Name\_\_\_\_\_\_Kan Katpark\_\_\_\_\_\_\_\_\_\_ID\_\_\_\_\_\_642115003\_\_\_\_\_Group\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tutorial 4 Algorithm Efficiency and Midterm review**

1. What is the big-O of the following snippet

1.1

|  |
| --- |
| **int** result = 0  **int** i = 1  **while** i < n  **if** n % i == 0  result += i  **end**  i += 1  **end**  **return** result |

O(n)

1.2

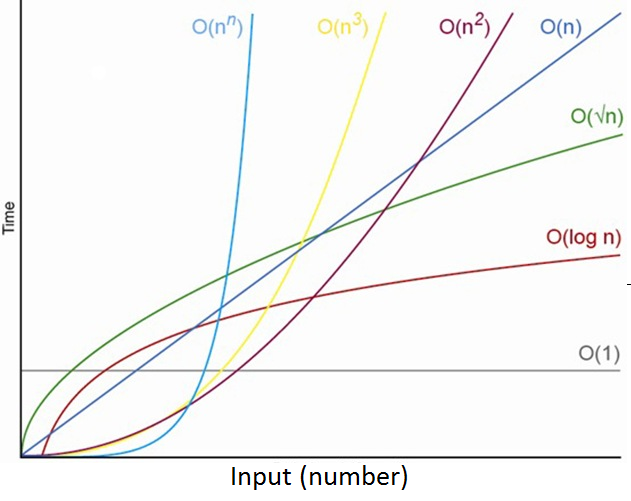
|  |
| --- |
| **if** array[0] == null  return true  **else**  return false  **end** |

O(1)

1.3

|  |
| --- |
| **public** static **int** doSomething(**int**[] arr, **int** x){    **int** size = arr.length;  **for**(**int** i=0;i<size;i++){  if(arr[i] ==x){  **return** i;  }  }  **return** -1;  } |

O(n)



1.4 According to above comparison figure, which function represents the fastest algorithm?

O(1)

1. Write a Java program that read a data file (you can download from the link here <https://www.dropbox.com/s/chnpp0kkvpbbyfb/data.txt?dl=0>

Your program must have a method call “mySearch” which responses to find for all the value in the given data file that are greater than 0.5. Below is an example output of the program from a different data file.

|  |
| --- |
| >Total number of values read: 15103  >Number of value > 0.5 is: 1343 |

What is the Big O of your method mySearch?

n + n log n



Working process

Read data -> sort data with quick sort -> create sub array of data starting with index of last number that smaller than 0.5 to end of array -> display sub array size

Copy and paste your java source code here

|  |
| --- |
| import java.util.\*;  import java.io.\*;  public class MySearch {      public static void main(String[] args) {          String location = "data.txt";          File file = new File(location);          Vector<Double> data = new Vector<Double>();          readData(data, file);          System.out.println("Total data number : " + data.size());          quickSort(data, 0, data.size());          int index = search(data, 0.5 );          Vector<Double> greaterThanTarget = new Vector<Double>(data.subList(index,data.size()));          System.out.println("Greater than 0.5 value is "+ greaterThanTarget.size());      }// end main      public static int search(Vector<Double> data , double target) {          for(int i = 0 ; i < data.size() ; i++){              if(data.elementAt(i) > target) return i ;          }          return 0 ;      }// end search      public static void quickSort(Vector<Double> data, int l, int r) {          if (l >= r)              return;          int part = partition(data, l, r);          quickSort(data, l, part);          quickSort(data, part+1, r);      } // end quick sort      public static int partition(Vector<Double> data, int l, int r) {          int i = l - 1;          double pivot = data.elementAt(r - 1);          for (int j = l; j < r - 1; j++) {              if (data.elementAt(j) < pivot) {                  i++;                  swap(data, i, j);              }          }          swap(data, i + 1, r - 1);          return i + 1;      } // end partition      public static void swap(Vector<Double> data, int i, int j) {          double temp = data.elementAt(i);          data.set(i, data.elementAt(j));          data.set(j, temp);      } // end swap      public static void readData(Vector<Double> data, File file) {          try {              Scanner sc = new Scanner(file);              while (sc.hasNextLine()) {                  double num = Double.parseDouble(sc.nextLine());                  data.addElement(num);              }          } catch (FileNotFoundException e) {              System.out.println(e);          }      }  } // end read data |