**CHƯƠNG 1: ĐỘ PHỨC TẠP CỦA GIẢI THUẬT**

**Câu 1:** Đánh giá độ phức tạp của đoạn code chạy trên C++ sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int search(int a[], int n, int x)  {  int i = 0;  while (i < n && a[i]!= x) {  i++;  if (i == n)  return -1;  }  return i;  } |

So sánh: 3

Gán: 1

Độ phức tạp: O(n)

**Câu 2:** Hãy đánh giá độ phức tạp của đoạn chương trình sau:

int Timkiem(int a[], int n, int x )

{

for(int i = 0; i < n ; i++)

{

if(a[i] == x)

return i;

}

return -1;

}

Độ phức tạp: O(n)

**Câu 3:** Hãy đánh giá độ phức tạp của đoạn chương trình sau:

*for (i = 0; i < n; i++)*

*for (j = 0; j < m; j++)*

*if (a[ i ][ j ] = = x) return 1;*

*return -1;*

Độ phức tạp: O(n2)

**Câu 4:** Hãy đánh giá độ phức tạp của đoạn chương trình sau:

*sum = 0;  
for( i = 0; i < n ; i++)*

*for( j = 0; j < i ; j++)*

*sum++;*

độ phức tạp: O(n2)

**Bài 5:** Đánh giá độ phức tạp của hàm tính dãy FIBONACCI sau

int Fibo(int n)

{

if(n <=1)

return n;

return Fibo (n-1)+Fibo(n-2)

}

**T(n)=** 1+ T(n-1)+T(n-2) **<** T(n-1)+T(n-1) **<** 2|T(n-1)| = 2 [T(n-2) + T(n-3)] **<** 2[T(n-2)+ T(n-2)]

**T(n)** < 2 \*2 \*T(n-2) = 2^i \* T(n-i)= 2^n \*T(n-n)= **2^n**