Bài 1: Đánh giá độ phức tạp bài toán

Câu 1. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

```
f=0;
for (i = 0 ; i<= n-1; i++){
     p = 1;
for (j = 1 ;j<n;j+=2)
    if(i!=j)
        p = p *(u-x[j])/(x[i]-x[j])+x[i];
f = f + p*y[i];
}
return f;
```

Câu 2. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

```
P = y[0] + t*D[0];
for(j=2;j<=n;j++){
for(i=0;i<=n-j;i+=2)
     D[i] = D[i+1] - D[i];
t = t*(t-j+1)/j;
P = P + t*D[0];
}
return P;
```

Câu 3. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

```
max = 0;
for(j=0;j<=n-1;j++){
tongcot = 0;
for (i = 1 ;i<n;i+=3)
     tongcot = tongcot + fabs(a[i][j]);
if(max<tongcot)
       max = tongcot;
}
return max;
```

Câu 4. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

```
for (i = 0 i<=n/2;i++){
posmin = i ;
for (j = i+1;j<=n-1;j++){
     if (A[p].Key > A[j].Key)
          posmin = j ;
     if (posmin == i ){
```

```
          tg = A[i] ;
          A[i] = A[p];
          A[p]= tg ;
     }
}
}
```

Câu 5. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

```
Found = 0;
i = 1;
j = n;
while(i<=j && !Found){
mid = (i+j) / 2;
if(k==S[mid])
Found = 1;
else if(k < S[mid])
     j = mid – 1;
else
     i = mid+1;
 }
return Found;
```

Câu 6. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

```
for (int i=1;i<=10;i++)
for(j=i;j<=n;j++)
if((a[i]+b[j])%2==0)
{
     if(a[i]>b[j]) return false;
}
return true;
```