1.1

We know that f(n) = Θ(g(n)) and g(n) = Θ(h(n))

as A and

as B

Simply, c1\*g(n) ≤ f(n) ≤ c2\*g(n) and c3\*h(n) ≤ g(n) ≤ c4\*h(n) when n>n0. We can combine the A and B, and we can get:

0 ≤ c1\*c3\*h(n) ≤ f(n) ≤ c2\*c4\*h(n) when n exceeds some point.

Let c5 = c1\*c3 and c6 = c2\*c4, and c5 and c6 both are constant

According to definition, f(n) = Θ(h(n)).

1.2

If f(n) = Θ(f()), then we have

Simply, when n>.

And we know that the , if f(n) = , and let g(n) = .

. To conclude, f(n) is not .

1.3

(a) for any positive constant k.

If k1:

When c=1, let ,

If the x is basic of log, and we can get: .

So, f(n) = . In other hand, .

Because , . So .

So, .

If k<1:

.

Because , . So .

So, .

In other hand, when c=1, let ,

If the x is basic of log, and we can get: .

So, f(n) =

Above all, whether k0 or k<1, f(n) = Θ(g(n))

(b)

In the f(n), the

However, the function is changed from time to time, we cannot know that what is the result when it limits to . g(n) = , let h(n)=f(n)-g(n) the range of h(n) is [, n], we cannot find a constant c to always make h(n) >= 0 or <=0. So there are not relationship between f(n) and g(n).

(c)

And log two sizes:

They have the same in log(n), so we just compare the n and

When , . To conclude, c = 1 and , f(n) = O(g(n)).

(d)

We can find that log(f(n)) equal to log(g(n)).

Therefore, f(n) = Θ(g(n)) and g(n) = Θ(f(n)).

(e)

f(n)=;

g(n)= (Sum arithmetic sequence)

if , we assume that ;

Solve this function

We can calculate the result, because we know that n is an integer, which means that n, f(n)g(n), c=1, so f(n) =

If c = , the function will lose the term to the third power.

Let the g(n) sizes equal to each other, we can get and solve:

So, when , c\*g(n)f(n), c =. f(n) =

Above all, f(n) = Θ(g(n))

2.1

T(n) = when and T(n) = 1

The Case 1, k 1:

T(n) +

……………………

The i is the times of the traversal, and the k is just a value.

Then we can solve the following equation:

let i = ,

T(n) =

The tight asymptotic for T(n) is when k

The Case 2, k=1:

T(n) +

……………………(k=1)

Then we can solve the following equation:

let i = ,

T(n) = +

The tight asymptotic for T(n) is when k

2.2

Through induction, we know that the time of the recursion is . For the base case, the if statement should be executed every time and complexity is 1, and the return statement will be executed only one time. In other case, if is an even, then it will take four complexity (if statement is 2 and the return statement is 2).

Else it will take five complexity (the if statement is 2 and return statement is 3).

int pow(int n) {

if(n == 0) --------------------------------------- 1

return 1; ---------------------------------- 1

int pmid = pow(n / 2); --------------------- 1, recursion times:

if(n % 2 == 0) { ------------------------------ 2

return pmid \* pmid; ------------------- 2

} else {

return 2 \* pmid \* pmid; --------------- 3

}

}

At first, if statement takes one time and the time of the recursion by the if is , return 1 is only one time. All steps are the even number case is smallest case (like n = 8), and 1 is odd, the later if statement is 1+2+3 = 6. The even time is , the complexity is 4+1 (assign pmid at a time). To conclude, the sum of them is + 8

All steps are the odd number case is smallest case (like n = 7). the later if statement is 1+2+3 = 6. There is not even time, the complexity is 4+1 (assign pmid at a time). To conclude, the sum of them is +9.

To conclude, the result is + 8 ≤ T(n) ≤ + 9.

2.3

We know that T(n) = Θ(log(n)). In the function, we can find that the n is not changed every times. It will divide to the left for a while and divide to the right for a while, and the loop times is .

We should also pay attention that the recursion is in the condition of the loop. So we can get that:

U(n)=\*(T(n)+O(1)).

\*)

)

So, ， c = .