FNU ANIRUDH

classmate
Date
Page

Assignment-4

① The number of internal nodes complete binary tree has is $2^h - 1$ where h is the height of the tree. A heap of height h has at least one additional node (otherwise it would be a heap of length h-1) and at most 2^h additional nodes (otherwise it would be a heap of length h+1)

Thus, if $n \in (2^h, 2^{h+1} - 1)$, then height will be $\lfloor \lg n \rfloor$.

② Number of Leaves in a heap is $\lceil n/2 \rceil$.

Base: h = 0. The number of leaves is $\lceil n/2 \rceil = \lceil n/2^{0+1} \rceil$.

Step: Let's assume it holds for nodes of height h-1. Let's take a tree and remove all it's leaves. We get a new tree with $n - \lceil n/2 \rceil = \lfloor n/2 \rfloor$ elements. Note that the nodes with height h in the old tree have height h-1 in the new one.

We will calculate the number of such nodes in the new tree. By the inductive assumption we have that T, the number of nodes with height h-1 in the new tree, is:

$$T = \left\lceil \lfloor n/2 \rfloor 2^{h-1+1} \right\rceil \leq \left\lceil (n/2) 2^h \right\rceil = \left\lceil \frac{n}{2^{h+1}} \right\rceil$$

This is also the number of nodes with height h in the old tree.

(3)

```
function PARTITION (A, p, r)
{
    if 'r'
        last ← r - 1
    else
        len (A) - 1

    value ← A(last)
    pivot ← p
    repeat ← 0

    for i from 'p' to 'last'
    {
        var ← A[i]
        if var = value
            repeat ← repeat + 1

        if var <= value
            A[pivot], A[i] ← A[i], A[pivot]
                                //SWAP
            pivot ← pivot + 1
        A[pivot], A[last] ← A[last], A[pivot]
                                //SWAP
        return pivot - repeat // 2
    }
```

This returns r.

We are counting the number of comparisons where A[j] = A[r] and then subtracting half that number from pivot.

(4) If the elements in A are the same then the returned element from each call to PARTITION(A,p,r) is r-1 thus yielding the worst-case partioning. The total running time is easily seen to be $\Theta(n^2)$. (one of the partions is always empty).

(5) If the elements in A are distinct and sorted in decreasing order then, as in the previous question, we have worst-case partioning and we get one empty partition thus total running time is easily seen to be $\Theta(n^2)$.

(6)

(a) $T(n) = \Theta(n^2)$ because each split will be (n-1) to 1.

(b) PARTITION(A, p, r)

```
q = p
t = r
i = p + 1
x = A[p]
while i < t + 1
    if A[i] < x
        exchange A[q] and A[i]
        q = q + 1
    else if A[i] > x
        exchange A[t] and A[i]
        t = t - 1
    else
        i = i + 1
return (q, t)
```

(c) RANDOMIZED-PARTITION (A, p, r)
    i = RANDOM (p, r)
    exchange A[r] with A[i]
    return PARTITION (A, p, r)

QUICKSORT (A, p, r)
    if p < r
        q, t = RANDOMIZED-PARTITION (A, p, r)
        QUICKSORT (A, p, q-1)
        QUICKSORT (A, t+1, r)

(d) Adjustment :- If there are duplicates in the array, once we choose one of the duplicate elements as pivot, we will never choose other duplicate in the sub array. In short, once a pivot x is chosen with $z\_i \leq x \leq z\_j$, we know that z_i and z_j cannot be compared at any subsequent time.