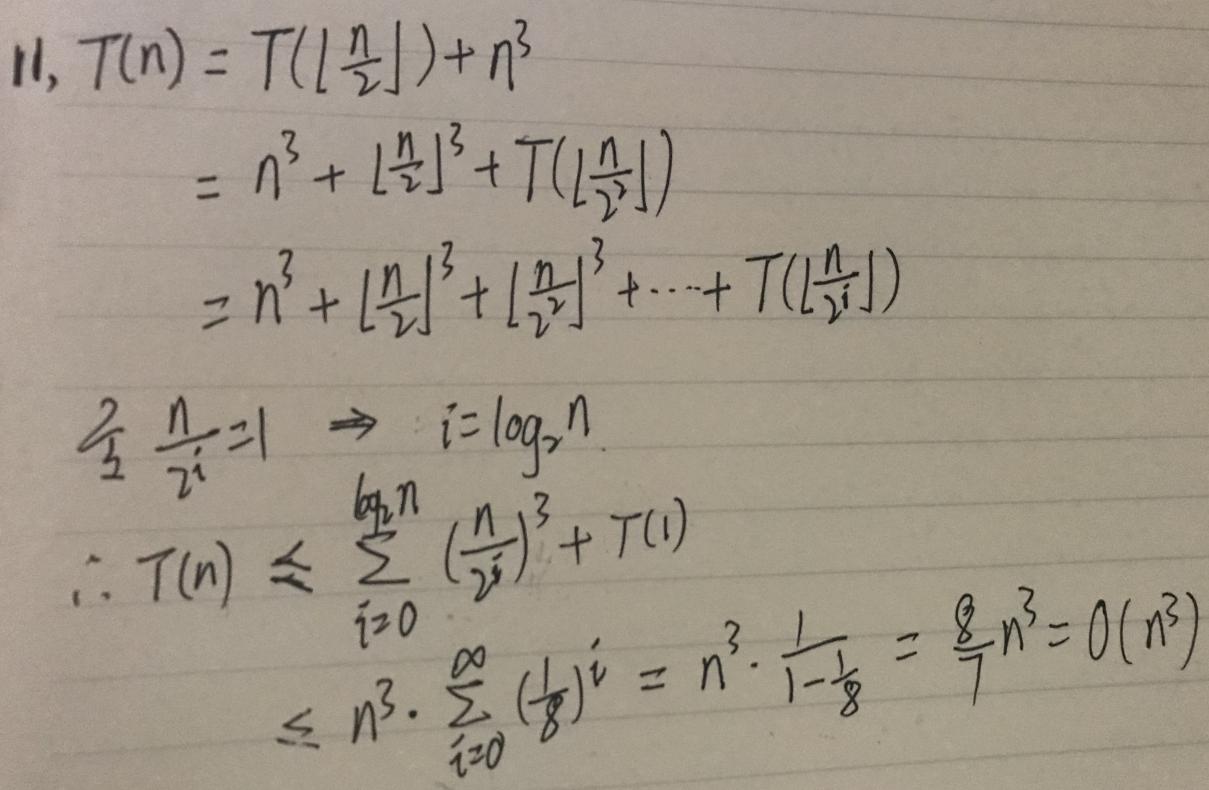
则原式 = T(1)+logn

= O(logn)

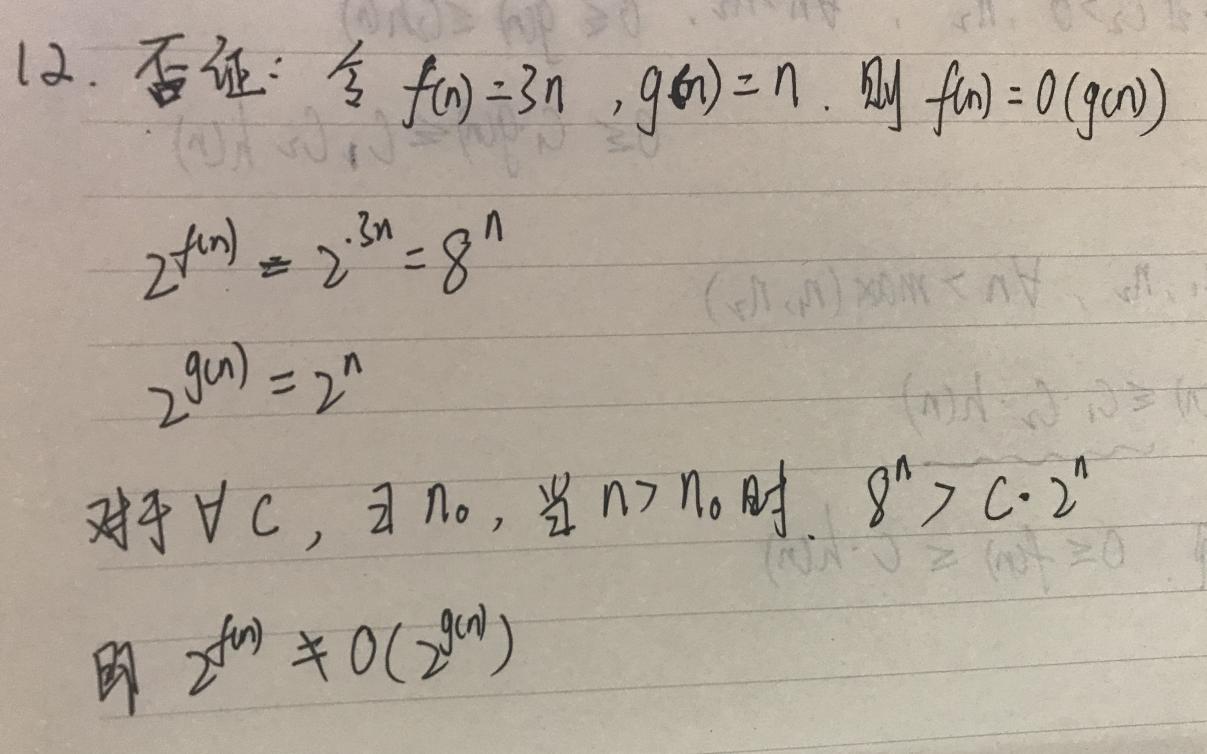
(10) 解方程 T(n)=9T(n/3)+n

a=9, b=3, f(n)=n,

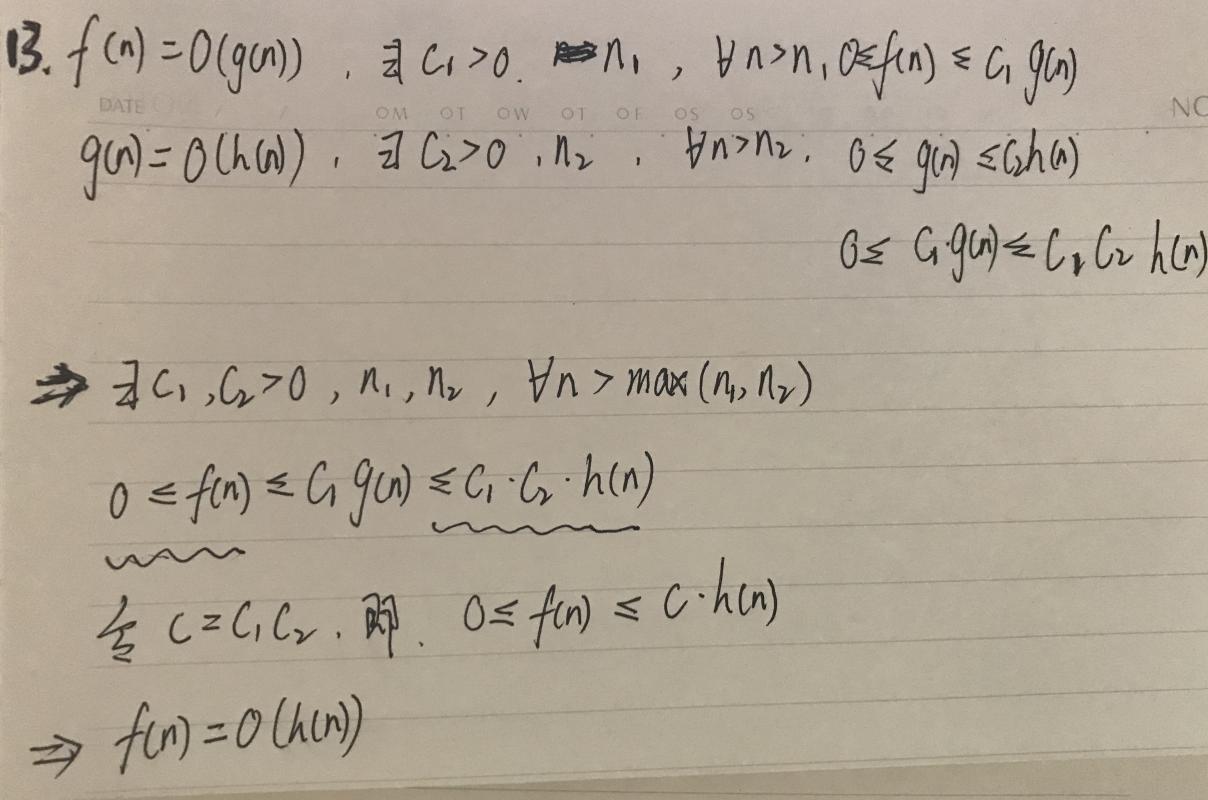
(11) 解方程 T(*n*)=T( ⎣*n*/2⎦ )+*n*3



(12) 证明或否证：如果*f*(*n*)=O(*g*(*n*))，那么2*f*(*n*)=O(2*g*(*n*))

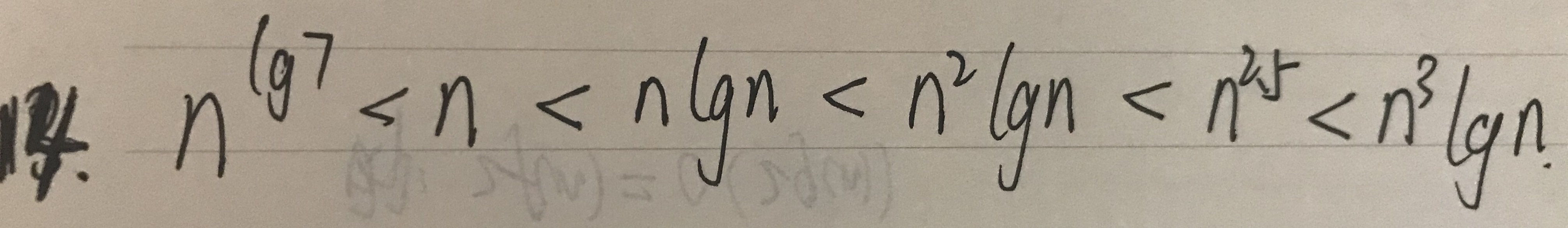


(13) 令f, g和h为定义在正整数上的正实数函数。假设f(n)=O(g(n))，g(n)=O(h(n))，证明f(n)=O(h(n)).



(14) 将下列函数按它们在n→∞时的无穷大阶数，从小到大排序。

*n*, *nlg*7, *n*2.5, *n*3*lgn*, *n*2*lgn*, *nlgn*



(15) 答：不一定。因为时间复杂性只是表示算法基本操作的次数，而不同算法的基本操作时间是不一样的。具体来说，时间复杂度为O(n2)的算法基本操作时间可能远小于时间复杂度为O(n)的算法，故运行时间不一定长。