

Instructions:

Set testArray[] to any of the above sets. Next, run the program in an IDE or command prompt, or however you execute Java files.

The program will output the median, if there is one.

//This algorithm fins the median of an array with i and n as inputs.

int findMedian(int[] a, int i,int n) {

Arrays.sort(a, i, i + n);

return a[i + n / 2]; // Return middle element

}

//This algorithm finds the smallest element at index with an array, left, right, and index as inputs

int smallestAtIndex(int[] a, int left, int right, int index) {

if (index > 0 && index <= right - left + 1) {

int n = right - left + 1 ; // Number of elements in a[left..right]

int i;

for (i = 0; i < n/5; i++) {

median[i] = findMedian(a, left + i \* 5, 5);

}

if (i\*5 < n) {

median[i] = findMedian(a, left + i \* 5, n % 5);

i++;

}

int medOfMed = median[i - 1];

if (i != 1)

medOfMed = smallestAtIndex(median, 0, i - 1, i / 2);

int pos = partition(a, left, right, medOfMed);

if (pos-left == index - 1)

return a[pos];

if (pos-left > index - 1) // If position is more, recur for left

return smallestAtIndex (a, left, pos - 1, index);

return smallestAtIndex (a, pos + 1, right, index - pos + left - 1);

}

return -1234567;

}

//This algorithm partitions with an array, left, right, and x as inputs

int partition(int[] a, int left, int right, int x) {

int i;

int temp;

for (i = left; i < right; i++) {

if (a[i] == x)

break;

}

temp = a[i];

a[i] = a[right];

a[right] = temp;

i = left;

for (int j = left; j < right; j++) {

if (a[j] <= x) {

temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

}

}

temp = a[i];

a[i] = a[right];

a[right] = temp;

return i;

}

main {

int[] testArray = {8, 1, 2, 45, 2, 11, 86, 21, 22, 11, 3, 4, 99, 34, 1294, 13, 1100};

int n = testArray.length;

int medIndex = n/2;

if (n%2 == 0) {

int x = smallestAtIndex(testArray, 0, n-1, medIndex);

int y = smallestAtIndex (testArray, 0, n-1, (medIndex + 1));

if (x != -1234567 && y != -1234567)

Print("Median is "+

+ (double)(x+y)/2);

else

Print("Error -- list is empty!");

} else {

medIndex = n/2 + 1;

int median = smallestAtIndex (testArray, 0, n-1, medIndex);

if (median != -1234567)

Print("Median is "+

+ median);

else

Print("Error -- list is empty!");

}

}

Test Cases:

{1, 2, 6, 11, 14, 14, 20, 25}

{}

{14, 6, 2, 25, 11, 14, 20, 1}

{2, 1}

{6}

{1, 2, 3}

{8, 1, 2, 45, 2, 11, 86, 21, 22, 11, 3, 4, 99, 34, 1294, 13, 1100}

{4, 4, 4, 4, 4, 4}

{0, -1, 2, -5, -10, 4}

{-2, -6, -6, -3, -78, -23, -8}

{-9, -4, -14, -5, -1, -3}

The recurrence for our function looks like this:

Using the Master Theorem:

Any recurrence in the form ,

if , the recurrence will become

In our case,