

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 of 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 n th 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 n th 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

```

1: procedure LARGESTELEMENT (int X[n], int Y[n])
2:   return  $\max(X[n-1], Y[n-1])$ 
3: end procedure

```

(b)

Algorithm 2 Calculate second largest element of two sorted arrays

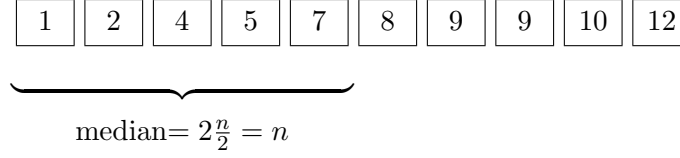
```

1: procedure SECONDLARGEST (int X[n], int Y[n])
2:   if  $X[n-1] == Y[n-1]$  then
3:     return  $\max(X[n-2], Y[n-2])$ 
4:   else if  $X[n-1] < Y[n-1]$  then
5:     return  $\max(X[n-1], Y[n-2])$ 
6:   else
7:     return  $\max(X[n-2], Y[n-1])$ 
8:   end if
9: end procedure

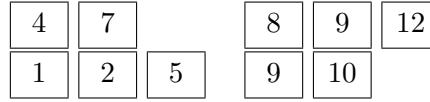
```

(c)

By looking at the two sorted arrays, we see that the median value will correspond to the n th 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 k th element.



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



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 k th 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 k th 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 $\mathcal{O}(\log n)$.

Algorithm 3 Calculate median element of two sorted arrays

```
1: procedure MEDIANELEMENT (int X[n], int Y[n])
2:   int low = 0
3:   int high = X.length
4:   while low  $\leq$  high do
5:     int mid1 = (low + high)/2
6:     int mid2 = X.length - mid1
7:     lx = X[mid1 - 1]
8:     ly = Y[mid2 - 1]
9:     rx = X[mid1]
10:    ry = Y[mid2]
11:    if  $\text{lx} \leq \text{ry} \ \&\& \ \text{ly} \leq \text{rx}$  then
12:      return  $\max(\text{lx}, \text{ly})$ 
13:    else if  $\text{lx} > \text{ry}$  then
14:      high = mid1-1
15:    else if  $\text{ly} > \text{rx}$  then
16:      low = mid1 + 1
17:    end if
18:  end while
19: end procedure
```

Problem 2

1-to-2 PARTITION:

Instance: A finite set of positive integers $Z = z_1, z_2, \dots, z_n$.

Question: Is there a subset Z' of Z such that $\text{Sum of all numbers in } Z' = 2 \times \text{Sum of all numbers in } Z - Z'$

- (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

(a)

For a solution to exist, two conditions must be satisfied:

- The sum of the total number integers $\sum_{i=1}^n z_i$ must be even. If it is odd, then the set cannot be divided into two equal subsets.
- If \exists a sum $S_1 = \sum_{i=1}^m z_i = \frac{1}{2} \sum_{i=1}^n z_i$, where $m < n$, then we are guaranteed to have the remaining $Z - Z'$ elements equal to $S_2 = \frac{1}{2} \sum_{i=1}^n z_i$. The set can be broken into two equal sums.

Consider the example of $Z = \{1, 3, 5, 11\}$. The first condition passes with $\sum_{i=1}^n z_i = 20$ ✓ however, the second condition requires us to find a subset whose sum is equal to $S_1 = \frac{1}{2} \sum_{i=1}^n z_i = 10$. The best we can do in this case is $Z_1 = \{1, 3, 5\} \rightarrow S_1 = 9 \neq 10$ and $Z_2 = \{11\} \rightarrow S_2 = 11$ ✗. \therefore this Z input will return false.

Now consider $Z = \{3, 1, 1, 2, 2, 1\}$. The first condition passes with $\sum_{i=1}^n z_i = 10$ ✓, and the second condition passes since $Z_1 = \{3, 1, 1\} \rightarrow S_1 = 5$ and $Z_2 = \{2, 2, 1\} \rightarrow S_2 = 5$ ✓. \therefore this Z input will return true.

Initial						
Sum (S) → Z ↓	0	1	2	3	4	5
	1	0	0	0	0	0
$z_1=3$	1					
$z_2=1$	1					
$z_3=1$	1					
$z_4=2$	1					
$z_5=2$	1					
$z_6=1$	1					

 \Rightarrow

Final						
Sum (S) → Z ↓	0	1	2	3	4	5
	1	0	0	0	0	0
$z_1=3$	1	0	0	1	0	0
$z_2=1$	1	1	0	1	1	0
$z_3=1$	1	1	1	1	1	1
$z_4=2$	1	1	1	1	1	1
$z_5=2$	1	1	1	1	1	1
$z_6=1$	1	1	1	1	1	1

Starting with the first empty cell of the matrix:

$1 - 3 = \text{undefined} \rightarrow \text{false} \therefore$ we cannot form the sum=1 using integer 3

$2 - 3 = \text{undefined} \rightarrow \text{false} \therefore$ we cannot form the sum=2 using integer 3

$3 - 3 = 0 \rightarrow \text{true}^1 \therefore$ we CAN form S=3 using z=3 (populate cell with 1) Note: $s(3) = 0$ and $s(0) = 1$

$4 - 3 = 1 \rightarrow \text{false}$ Note: $s(4) = 0$ and $s(1) = 0$

$5 - 3 = 2 \rightarrow \text{false}$ Note: $s(5) = 0$ and $s(2) = 0$

The emerging pattern can be summarized as: if the i th-1 cell is equal to true OR the difference between s and z for that particular row and column is true, then we take true. ie if $f(s) = 1 || f(s - z) = 1$. Therefore, our functional equation can be expressed as

$$f_i[s, z_i] = \max\{f_{i-1}(s), f_{i-1}(s - z_i)\}$$

If we find the last cell to be true, then it IS possible to form the value $\frac{S}{2}$ with the given assortment of Z .

¹We take a step up to the previous row and traverse to the left until we reach the column value that is equal to the difference between the sum and integer.