• Homework 1

Jakob Symchych
nr2473
02/02/23

1) Longest Sorted SubArray

1.1) returns K=5

=> [110, 129, 130, 140, 142]

1.2) Base case (prove (I) true)

=> K = n

=> LIS(A) := longest inc. subArray of A

=> LIS(A) length $\leq$ K

=> LIS(A) [1 ... K]

1.2 cont.) Inductive step

=> Assume: (I) is true during last iteration of loop

prove: (I) true on next iteration

=> LIS(A) length $\leq$ K_{old} (assumption)

LIS(A) length $\leq$ K_{new} (prove)

$\therefore K_{new} = K_{old} + 1$

=> LIS(A) length $\leq K_{old} + 1 = K_{new}$

↳ Algorithm checks all elements in search of LIS(A), thus t/a iteration will increase size of LIS(A) by (K_{old} +1) until LIS(A) is found

1.2 cont.) Termination Step (when we leave loop)

=> We leave the loop when isIncreasing(int[]a...b]) returns true. That is when our k value is the LIS(A). Which we get to by decrementing the length of A[1....k]

1.3) Best case

=> Best case is when Array is already in increasing order

$$\overset{1 \ldots\ldots n}{\text{e.g: } A = [1, 2, 3, 4, 5]}$$

=> We would just need to walk through array n times

=> O(n)

1.4) Worst case

=> Worst case would be when Array is in descending order

e.g: A = [5, 4, 3, 2, 1]

=> isIncreasing() would return false on e/a iteration which we do n # of times. We also have the other loop while (high ≤ n) which also runs for n times

=> $O(n^2)$

1.5)

=> Algorithm is not the most efficient as this seems to be a brute force approach. On average algorithm will be $O(n^2)$. A better algorithm may loop through the array 1 time, checking for increasing sub Array.

2) Asymptotic Notation

2.1) $2n^3 + n^2 + 4 \in \Theta(n^3)$

$\therefore f(n) = 2n^3 + n^2 + 4$

$f(n) \leq c \cdot g(n)$

- looking for $f(n) \in O(n^3)$

$\Rightarrow 2n^3 + n^2 + 4 \leq c \cdot n^3$ // multi. 4 by n^2

$\Rightarrow 2n^3 + n^2 + 4n^2 = 2n^3 + 5n^2 \leq c \cdot n^3$ // replace 5 w/ n ($n \geq 5$)

$\Rightarrow 2n^3 + n \cdot n^2 \leq c \cdot n^3 = 2n^3 + n^3 = 3n^3$

$\Rightarrow 3n^3 \leq c \cdot n^3$

$\Rightarrow C = 3$

$n_0 = 5$

$f(n) \in O(n^3)$

- looking for $f(n) \in \Omega(n^3)$

$\Rightarrow 2n^3 + n^2 + 4 \geq c \cdot n^3$

// subtract 4

$\Rightarrow 2n^3 + n^2 + 4 - 4 \geq c \cdot n^3$

$\Rightarrow 2n^3 + n^2 \geq c \cdot n^3$ // replace 1 w/ n $(n \geq 1)$

$\Rightarrow 2n^3 + n \cdot n^2 \geq c \cdot n^3$

$\Rightarrow 2n^3 + n^3 \geq c \cdot n^3 \Rightarrow 3n^3 \geq c \cdot n^3$

$c = 3$
$n_0 = 1$

$f(n) \in \Omega(n^3)$

Therefore, $f(n) \in \Theta(n^3)$ □

2.2) $3n^4 - 9n^2 + 4n \in \Theta(n^4)$

· looking for $f(n) \in O(n^4)$

$\Rightarrow 3n^4 - 9n^2 + 4n \leq c \cdot n^4$ // add $9n^2$

$\Rightarrow 3n^4 + 4n \leq c \cdot n^4$ // replace 4 w/ n^3 $(n^3 \geq 4)$

$\Rightarrow 3n^4 + n^4 \leq c \cdot n^4$

$= 4n^4 \leq c \cdot n^4$

$c = 4$
$n_0 = \sqrt[3]{4} = 1.5 \approx \frac{3}{2}$

• looking for $f(n) \in \Omega(n^4)$

$\Rightarrow 3n^4 - 9n^2 + 4n \leq c \cdot n^4$ // subtract $4n$

$\Rightarrow 3n^4 - 9n^2 \leq c \cdot n^4$ // replace 9 w/ n^2 $(n^2 \geq 9)$

$\Rightarrow 3n^4 - n^4 \leq c \cdot n^4 \Rightarrow 2n^4 \leq c \cdot n^4$

$c = 2$

$n_0 = \sqrt{9} = 3$ $f(n) \in \Omega(n^4)$

Therefore, $3n^4 - 9n^2 + 4n \in \Theta(n^4)$ □

3) Summations

3.1) $\sum_{i=10}^{n}(5i+3) = O(n^2)$

$$\Rightarrow \sum_{i=10}^{n}(5i+3) = \sum_{i=1}^{n}(5i+3) - \underbrace{\sum_{i=1}^{10}(5i+3)}_{= \text{some constant } \#}$$

$$\Rightarrow 5\underbrace{\sum_{i=1}^{n} i}_{= \frac{n(n+1)}{2}} + \underbrace{\sum_{i=1}^{n} 3}_{= 3n} - C \qquad C := \sum_{i=1}^{10}(5i+3)$$

$$\Rightarrow \left(5 \cdot \frac{n(n+1)}{2}\right) + 3n - C$$

$$\Rightarrow \frac{5n^2 + 11n}{2} - C = O(n^2)$$

3.2) $\sum_{i=0}^{\log_2(n)} 2^i = O(n)$ $\quad \therefore a^{\log_a n} = n$

$\Rightarrow 2^0 + 2^1 + \ldots + 2^{\log_2 n}$ log rules

$\Rightarrow \sum_{i=0}^{\log_2 n} 2^i = \frac{2^{\log_2 n + 1} - 1}{2 - 1}$ $\quad \therefore a^{b+c} = a^b \cdot a^c$

$\Rightarrow 2^{\log_2 n + 1} - 1$

$\Rightarrow 2^{\log_2 n} \cdot 2^1 - 1$

$\Rightarrow n \cdot 2 - 1 = 2n + 1$

$= O(n)$