Lab Assignment No 1 - Introduction and Advanced Data Structures

Nihar Sodhaparmar - P22CS013

September 2022

PART - B

(a) IF $f(n) = 100 * 2^n + 8n^2$, prove that $f(n) = O(2^n)$. Can you claim that $f(n) = \Theta(2^n)$. IF so, prove the same.

Answer:

$$\begin{aligned} f(n) &= 100 * 2^n + 8n^2 \\ &\leq 100 * 2^n + 8 * 2^n \\ &\leq 108 * 2^n \\ &= O(n^2), \text{where } c = 108 \text{ and } \forall n > 0 \end{aligned}$$

(b) Is it correct to say that $f(n) = 3n + 8 = \Omega(1)$?. Given the facts that $f(n) = 3n + 3 = \Omega(n)$ and $f(n) = 3n + 3 = \Omega(1)$, which one is correct? Which one would you choose to prescribe the growth rate of f(n) ?

Answer:

(i) $f(n) = 3n + 8 = \Omega(1)$ is correct.

(ii) $f(n) = 3n + 8 = \Omega(1)$ and $f(n) = 3n + 3 = \Omega(n)$ both are correct.

$$\begin{aligned} f(n) &= 3n + 8 \\ &\geq 3n \\ &= \Omega(n), \quad \text{where } c = 3 \text{ and } \forall n > 0 \end{aligned}$$

(iii) $f(n) = 3n + 3 = \Omega(n)$ is use for showing growth rate because it is tight lower bound.

(c) Consider the two functions viz. $f(n) = n^2$ and $g(n) = 2n^2$. Which functions growth rate is higher? Use appropriate asymptotic notation to specify the time complexity of the two functions.

Answer:

$$f(n) = n^2 = O(n^2), \quad \text{where } c = 1 \text{ and } \forall n > 0$$
$$g(n) = 2n^2 = O(n^2), \quad \text{where } c = 2 \text{ and } \forall n > 0$$

So, both functions have same big O complexity and growth rate is same.

(d) Prove the following: For any two functions $f(n)$ and $g(n)$, $f(n) = \Theta(g(n))$ only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Answer:

(i) $f(n) = O(g(n))$
$f(n) \leq c_1 g(n), \quad \forall n > n_1$

(ii) $f(n) = \Omega(g(n))$
$c_2 g(n) \leq f(n), \quad \forall n > n_2$

from (i) and (ii),
$c_1 g(n) <= f(n) <= c_1 g(n), \quad \forall n > max(n_1, n_2)$
therefore, $f(n) = \Theta(g(n))$

(e) Solve the following problems:

(i) Show that $T(n) = 1 + 2 + 3 + \ldots..n = \Theta(n^2)$
Answer:

$$n/2 + n/2 + n/2 + \ldots..n/2 \leq T(n) \leq n + n + n + \ldots..n$$
$$n * n/2 \leq T(n) \leq n * n$$
$$n^2/2 \leq f(n) \leq n^2$$

therefore $f(n) = \Theta(n^2)$, $\quad$ where $c_1 = 1/2$ and $c_2 = 2$ $\forall n > 0$

(ii) Prove or disprove: $2n^3 - n^2 = O(n^3)$
Answer:

$$\begin{aligned} f(n) &= 2n^3 - n^2 \\ &\leq 2n^3 \\ &= cn^3, \text{where } c = 2 \text{ and } \forall n > 0 \\ &= O(n^3) \end{aligned}$$

(iii) Prove that $7n^2logn + 25000n = O(n^2logn)$

Answer:

$$\begin{aligned} f(n) &= 7n^2logn + 25000n \\ &\leq 7n^2logn + 25000n^2logn \\ &\leq 25007n^2logn \\ &= O(n^2logn), \text{where } c = 25007 \text{ and } \forall n > 0 \end{aligned}$$

(f) If $T1(n) = O(f(n))$ and $T2(n) = O(g(n))$ then show that (a) $T1(n) + T2(n) = max(O(g(n), O(f(n))$ (b) $T1(n) * T2(n) = O((g(n) * (f(n))$.

Answer:

$T1(n) = O(f(n)) \leq c_1 f(n) \ldots (i)$
$T2(n) = O(g(n)) \leq c_2 g(n) \ldots (ii)$

(a) $T1(n) + T2(n) = max(O(g(n), O(f(n))$

$$\begin{aligned} T1(n) + T2(n) &\leq c_1 f(n) + c_2 g(n), \quad \text{where } n_0 = max(n_1, n_2) \\ &\leq c_3 f(n) + c_3 g(n), \quad \text{where } c_3 = max(c_1, c_2) \\ &\leq 2c_3 max(f(n), g(n)) \\ &\leq cmax(f(n), g(n)) \\ &= O(max(f(n), g(n))) \end{aligned}$$

(b) $T1(n) * T2(n) = O((g(n) * (f(n))$

$$\begin{aligned} T1(n) * T2(n) &\leq c_1 c_2 f(n) * g(n) \\ &\leq cf(n) * g(n) \\ &= O(f(n) * g(n)) \end{aligned}$$

(g) Show that $max\{f(n), g(n)\} = \Theta(f(n) + g(n))$

Answer:

First prove, $maxf(n), g(n) = \Omega(f(n) + g(n))$

$$\begin{aligned} max(f(x), g(x)) &\geq g(x) \ldots (i) \\ max(f(x), g(x)) &\geq f(x) \ldots (ii) \\ \text{Now (i) + (ii),} \\ 2max(f(x), g(x)) &\geq f(x) + g(x) \\ max(f(x), g(x)) &\geq 1/2(f(x) + g(x)) \\ max(f(x), g(x)) &= \Omega(f(x) + g(x)) \ldots (I) \end{aligned}$$

Now prove, $maxf(n), g(n) = O(f(n) + g(n))$

$$f(x) \leq f(x) + g(x) \text{ and } g(x) \leq f(x) + g(x)$$
$$f(x) = O(f(x) + g(x)) \text{ and } g(x) = O(f(x) + g(x))$$

therefore, $max(f(x), g(x)) = O(f(x) + g(x)) \dots (II)$

from (I) and (II),
$max\{f(n), g(n)\} = \Theta(f(n) + g(n))$

(h) Prove or disprove: (a) $n^2 2^n + n^{100} = \Theta(n^2 2^n)$ (b) $n^2/logn = \Theta(n^2)$

Answer:

(a) Given, $n^2 2^n + n^{100} = \Theta(n^2 2^n)$, we need to find c_1 and c_2
such that $c_1 * n^2 2^n \leq f(n) \leq c_2 * n^2 2^n$ where $f(n) = n^2 2^n + n^{100}$
So that for $c_1 = \frac{1}{2}$ and $c_2 = 2$ above inequality holds.
Hence proved, $n^2 2^n + n^{100} = \theta(n^2 2^n)$

(b) Given, $\frac{n^2}{\log(n)} = \Theta(n^2)$, we need to find c_1 and c_2
such that $c_1 * n^2 \leq f(n) \leq c_2 * n^2$ where $\frac{n^2}{\log(n)} = \Theta(n^2)$
But for $n \geq n_0 (= 1)$ no such c_1 and c_2 exist.
Hence we can say, $n^2 2^n + n^{100} \neq \Theta(n^2 2^n)$

(i) Prove that if T(x) is a polynomial of degree n, then $T(x) = \Theta(x^n)$.

Answer:

$$\begin{aligned} T(x) &= a_0 + a_1 x + a_2 x^2 + \dots\dots\dots + a_m x^m \\ &\leq a_0 x^m + a_1 x^m + a_2 x^m + \dots\dots\dots + a_m x^m \\ &\leq (a_0 + a_1 + a_2 + \dots\dots\dots + a_m) x^m \\ &= cx^m \\ &= O(x^m) \end{aligned}$$

$$\begin{aligned} T(x) &= a_0 + a_1 x + a_2 x^2 + \dots\dots\dots + a_m x^m \\ &\geq a_m x^m \\ &= cn^m \\ &= \Omega(n^m) \end{aligned}$$

So, we can say that $\Theta(x^m)$

(j) If P(n) is any polynomial of degree m or less then show that $P(n) = a^0 + a^1n + a^2n^2 + ... + a^mn^m$ then $P(n) = O(n^m)$.

Answer:

$$
\begin{aligned}
P(n) &= a_0 + a_1n + a_2n^2 + ... + a_mn^m \\
&\leq a_0n^m + a_1n^m + a_2n^m + ... + a_mn^m \\
&\leq (a_0 + a_1 + a_2 + ... + a_m)n^m \\
&= cn^m \\
&= O(n^m)
\end{aligned}
$$

(k) Find the running time of the following algorithm in terms of the asymptotic notations:

```
Algorithm SUM( n )
1 .   answer = 0 ;
2 .   for i= 1 t o n do
3 .     for j= 1 t o i do
4 .       for k = 1 t o j do
5 .         answer++;
6 .   print(answer);
```

Answer:

Line 2 runs n times
Line 3 runs $n * n$ times
Line 4 runs $n * n * n$ times
So, running time of algorithm is $O(n^3)$

(l) Let A and B be two programs that perform the same task. Let $t_{A(n)}$ and $t_{B(n)}$ respectively denote their values. For each of the following pairs, find the range of n value for which program A is faster than program B :

(i) $t_{A(n)} = 1000n$ and $t_{B(n)} = 10n^2$

$$
\begin{aligned}
1000n &< 10n^2 \\
1000n - 10n^2 &< 0 \\
n(1000 - 10n) &< 0 \\
1000 - 10n &< 0 \\
n &> 100
\end{aligned}
$$

So that, $n \in (100, \infty)$

(ii) $t_{A(n)} = 1000nlog_2n$ and $t_{B(n)} = n^2$

$$\begin{aligned}
1000nlog_2n &< n^2 \\
1000nlog_2n - n^2 &< 0 \\
n(1000log_2n - n) &< 0 \\
1000log_2n &< n \\
log_2n^{1000} &< n \\
n^{1000} &< 2^n \\
2^n - n^{1000} &> 0
\end{aligned}$$

So that, $n \in (0,)$

(iii) $t_{A(n)} = 2n^2$ and $t_{B(n)} = n^3$

$$\begin{aligned}
2n^2 &< n^3 \\
2n^2 - n^3 &< 0 \\
n^2(2 - n) &< 0 \\
n &> 2
\end{aligned}$$

So that, $n \in (2, \infty)$

(iv) $t_{A(n)} = 2n$ and $t_{B(n)} = 100n$

$$\begin{aligned}
2n &< 100n \\
2n - 100n &< 0 \\
-98n &< 0
\end{aligned}$$

So that, A is always faster than B.

(m) Consider an input array A of n elements. Each element is an n-bit integer except 0. Which sorting algorithm would you recommend for sorting the array ? Why ? What will be the complexity your sorting algorithm ? [Hint: What is the range in which each array value (i.e. a number) i.e. an integer falls into ?]

Answer:

(n) Given the following statement viz. Consider an input array a[1..n] of arbitrary numbers. It is given that the array has only $O(1)$ distinct elements. What does the statement imply?

Answer:

The statement shows that no matter what size of array is, the array has only fixed number of distinct constant element.