Bài 1:

In biểu diễn nhị phân của một số nguyên.

void nhi\_phan(int n)

{

if(n>0)

{

nhi\_phan(n/2); // f(n) = n/2; a = 1

printf("%d",n%2); // g(n) = 1

}

}

Độ phức tạp của giải thuật trên:

Ta có: T(n) = T(n/2) + 1

= T(n/2.2) + 1 + 1

=…= lognT(1) + 1.logn

= 2.logn = O(logn).

Bài 2:

Phân tích một số nguyên thành tích các thừa số nguyên tố.

int phan\_tich(int n)

{

int d=2;

if(n==1)

return 0; // trường hợp suy biến

else // g(n) = g1(n) + g2(n) + g3(n) + g4(n)

{

while((n%d)!=0) // g1(n) = 1

d++; // g2(n) = 1

if (n/d == 1) // g3(n) = 1

printf("%d",d); // g4(n) = 1

else

printf("%d.", d); // g4(n) = 1

phan\_tich(n/d); // f(n) = n/d; a = 1

}

}

Độ phức tạp của thuật giải trên :

Ta có: T(n) = T() + 4

= T(.d) + 4 + 4

= … = logdn.T(1) + 4.logdn

= 5.logdn = 5.logn/logd = O(logn).

Bài 3:

Tìm số Fibonacci

int fibonacci( int n)

{

if(n==1 || n==2) return 1; // trường hợp suy biến

return (fibonacci(n-1) + fibonacci(n-2));

// f(n) = n – 1 và f(n) = n – 2 ; g(n) = 1

}

Độ phức tạp thuật giải trên là:

Ta có: T(n) = T(n – 1) + T(n – 2) + 1

= T(n – 2) + T(n – 3) + 1 + T(n – 3) + T(n – 4) + 1 + 1

= T(n – 2) + 2.T(n – 3) + T(n – 4) + 2 + 1

=…= T(3) + 2T(2) + T(1) + 2(n – 4 ) + 1

= T(2) + T(1) + 1 + 2T(2) + T(1) + 2(n – 4) + 1

= 3T(2) + 2T(1) + 2(n – 3)

= 5T(1) + 2(n – 3)

= 5 + 2n – 6 = 2n – 1 = O(n).

Bài 4:

Bài toán tháp Hà Nội

void chuyen\_thap(int n, char x, char t, char d)

{

if(n==1) // trường hợp suy biến

printf("%c -> %c\n",x,d);

else

{

chuyen\_thap(n-1,x,d,t); // f(n) = n - 1

chuyen\_thap(1,x,t,d); // g(n) = 1

chuyen\_thap(n-1,t,x,d); // a = 2

}

}

Độ phức tạp giải thuật trên là:

Ta có: T(n) = 2T(n – 1) + 1

= 2[2T(n – 2) + 1] + 1

=…= 2nT(1) + 2n-1 + …+1

= 2n + = 2n + 2n – 1 = 2.2n – 1 = O(2n).



f(x): g(x): n h(x): 2n