BANNER ID : B00872269

DALHOUSIE UNIVERSITY
CSCI 6057 - ADVANCED DATA STRUCTURES
WINTER 2022
ASSIGNMENT 2 - SOLUTION [DUE: February 9, 2022]

① SOLUTION:

Given:

$$\sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor} = \begin{cases} 2 \times (2^{K/2} - 1) & \text{if } K \text{ is even} \\ 3 \times 2^{(K-1)/2} - 2 & \text{otherwise} \end{cases}$$

(i) To prove it by induction method let there be case where (1) K = even and (2) K = odd.

Case (1): $K = 2K$

$$\text{let } S(2K) = \sum_{j=0}^{2K-1} 2^{\lfloor j/2 \rfloor}$$

Now, to adjust the summation sequence,

$$\therefore S(2K) = \sum_{j=0}^{K-1} \left(2^{\lfloor 2j/2 \rfloor} + 2^{\lfloor (2j+1)/2 \rfloor} \right)$$

$$= \sum_{j=0}^{K-1} \left(2^{j} + 2^{\lfloor j + (1/2) \rfloor} \right)$$

$$= \sum_{j=0}^{K-1} \left(2^{j} + 2^{j} \right) \quad \left[\lfloor j + (1/2) \rfloor = j \right.$$

$$= \sum_{j=0}^{K-1} 2^{j} + \sum_{j=0}^{K-1} 2^{j}$$

$$= 2 \times \sum_{j=0}^{K-1} 2^{j}$$

(2)

now $\sum_{i=0}^{n} 2^n = 2^{n+1} - 1$

So, $S(2K) = 2 * (2^{(K-1)+1} - 1)$

$$\therefore S(2K) = 2 * (2^K - 1) \quad -(1)$$

$$\therefore \text{for } S(K) = 2 * (2^{K/2} - 1) \quad -(2)$$

Case (2) : $K+1 = 2K+1$

$\therefore S(2K+1) = S(2K) + 2^K$

Substituting equation (1);

$= 2 * (2^K - 1) + 2^K$

$= 4^K - 2 + 2^K$

$= 6^K - 2$

$$S(2K+1) = 3.2^K - 2 \quad -(3)$$

$$\therefore S(K+) = 3 * 2^{(K-1)/2} - 2 \quad -(4)$$

Therefore, equality holds for any positive integer K.

(3)

(ii)

Given: $\sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor}$

(i) for K = even

let K = 8

$$\sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor} = \sum_{j=0}^{7} 2^{\lfloor j/2 \rfloor}$$

$$= 2^0 + 2^0 + 2^1 + 2^1 + 2^2 + 2^2 + 2^3 + 2^3$$

K = 0 1 2 3 4 5 6 7

$$= (2^0 + 2^1 + 2^2 + 2^3) + (2^0 + 2^1 + 2^2 + 2^3$$

$$= 2 \times (2^0 + 2^1 + 2^2 + 2^3)$$

Generalizing the formula we get,

$$\sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor} = 2\left(2^{K/2} - 1\right) \quad \text{for } K = \text{even}$$

Hence proved for the even case.

(4)

(2) For K = odd

let K = 7

$$\sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor} = \sum_{j=0}^{6} 2^{\lfloor j/2 \rfloor}$$

$$= 2^0 + 2^0 + 2^1 + 2^1 + 2^2 + 2^2 + 2^3$$

K = 0 1 2 3 4 5 6

$$= (2^0 + 2^1 + 2^2) + (2^0 + 2^1 + 2^2) + 2^3$$

$$= 2 * (2^0 + 2^1 + 2^2) + 2^3$$

Now generalizing the sequence we get

$$= 2 * (2^{(K-1)/2} - 1) + 2^{(K-1)/2}$$

[since K is even, we take (K-1) as odd]

$$= 2 * 2^{(K-1)/2} + 2^{(K-1)/2} - 2$$

$$\sum_{j=0}^{K-1} 2^{\lfloor j/2 \rfloor} = 3 * 2^{(K-1)/2} - 2 \quad \text{for } K = \text{odd}$$

Hence, proved for the odd case.

(5)

(2) SOLUTION:

Idea: The worst-case running time of any comparision based algorithm that sorts given sequence of 'n' elements is $\Omega(n \lg n)$.

Proof: Suppose a decision tree of height 'h' sorts 'n' elements.

The tree must contain at least 'n!' leaves because there are n! permutations of 'n' elements, each permutation indicating a different sorted order.

However, a binary tree of height 'h' can only have '2^h' leaves.

$\therefore n! \leq 2^h$

$\therefore h \geq \log(n!)$

The stirling's approximation states, that

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left[1 + \Theta\left(\frac{1}{n}\right)\right]$$

$$\therefore \log(n!) = \frac{1}{2}\log(2\pi n) + n\log(n/e) + \log\left[1 + \Theta\left(\frac{1}{n}\right)\right]$$

(6)

≥ 0

$= \pi \log n + n \log n - n \log e + \log 1 + \log\left[\Theta\left(\frac{1}{n}\right)\right]$

≥ 0

$\geq n \log n - n \log e$

$= \Omega(n \log n)$

∴ Therefore, any comparision-based sorting algorithm must make at least $O(n \log n)$ comparisions to sort the input array or given sequence of elements.

(7)

(3) SOLUTION:

Given:

$$S(u) = (\sqrt{u}+1)\,S(\sqrt{u}) + \sqrt{u}$$

(i) Using substitution method to prove that $S(u) = O(u)$

Let, $S(u) \leq c.(u-a)$ where $a < u$

$$\therefore S(u) \leq (\sqrt{u}+1)\,c.(\sqrt{u-a}) + \sqrt{u}$$

$$\leq c\left[\sqrt{u^2-ua} + \sqrt{u-a}\right] + \sqrt{u}$$

let $u^2 > ua$

$$\leq c\left[\sqrt{u^2} + \sqrt{u-a}\right] + \sqrt{u}$$

$$\leq cu + c\sqrt{u-a} + \sqrt{u}$$

$$\therefore S(u) \leq cu$$

$$\boxed{\therefore S(u) = O(u)} \qquad -(1)$$

(8)

(ii) Using substitution method to prove that $S(u) = \Omega(u)$.

Let, $S(u) \geq c(u+a)$ where $a < u$

$\therefore S(u) \geq (\sqrt{u} + 1)\, c(\sqrt{u+a}) + \sqrt{u}$

$\geq c\left[\sqrt{u^2+ua} + \sqrt{u+a}\right] + \sqrt{u}$

$\geq c\left[\sqrt{u^2} + \sqrt{u+a}\right] + \sqrt{u}$

(Assuming $u^2 > ua$)

$\geq cu + c\sqrt{u+a} + \sqrt{u}$

$\therefore S(u) \geq c \cdot u$

$\therefore S(u) = \Omega(u)$ - (2)

(4) SOLUTION:

Given: The variant of van Emde Boas trees has $u^{1/3}$ subwidgets instead of $u^{1/2}$ and the universe size of subwidget $u^{2/3}$ instead of $u^{1/2}$.

(i) For all the given operations: Member, Successor, Insert and Delete; following changes take place given the subwidget of size $u^{2/3}$.

$\therefore high = \lfloor x / \sqrt[3]{u^2} \rfloor$ — (1)

$\therefore low = x \bmod \sqrt[3]{u^2}$ — (2)

$\therefore$ Whereever high and low are used in the psuedocode, they will be updated as in equation (1) and (2).

Also, $\sqrt{u}$ term will change to $\sqrt[3]{u^2}$ in the psuedo code, like for instance in successor (w, x) the following will be updated in place of $\sqrt{u}$.

if low (x) < max [w']

return high(x). $\sqrt[3]{|w'|^2}$ + ...

(18)

(ii) Given the variant, the running time for all the operations is :

$\therefore T(u) = T(\sqrt[3]{u^2}) + O(1)$

Let $m = \log u$, $\Rightarrow u = 2^m$

$\therefore T(2^m) = T(2^{2m/3}) + O(1)$

Now, $S(m) = T(2^m)$

$\therefore S(m) = S(2m/3) + O(1)$ — (1)

Now using master theorem :

$T(n) = aT\left(\left[\frac{n}{b}\right]\right) + O(n^d)$

$a = 1 \quad b = \frac{3}{2} \quad d = 0$

$\therefore n^{\log_b a} = 0$

$\therefore d = n^{\log_b a}$ as per second case

$\therefore S(m) = O(\log m)$

$= O(\log \log u)$