Data Structures and Data Management

CS 240

Éric Schost

LaTeXed by Sibelius Peng

Preface

Disclaimer Much of the information on this set of notes is transcribed directly/indirectly from the lectures of CS 240 during Spring 2019 as well as other related resources. I do not make any warranties about the completeness, reliability and accuracy of this set of notes. Use at your own risk.

This set of notes is quite incomplete. I recompiled the old tex code using the new template. I will most likely make a new set of notes sooner or later. Stay tuned!

For any questions, send me an email via `https://notes.sibeliusp.com/contact`.

You can find my notes for other courses on `https://notes.sibeliusp.com/`.

Sibelius Peng

Contents

1

May 7

Problem count positive integers in an array.

An Instance `[-5, 10, -5, 20]`

The Solution 2

Size of the input length of the array

```
Count(A) // A is an array of length n
res = 0
for i = 0 ... n-1
if A[i] > 0
res++
return res
```

1.1 Order Notation

Example $f(n) = 2n^2 + 3n + 11 \qquad g(n) = n^2$

Proof For $n \geq 1$,

$$\begin{aligned} 2n^2 &\leq 2n^2 \\ 3n &\leq 3n^2 \quad \implies f(n) \leq 16n^2 \\ 11 &\leq 11n^2 \end{aligned}$$

Taking $c = 16, n_0 = 1$ this proves that $f(n) \in O(n^2)$

2

May 9

$f(n) = 75n + 500, g(n) = 5n^2$?

Proof

1. For $n \geq 20, 100n \leq 5n^2$
2. For $n \geq 20, 500 \leq 25n$

So if $n \geq 20, f(n) = 500 + 75n \leq 25n + 75n \leq 5n^2 = g(n)$. Since also $f(n) \geq 0$ for all n, taking $n_0 = 20$ and $c = 1$, this proves $f(n) \in O(g(n))$

Another Proof for $n \geq 1, 75n \leq 75n^2, 500 \leq 500n^2$
$f(n) \leq 575n^2 = 115g(n)$
So taking $n_0 = 1, c = 115$, this proves $f(n) \in O(g(n))$

Prove that $f(n) = 2n^2 + 3n + 11 \in \Omega\left(n^2\right)$ from first principles.

Proof

$$
\begin{aligned}
2n^2 &\geq 2n^2 \\
3n &\geq 0 \\
11 &\geq 0
\end{aligned}
$$

$f(n) \geq 2n^2 = 2g(n)$. Taking $n_0 = 1, c = 2$, this completes the proof. ($n_0 = 1, c = 1$ work as well)

Prove that $\frac{1}{2}n^2 - 5n \in \Omega\left(n^2\right)$ from first principles.

Proof For $n \geq 20, n^2 \geq 20n$, then $-5n \geq \frac{-1}{4}n^2$
add $\frac{1}{2}n^2$, $\quad \frac{1}{2}n^2 - 5n \geq \frac{1}{2}n^2 - \frac{1}{4}n^2 = \frac{1}{4}g(n)$
$f(n) \geq \frac{1}{4}g(n)$
So taking $n_0 = 20, c = \frac{1}{4}$, this completes the proof.

Prove that $\log_b(n) \in \Theta(\log n)$ for all $b > 1$ from first principles.

Proof

$$f(n) = \frac{\log n}{\log b} = \frac{g(n)}{\log b}$$

$$\frac{g(n)}{\log b} \leq f(n) \leq \frac{g(n)}{\log b}$$

Taking $n_0 = 1, c_1 = c_2 = \frac{1}{\log b}$, this completes the proof.

Example $f(n) = 2000n^2, g(n) = n^n$.
Given $c > 0$, we have to find n_0, (depend on c), such that for $n \geq n_0$, $|f(n)| < |cg(n)| \iff 2000n^2 < cn^n$ $(*)$

$(*)$ is equivalent to $2000 < cn^{n-2}$

1. For $n \geq 3, n-2 \geq 1$, so $n^1 \leq n^{n-2}$
2. For $n \geq 3$ and $n \geq \frac{2000}{c} + 1$
$$\frac{2000}{c} < \frac{2000}{c} + 1 \leq n \leq n^{n-2}$$
So taking $n_0 = \max\left(3, \frac{2000}{c} + 1\right)$, this proves $f(n) \in o(g(n))$

Example Let $f(n)$ be a polynomial of degree $d \geq 0$,

$$f(n) = c_d n^d + c_{d-1} n^{d-1} + \cdots + c_1 n + c_0$$

for some $c_d > 0$, prove $f(n) \in \Theta(n^d)$

Proof Then
$$\frac{f(n)}{g(n)} = \frac{c_d n^d + c_{d-1} n^{d-1} + \cdots + c_1 n + c_0}{n^d} = c_d + c_{d-1}\frac{1}{n} + \ldots + \frac{c_0}{n^d}$$
Then $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists, and is equal to

$$c_d + 0 + \ldots + 0 = c_d > 0$$

By the limit test, $f(n) \in \Theta(g(n))$

Example Prove that $n(2 + \sin n\pi/2)$ is $\Theta(n)$. Note that $\lim_{n\to\infty}(2 + \sin n\pi/2)$ does not exist.

Proof for $n \geq 1, -1 \leq \sin n\pi/2 \leq 1$ $\quad \ldots \quad$ $n \leq f(n) \leq 3n$. So taking $n_0 = 1, c_1 = 1, c_2 = 3$, this completes the proof.
On the other hand,
$$\frac{f(n)}{g(n)} = 2 + \sin n\pi/2 \qquad \text{has no limit at } n = \infty$$
the limit test does not apply

3

May 14

Example 3 $f(n) = \log(n) = \frac{\ln n}{\ln 2} \to f'(n) = \frac{1}{\ln 2 \cdot n}$ $\qquad g(n) = n \to g'(n) = 1$

So

$$\lim_{n\to\infty} \frac{f'}{g'} = 0 \implies \lim_{n\to\infty} \frac{f}{g} = 0 \implies f(n) \in o(g(n))$$

$f(n) = \log n \to f'(n) = \frac{1}{\ln n} \cdot \frac{1}{n}$
$g(n) = n^a$

$$\implies \frac{f'}{f'} = \frac{1}{\ln 2} \frac{1}{a} \frac{1}{n^a}$$

As before, limit f′/g′ = 0 $\implies$ limit f/g = 0. Therefore $f(n) \in o(g(n))$

$f(n) = (\log n)^c, \qquad g(n) = n^d$

$$\frac{f}{g} = \left(\frac{\log n}{n^{d/c}}\right)^c$$

Taking $a = \frac{d}{c}$, we saw that $\lim_{n\to\infty} \frac{\log n}{n^{d/c}} = 0$, so lim f/g = 0. So $f(n) \in o(f(n))$

3.1 Algorithm Analysis

```
Test1(n)
1. sum <- 0
2. for i <- 1 to n do
3. for j <- i to n do
4. sum <- sum + (i-j)^2
5. return sum
```

Let $T_1(n)$ be the runtime of Test1(n). Then $T_1(n) \in \Theta(S_1(n))$ where $S_1(n)$ is the number of time we enter Step4.

$$S_1(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} 1$$

1. $\sum_{j=1}^{n} 1 = n - i + 1$

2. So

$$S_n = \sum_{i=1}^{n}(n-i+1) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1 = n^2 - \frac{n(n+1)}{2} + 2 = \frac{1}{2}n^2 + \frac{1}{2}n \in \Theta(n^2)$$

So $T_1(n) \in \Theta(n^2)$

3.2 two strategies

```
Test2(A, n)
max <- 0
for i <- 1 to n do
for j <- i to n do
sum <- 0
for k <- i to j do
sum <- sum + A[k]
max <- max (max, sum)
return max
```

Insertion sort: sorting A in a descending order.

Worst case A sorted in increasing order.
Then for all i, A[i] goes to order 0 in i steps -> worst case runtime $\Theta(\sum_{i=1}^{n} i) = \Theta(n^2)$.

Best Case A sorted in decreasing order.
Then for all i, we exit the while loop immediately -> best case runtime $\Theta(\sum_{i=1}^{n} 1) = \Theta(n)$

4

May 16

$T(n) = 2T\left(\frac{n}{2}\right) + cn, \qquad n > 1\ (*)$
$T(1) = c$

$n = 2^k \to T(2^k) = 2T(2^{k-1}) + c2^k = 2(2T(2^{k-2}) + c2^{k-1}) + c2^k$ by $(*)$
$= 2^2T(2^{k-2}) + 2c2^k$
$= 2^2(2T(2^{k-3}) + c2^{k-2}) + 2c2^k$ by $(*)$
$= 2^3T(2^{k-3}) + 3c2^k$
$= 2^4T(2^{k-4}) + 4c2^k$
$= \ldots = 2^kT(2^{k-k}) + kc2^k$
$= 2^kT(1) + kc2^k = c2^k(k+1)$

Since $n = 2^k$, $\log n = k$

$$T(2^k) = c2^k(k+1)$$

$$T(n) = cn(\log n + 1)$$

5

May 21

Insert(A, k)

- if A is full, double its size
- copy k into A

$$\text{cost of insert, if length(A)=n} \begin{cases} 1 & \text{copy if A not full} \\ 1+n & \text{new key + doubling. otherwise} \end{cases}$$

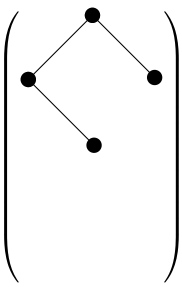
Suppose we start with length(A)=1. Total cost of n inserts (n a power of 2) is

$$\underbrace{1+1+\ldots 1}_{n \text{ (new key)}} + \underbrace{1+2+4+8+\ldots+n}_{\text{doubling}} = n + 2n - 1 = 3n - 1$$

5.1 Binary heaps

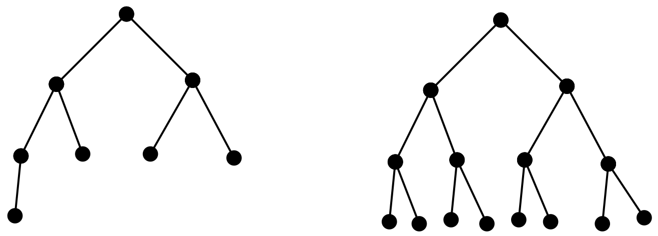
height of binary tree is length of the longest path from the root to a node.

height(•)=0

height  = 2

height (∅) = -1

number of nodes in a heap of height 3



at least 8 nodes

at most 15 nodes

true for any binary tree

$8 \leq n \leq 15$ if $h = 3$
$2^h \leq n \leq 2^{h+1} - 1$ any h true for any binary trees
$h \leq \log n$ and $h \geq \log n + 1$

Number of nodes in a heap of height h is

- at least
$$1 + 2 + 4 + \ldots + 2^{h-1} + 1 = 2^h$$
- is at most $1 + \ldots + 2^h = 2^{h+1} - 1$

6

May 23

recursive_heapify(T, n)

1. if n = 1, return
2. recursive_heapify (left child of T, # elements in left child)
3. recursive_heapify (right child, # in right)
4. fix down the root

7

Jun 18

7.1 Proof for slide 2 mod 6

Lower bound for search in a dictionary of size n, with keys $k_1, \ldots, k_n$, values $v_1, \ldots, v_n$. We count, comparisons between input key k and k_i's. (comparisons can be $<, >$ or $=$).

The decision tree associated to a given search algorithm in size n has $n+1$ leaves. $\begin{cases} (v_1, \ldots, v_n) \\ \text{“not found"} \end{cases}$

$$n + 1 = \#\, leaves \leq \#\, nodes \leq 2^{h+1} - 1$$

$$\implies h \geq \log(n+1) - 1$$

(and the height h is the most case # comparisons for this algorithm)

Suppose $A[0]$ and $A[n-1]$ are fixed $A[1]...A[n-2]$ chosen uniformly at random in $\{A[0]...A[n-1]\}$

Can prove to interpolation search in an array of length n with probability $\geq 1/4$, we do a recursive call in length $\leq \sqrt{n}$

$$\implies T^{avg}(n) \leq c + \frac{1}{4} T^{avg}(\sqrt{n}) + \frac{3}{4} T^{avg}(n)$$