PHÂN TÍCH THUẬT TOÁN

SOLUTION

Câu 1

Chứng minh độ phức tạp tính toán sau đây:

a) $(n+1)^5$ thuộc $O(n^5)$

$(n+1)^5 = n^5+5n^4+10n^3+10n^2+5n+1 \leq (1+5+10+10+5+1)n^5 = cn^5$

$\Rightarrow c=32, n_0=1$

b) 2^{n+1} thuộc $O(2^n)$
$2^{n+1} = 2^n.2 = c2^n \Rightarrow c=2, n_0=1$

c) $5n^2 + 3n\log n + 2n + 5$ thuộc $O(n^2)$
$5n^2+3n\log n+2n+5 \leq (5+3+2+5)n^2 = cn^2$
$\Rightarrow c=15, n_0=1$

d) $3\log n + 2$ thuộc $O(\log n)$
$3\log n + 2 = 3\log n + 2\log 1 \leq (3+2)\log n = c\log n$
$\Rightarrow c=5, n_0=2$

e) $3n\log n - 2n$ thuộc $\Omega(n\log n)$
$3n\log n - 2n = n\log n + 2n(\log n-1) \geq n\log n$
$\Rightarrow c=2, n_0=1$

f) n^2 thuộc $\Omega(n\log n)$
$n^2 = n.n \geq n\log n \Rightarrow c=1, n_0=1$

g) $3n\log n + 4n + 5\log n$ thuộc $\Theta(n\log n)$
$3n\log n \leq 3n\log n + 4n + 5\log n \leq (3+4+5)n\log n$
$\Rightarrow 3n\log n \leq 3n\log n + 4n + 5\log n \leq 12n\log n$
$\Rightarrow c_1=3, c_2=12, n_0=2$

Câu 2

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

a) $4n\log n + 2n$ $\quad$ 2^{10} $\quad$ $2^{\log n}$

$4n\log n + 2n$ is $O(n\log n)$ $\quad$ 2^{10} is $O(1)$ $\quad$ $2^{\log n}=n$ is $O(n)$

=> $4n\log n + 2n \leq 2^{\log n} \leq 2^{10}$

b) $n^2 + 10$ $\quad n^3$ $\quad n\log n$

$n^2 + 10$ is $O(n^2)$ $\quad n^3$ is $O(n^3)$ $\quad n\log n$ is $O(n\log n)$

=> $n\log n \leq n^2 + 10 \leq n^3$

c) $4^{\log n}$ $\quad 4n$ $\quad n^{1/\log n}$

$4^{\log n}$ is $O(n^2)$ $\quad 4n$ is $O(n)$ $\quad n^{1/\log n}$ is $O(1)$

=> $n^{1/\log n} \leq 4n \leq 4^{\log n}$

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
```

$T(n) = (c_2+c_3+c_4+c_5)\log n + (c_0+c_1+c_6)$
is $O(\log n)$

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

$T(n) = (c_2+c_3+c_4+c_5)\log_3 n + (c_0+c_1+c_6)$
is $O(\log_3 n)$

b)

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

$T(n) = (c_2+c_3)n + (c_0+c_1+c_2+c_4)$
is $O(n)$

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

$T(n) = (c_2+c_3)n/2 + (c_0+c_1+c_2+c_4)$
is $O(n)$

```
def sum_example3(S):
    n = len(S)
    total = 0
```

```
def sum_example4(S):
    n = len(S)
    prefix = 0
```

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

$T(n) = (c_3+c_4)n^2/2 + (2c_2+c_3+c_4)n/2 + (c_0+c_1+c_2+c_5)$
is $O(n^2)$

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

$T(n) = (c_3+c_4+c_5)n + (c_0+c_1+c_2+c_3+c_6)$
is $O(n)$

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
```

$T(n) = (c_2+c_3+c_4)n^2/2 + (2c_1-c_2-c_3-c_4)n/2 + (c_0+c_1+c_5)$
is $O(n^2)$

```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
```

$T(n) = c_1 n\log n\ (c_2+c_3+c_4)n + (c_0+c_5-c_3-c_4)$
is $O(n\log n)$

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
```

1. def insertion_sort(s):	cost	times
2. n = len(s)	c_0	1
3. for step in range(1, n):	c_1	n
4. key = s[step]	c_2	n-1
5. i = step – 1	c_3	n-1
6. while i >= 0 and key < s[i]	c_4	$\sum_{i=1}^{n-1} t_i$
7. s[i+1] = s[i]	c_5	$\sum_{i=1}^{n-1}(t_i-1)$
8. i = i – 1	c_6	$\sum_{i=1}^{n-1}(t_i-1)$
9. s[i+1] = key	c_7	n-1
10. return s	c_8	1

t_i is the number of times while loop test in line 6 is executed for that value of i

$T(n) = c_0 + c_1n + c_2(n-1) + c_3(n-1) + c_4\sum_{i=1}^{n-1} t_i + c_5\sum_{i=1}^{n-1}(t_i-1) + c_6\sum_{i=1}^{n-1}(t_i-1) + c_7(n-1) + c_8$

- Best case: already ordered numbers
  - $t_i=1$, line 7 and 8 will be executed 0 times
  - $T(n) = c_0 + c_1n + c_2(n-1) + c_3(n-1) + c_4(n-1)+ c_7(n-1) + c_8$
    $= (c_1 + c_2 + c_3 + c_4 + c_7)n + (c_0 + c_8 - c_2 - c_3 - c_4 - c_7)$
    $= cn + c'$
    $\Rightarrow O(n)$
- Worst case: reverse numbers
  - $t_i=i$, line 7 and 8 will be executed i times
  - $\sum_{i=1}^{n-1} t_i = \sum_{i=1}^{n-1} i = n(n+1)/2-1$, and $\sum_{i=1}^{n-1}(t_i-1) = \sum_{i=1}^{n-1}(i-1) = n(n-1)/2$
  - $T(n) = c_0 + c_1n + c_2(n-1) + c_3(n-1) + c_4 (n(n+1)/2-1) + c_5n(n-1)/2$
    $+ c_6n(n-1)/2 + c_7(n-1) + c_8$
    $= an^2 + bn + c$
    $\Rightarrow O(n^2)$
- Average case: random numbers
  - $t_i = i/2 \Rightarrow$ The same worst case: $O(n^2)$

Câu 5

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

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

$\sum_{i=1}^{n} \log i = \log 1 + \log 2 + \ldots + \log n = \log(1*2*\ldots*n)$

$\leq \log(n*n*\ldots*n) = n\log n$

$\Rightarrow \sum_{i=1}^{n} \log i$ is $O(n\log n)$

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

$\sum_{i=1}^{n} \log i \geq \sum_{i=n/2}^{n} \log i = \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2}+1\right) + \ldots + \log(n)$

= logn – log2 + log(n+3) – log2 + … + logn

≥ logn – log2 + logn – log2 + … + logn

= n/2logn – (n/2-1)log2

=> $\sum_{i=1}^{n}$ logi is Ω(nlogn)

=> $\sum_{i=1}^{n}$ logi is Θ(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 ≥ d, thì p(n) = $O(n^k)$

p(n) is $O(n^k)$, if k ≥ d

$\sum_{i=0}^{d} a_i n^i \le cn^k$
divide n^k => $\sum_{i=0}^{d} a_i n^{i-k} \le c$
k >= d => i-k ≤ 0 => $n^{i-k} \le 1$
=> Choose c = $\sum_{i=0}^{d} a_i$

(ii) Nếu k = d, thì p(n) = $\Theta(n^k)$
p(n) is $\Theta(n^k)$, if k = d
$c_1 n^k \le p(n) \le c_2 n^k$
Note (i) => $c_2 = \sum_{i=0}^{d} a_i$
Find c_1
$c_1 n^k \le \sum_{i=0}^{d} a_i n^i$
divide n^k => $c_1 \le \sum_{i=0}^{d} a_i n^{i-k}$
k = d => i-k ≤ 0 => $n^{i-k} \le 1$
=> Choose $c_1 = a_d$
=> $c_1 = a_d$, $c_2 = \sum_{i=0}^{d} a_i$

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