EL9343 Homework 3
(Due Oct. 15th, 2019

All problem/exercise numbers are for the third edition of CLRS text book

.

---

1. Exercise 7.4-5 in CLRS Textbook

2. Problem 7-2 in CLRS Text book

3. Similar to Figure 8.2, illustrate the operation of COUNTING-SORT on

$$[2, 4, 7, 3, 6, 1, 3, 4, 5, 7]$$

4. Exercise 8.2-3 in CLRS Textbook

5. Exercise 8.3-3 in CLRS Textbook

6. Problem 9-1 in CLRS Textbook

7. Exercise 11.2-1 in CLRS Textbook

8. Exercise 12.2-9 in CLRS Textbook

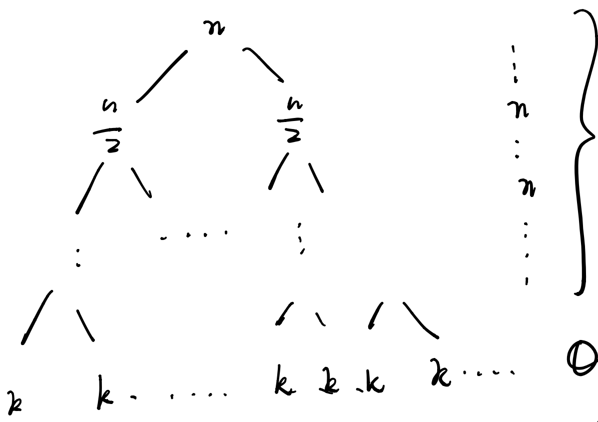
1. Exercise 7.4-5 in CLRS Textbook

Sol:

When input "nearly" sorted, when sorting subarray with fewer than k elements. run insertion sort.

Therefore, for $(\frac{n}{k})$ subarray the running time is $O(k)$ as the elements are "nearly" sorted.

While the partition is called. x times so that subarray's elements are fewer than k. (Consider the best case of partitioning). at each stage it takes $\Theta(n)$ time to partition.



$$\frac{n}{2^x} = k,$$

$$x = \lg\left(\frac{n}{k}\right)$$

$$T_1 = n\lg\left(\frac{n}{k}\right).$$

$O(k)\cdot n$.

if nearly sorted then $O(k)\cdot\frac{n}{k}$

thus: $T = O(nk + n\lg\frac{n}{k})$.

Theoretically: $\frac{\partial T}{\partial k} = O(n - \frac{n}{k}) = 0$.

$$k = 1$$

which means we abandon the insertion sort.

but in practice who knows. since it's a nearly sorted array. why don't we abandon QuickSort, instead and choose k as large as possible since

$$\frac{\partial T}{\partial k} < 0.\quad \forall k.$$

2. Problem 7-2 in CLRS Text book

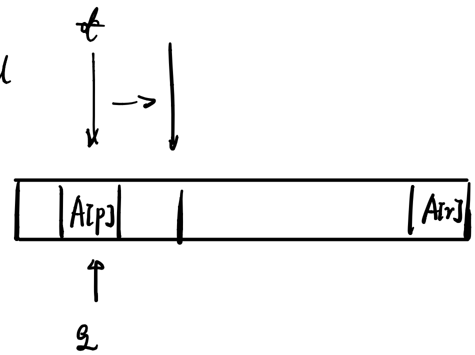
Quick sort with equal element values.

a. All elements are equal.

We have the same comparison times, since the randomized pivot changes nothing of the array, the analysis remains the same; since the array is almost ascending the overall running time is $O(n^2)$.

b. partition the array as

$A[p..q-1]$ $A[q..t]$ $A[t+1..r]$.



PARTITION:(A, p, r) # modified

Do if p < r:

    # choose the right element as pivot

    x = A[r]

    q = p

    t = p

    # q is the left boundary t is the right boundary.

    while. (t < r).

        do if (A[t] < x).

            then. A[t] <-> A[q].

                t ← t+1

                q ← q+1

        else if (A[t] = x)

            then t ← t+1

        else. A[t] <-> A[r].

                r ← r-1.

    end.

    return. q, r.

c. RANDOMIZED-PARTITION (A, p, r)

i = RANDOM (p, r)

A[i] <-> A[r]

call function : PARTITION (A, p, r)

d. Quick Sort (A, p, r).

if p < r then

q, t = RANDOMIZED-PARTITION (A, p, r)

Quick Sort (A, p, q-1)

Quick Sort (A, t+1, r)

3. Similar to Figure 8.2, illustrate the operation of COUNTING-SORT on

$$[2, 4, 7, 3, 6, 1, 3, 4, 5, 7]$$

2	4	7	3	6	1	3	4	5	7

0	1	2	3	4	5	6	7
0	1	1	2	2	1	1	2

0	1	2	3	4	5	6	7
0	1	2	4	6	7	8	10

									7

0	1	2	3	4	5	6	7
0	1	2	4	6	7	8	9

						5			7

0	1	2	3	4	5	6	7
0	1	2	4	6	6	8	9

					4	5			7

0	1	2	3	4	5	6	7
0	1	2	4	5	6	8	9

1	2	3	3	4	4	5	6	7	7

4. Exercise 8.2-3 in CLRS Textbook

Still works properly, but not stable.

1. the $C[k-1]$, $C[k]$ still contains all the elements within their interval.

2. the order of the key would be reversed.

5. Exercise 8.3-3 in CLRS Textbook

Proof of Radix Sort:

suppose the max number of digits is d. the base is x.

after sorting i digits. that $A[\,]_{i:d}$ is sorted holds.

then according to the algorithms, $A[\,]_{i-1}$ is sorted, (where the subscript means the $i-1$ digits from right to left), since $v \cdot x^{d-i+1}$ (v is the number of $A[\,]_{i-1}$) is greater then the presorted digits, so for elements having different v, they're already sorted, thus the question boils down to elements with the same number of v, since the last $d-i+1$ digits can be viewed as the key and we're using stable algorithm to sort, thus elements with the same v are also sorted. Therefore that the array $A[\,]_{i-1:d}$ is sorted still holds.

Q.E.D.

6. Problem 9-1 in CLRS Textbook

Largest i number in sorted order.

a. $\Theta(n \lg n) \rightarrow \Theta(i)$ list ith largest takes $\Theta(i)$ time

b. $\Theta((n+i)\lg n)$ build a tree $n \lg n$. extract $\Theta(\lg n)$ i times

c. $\Theta(n + i\lg i)$ to find the ith largest number and partition it takes n times, sort i numbers takes $\Theta(i \lg i)$ time.

7. Exercise 11.2-1 in CLRS Textbook

$h: K[n] \rightarrow T[m]$ define the collision set C

$$C = \{ \{k, l\} : k \neq l \text{ and } h(k) = h(l) \}$$

$$\mathbb{E}[|C|] = \sum_{i=1}^{n} \mathbb{E}[n_i] \quad \text{where. } n_i = \sum_{j>i} \mathbb{I}_{h(k_i)}(h(k_j))$$

$$\mathbb{E}[n_i] = (n-i)\frac{1}{m}$$ since we have the uniform hashing

$$\Rightarrow \quad \mathbb{E}[|C|] = \sum_{i=1}^{n} \frac{n-i}{m}$$

$$= \frac{n^2 - n}{2m}.$$

8. Exercise 12.2-9 in CLRS Textbook

Showing y.key is either the smallest key in T larger than x.key. or the largest key in T smaller than x.key is equivalent to show that y is either successor(x) or predecessor(x), which holds because y is the parent of leaf node x.