CMPS101: Homework #2 Solutions

TA: Krishna (vrk@soe.ucsc.edu)

Due Date: April 19, 2011

1 3.1-1

Problem

Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that $max(f(n), g(n)) = \Theta(f(n) + g(n))$.

Solution 1

The functions $f(n)$ and $g(n)$ are asymptotically non negative, there exists n_0 such that $f(n) \geq 0$ and $g(n) \geq 0$ for all $n \geq n_0$. Thus, we have that for all $n \geq n_0$, $f(n) + g(n) \geq f(n) \geq 0$ and $f(n) + g(n) \geq g(n) \geq 0$. Adding both inequalities (since the functions are nonnegative), we get $f(n) + g(n) \geq max(f(n), g(n))$ for all $n \geq n_0$. This proves that $max(f(n), g(n)) \leq c(f(n) + g(n))$ for all $n \geq n_0$ with $c = 1$, in other words, $max(f(n), g(n)) = O(f(n) + g(n))$.

Similarly, we can see that $max(f(n), g(n)) \geq f(n)$ and $max(f(n), g(n)) \geq g(n)$ for all $n \geq n_0$. Adding these two inequalities, we can see that

$$2max(f(n), g(n)) \geq (g(n) + f(n))$$

, or

$$max(f(n), g(n)) \geq \frac{1}{2}(g(n) + f(n))$$

for all $n \geq n_0$. Thus $max(f(n), g(n)) = \Omega(g(n) + f(n))$ with constant $c = \frac{1}{2}$.

2 3.1-2

Problem

Show that for any real constants a and b, where $b > 0$, $(n + a)^b = \Theta(n^b)$.

Solution

By the definition of $\Theta(\cdot)$, we need find the constants c_1, c_2, n_0 such that $0 \leq c_1 n^b \leq (n+a)^b \leq c_2 n^b$ for all $n \geq n_0$.

Note that for large values of n, $n \geq |a|$ we have

$$n + a \leq n + |a| \leq 2n$$

and for further large values of n, $n \geq 2|a|$, (i.e., $|a| \leq \frac{1}{2}n$)

$$n + a \geq n - |a| \geq \frac{1}{2}n$$

.

Thus, when $n \geq 2|a|$, we have

$$0 \leq \frac{1}{2}n \leq n + a \leq 2n$$

.

Since b is a positive constant, we can raise the quantities to the b^{th} power with out affecting the inequality. thus

$$0 \leq (\frac{1}{2}n)^b \leq (n+a)^b \leq (2n)^b$$
$$0 \leq (\frac{1}{2})^b n^b \leq (n+a)^b \leq (2)^b (n)^b$$

Thus, with $c_1 = (\frac{1}{2})^b, c_2 = 2^b$, and $n_0 = 2|a|$ we satisfy the definition.

3 3.1-4

Problem

Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

Solution

$2^{n+1} = O(2^n)$, but $2^{2n} \neq O(2^n)$. To show that $2^{n+1} = O(2^n)$, we must find constants $c, n_0 > 0$ such that $0 \leq 2^{n+1} c 2^n$ for all $n \geq n_0$. Since $2^{n+1} = 22^n$ for all n, we can satisfy the definition with $c = 2$ and $n_0 = 1$.

To show that $2^{2n} \neq O(2^n)$, assume there exist constants $c, n_0 > 0$ such that $0 \leq 2^{2n} \leq c2^n$ for all $n \geq n_0$. Then $2^{2n} = 2^n \times 2^n \leq c2^n \implies 2^n \leq c$. But no constant is greater than 2^n for all n, and so the assumption leads to a contradiction.

4 3.2

Problem

Indicate, for each pair of expressions (A, B) in the table below, whether A is $O, o, \Omega, \omega, \Theta$ of B. Assume that $k \geq 1$, $\epsilon > 0$, and $c > 1$ are constants. Your answer should be in the form of the table with "yes" or "no" written in each box.

Note that we treat $lg\ n$ as a natural logarithm. Dealing with logarithms with different base is simple as they just add a constant factor, for example, $log_b\ n = \frac{log\ n}{\log\ b}$.

	A	B	O	o	Ω	ω	Θ
a.	$lg^k n$	n^ϵ	yes	yes	no	no	no
b.	n^k	c^n	yes	yes	no	no	no
c.	$\sqrt{n}$	$n^{sin\ n}$	no	no	no	no	no
d.	2^n	$2^{n/2}$	no	no	yes	yes	no
e.	n^{lgc}	c^{lgn}	yes	no	yes	no	yes
f.	$lg(n!)$	$lg(n^n)$	yes	no	yes	no	yes

(a) Apply L'Hospital's rule repeatedly to see that $\lim_{n\to\infty} \frac{(lgn)^k}{n^\epsilon} = 0$ to conclude that $(lgn)^k = o(n^\epsilon)$.

$$\begin{aligned}\lim_{n\to\infty} \frac{(lgn)^k}{n^\epsilon} &= \lim_{n\to\infty} \frac{k(lgn)^{k-1}\frac{1}{n}}{\epsilon n^{\epsilon-1}} \\ &= \lim_{n\to\infty} \frac{k(lgn)^{k-1}}{\epsilon n^\epsilon} \\ &= \lim_{n\to\infty} \frac{k\frac{d(lgn)^{k-1}}{dn}}{\epsilon\frac{dn^\epsilon}{dn}} \\ &= \lim_{n\to\infty} \frac{k(k-1)(lgn)^{k-2}\frac{1}{n}}{\epsilon^2 n^{\epsilon-1}}\end{aligned}$$

After k applications of the rule, we get

$$\lim_{n\to\infty} \frac{k(k-1)(k-2)....1}{\epsilon^k n^\epsilon} = 0$$

(b) Apply L'Hospital's rule repeatedly to see that $\lim_{n\to\infty} \frac{n^k}{c^n} = 0$ to conclude that $n^k = o(c^n)$.

(c) You can visually inspect the plots to see that $n^{sin\ n}$ is an oscillating function. $sin\ n$ oscillates between 1 and -1. When at its maximum value, $n^{sinn} > c\sqrt{n}$ and thus $n^{sin\ n} \neq O(\sqrt{n})$. When $sin\ n$ is at its minimum, $n^{sin\ n} < c\sqrt{n}$ and thus $n^{sin\ n} \neq \Omega(\sqrt{n})$.

(d) $\lim_{n\to\infty} \frac{2^n}{2^{n/2}} = \infty$ and therefore $2^n = \omega(2^{n/2})$.

(e) Recall that $n^{lgc} = c^{lgc}$.

(f) Note $lg(n^n) = nlg(n)$, and using Stirling's formula it is shown in the text that $lg(n!) = \Theta(nlg(n))$.

Solution

5 4.3-2

Problem

Show that the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(lgn)$.

Solution

We will use substitution method to verify the given solution. We start by trying to prove that $T(n) \le clg(n)$.

$$\begin{aligned} T(n) &\le clg(\lceil n/2 \rceil) + 1 \\ &< clg(n/2 + 1) + 1 \\ &= clg(\frac{n+2}{2}) + 1 \\ &= clg(n+2) - c + 1 \end{aligned}$$

This is inconclusive, so we will modify the solution a bit. We will try to prove that $T(n) \le clg(n - b)$. Note that this still would imply that $T(n)$ is $O(lgn)$.

$$\begin{aligned} T(n) &\le clg(\lceil n/2 - b \rceil) + 1 \\ &< clg(n/2 - b + 1) + 1 \\ &= clg(\frac{n - b - (b-2)}{2}) + 1 \\ &= clg(n - b - (b-2)) - clg2 + 1 \\ &\le clg(n - b) \end{aligned}$$

The last inequality is true if $b \ge 2$ and $c \ge 1$. Note that this still does not work for $n = 1$ as it involves computing $clog(1 - 2)$. It also does not work for $n = 2$. So the simple solution is to move up the base case.

6 4.4-6

Problem

Argue that the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + cn$, where c is a constant, is $\Omega(nlgn)$ by appealing to a recursion tree.

Solution

We are trying to prove a lower bound to the recurrence. Consider the smallest path of the recursion tree, $n \rightarrow 1/3n \rightarrow (1/3)^2n... \rightarrow 1$. This recursion bottoms out at level k at which $n/3^k = 1$, i.e., $k = log_3 n$. Since each node has two children and each level contributes cn, overall contribution from the internal nodes is at least $nlog_3 n = \Omega(nlogn)$.

7 Algorithm

Problem

You are given an $n \times n \times n$ array $A(i, j, k)$ of numbers. After $\Theta(n^3)$ preprocessing, show how to compute queries of the following form in O(1) time:

Input: $1 \leq i_1 \leq i_2 \leq n$, $1 \leq j_1 \leq j_2 \leq n$, $1 \leq k_1 \leq k_2 \leq n$
Output: $\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \sum_{k=k_1}^{k_2} A(i, j, k)$

Hint: We discussed similar problem for 1 and 2 dimensions in class

Solution

For the one dimensional case, the preprocessing involves computing the following sums, $S(1) = A(1), S(2) = \sum_{i=1}^{2} A(i), \ldots, S(n) = \sum_{i=1}^{n} A(i)$. This can be computed in linear time by observing that $S(i+1) = S(i)+A(i+1)$. Any query of from $\sum_{i=i_1}^{i_2} A(i)$ can be computed as $S(i_2) - S(i_1 - 1)$ in constant time.

In case of 2-d case, we compute the sums $S(r, s) = \sum_{i=1}^{r} \sum_{j=1}^{s} A(i, j)$ in $O(n^2)$ by noting the recurrence $S(i + 1, j + 1) = A(i, j) + S(i + 1, j) + S(i, j + 1) - S(i, j)$. The computation can be done in $O(n^2)$ as we spend constant time on each element by making use of previously computed results and there are n^2 elements to compute. The query of form $\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A(i, j)$ can be computed as $S(i_2, j_2)-S(i_1-1, j_2)-S(i_2, j_1-1)+S(i_1-1, j_1-1)$.

For the 3 dimensional case, compute the sums

$$S(r, s, t) = \sum_{i=1}^{r} \sum_{j=1}^{s} \sum_{k=1}^{t} A(i, j, k)$$

which can be done in $O(n^3)$. This computation can be performed using the following recursion,

$$\begin{aligned}
&S(i+1,j+1,k+1)\\
&= A(i+1,j+1,k+1)\\
&\ + S(i,j+1,k+1) + S(i+1,j,k+1) + S(i,j+1,k+1)\\
&\ - S(i+1,j,k) - S(i,j+1,k) - S(i,j,k+1)\\
&\ + S(i,j,k)
\end{aligned}$$

note that when applying this recursion, we observe the following initial conditions. $S(i,j,k) = 0$ when atleast one of i,j,k is zero. For example,

$$\begin{aligned}
&S(1,1,1)\\
&= A(1,1,1)\\
&\ + S(0,1,1) + S(1,0,1) + S(1,1,0)\\
&\ - S(1,0,0) - S(0,1,0) - S(0,0,1)\\
&\ + S(0,0,0)\\
&= A(1,1,1) + 0 + 0 + 0 - 0 - 0 - 0 + 0
\end{aligned}$$

Queries of the following form can now be computed in $O(1)$ time

$$\begin{aligned}
&\sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j_2}\sum_{k=k_1}^{k_2} A(i,j,k)\\
&= S(i_2,j_2,k_2)\\
&\ - S(i_1-1,j_2,k_2) - S(i_2,j_1-1,k_2) - S(i_2,j_2,k_1-1)\\
&\ + S(i_1-1,j_1-1,k_2) + S(i_1-1,j_2,k_1-1) + S(i_2,j_1-1,k_1-1)\\
&\ - S(i_1-1,j_1-1,k_1-1).
\end{aligned}$$

As an example of how these expressions are derived, consider the 2d case.

$$\sum_{i=1}^{i_2}\sum_{j=1}^{j_2} A(i,j) = \sum_{i=1}^{i_1-1}\sum_{j=1}^{j_2} A(i,j) + \sum_{i=i_1}^{i_2}\sum_{j=1}^{j_2} A(i,j)$$

$$
\begin{aligned}
\boldsymbol{S(i_2, j_2)} &= \boldsymbol{S(i_1 - 1, j_2) + \sum_{i=i_1}^{i_2} \sum_{j=1}^{j_2} A(i,j)} \\
&= \boldsymbol{S(i_1 - 1, j_2) + \sum_{i=i_1}^{i_2} \sum_{j=1}^{j_1-1} A(i,j) + \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A(i,j)} \quad \text{last term is the quantity we need} \\
&= \boldsymbol{S(i_1 - 1, j_2) + \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A(i,j) + \sum_{i=i_1}^{i_2} \sum_{j=1}^{j_1-1} A(i,j) + \sum_{i=1}^{i_1-1} \sum_{j=1}^{j_1-1} A(i,j) - \sum_{i=1}^{i_1-1} \sum_{j=1}^{j_1-1} A(i,j)} \\
&= \boldsymbol{S(i_1 - 1, j_2) + \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A(i,j) + \sum_{i=1}^{i_2} \sum_{j=1}^{j_1-1} A(i,j) - \sum_{i=1}^{i_1-1} \sum_{j=1}^{j_1-1} A(i,j)} \\
&= \boldsymbol{S(i_1 - 1, j_2) + \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A(i,j) + S(i_2, j_1 - 1) - S(i_1 - 1, j_1 - 1)}
\end{aligned}
$$

Rearranging terms we get the desired expression.

$$
\boldsymbol{\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A(i,j) = S(i_2, j_2) - S(i_1 - 1, j_2) - S(i_2, j_1 - 1) + S(i_1 - 1, j_1 - 1)}
$$

The expression for the 3-d problem can be obtained in a similar fashion.