Solutions 1

(a) $\log_2 n^2 + \log_2 2 = \log_2 2n^2 \Rightarrow \log_2 2n^2 \leq c.n$

* Big O upper bound

$\Rightarrow 2n^2 \leq 2^{c.n}$ where $c=2$ for all $n \geq 0$ $\boxed{\text{true}}$

(b) * omega lower bound

$c.n \leq \sqrt{n}\sqrt{n+1} \Rightarrow c.\sqrt{n} \leq \sqrt{n+1}$ where $c=1$ for all $n \geq 0$ $\boxed{\text{true}}$

(c) if and only if $n^{n-1} \in \Omega(n^n)$, $n^{n-1} \in O(n^n)$

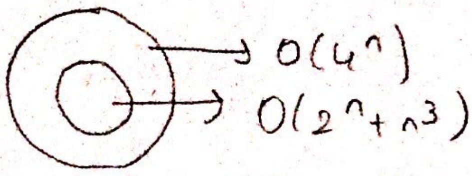
$n^n.c_1 \leq n^{n-1} \leq n^n.c_2$

for the omega approximation, c_1 must be 0 or less.
The omega approximation does not matter if c_1 takes a value of 0 or less $\Rightarrow$ $\boxed{\text{So it is false}}$

CamScanner ile tarandı

(d) $O(2^n + n^3) \subset O(4^n)$

$\Rightarrow$



$\Rightarrow c_1(2^n + n^3) \leq c_2 4^n$

$=$ if $\lim\limits_{n\to\infty} \dfrac{4^n}{2^n + n^3} = \infty$, then the expression is $\boxed{\text{true}}$

$\Rightarrow \lim\limits_{n\to\infty} \dfrac{4^n}{2^n+n^3} = \lim\limits_{n\to\infty} \dfrac{2^n}{\frac{1}{2^n}(2^n+n^3)}$

$\Rightarrow \lim\limits_{n\to\infty} \dfrac{2^n}{\left(1+\frac{n^3}{2^n}\right)} = \dfrac{\lim\limits_{n\to\infty} 2^n}{\lim\limits_{n\to\infty}\left(1+\frac{n^3}{2^n}\right)} = \dfrac{\infty}{\lim\limits_{n\to\infty} 1 + \lim\limits_{n\to\infty}\frac{n^3}{2^n}} = \dfrac{\infty}{1+\lim\limits_{n\to\infty}\frac{n^3}{2^n}}$

Rule of L'hopital

$\lim\limits_{n\to\infty} \dfrac{n^3}{2^n} = \lim\limits_{n\to\infty} \dfrac{3n^2}{\ln 2 . 2^n} = \lim\limits_{n\to\infty} \dfrac{6n}{\ln^2 2 . 2^n} = \lim\limits_{n\to\infty} \dfrac{6}{\ln^3 2 . 2^n}$

$= \lim\limits_{n\to\infty} \dfrac{0}{\ln^4 2 . 2^n}$

$\boxed{=0}$

$\dfrac{\infty}{0+1} = \infty$

the expression that is $O(2^n+n^3) \subset O(4^n)$ is true

(e) $O(2\log_3 \sqrt[3]{n}) \subset O(3\log_2 n^2)$

$c_1 \cdot 2\log_3 \sqrt[3]{n} \leq 3\log_2 n^2 \cdot c_2$

$\frac{2 \cdot c_1}{3} \log_3 n \leq 6 c_2 \log_2 n$

$2 c_1 \log_3 n \leq 18 c_2 \log_2 n$ $\Big\rangle$ $c_1 \log_3 n \leq 9 c_2 \log_2 n$

$= c_1 . 2^n \leq 9 c_2 3^n$

$= c_1 . 2^n \leq c_2 . 3^{n+2}$

where $c_1 = 1$ $c_2 = 1$

for all $n \geq 1$ $\boxed{\text{true}}$

(f) $\log_2 \sqrt{n}$ and $(\log_2 n)^2$ are of the same asymptotical order

(f.1)

$\log_2 \sqrt{n} = \frac{1}{2}\log_2 n = O(\log n) = \Omega(\log n) = \Theta(\log n)$

$\frac{1}{2}\log_2 n \leq c . \log_2 n$

where $c = \frac{1}{2}$

for all $n \geq 1$ true

$c . \log_2 n \leq \frac{1}{2}\log_2 n$

where $c = \frac{1}{2}$

for all $n \geq$ true

if and only if $O(\log n) = \Omega(\log n)$ true

(f.2) $\log_2 n . \log_2 n = O(\log n . \log n) = \Omega(\log n . \log n) = \Theta(\log n \log n)$

$\boxed{\text{It is not true}}$ Because $\log_2 \sqrt{n}$ $(\log_2 n)^2$ are not same asymptotical order. $\boxed{\text{false}}$

Solutions 2

n^2, n^3, $n^2 \log n$, $\sqrt{n}$, $\log n$, 10^n, 2^n, $8^{\log n}$

ordered situations (great to small)

10^n, 2^n, $n^3 = 8^{\log n}$, $n^2 \log n$, n^2, $\sqrt{n}$, $\log n$

Proof

In this order, I will compare two by two, if there is no problem, they are ordered correctly.

① 10^n vs 2^n

$10^n = 2^n . 5^n$

$2^n . 5^n \geq 2^n$

$5^n \geq 1$

$\forall n \geq 0$

② 2^n vs n^3

*exponential growth rate greater than cubic growth rate

$$\lim_{n \to \infty} \frac{2^n}{n^3} \xrightarrow{L'H} \lim_{n \to \infty} \frac{\ln 2 . 2^n}{3n^2}$$

$$\xrightarrow{L'H} \lim_{n \to \infty} \frac{\ln^2 2 . 2^n}{6n} \xrightarrow{L'H} \frac{\ln^3 2 . 2^n}{6} = \infty$$

③ n^3 vs $8^{\log n}$

$n^3 = 8^{\log_2 n}$

$n^3 = 2^{3 \log_2 n}$

$n^3 = \cancel{2}^{\log_2 n^3}$

$\Rightarrow n^3 = n^3$

④ n^3 vs $n^2 \log n$

$n^3 \geq n^2 \log n$

$n \geq \log n$ ✓

$\forall n \geq 1$

⑤ $n^2 \log n$ vs n^2

$n^2 \log n \geq n^2$

$\log n \geq 1$

$\forall n \geq 2$

⑥ n^2 vs $\sqrt{n}$

$n^2 \geq n^{\frac{1}{2}}$

$\forall n \geq 0$

⑦ $\sqrt{n}$ vs $\log n$

$$\sqrt{n} \geq \log n \Rightarrow \lim_{n \to \infty} \frac{\sqrt{n}}{\log n} = \lim_{n \to \infty} \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n \ln 2}} = \lim_{n \to \infty} \frac{1}{2} \ln 2 \sqrt{x} = \infty$$

CamScanner ile tarandı

Solution 3

a)
```
void f(int my_array[]){
    for(int i = 0 ;  i< sizeOfArray ; ++i){
        if(my_array[i] < first_element){
            second_element = first_element);
            first_element = my_array[i];
        }
        else if (my_array[i] < second_element){
            if(my_array[i] != first_element){
                second_element = my_array[i];
            }
        }
    }
}
```

* It is the sizeofArray that determines the number of iterations of the for loop.

This sizeofArray value does not chage within the loop. The number of iteration of the loop will only increase or decrease according to sizeof Array.

So total time complexity is O(sizeofArray)

b)
```
void f(int n){
    int count = 0;
    for(int i = 2; i <= n; ++i){
        if(i % 2 == 0)
            ++count;
        else
            i = (i-1) * i;
    }
}
```

* Let's watch the growth rate of i value

i = 2, $\boxed{i=3,\ i=6}$ $\boxed{i=7,\ i=42}$ $\boxed{i=43,\ i=42*43}$ . . .

1 iterate   1 iterate   1 iterate . . .

Could the actual increase of i at each iterate be like this?

i = 2, i = 6, i = 42 . . . $i = i^2 + i$

So the loop can be:

```
void f(int n){
    int count = 0;
    for(int i = 2; i <= n; i = i^2 + i)
        ++count;
}
```

i values for $i = i^2 + i$ (ignored $+i$)

$2^1, 2^2, 2^4, 2^8 \ldots 2^{2^x} = n$

$x=0 \quad x=1 \quad x=2 \quad x=3 \ldots$

$\boxed{\text{! } x \text{ is number of iterate}}$

$2^{2^x} = n$

$2^x = \log n$

$\boxed{x = \log\log n}$

$\boxed{O(\log\log N)}$

total time complexity of function

Solution 4

(4) $\underbrace{\sum_{i=1}^{n} i^2 \log i}_{\text{increases}} \Rightarrow \int_0^n i^2 \log i \, di \leq f(n) \leq \int_1^{n+1} i^2 \log i \, di$

$\Rightarrow \underbrace{\frac{i^3(3\ln i - 1)}{9} \Big|_0^n}_{} \leq f(n) \leq \underbrace{\frac{i^3(3\ln i - 1)}{9} \Big|_1^{n-2}}_{}$

undefined because logarithm does not take "0"

$\frac{(n+1)^3(3\ln(n+1) - 1)}{9} - \frac{2}{9}$

$n^3 \ldots$ $3\ln \ldots$

$f(n) \in O(n^3 \log n)$

For lower bound

$f(n) = \sum_{i=2}^{n} i^2 \log i + 1 \rightarrow 1 + \int_1^n i^2 \log i \, di \leq f(n)$

$= 1 + \frac{n^3(3\ln n - 1)}{9} \leq f(n)$

$\boxed{f(n) = \Theta(n^3 \log n)} \leftarrow f(n) \in \Omega(n^3 \log n)$

(5) $\underbrace{\sum_{i=1}^{n} i^3}_{\text{increases}} \Rightarrow \int_0^n i^3 \, di \leq f(n) \leq \int_1^{n+1} i^3 \, di$

$= \frac{i^4}{4} \Big|_0^n \leq f(n) \leq \frac{i^4}{4} \Big|_1^{n+1}$

$= \frac{i^4}{4} \leq f(n) \leq \frac{(n+1)^4 - 1}{4}$

$\left.\begin{array}{l} f(n) \in O(n^4) \\ f(n) \in \Omega(n^4) \end{array}\right\} \boxed{f(n) \in \Theta(n^4)}$

ⓒ $\underbrace{\sum_{i=1}^{n} \frac{1}{2\sqrt{i}}}_{\text{decreases}} \Rightarrow \int_{1}^{n+1} \frac{1}{2\sqrt{i}} \leq f(n) \leq \int_{0}^{n} \frac{1}{2\sqrt{i}}\, di$

$$= \frac{\sqrt{i}}{4} \Big|_{1}^{n+1} \leq f(n) \leq \frac{\sqrt{i}}{4} \Big|_{0}^{n}$$

$$= \frac{\sqrt{n+1} - 1}{4} \leq f(n) \leq \frac{\sqrt{n}}{4}$$

$$\left.\begin{array}{l} f(n) \in O(\sqrt{n}) \\ f(n) \in \Omega(\sqrt{n}) \end{array}\right\} \boxed{f(n) \in \Theta(\sqrt{n})}$$

ⓓ $\underbrace{\sum_{i=1}^{n} \frac{1}{i}}_{\text{decreases}} \Rightarrow \int_{1}^{n+1} \frac{1}{i}\, di \leq f(n) \leq \int_{0}^{n} \frac{1}{i}\, di$

$$= \underbrace{\ln i \Big|_{1}^{n+1}}_{} \leq f(n) \leq \underbrace{\ln i \Big|_{0}^{n}}_{}$$

$\ln(n+1) \leq f(n)$

undefined because logarithm function does not take "0"

$\boxed{f(n) \in \Omega(\log n)}$

Upper Bound.

$f(n) = \sum_{i=2}^{n} \frac{1}{i} + 1 \Rightarrow f(n) \leq 1 + \int_{1}^{n} \frac{1}{i}\, di$

$= f(n) \leq 1 + \ln i \Big|_{1}^{n}$

$= f(n) \leq 1 + \ln n$

$\boxed{= f(n) \in O(\log n)}$

$\boxed{f(n) \in \Theta(\log n)}$

CamScanner ile tarandı

Solutions 5

The python3 code of the algorithm mentioned is as follows

```
def search(list, target):
    for i in range(len(list)):
        if list[i] == target:
            return True
    return False
```

Worst Case Complexities

When target is not represent, then search() method compare tarteg with all the elements of list one by one
Therefore, the worst case time complexity of linear search be $\Theta(n)$

Best Case Complexities

In the linear search problem, the best case occurs when target is present at the first location. So time complexity in the best case would be $\Theta(1)$