

Data Structure and Algorithm HW1

R11522709 機械所 碩一 石翊鵬

April 13, 2023

1. What if you became a DSA TA?

Proof of conjecture 1

Consider $f(n) \geq 0$ and $g(n) \geq 0$ that $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c$, where $c \in R$

To prove $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \geq 0$ for all $n \in R$, we can consider $h(n) = 0$

Since $f(n) \geq 0$ and $g(n) \geq 0$ for all $n \in R$, $\frac{f(n)}{g(n)} \geq h(n)$ for all n therefore,

$$\lim_{n \rightarrow \infty} h(n) \geq \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$$
$$\lim_{n \rightarrow \infty} h(n) = 0 \Rightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c \geq 0$$

By definition of limit, there exists $\epsilon > 0$, $n_0 > 0$ such that for all $n > n_0$,

$$\left| \frac{f(n)}{g(n)} - c \right| < \epsilon$$
$$-\epsilon < \frac{f(n)}{g(n)} - c < \epsilon$$
$$-\epsilon + c < \frac{f(n)}{g(n)} < \epsilon + c$$

In case that $0 < c < \epsilon$

$$-\epsilon + c < 0 < \frac{f(n)}{g(n)} < \epsilon + c$$

In case that $c \geq \epsilon$

$$0 \leq -\epsilon + c < \frac{f(n)}{g(n)} < \epsilon + c$$
$$0 \leq f(n) < (\epsilon + c)g(n)$$

Let $(\epsilon + c) = c_1$

$$0 \leq f(n) \leq c_1 g(n)$$

which shows $f(n) = O(g(n))$, QED

1.1 - all by myself

Consider $c_1, c_2 > 0$ for all $n > 0$

if

$$\frac{c_1}{c_2}n \geq 1$$

then

$$c_1n^3 \geq c_2n^2$$

let

$$n_0 = \frac{c_2}{c_1}$$

$$\frac{c_1}{c_2}n_0 = 1$$

consider $k > 0$

$$\frac{c_1}{c_2}(n_0 + k) = 1 + \frac{c_1}{c_2}k$$

where

$$\frac{c_1}{c_2}k > 0$$

that is, for all $n \geq n_0$,

$$0 \leq c_2n^2 \leq c_1n^3$$

therefore, if

$$0 \leq f(x) \leq c_3n^2$$

for all $n > n_1$ where $n_1, c_3 > 0$

then

$$0 \leq f(n) \leq c_3n^2 \leq \frac{c_3c_1}{c_2}n^3$$

for all $n > \max(n_0, n_1)$, QED

1-2 - all by myself

Let $t_m =$ the number of while checks, $k =$ the index of key, $d_n =$ time cost of each line to execute once
Since the value of m is assigned to l in the beginning of each while loop, m will start from 1 and increase by 1 when while loop is executed once, moreover, $A[m] \leq \text{key}$ will always be satisfied.
therefore,

$$t_m = k$$

in worst case, key is not in the array, in this case

$$T(n) = d_1n + d_2(n-1) + d_3(n-1) + d_4 * 0 + d_5(n-1) + d_6 * 0 + d_7(n-1) + d_8(n-1) + d_9$$

$$T(n) = c_1n + c_2$$

where $c_1, c_2 > 0$

$$\lim_{n \rightarrow \infty} \frac{c_1n + c_2}{n} = c_1$$

By conjecture 1, time complexity of the algorithm is $O(n)$

1-3

ref: https://math.stackexchange.com/questions/925053/using-limits-to-determine-big-o-big-omega-and-big-theta#comment6149810_925053

$f(n) = \Theta(n^2) \iff$ there exists positive (n_0, c_1, c_2) such that $c_1 n^2 \leq f(n) \leq c_2 n^2$ for all $n \geq n_0$

Let $c_1 = 1$, $c_2 = 10$ and $f(n) = n^2(\sin(n) + 2) + n$

for all $n > 0$, $n^2 \leq f(n) \leq 10n^2$

Suppose

$$\lim_{n \rightarrow \infty} f(n) = L$$

where $L \in \mathbb{R}$, then there exist $\epsilon > 0$, $n > 0$ such that for all $n > n_0$,

$$|f(n) - L| < \epsilon$$

Let $f(n_1) = \epsilon_1$ where $n_1 > n_0$ and $\epsilon_1 < \epsilon$

$$f(n_1 + 2\pi k) - f(n_1) = ((n_1 + 2\pi k)^2 - n_1^2)(\sin(n_1) + 2) + 2\pi k$$

where $k \in \mathbb{N}$

$$\lim_{k \rightarrow \infty} f(n_1 + 2\pi k) - f(n_1) \neq 0$$

$$\Rightarrow |f(n + 2\pi k) - L| > \epsilon$$

for some $k > k_0$ where $k_0 \in \mathbb{N}$

That is, $\lim_{n \rightarrow \infty} \frac{f(n)}{n^2}$ does not exist, therefore, the proposition is WRONG.

1-4 - all by myself

$$lg(n) = 2lg(\sqrt{n}) = 2 \frac{\ln(\sqrt{n})}{\ln(2)}$$

Consider

$$\lim_{n \rightarrow \infty} \frac{lg(n)}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{2}{\ln(2)} \frac{\ln(\sqrt{n})}{\sqrt{n}}$$

Let $\sqrt{n} = x$

$$\lim_{n \rightarrow \infty} \frac{2}{\ln(2)} \frac{\ln(\sqrt{n})}{\sqrt{n}} = \lim_{x \rightarrow \infty} \frac{2}{\ln(2)} \frac{\ln(x)}{x}$$

By L'Hopital's rule

$$\lim_{x \rightarrow \infty} \frac{2}{\ln(2)} \frac{\ln(x)}{x} = \lim_{x \rightarrow \infty} \frac{2}{\ln(2)} \frac{\frac{d}{dx} \ln(x)}{\frac{d}{dx} x} = \lim_{x \rightarrow \infty} \frac{2}{\ln(2)} \frac{\frac{1}{x}}{1} = 0 \in \mathbb{R}$$

By conjecture 1, $lg(n) = O(\sqrt{n})$, QED

1-5 - all by myself

Consider

$$\frac{\lim_{n \rightarrow \infty} \sum_{i=1}^n i^n}{\lim_{n \rightarrow \infty} \sum_{i=1}^n n^n} = \lim_{n \rightarrow \infty} \sum_{i=1}^n \left(\frac{i}{n}\right)^n$$

and

$$\lim_{n \rightarrow \infty} \int_1^n \left(\frac{i}{n}\right)^n di$$

and

$$\lim_{n \rightarrow \infty} \int_0^{n-1} \left(\frac{i}{n}\right)^n di$$

By definition of Σ ,

$$\lim_{n \rightarrow \infty} \int_0^{n-1} \left(\frac{i}{n}\right)^n di < \lim_{n \rightarrow \infty} \sum_{i=1}^n \left(\frac{i}{n}\right)^n < \lim_{n \rightarrow \infty} \int_1^n \left(\frac{i}{n}\right)^n di$$

$$\lim_{n \rightarrow \infty} \int_1^n \left(\frac{i}{n}\right)^n di = \lim_{n \rightarrow \infty} \frac{1}{n^n} \int_1^n i^n di = \lim_{n \rightarrow \infty} \frac{1}{n^n} [i^n]_1^n = \lim_{n \rightarrow \infty} \frac{n^n}{n^n} - \lim_{n \rightarrow \infty} \frac{1^n}{n^n} = 1 - 0 = 1$$

$$\lim_{n \rightarrow \infty} \int_0^{n-1} \left(\frac{i}{n}\right)^n di = \lim_{n \rightarrow \infty} \frac{1}{n^n} \int_0^{n-1} i^n di = \lim_{n \rightarrow \infty} \frac{1}{n^n} [i^n]_0^{n-1} = \lim_{n \rightarrow \infty} \frac{(n-1)^n}{n^n} - \lim_{n \rightarrow \infty} \frac{1^n}{n^n} = 1 - 0 = 1$$

By squeeze theorem,

$$\lim_{n \rightarrow \infty} \sum_{i=1}^n \left(\frac{i}{n}\right)^n = 1 \in R$$

By conjecture 1, $\sum_{i=1}^n i^n = O(n^n)$

1-6 - all by myself

In second last line, since c can be positive or negative, it is possible that $\frac{1}{2^c} > 2^c$ and therefore $f(n) > \frac{1}{2^c}$

The correct proof:

$$|lg(f(n)) - lg(g(n))| = c \Rightarrow lg\left(\frac{f(n)}{g(n)}\right) = c \Rightarrow \frac{f(n)}{g(n)} = 2^c$$

$$\Rightarrow |lg(f(n)) - lg(g(n))| = c \Rightarrow lg\left(\frac{f(n)}{g(n)}\right) = c_1 \Rightarrow \frac{f(n)}{g(n)} = 2^{c_1}$$

where $c_1 = \pm c$

Take $c' = 2^c - 1$, we have $f(n) \leq c'g(n)$ for all $n > n_0$, QED

2. DSA Judge

2-1

ref: chatGPT

psuedo code:

```
procedure FINDLOST( $A, n$ )  
  for  $i$  from 0 to 1 do  
     $m = n/2$   
     $l = 0$   
     $r = n$   
    while  $(r - l) > 1$  do  
      if  $A[m] < m + (2 - k)$  then  
         $l = m$   
      else  
         $r = m$   
      end if  
    end while  
  end for  
   $\max(r_1, r_2) + = 1$   
  return  $r_1, r_2$   
end procedure
```

time complexity is almost the same as binary sort in worst case, the code will be run $\lg(n)$ times, the time complexity $T(n) = O(\lg(n))$

2-2 - all by myself

```
procedure COUPLEDDOUBLE( $A, n$ )  
   $isCouple = 0$   
   $m = A[n]/2$   
  for  $i$  from 0 to  $n - 1$  do  
    if  $A[n - i] < m$  then  
       $m = m/2$   
       $k = -k$   
    end if  
     $isCouple + = k$   
  end for  
  if  $isCouple == 0$  then  
    return True  
  else  
    return False  
  end if  
end procedure
```

2-3 - all by myself

```
procedure REVERSELYSORT(root1, root2)
  if root1.ID > root2.ID then
    newNext = root1
  else
    newNext = root2
  end if
  while root1.next or root2.next != NIL do
    if root2.ID > root1.ID then
      switch(root1, root2)
    end if
    if root1.next != NIL then
      newPrev = root1.next
      root1.next = newNext
      newNext = root1
      root1 = newPrev
    else
      root1.next = newNext
      newNext = root1
      while root2.next != NIL do
        newPrev = root2.next
        root2.next = newNext
        newNext = root1
        root2 = newPrev
      end while
    end if
  end while
end procedure
```

2-4 - all by myself

```
procedure REVERSELYSORT(root)
end procedure
```