Câu 0:

Cho một mảng A có n phần tử số nguyên. Với n > 1.

Tìm hiệu lớn nhất khi lấy bất kỳ A[x] – A[y] với x > y.

Cho biết độ phức tạp. (hãy tìm thuật toán có độ phức tạp thấp nhất có thể)

int maxDiff(int A[], int n)

{

//…

}

Vd: Mảng A = [ 9, 1, 2, 8, ], n = 4

maxDiff(A,n) = 7 // bởi vì A[3] – A[1] = 8 -1 = 7

Câu 1: Tính độ phức tạp của các thuật toán sau:

a.

int a = 0, b = 0;

for (i = 0; i < N; i++) {

    a = a + rand();

}

for (j = 0; j < M; j++) {

    b = b + rand();

}

b.

int a = 0;

for (i = 0; i < N; i++) {

    for (j = N; j > i; j--) {

        a = a + i + j;

    }

}

c.

int a = 0, i = N;

while (i > 0) {

    a += i;

    i /= 2;

}

d.

int i, j, k = 0;

for (i = n / 2; i <= n; i++) {

    for (j = 2; j <= n; j = j \* 2) {

        k = k + n / 2;

    }

}

Câu 2: Tính độ phức tạp của giải thuật tìm kiếm nội suy (Interpolation Search):

def interpolationSearch(arr, n, x):

# Find indexs of two corners

lo = 0

hi = (n - 1)

# Since array is sorted, an element present

# in array must be in range defined by corner

while lo <= hi and x >= arr[lo] and x <= arr[hi]:

if lo == hi:

if arr[lo] == x:

return lo;

return -1;

# Probing the position with keeping

# uniform distribution in mind.

pos = lo + int(((float(hi - lo) /

( arr[hi] - arr[lo])) \* ( x - arr[lo])))

# Condition of target found

if arr[pos] == x:

return pos

# If x is larger, x is in upper part

if arr[pos] < x:

lo = pos + 1;

# If x is smaller, x is in lower part

else:

hi = pos - 1;

return -1