PHÂN TÍCH THUẬT TOÁN

Câu 1

Chứng minh độ phức tạp tính toán sau đây:

a) $(n+1)^5$ thuộc $O(n^5)$
b) 2^{n+1} thuộc $O(2^n)$
c) $5n^2 + 3n\log n + 2n + 5$ thuộc $O(n^2)$
d) $3\log n + 2$ thuộc $O(\log n)$
e) $3n\log n – 2n$ thuộc $\Omega(n\log n)$
f) n^2 thuộc $\Omega(n\log n)$
g) $3n\log n + 4n + 5\log n$ thuộc $\Theta(n\log n)$

Câu 2

Sắp xếp tăng dần độ phức tạp thuật toán các hàm sau đây:

a)	$4n\log n + 2n$	2^{10}	$2^{\log n}$
b)	$n^2 + 10$	n^3	$n\log n$
c)	$4^{\log n}$	$4n$	$n^{1/\log n}$

Câu 3

Các định số lượng phép tính phù hợp và độ phức tạp thuật toán cho các đoạn code sau đây:

a)

```
def step_example1(n):
    i = 1
    count = 0
    while i < n:
        print(i)
        i *= 2
        count += 1
    return count
```

```
def step_example2(n):
    i = 1
    count = 0
    while i < n:
        print(i)
        i *= 3
        count += 1
    return count
```

b)

```
def sum_example1(S):
    n = len(S)
    total = 0
    for i in range(n):
        total += S[i]
    return total
```

```
def sum_example2(S):
    n = len(S)
    total = 0
    for i in range(0, n, 2):
        total += S[i]
    return total
```

```python
def sum_example3(S):
    n = len(S)
    total = 0
    for i in range(n):
        for j in range(1+i):
            total += S[j]
    return total
```

```python
def sum_example4(S):
    n = len(S)
    prefix = 0
    total = 0
    for i in range(n):
        prefix += S[i]
        total += prefix
    return total
```

c)

```python
def uniq_example1(S):
    n = len(S)
    for i in range(n):
        for j in range(i+1, n):
            if S[i] == S[j]:
                return False
    return True
```

```python
def uniq_example2(S):
    n = len(S)
    S_temp = sorted(S)
    for i in range(n-1):
        if S_temp[i] == S_temp[i+1]:
            return False
    return True
```

Câu 4

Đánh giá độ phức tạp thời gian của thuật toán sắp xếp chèn (Insertion Sort) sau đây trong các trường hợp tốt nhất (best case), tệ nhất (worst case) và trung bình (average case):

```python
def insertion_sort(S):
    n = len(S)
    for step in range(1, n):
        key = S[step]
        i = step - 1
        while i >= 0 and key < S[i]:
            S[i + 1] = S[i]
            i = i - 1
        S[i + 1] = key
    return S
```

Câu 5

(Mở rộng) - Chứng minh các độ phức tạp sau đây:

a) $\sum_{i=1}^{n} \log i$ thuộc O(nlogn)

b) $\sum_{i=1}^{n} \log i$ thuộc Ω(nlogn)

c) Giả sử $p(n) = \sum_{i=0}^{d} a_i n^i$, trong đó $a_d > 0$. Cho k là một hằng số. Chứng minh các tính chất sau:

(i) Nếu $k \geq d$, thì $p(n) = O(n^k)$

(ii) Nếu $k = d$, thì $p(n) = \Theta(n^k)$

RUBRIC

	Mức độ	Kiến thức	Đánh giá
Câu 1	1	Chứng minh độ phức tạp tính toán	Khả năng xác định được độ phức toán dựa vào các hàm số cho trước
Câu 2	2	Sắp xếp độ phức tạp tính toán	Xác định được độ phức tạp tính toán dựa vào các hàm cho trước Hiểu mối tương quan giữa chúng để so sánh độ phức tạp tính toán
Câu 3	2	Phân tích độ phức tạp tính toán các đoạn code	Phân tích các đoạn code Ước lượng số lượng các phép tính toán Đánh giá độ phức tạp tính toán
Câu 4	3	Phân tích độ phức tạp tính toán trong các trường hợp: tốt nhất, tồi nhất và trung bình	Phân tích các đoạn code Ước lượng số lượng các phép tính toán Đánh giá độ phức tạp tính toán trong các trường hợp tốt, tồi nhất và trung bình Hiểu thuật toán sắp xếp cơ bản đầu tiên: thuật toán sắp xếp chèn
Câu 5	4	Chứng minh độ phức tạp tính toán	Hiểu sâu hơn về xác định độ phức tạp thuật toán trong các ví dụ phức tạp hơn