CS 170 Dis 0

Released on 2017-08-27

1 Asymptotic Bound Practice

Prove that for any $\epsilon > 0$ we have $\log x \in O(x^\epsilon)$.

$\lim_{x\to\infty} \frac{x^\epsilon}{\log x} = \lim_{x\to\infty} \frac{\epsilon x^{\epsilon-1}}{\frac{1}{x}} = \lim_{x\to\infty} \epsilon x^\epsilon = \infty$ → as we can see x^ϵ grows faster than $\log x$

using l'Hopital Rule

✓

- therefore x^ϵ is upper bound of $\log x$ for any $\epsilon > 0$

$\Rightarrow \log x \in O(x^\epsilon)$

2 Bounding Sums

Let $f(\cdot)$ be a function. Consider the equality

$$\sum_{i=1}^{n} f(i) \in \Theta(f(n)),$$

Give a function f_1 such that the equality holds, and a function f_2 such that the equality does not hold.

✓

$\Rightarrow$ equality doesn't hold $\quad \sum_{i=1}^{n} f(i) = 1+2+\dots+n = \frac{n(n+1)}{2} = \frac{n^2}{2}+\frac{n}{2} = \Theta(n^2) \neq \Theta(n)$

$f_2(n) = n$ $\quad \neq \Theta(f(n))$

$\Rightarrow$ equality holds $\quad \sum_{i=1}^{n} f(i) = c^1 + c^2 + \dots + c^n = \Theta(c^n) \cancel{= \Theta(f(n))}$

$f_1(n) = c^n$

where $c > 1$

$= c^{n+1} - 2 = \Theta(c^n) = \Theta(f(n))$

3 In Between Functions

✓

Prove or disprove: If $f : \mathbb{N} \to \mathbb{N}$ is any positive-valued function, then either (1) there exists a constant $c > 0$ so that $f(n) \in O(n^c)$, or (2) there exists a constant $\alpha > 1$ so that $f(n) \in \Omega(\alpha^n)$.

$\Rightarrow$ not true

- for example $f(n) = n^c \cdot \log n$

- $\lim_{n\to\infty} \left(\frac{n^c \log n}{n^c}\right) = \lim_{n\to\infty} \log n = \infty \Rightarrow$ therefore $f(n) = \Omega(n^c)$
- $\lim_{n\to\infty} \left(\frac{n^c \log n}{\alpha^n}\right) = \lim_{n\to\infty} \frac{c \cdot n^{c-1} \log n + n^c \cdot \frac{1}{n}}{\log \alpha \cdot \alpha^n} = \lim_{n\to\infty} \frac{c \cdot n^{c-1} \log n + n^{c-1}}{\log \alpha \cdot \alpha^n}$

$= \frac{c}{\log \alpha} \lim_{n\to\infty} \frac{n^{c-1} \cdot \log n}{\alpha^n} + \frac{1}{\log \alpha} \lim_{n\to\infty} \frac{n^{c-1}}{\alpha^n} = 0 \Rightarrow$ therefore $f(n) = O(\alpha^n)$

0 0

$\Rightarrow$ we can see that $f(n) = n^c \log n$ is neither $f(n) \neq O(n^c)$ or $f(n) \neq \Omega(\alpha^n)$

$\Rightarrow$ we can also consider $f(n) = 2^{\sqrt{n}}$

- for any constant $c > 0$, $f(n) \in \Omega(n^c)$
- for any constant $\alpha > 1$, $f(n) \in O(\alpha^n)$

→ there are algorithms whose running time grows faster than any polynomial but slower than any exponential

4 Recurrence Relation Practice

Derive an asymptotic *tight* bound for the following $T(n)$. Cite any theorem you use.

↳ use Master Theorem

(a) $T(n) = 2 \cdot T(\frac{n}{2}) + \sqrt{n}$.

√

$a = 2$ $\quad \frac{a}{b^d} = \frac{2}{2^{\frac{1}{2}}} = \sqrt{2} > 1 \quad \Theta(n^{\log_2 2}) = \underline{\Theta(n)}$

$b = 2$

$d = \frac{1}{2}$

(b) $T(n) = T(n-1) + c^n$ for constants $c > 0$.

⊗

(c) $T(n) = 2T(\sqrt{n}) + 3$, and $T(2) = 3$.

$T(n) = 2 \cdot T(\sqrt{n}) + 3$

$= 2(2T(n^{\frac{1}{4}}) + 3) + 3$

$\underline{\Theta(\log n)}$

~~$T = 3 + 2^1 \cdot 3 + \ldots + 2^i \cdot 3 = 3 \sum_{i=0} 2^i$~~

- work done on every node is constant

→ total work done is simply number of nodes of tree

$2^{h+1} - 1 = 2^{\log_2(\log_2(n))} - 1$

$= (\log_2(n))^{\log_2 2} - 1 = \log_2(n) - 1 = \Theta(\log n)$

height $= \log(\log n)$

1 problem of size n each at cost 3 | 3

2^1 $\quad n^{\frac{1}{2}}$ $\quad 3$ | $2^1 \cdot 3$

2^i $\quad n^{(\frac{1}{2})^i}$ $\quad 3$ | $2^i \cdot 3$

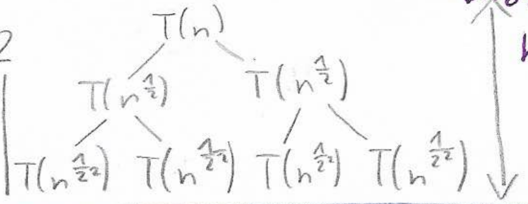
⋮

⇒ it will stop when $n^{\frac{1}{2^j}} = 2$

$(\frac{1}{2})^j \log_2 n = \log_2 2$ | $j = \log_2(\log_2 n)$

$\frac{1}{2^j} \log_2 n = 1$

$\log_2(n) = 2^j$



b) $\underline{T(n) = T(n-1) + c^n}$

~~1 problem of size n each at cost c^n | $n c^n$~~

~~1 $\quad n-1 \quad c^{n-1}$ | $(n-1) c^{n-1}$~~

~~⋮~~

~~1 $\quad n-i \quad c^{n-i}$ | $(n-i) c^{n-i}$~~

~~$T(n) = \cancel{x} c^n + (\cancel{n-1}) c^{n-1} + \ldots + (\cancel{n-i}) c^{n-i}$~~

~~• CASE 1 : $c > 1$~~

~~$O(\cancel{x} c^n)$~~

~~• CASE 2 : $c = 1$~~

~~$n + (n-1) + \ldots + 2 + 1 = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$~~

~~$O(n^{\cancel{2}})$~~

~~• CASE 3 : $c < 1$~~

~~$O(1)$~~

⇒ expand recurrence relation

$T(n) = T(n-1) + c^n$

$= T(n-2) + c^{n-1} + c^n$

$= T(n-3) + c^{n-2} + c^{n-1} + c^n$

→ therefore

$T(n) = 1 + c^1 + \ldots + c^{n-1} + c^n$

$T(n) = \sum_{i=0}^{i=n} c^i$

• CASE 1 : $c < 1$

$1 > c^1 > \ldots > c^n$ $\quad \underline{\Theta(1)}$

• CASE 2 : $c = 1$

$\underbrace{1 + 1 + 1 + \ldots + 1}_{n+1}$ $\quad n + 1 = \underline{\Theta(n)}$

• CASE 3 $c > 1$

$1 < c^1 < \ldots < c^n$ $\quad \underline{\Theta(c^n)}$

CORRECTIONS

4b $T(n) = T(n-1) + c^n$ for constants $c > 0$

WAY 1 $T(n) = T(n-1) + c^n = T(n-2) + c^{n-1} + c^n$

$1 = c^1 + \ldots + c^{n-1} + c^n = \sum_{i=0}^{n} c^i - 1 = \frac{c^{n+1} - 1}{c-1} - 1$

WAY 2

1 problem of size	n	each at cost c^n	c^n
1	n-1	c^{n-1}	c^{n-1}
1	1	c^1	c^1

$T(n) = c^n + c^{n-1} + \ldots + c^1 = \sum_{i=0}^{n} c^i - 1 = \frac{c^{n+1}-1}{c-1} - 1 = \frac{c(c^n - 1)}{c-1}$

=> discussion of cases

1) $c < 1$

$T(n) = \boxed{\Theta(1)}$

NOTE $\frac{1}{1-c} > \frac{c(c^n-1)}{c-1} = T(n) > \Theta(1)$

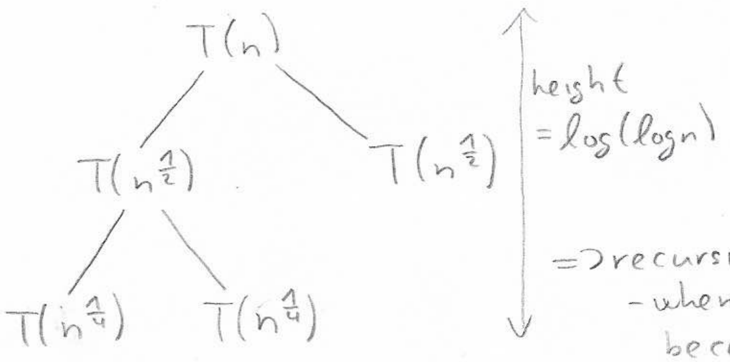
2) $c = 1$

$T(n) = \underbrace{1 + 1 + \ldots + 1}_{n} = \boxed{\Theta(n)}$

3) $c > 1$

$T(n) = \frac{c(c^n-1)}{c-1} = \boxed{\Theta(c^n)}$

4c $T(n) = 2 \cdot T(\sqrt{n}) + 3$ and $T(2) = 3$



1 problem of size	n	each at cost 3	$3 \cdot 1$
2	$n^{\frac{1}{2}}$	3	$3 \cdot 2$
2^i	$n^{\frac{1}{2^i}}$	3	$3 \cdot 2^i$

=> recursion will stop when $n^{(\frac{1}{2})^i} = 2$

- when the size of problem is 2 because $T(2) = 3$

$n^{(\frac{1}{2})^i} = 2$

$(\frac{1}{2})^i \log_2 n = \log_2 2$

$(\frac{1}{2})^i \log_2 n = 1$

$\log_2 n = 2^i$

$\log_2(\log_2 n) = i \log_2 2$

$i = \log_2(\log_2 n)$

-> note that $\log(\log n)$ is a height of recursion tree because recursion stops after $\log(\log n)$ steps

-> note that work done on every note is constant (3)

-> therefore total work done is $3 \cdot$(number of nodes)

number of nodes in tree $= 2^{h+1} - 1 = 2^{\log(\log n)+1} - 1$

$= 2 \cdot 2^{\log(\log n)} - 1 = 2 \cdot (\log n)^{\log 2} - 1 = 2 \cdot \log n = \boxed{\Theta(\log n)}$