CS545 Spring, 2015

Homework Assignment #2
Due: Feb 26
Student: Shuo Yang

---

1. **Algorithm**

Function $FindClosePair(A[1:n])$
$distance := ComputeCloseDistance(A[1:n])$
$(x, y) := ComputeClosePair(A[1:n], distance)$

Function $ComputeCloseDistance(A[1:n])$
Linearly scan the input array $A[1:n]$ from left to right, find the minimum and maximum numbers, put them into variables min and max respectively, and let
$distance := \lfloor (max - min)/(n-1) \rfloor$.

Function $ComputeClosePair(A[1:n], distance)$
(1) Linearly scan the input array A from left to right, find the minimum and maximum numbers, put them into variables min and max respectively.
(2) Divide the n elements into $\lfloor n/5 \rfloor$ groups of 5 elements, and ≤ 1 group of < 5 elements.
(3) Find the median of each group using selection sort.
(4) Recursively find the median med of the $\lfloor n/5 \rfloor$ medians found in step (3).
(5) Partition the input array A around med from step (4) into two subarrays A_{low} and A_{high} such that A_{low} contains all elements $\leq med$ and A_{high} contains all elements $\geq med$.
(6) **if** $med - min \leq distance$
return (med, min)
if $max - med \leq distance$
return (max, med)
if n is odd
if $max - med \geq med - min$
$(x, y) := ComputeClosePair(A_{low}, distance)$
else
$(x, y) := ComputeClosePair(A_{high}, distance)$
else // n is even
if $max - med - distance \geq med - min$ // med is lower median
$(x, y) := ComputeClosePair(A_{low}, distance)$
else
$(x, y) := ComputeClosePair(A_{high}, distance)$

Run time analysis
The sub-routine $ComputeCloseDistance$ takes $\Theta(n)$ time. Let the total run time for $ComputeClosePair$ be $T(n)$ where n is the number of elements in the input array A. Step (1) takes $\Theta(n)$ time, step (2) takes $\Theta(n)$ time, step (3) takes $\Theta(n)$ time, step (4) takes $T(\lfloor (n/5) \rfloor)$ time, step (5) takes $\Theta(n)$ time, step (6) takes $T(n/2)$ time because for each recursive call, we reduce the size of the input for the sub-problem into half, that is, either recurse on lower partition or higher partition. Thus, we have:

$$\begin{aligned} T(n) &= T(n/5) + T(n/2) + \Theta(n) \\ &= \Theta(n) \end{aligned} \tag{1}$$

So the total run time for $FindClosePair$ is $T(n) + \Theta(n) = \Theta(n)$.

Correctness

Lemma1: Given an unsorted array A of n distinct numbers, a close pair always exists.

Proof. Prove by contradiction, that is, no such a close pair exists, this means for any pair (x, y) where $x > y$, we have:

$$x - y > \frac{1}{n-1}(max - min) \tag{2}$$

Suppose we order numbers in the array A in ascending order as a sequence: $a_1, a_2, \cdots, a_n$ where $a_i > a_j$ if $i > j$. Let $distance = \frac{1}{n-1}(max - min)$, according to the assumption, we have:

$$\begin{aligned} a_n - a_{n-1} &> distance \\ a_{n-1} - a_{n-2} &> distance \\ &\cdots \\ a_3 - a_2 &> distance \\ a_2 - a_1 &> distance \end{aligned} \tag{3}$$

Summing the above equations together produces:

$$\begin{aligned} a_n - a_1 &> distance \times (n-1) \\ &= \frac{1}{n-1}(max - min) \times (n-1) \\ &= max - min \end{aligned} \tag{4}$$

Since we have sorted the array A in ascending order, this means that $a_n = max$ and $a_1 = min$, therefore the above equation says that $max - min > max - min$, this is clearly a contradiction, thus the lemma1 must be true. □

Lemma2: Given an unsorted array A of n distinct numbers partitioned around its median into two subarrys A_{low} and A_{high}. A_{low} contains all elements $\leq$ median and A_{high} contains all elements $\geq$ median. Let the maximum element be max, minimum element be min and median be med. A close pair must exist in A_{low} if $max - med \geq med - min$ (when n is odd) or $max - med - distance \geq med - min$ (when n is even), and in A_{high} otherwise.

Proof. Prove by contradition. If $max - med \geq med - min$ (when n is odd) or $max - med - distance \geq med - min$ (when n is even), assume that A_{low} does not contain any close pairs. There are two cases to consider:

(a) n is odd.
In this case, A_{low} contains $\frac{n-1}{2} + 1 = \frac{n+1}{2}$ elements, including the med itself. Applying the same method used in proving Lemma1, we sort A_{low} as: $a_0, a_1, \cdots, a_{(n+1)/2}$ in ascending order. According to the assumption, we must have,

$$\begin{aligned} a_{\frac{n+1}{2}} - a_{\frac{n+1}{2}-1} &> distance \\ &\cdots \\ a_3 - a_2 &> distance \\ a_2 - a_1 &> distance \end{aligned} \tag{5}$$

Summing the above equations together produces:

$$\begin{aligned} a_{\frac{n+1}{2}} - a_1 &> distance \times (\frac{n+1}{2} - 1) \\ &= \frac{1}{n-1}(max - min) \times (\frac{n+1}{2} - 1) \\ &= \frac{1}{n-1}(max - min) \times \frac{n-1}{2} \\ &= \frac{max - min}{2} \end{aligned} \tag{6}$$

Since A_{low} is sorted in ascending order, $a_{\frac{n+1}{2}} = med$ and $a_1 = min$. Thus,

$$\begin{aligned} med - min &> \frac{max - min}{2} \\ med &> \frac{max + min}{2} \end{aligned} \tag{7}$$

And because $max - med \geq med - min$, we have $med \leq \frac{max+min}{2}$. But we have just proved that $med > \frac{max+min}{2}$, clearly it is a contradiction. Thus, there must exist a close pair in A_{low}.

(b) n is even.

In this case, there are two medians, left median and right median. Assume that we pick the low median. A_{low} contains $\frac{n}{2}$ elements, including the med itself. Applying the same method used in proving Lemma1, we sort A_{low} as: $a_1, a_2, \cdots, a_{n/2}$ in ascending order. According to the assumption, we must have,

$$\begin{aligned} a_{\frac{n}{2}} - a_{\frac{n}{2}-1} &> distance \\ &\cdots \\ a_3 - a_2 &> distance \\ a_2 - a_1 &> distance \end{aligned} \tag{8}$$

Summing the above equations together produces:

$$\begin{aligned} a_{\frac{n}{2}} - a_1 &> distance \times (\frac{n}{2} - 1) \\ &= \frac{1}{n-1}(max - min) \times (\frac{n}{2} - 1) \\ &= \frac{max - min}{2} \times \frac{n-2}{n-1} \end{aligned} \tag{9}$$

Since A_{low} is sorted in ascending order, $a_{\frac{n}{2}} = med$ and $a_1 = min$, we have:

$$\begin{aligned} med - min &> \frac{max - min}{2} \times \frac{n-2}{n-1} \\ med &> \frac{max + min}{2} - \frac{max - min}{2(n-1)} \end{aligned} \tag{10}$$

Substituting $\frac{max-min}{n-1}$ with $distance$, we have:

$$med > \frac{max + min}{2} - \frac{distance}{2} \tag{11}$$

And because $max - med - distance \geq med - min$, we have $med \leq \frac{max+min}{2} - \frac{distance}{2}$. But we have just proved that $med > \frac{max-min}{2} - \frac{distance}{2}$, clearly it is a contradiction. Thus, there must exist a close pair in A_{low}.

The proof for the case when $max - med < med - min$ (when n is odd) or $max - med - distance < med - min$ (when n is even) is symmetrical. □

So combining Lemma1 and Lemma2, we can conclude that our algorithm can always find a pair of close elements.

2.

3. **Algorithm**

Function $FindSmallestInMerge(A[1:m], B[1:n], k)$
$medA := \lfloor m/2 \rfloor$ // index of median of A
$medB := \lfloor n/2 \rfloor$ // index of median of B
$medM := medA + medB$ // index of median of merge of A and B
if $A[medA] > B[medB]$
if $k == medM$
return $A[medA]$
else if $k < medM$
$x := FindSmallestInMerge(A[1 : medA - 1], B[1 : medB], k)$
else // $k > medM$
$x := FindSmallestInMerge(A[medA + 1 : m], B[medB + 1 : n], k - medM)$
else // $A[medA] < B[medB]$
if $k == medM$
return $B[medB]$
else if $k < medM$
$x := FindSmallestInMerge(A[1 : medA], B[1 : medB - 1], k)$
else // $k > medM$
$x := FindSmallestInMerge(A[medA + 1 : m], B[medB + 1 : 1], k - medM)$

Run time analysis

In each recursive call, we reduce the problem into half of its original size, and other operations executes in constant time, thus the recurrence equation is:

$$\begin{aligned} T(m+n) &= T(\frac{m+n}{2}) + \Theta(1) \\ &= \Theta(\log(m+n)) \end{aligned} \tag{12}$$