

Homework Assignment 5

Tyler Paulley

11/14/2016

Problem 1:

2. a. Write pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers.

b. Set up and solve (for $n = 2k$) a recurrence relation for the number of key comparisons made by your algorithm.

c. How does this algorithm compare with the brute-force algorithm for this problem?

a.

// Finds the values of the smallest and largest elements in an array

// Input: An array $A[0..n - 1]$ between indices left and right where $\text{left} \leq \text{right}$

// Output: Values of the smallest and largest elements in $A[0..n - 1]$

SearchMinMax($A[0..n - 1]$, min, max)

 if(right = left)

 min = $A[\text{left}]$;

 max = $A[\text{left}]$;

 else if(right - left = 1)

 if($A[\text{left}] \leq A[\text{right}]$)

 min = $A[\text{left}]$;

 max = $A[\text{right}]$;

 else

 min = $A[\text{right}]$;

 max = $A[\text{left}]$;

 else

 SearchMinMax($A[\text{left}..\text{floor}((\text{left} + \text{right}) / 2)$, min, max);

 SearchMinMax($A[\text{floor}((\text{left} + \text{right}) / 2) + 1..\text{right}]$, min2, max2);

 if(min2 < min)

 min = min2

 if(max2 > max)

 max = max2

b.

$$C(n) = 2C(n/2) + 2 \text{ for } n > 2, C(2) = 1, C(1) = 0$$

$$C(2^k) = 2C(2^{k-1}) + 2$$

$$= 2[2C(2^{k-2}) + 2] + 2 = 2^2C(2^{k-2}) + 2^2 + 2$$

$$= 2^2[2C(2^{k-3}) + 2] + 2^2 + 2 = 2^3C(2^{k-3}) + 2^3 + 2^2 + 2$$

$$= 2^iC(2^{k-i}) + 2^i + 2^{i-1} + \dots + 2$$

$$= 2^{k-1}C(2) + 2^{k-1} + \dots + 2 = 2^{k-1} + 2^k - 2$$

$$= (3n/2) - 2$$

c)

This algorithm makes a quarter of the comparisons of the brute force algorithm.

Problem 2:

5. Find the order of growth for solutions of the following recurrences.

a. $T(n) = 4T(n/2) + n$, $T(1) = 1$

b. $T(n) = 4T(n/2) + n^2$, $T(1) = 1$

c. $T(n) = 4T(n/2) + n^3$, $T(1) = 1$

Using the Master Theorem:

a. $T(n) = 4T(n/2) + n$.

$a = 4$, $b = 2$, $d = 1$.

$T(n) \in \Theta(n^{\log_2 4}) = \Theta(n^2)$.

b. $T(n) = 4T(n/2) + n^2$.

$a = 4$, $b = 2$, $d = 2$.

$T(n) \in \Theta(n^2 \log n)$.

c. $T(n) = 4T(n/2) + n^3$.

$a = 4$, $b = 2$, $d = 3$.

$T(n) \in \Theta(n^3)$.

Problem 3:

7. a. Estimate how many times faster quicksort will sort an array of one million random numbers than insertion sort.

b. True or false: For every $n > 1$, there are n -element arrays that are sorted faster by insertion sort than by quicksort?

a. Average Case for Insertion: $\Theta(n^2) = 10^{12}$

Average case for Quicksort: $\Theta(n \log n) = 19,931,569$

$10^{12} / 19,931,569 = 50171.6$ times faster

b. True, for each n , quicksort's average case will always consume less time than insertion sort's.

Problem 4:

2. The following algorithm seeks to compute the number of leaves in a binary tree.

ALGORITHM *LeafCounter*(T)

//Computes recursively the number of leaves in a binary tree

//Input: A binary tree T

//Output: The number of leaves in T

if $T = \emptyset$ return 0

else return *LeafCounter*(T_{left}) + *LeafCounter*(T_{right})

Is this algorithm correct? If it is, prove it; if it is not, make an appropriate correction.

The algorithm is not correct since it returns 0 rather than 1 for a one node binary tree.

After correcting the code it looks like this:

```
Algorithm LeafCounter (T)
//Computes recursively the number of leaves in a binary tree
//Input: A binary tree T
//Output: The number of leaves in T
if T =  $\emptyset$  return 0
else if TL =  $\emptyset$  and TR =  $\emptyset$  return 1
else return LeafCounter(TL)+ LeafCounter(TR)
```

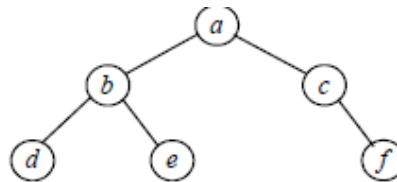
Problem 5:

5. Traverse the following binary tree

a. in preorder.

b. in inorder.

c. in postorder.



a. Preorder = a b d e c f

b. Inorder = d b e a c f

c. Postorder = d e b f c a

Problem 6:

2. Compute $2101 * 1130$ by applying the divide-and-conquer algorithm outlined in the text.

Looking at $2101 * 1130$:

$$c_2 = 21 * 11$$

$$c_0 = 01 * 30$$

$$c_1 = (21 + 01) * (11 + 30) - (c_2 + c_0) = 22 * 41 - 21 * 11 - 01 * 30.$$

Looking at $21 * 11$:

$$c_2 = 2 * 1 = 2$$

$$c_0 = 1 * 1 = 1$$

$$c_1 = (2 + 1) * (1 + 1) - (2 + 1) = 3 * 2 - 3 = 3.$$

$$21 * 11 = 2 \cdot 102 + 3 \cdot 101 + 1 = 231.$$

Looking at $01 * 30$:

$$c_2 = 0 * 3 = 0$$

$$c_0 = 1 * 0 = 0$$

$$c_1 = (0 + 1) * (3 + 0) - (0 + 0) = 1 * 3 - 0 = 3.$$

$$01 * 30 = 0 \cdot 102 + 3 \cdot 101 + 0 = 30.$$

Looking at $22 * 41$:

$$c2 = 2 * 4 = 8$$

$$c0 = 2 * 1 = 2$$

$$c1 = (2 + 2) * (4 + 1) - (8 + 2) = 4 * 5 - 10 = 10.$$

$$22 * 41 = 8 \cdot 102 + 10 \cdot 101 + 2 = 902.$$

Therefore...

$$2101 * 1130 = 231 \cdot 104 + (902 - 231 - 30) \cdot 102 + 30 = 2,374,130$$

Problem 7:

6. Verify the formulas underlying Strassen's algorithm for multiplying 2×2 matrices.

$$m_1 + m_4 - m_5 + m_7 =$$

$$(a_{00} + a_{11})(b_{00} + b_{11}) + a_{11}(b_{10} - b_{00}) - (a_{00} + a_{01})b_{11} + (a_{01} - a_{11})(b_{10} + b_{11}) =$$

$$a_{00}b_{00} + a_{11}b_{00} + a_{00}b_{11} + a_{11}b_{11} + a_{11}b_{10} - a_{11}b_{00} - a_{00}b_{11} - a_{01}b_{11} + a_{01}b_{10} -$$

$$a_{11}b_{10} + a_{01}b_{11} - a_{11}b_{11}$$

$$= a_{00}b_{00} + a_{01}b_{10}$$

$$m_3 + m_5 =$$

$$a_{00}(b_{01} - b_{11}) + (a_{00} + a_{01})b_{11} = a_{00}b_{01} - a_{00}b_{11} + a_{00}b_{11} + a_{01}b_{11} =$$

$$a_{00}b_{01} + a_{01}b_{11}$$

$$m_2 + m_4 =$$

$$(a_{10} + a_{11})b_{00} + a_{11}(b_{10} - b_{00}) = a_{10}b_{00} + a_{11}b_{00} + a_{11}b_{10} - a_{11}b_{00} =$$

$$a_{10}b_{00} + a_{11}b_{10}$$

$$m_1 + m_3 - m_2 + m_6 =$$

$$(a_{00} + a_{11})(b_{00} + b_{11}) + a_{00}(b_{01} - b_{11}) - (a_{10} +$$

$$a_{11})b_{00} + (a_{10} - a_{00})(b_{00} + b_{01}) =$$

$$a_{00}b_{00} + a_{11}b_{00} + a_{00}b_{11} + a_{11}b_{11} + a_{00}b_{01} - a_{00}b_{11} - a_{10}b_{00} - a_{11}b_{00} + a_{10}b_{00} -$$

$$a_{00}b_{00} + a_{10}b_{01} - a_{00}b_{01}$$

$$= a_{10}b_{01} + a_{11}b_{11}.$$

Problem 8:

1. a. For the one-dimensional version of the closest-pair problem, i.e., for the problem of finding two closest numbers among a given set of n real numbers, design an algorithm that is directly based on the divide-and-conquer technique and determine its efficiency class.
b. Is it a good algorithm for this problem?

a. We can find the closest pair by comparing the two closest points on the left half of the sorted list, the right half of the list, and the distance between the rightmost point in the first half and the leftmost point in the second half.

FindClosestPair($P[0..n - 1]$)

//Input: A subarray $P[l..r]$ ($l \leq r$) of a given array $P[0..n - 1]$

//Output: The distance between the closest pair of numbers

if $r = l$

return infinity

else if $r - l = 1$

return $P[r] - P[l]$

else

return $\min\{ \text{FindClosestPair}(P[l..\text{floor}((l + r)/2)]), \text{FindClosestPair}(P[(\text{floor}((l + r)/2 + 1)..r]), P[\text{floor}((l + r)/2 + 1)] - P[\text{floor}((l + r)/2)] \}$

The recurrence relation is $T(n) = 2T(n/2) + c$.

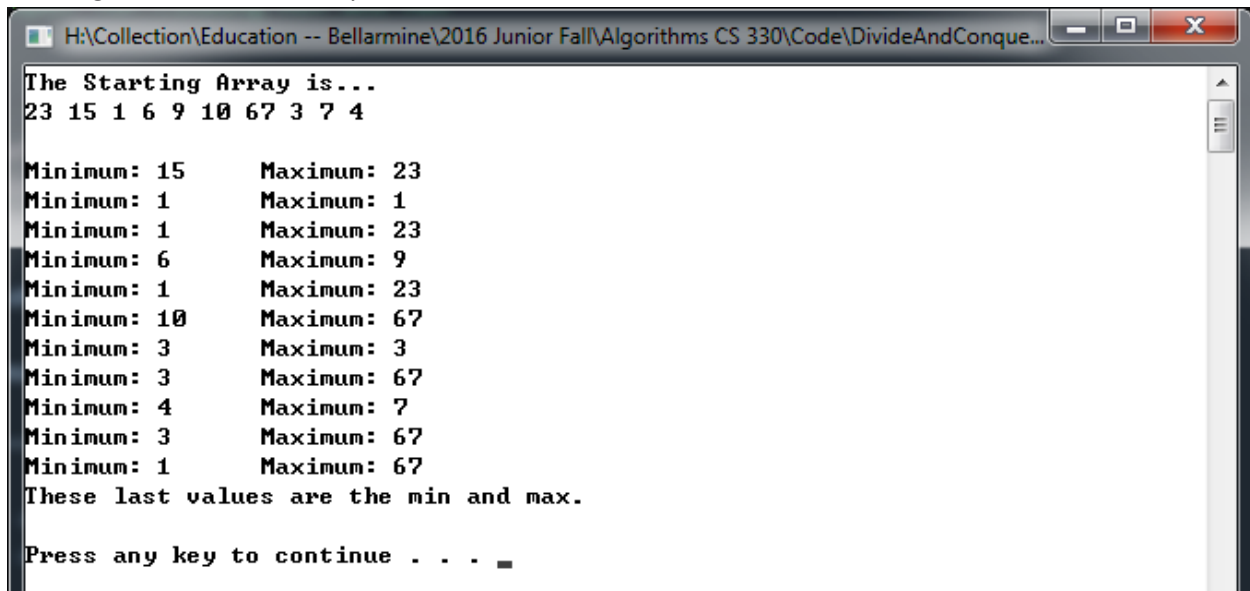
Using the master theorem, its solution is $\Theta(n \log 2) = \Theta(n)$.

However, if we are going to add in the time it takes to sort the list (mergesort) then the efficiency will be: $\Theta(n \log n) + \Theta(n) = \Theta(n \log n)$.

b. It is possible that we could sort the list while comparing the distances between the points however this would result in the same time efficiency. Because of this, the algorithm is efficient.

Code for Problem 1

Running the code for an array of size 10



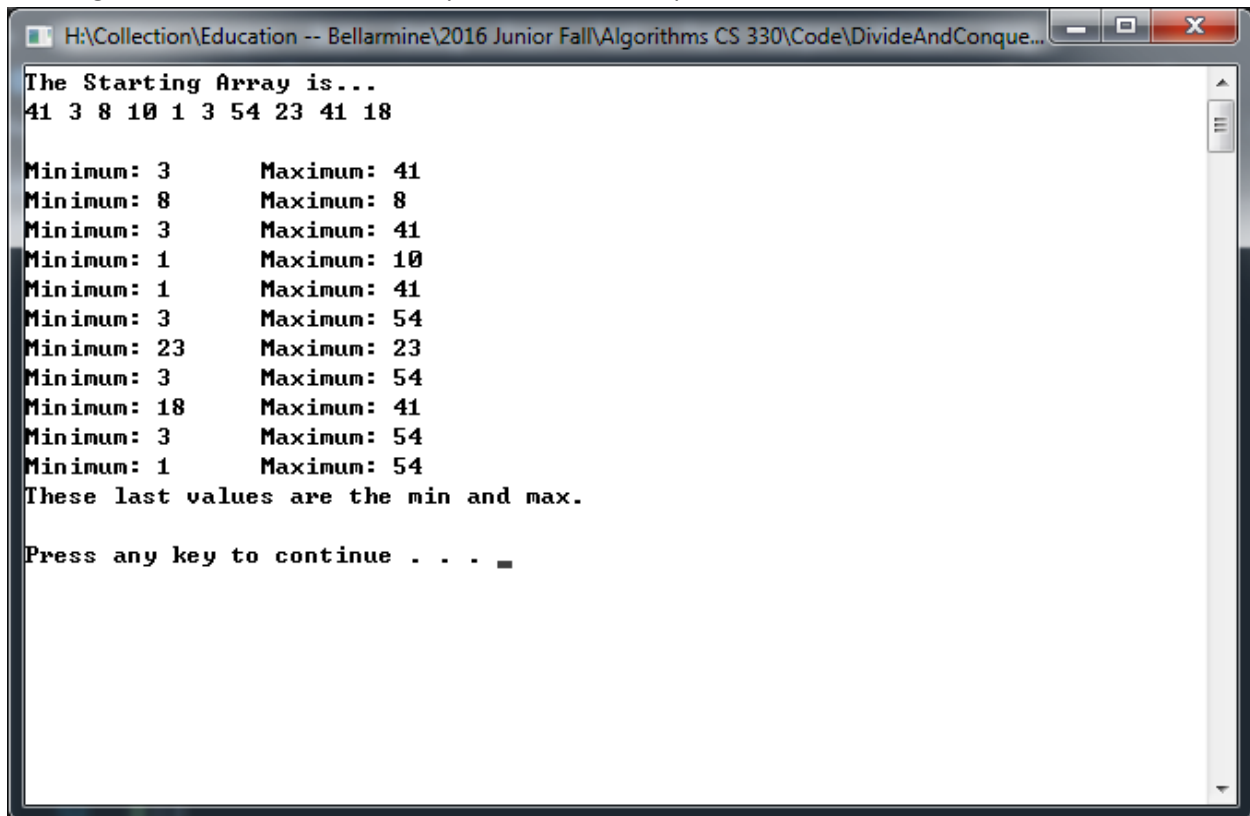
The screenshot shows a Windows command prompt window with the title bar "H:\Collection\Education -- Bellarmine\2016 Junior Fall\Algorithms CS 330\Code\DivideAndConque...". The window contains the following text:

```
The Starting Array is...
23 15 1 6 9 10 67 3 7 4

Minimum: 15      Maximum: 23
Minimum: 1       Maximum: 1
Minimum: 1       Maximum: 23
Minimum: 6       Maximum: 9
Minimum: 1       Maximum: 23
Minimum: 10      Maximum: 67
Minimum: 3       Maximum: 3
Minimum: 3       Maximum: 67
Minimum: 4       Maximum: 7
Minimum: 3       Maximum: 67
Minimum: 1       Maximum: 67
These last values are the min and max.

Press any key to continue . . .
```

Running the code for a different array of size 10 with duplicates



The screenshot shows a Windows command prompt window with the title bar "H:\Collection\Education -- Bellarmine\2016 Junior Fall\Algorithms CS 330\Code\DivideAndConque...". The window contains the following text:

```
The Starting Array is...
41 3 8 10 1 3 54 23 41 18

Minimum: 3      Maximum: 41
Minimum: 8      Maximum: 8
Minimum: 3      Maximum: 41
Minimum: 1      Maximum: 10
Minimum: 1      Maximum: 41
Minimum: 3      Maximum: 54
Minimum: 23     Maximum: 23
Minimum: 3      Maximum: 54
Minimum: 18     Maximum: 41
Minimum: 3      Maximum: 54
Minimum: 1      Maximum: 54
These last values are the min and max.

Press any key to continue . . .
```