EECS 4020 Algorithms

HW2

I. Quicksort

Q1

- Suppose all values in $A[1..n]$ are the same

Running time of Quicksort on A?

Q1 [solution]

Ans: It depends.

- If comparison of two numbers can have 3 results (<, >, =) ➔ O(n) time
- If comparison of two numbers only has 2 results (<, ≥) ➔ O(n^2) time

Q2

- Suppose the splits in every level of Quicksort are in proportion of α and $1 - \alpha$ $(0 < \alpha \leq 1/2)$

Running time of Quicksort on A?

Q2 [solution]

In the recursion tree:

- First $\log_{1/\alpha} n$ levels each has $\Theta(n)$ steps
  ➔ $\Omega(n \log n)$ time
- At most $\log_{1/(1-\alpha)} n$ levels, each has $O(n)$ steps
  ➔ $O(n \log n)$ time

$\boxed{\text{Running time is } \Theta(n \log n)}$

Q3

- n pairs of bolts and nuts, all different sizes
- Bolts and nuts are separated, and mixed
- One of the nuts, X, is now taken away

How to find the bolt that matches X ?

Q3 [solution]

1. Pick a random bolt, Y
2. If Y does not have a matching nut, return Y
3. Else, let Y' be the matching nut of Y
4. Use Y and Y' to separate the bolts and nuts into "small" and "large" groups
5. Recursively find the bolt from the group with the missing nut

II. Lower Bound in Comparison Sorts

Q1

- Total n/k groups, each with k distinct numbers
- Numbers in j^{th} group are all smaller than numbers in $j+1^{th}$ group

Lower bound to sort all the n numbers ?

Q1 [solution]

- Number of possible inputs : $(k!)^{n/k}$
- By decision tree model :

  \# comparisons = height of decision tree

  $$\geq \log [(k!)^{n/k}]$$

  $$= (n/k) \log (k!)$$

  $$= \Omega(n \log k)$$

Q2

- 25 horses, each runs with different speed
- One race can compare 5 horses
- Target : Find out the best 3 horses

How many races do we need ?

Q2 [solution]

At most 7 races

1. Partition horses into 5 groups (A, B, C, D, E);
   organize a race for each group [Race 1—5]
2. Get the winners (A_1, B_1, C_1, D_1, E_1);
   organize a race between them [Race 6]

Q2 [solution]

3. WLOG, suppose $A_1 > B_1 > C_1 > D_1 > E_1$
4. Get the horses A_2, A_3, B_1, B_2, C_1; organize a race between them [Race 7]
5. The best 3 horses are :

   A_1, 1^{st} of Race 7, 2^{nd} of Race 7

Q2 [solution]

$\boxed{\text{At least 7 races}}$

1. Suppose to the contrary that 6 races is enough
2. After first 5 races, we have at least 5 non-losers
3. Yet, we cannot have more than 5 non-losers; else, one further race cannot find the winner

   → After first 5 races, exactly 5 non-losers

Q2 [solution]

4. There are two cases :
   - Case 1: All winners in first 5 races are distinct
     - They must race in Race 6

     ➔ cannot find 2nd best (why?)
   - Case 2: Some winner X wins more than once
     - All non-losers must race in Race 6

     ➔ We let X wins ➔ cannot find 2nd best (why?)

III. Sorting in Linear Time

Q1

- n integers, with value between 0 and $n^2 - 1$

$\boxed{\text{How to sort them in } O(n) \text{ time ?}}$

Q1 [solution]

1. Treat each number as a 2-digit n-ary number
    - Represent a number r as (x, y), $r = nx + y$
2. Sort the numbers by Radix Sort

Q2

- Show how Radix Sort sorts the following :

COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB,
BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX

Q2 [solution]

Round 1 :

SEA, TEA, MOB, TAB, DOG, RUG, DIG, BIG, BAR, EAR, TAR, COW, ROW, NOW, BOX, FOX

Round 2 :

TAB, BAR, EAR, TAR, SEA, TEA, DIG, BIG, MOB, DOG, COW, ROW, NOW, BOX, FOX, RUG

Q2 [solution]

Round 3 :

BAR, BIG, BOX, COW, SEA, DIG, DOG, EAR, FOX,

MOB, NOW, ROW, RUG, SEA, TAB, TAR, TEA

Q3

- A set S of n integers, each in the range 0 to k
- Target: To support query below in $O(1)$ time:

  count(L, R) := reports # integers in S whose value is in the range [L, R]

How to do so in $O(n + k)$ time and space ?

Q3 [solution]

Build a prefix-sum structure :

1. Process S to get an array $Count[0..k]$ such that

   $$Count[\, j \,] = \#\text{ items in } S \text{ with value } j$$

2. Get an array $Sum[0..k]$ such that

   $$Sum[\, j \,] = Count[\, 0 \,] + \ldots + Count[\, j \,]$$

Total time and space : $O(\, n + k \,)$

Q3 [solution]

To answer count(L, R) :

we return

$$\text{Sum}[\,R\,] - \text{Sum}[\,L-1\,]$$

where $\text{Sum}[\,-1\,]$ is set to 0

$\boxed{\text{Query time : } O(1)}$

IV. Order Statistics

Q1

- n distinct numbers
- Get the 1st, 2nd, 4th, 8th, … smallest numbers

How to do so in $O(n)$ time ?

Q1 [solution]

1. Let k be the largest 2-power that is at most n
2. Get the k^{th} smallest number X
3. Use X to filter out the k smallest numbers
4. If $k > 1$, set k to be $k/2$, go to Step 2;

   Else, finish running

$$\boxed{\text{Total time} = n + n/2 + n/4 + \ldots = O(n)}$$

Q2

- n distinct numbers
- Get smallest sqrt(n) numbers in sorted order

How to do so in O(n) time ?

Q2 [solution]

1. Get the sqrt(n)th smallest number Y
2. Use Y to filter out the sqrt(n) smallest numbers
3. Sort the numbers

$$\boxed{\text{Total time} = n + \text{sqrt}(n) \log n = O(n)}$$

Q3

- A set S of n distinct numbers (n = odd)
- Get k numbers with values closest to median

How to do so in O(n) time ?

Q3 [solution]

1. Get the median M
2. Get a set S' by deducting each number of S by M, and taking the absolute value
3. Find the k^{th} smallest value V of S'
4. Use V to filter out the k smallest values of S'
5. Report original numbers in S that correspond to the k smallest values from Step 4