Fall 2021 Rick Ramirez

Computer Science 5300

Advanced Algorithm Design and Analysis
Assignment # 1

Problem 1

Let X(1..n) and Y(1..n) contain two lists of n integers, each sorted in nondecreasing order. Give the best (worst-case complexity) algorithm that you can think for finding

(a) the largest integer of all 2n combined elements.

(b) the second largest integer of all 2n combined elements.

(c) the median (or the nth smallest integer) of all 2n combined elements.

For instance, X = (4, 7, 8, 9, 12) and Y = (1, 2, 5, 9, 10), then median = 7, the nth smallest, in the combined list (1, 2, 4, 5, 7, 8, 9, 9, 10, 12). [Hint: use the concept similar to binary search]

Solution:

(a)

Algorithm 1 Calculates the largest element of two sorted arrays

```
procedure largestElement (int X[n], int Y[n])
    return max(X[n-1], Y[n-1])
end procedure
```

(b)

Algorithm 2 Calculate second largest element of two sorted arrays

```
procedure secondLargest (int X[n], int Y[n])
    if X[n-1] == Y[n-1] then
        return max(X[n-2], Y[n-2])
    else if X[n-1] < Y[n-1] then
        return max(X[n-1], Y[n-2])
    else
        return max(X[n-2], Y[n-1])
    end if
end procedure
```

(c)

By looking at the two sorted arrays, we see that the median value will correspond to the nth term in the list of length 2n, and that this portion of the list contains part of X[ ] and Y[ ]. Also, note that the last element of any subset of the two sorted lists, starting at index zero, will always correspond to the kth element.

$$\underbrace{\boxed{1}\ \boxed{2}\ \boxed{4}\ \boxed{5}\ \boxed{7}}_{\text{median}=\ 2\frac{n}{2}=n}\ \boxed{8}\ \boxed{9}\ \boxed{9}\ \boxed{10}\ \boxed{12}$$

If we look at the two list individually, we find that the elements that lead up to the partition of the kth element are

$$\begin{array}{llllll} \boxed{4} & \boxed{7} & & \quad \boxed{8} & \boxed{9} & \boxed{12} \\ \boxed{1} & \boxed{2} & \boxed{5} & \quad \boxed{9} & \boxed{10} & \end{array}$$

Let the elements to the left and right of the partition of the first array be l_x and r_x respectively. And similarly, for the second array, l_y and r_y. Notice that for the solution to be valid, ie, for all the elements to the left of the partition to be less than all the elements to the right, requires

$$l_x \leq r_y$$
$$l_y \leq r_x$$

Only these conditions need to be checked because we are guaranteed to have $l_x \leq r_x$ and $l_y \leq r_y$. Once these conditions are satisfied, the kth element will be $max(l_x, l_y)$ because we know that in the combined array, the largest element to the left of the partition corresponds to the kth element. To find the partition, we apply a binomial search to only one array while checking the conditions against both arrays. We start by taking the midpoint of the first array

$$mid1 = \frac{low + high}{2}$$

and let the remaining $mid2 = n - mid1$ elements be taken from the second array, where low and high are initially set to 0 and n respectively. if we find that $l_x > r_y$, we cut out the right half of the array by assigning $high = mid1 - 1$ and if we find that $l_y > r_x$, we remove the left half of the array by assigning $low = mid + 1$. Since the algorithm utilizes a binomial search, the time complexity will be $\mathcal{O}(n \log n)$

Algorithm 3 Calculate median element of two sorted arrays

```
procedure MEDIANELEMENT (int X[n], int Y[n])
    int low = 0
    int high = X.length
    while low ≤ high do
        int mid1 = (low + high)/2
        int mid2 = X.length - mid1
        lx = X[mid1 - 1]
        ly = Y[mid2 - 1]
        rx = X[mid1]
        ry = Y[mid2]
        if lx ≤ ry && ly ≤ rx then
            return max(lx, ly)
        else if lx > ry then
            high = mid1-1
        else if ly > rx then
            low = mid1 + 1
        end if
    end while
end procedure
```

Problem 2

1-to-2 PARTITION:

Instance: A finite set of positive integers Z = z1 , z2 , ... , zn .

Question: Is there a subset Z' of Z such that Sum of all numbers in Z' = 2 x Sum of all numbers in Z-Z'

(a) Obtain the dynamic programming functional equation to solve the 1-to-2 PARTITION problem.

(b) Give an algorithm to implement your functional equation.

(c) Give an example of 5 numbers with a total of 21 as an input instance for 1-to-2 PARTITION problem, and show how your algorithm works on this input instance.

(d) What is the complexity of your algorithm?

Solution

******* SOLUTION GOES HERE *******

Problem 3

Decide True or False for each of the followings. You MUST briefly justify your answer.

Satisfiability:

Instance: Set U of variables, collection C of clauses over U.

Question: Is there a satisfying truth assignment for C?

(a) If P $\neq$ NP, then no problem in NP can be solved in polynomial time deterministically.

(b) If a decision problem A is NP-complete, proving that A is reducible to B, in polynomial time, is sufficient to show that B is NP-complete.

(c) It is known that SAT (Satisfiability) is NP-complete, and 3SAT (all clauses have size 3) is NP-complete. 1SAT (all clauses have size 1) is also NP-complete.

Solution

******* SOLUTION GOES HERE *******

Problem 4

Solution

******* SOLUTION GOES HERE *******

Problem 5

Solution

******* SOLUTION GOES HERE *******

Problem 6

Solution

******* SOLUTION GOES HERE *******