Programming Assignment 2 WriteUp

Daniel Quiroga
Algorithms

April 16, 2019

1 Code

```
import java.util.*;
import java.text.*;
import java.lang.*;

// used JAVA SE 11

public class PA2 {
        public static void main(String[] args){
                Scanner array = new Scanner(System.in);
                System.out.println("Input an array of at least 2 elements:
                 ↪  ");
                String str = array.nextLine();
                // System.out.println(str);

                str = str.replace("[", "");
                str = str.replace("]", "");
                str = str.replace(" ", "");
                // System.out.println(str);

                String[] arr = str.split(",");

                // System.out.println(arr.length);

                while (arr.length <= 1){
                        System.out.println("Input an array of at least 2
                         ↪  elements: ");
                        str = array.nextLine();
                        // System.out.println(str);

                        str = str.replace("[", "");
                        str = str.replace("]", "");
                        str = str.replace(" ", "");
                        // System.out.println(str);

                        arr = str.split(",");

                        // System.out.println(arr.length);
                }

                // for (String s : arr){
                //         System.out.println(s);
                // }
                List a = new ArrayList();
                for (String i : arr){
                        a.add(Integer.parseInt(i));
                }
                // System.out.println(a);

                Scanner num = new Scanner(System.in);
                System.out.println("What is your k?");
                int k = (int) num.nextDouble();
                while (k < 1 || k > arr.length) {
                        System.out.println("What is your k?");
                        k = (int) num.nextDouble();
                }
                Sort s = new Sort();
                System.out.println("Select: " + s.select((ArrayList) a, k));
                System.out.println("Det_select: " + s.det_select((ArrayList)
                 ↪  a, k));
                System.out.println("Quicksort: " + s.quicksort((ArrayList)
                 ↪  a, k));
        }
}

class Sort {
        Sort(){
                int i = 0;
                List A = new ArrayList();
                Random random = new Random();

                while (i != 10000000){
                        A.add(random.nextInt(100000));
                        i++;
                }
                System.out.println("Testing my sorting alogirthms with an
                 ↪  enormous array with random numbers inputted");

                long startSelect = System.nanoTime();
                int select = select((ArrayList) A, (int) Math.ceil(((double)
                ↪  i) / 2));
                long endSelect = System.nanoTime();
                System.out.println("Select: " + select + " in " +
                 ↪  ((endSelect - startSelect)/1000000000) + " seconds.");

                long startDet = System.nanoTime();
                int det = det_select((ArrayList) A, (int)
                 ↪  Math.ceil(((double) i) / 2));
                long endDet = System.nanoTime();
                System.out.println("Det_select: " + det + " in " + ((endDet
                 ↪  - startDet)/1000000000) + " seconds. ");

                long startQuick = System.nanoTime();
                int quick = quicksort((ArrayList) A, (int)
                 ↪  Math.ceil(((double) i) / 2));
                long endQuick = System.nanoTime();
                System.out.println("Quicksort: " + quick + " in " +
                 ↪  ((endQuick - startQuick)/1000000000) + " seconds. " );
        }


        // Algorithm 1
        public int select(ArrayList S, int k){
                Random rand = new Random();
                int split = rand.nextInt(S.size());
                int splitter = (int) S.get(split);

                List lower = new ArrayList();
                List upper = new ArrayList();
                List equal = new ArrayList();

                for (int count = 0 ; count < S.size(); count++){
                        if ((int) S.get(count) < splitter){
                                lower.add((int) S.get(count));
                        }
                        else if ((int) S.get(count) > splitter){
                                upper.add((int) S.get(count));
                        }
                        else if ((int) S.get(count) == splitter){
                                equal.add((int) S.get(count));
                        }

                }

                if (lower.size() == k - 1){
                        return splitter;
                }

                else if (lower.size() >= k){
                        return select((ArrayList) lower, k);
                }

                // else if (lower.size() < k - 1){
                else{
                        if (lower.size() + equal.size() >= k){
                                return splitter;
                        }
                        return select((ArrayList) upper, k - lower.size() -
                         ↪  equal.size());
                }

        }



        // Algorithm 2
        public int det_select(ArrayList S, int k){
                if (S.size() <= 10) {
                        return quicksort(S, k);
                }
                List x = new ArrayList();
                List array = new ArrayList();

                for (int i = 0; i < S.size(); i++){
                        if(array.size() < 5 && i != S.size() - 1){
                                array.add(S.get(i));
                        }

                        if (array.size() == 5){
                                x.add(array);
                                array = new ArrayList();
                        }
                        else if (i == S.size() - 1){
                                array.add(S.get(i));
                                if (S.size() != 0){
```

```
                        x.add(array);
                }
        }
}

int count;
List X = new ArrayList();
for(count = 0; count < x.size(); count++){
        if (((ArrayList) x.get(count)).size() == 5){
                X.add(det_select((ArrayList) x.get(count),
                 ↪  3));
        }
        else{
                //
                 ↪  System.out.println(Math.ceil((double)((ArrayList)x.
                 ↪  / 2));
                //
                 ↪  System.out.println(((ArrayList)x.get(count)).size()
                 ↪  / 2);
                X.add(det_select((ArrayList) x.get(count),
                 ↪  (int)
                 ↪  Math.ceil((double)((ArrayList)x.get(count)).size()
                 ↪  / 2)));
        }
}

// System.out.println(S + " "+  k);

// System.out.println("x " + X);

int m = det_select((ArrayList) X, (int)
 ↪  Math.ceil(S.size()/10));

// System.out.println("M " + m);

List lower = new ArrayList();
List upper = new ArrayList();
List equal = new ArrayList();


for (count = 0 ; count < S.size(); count++){
        if ((int) S.get(count) < m){
                lower.add((int) S.get(count));
        }
        else if ((int) S.get(count) > m){
```

```
                                upper.add((int) S.get(count));
                        }
                        else if ((int) S.get(count) == m){
                                equal.add((int) S.get(count));
                        }

                }

                // System.out.println("lower: " + lower );
                // System.out.println("upper: " + upper );
                // System.out.println("equal: " + equal );

                if (k <= lower.size()){
                        return det_select((ArrayList) lower, k);
                }

                else if (k > lower.size() + equal.size()){
                        // System.out.println((lower.size() +
                         ↪  equal.size()));
                        return det_select((ArrayList) upper, k -
                         ↪  lower.size() - equal.size());
                }
                else{
                        return m;
                }

        }



        // Algorithm 3
        public int quicksort(ArrayList S, int k){
                List q = quicksortAlgo(S);
                return (int) q.get(k-1);
        }

        private ArrayList quicksortAlgo(ArrayList S){
                List lower = new ArrayList();
                List upper = new ArrayList();
                List equal = new ArrayList();

                if (S.size() <= 3){
                        Collections.sort(S);
                        return S;
                }
                else{
                        Random rand = new Random();
                        int split = rand.nextInt(S.size());
                        int splitter = (int) S.get(split);
                        for (int count = 0 ; count < S.size(); count++){
                                if ((int) S.get(count) < splitter){
                                        lower.add((int) S.get(count));
                                }
                                else if ((int) S.get(count) > splitter){
                                        upper.add((int) S.get(count));
                                }
                                else if ((int) S.get(count) == splitter){
                                        equal.add((int) S.get(count));
                                }

                        }
                        List combine = new ArrayList();
                        combine.addAll(quicksortAlgo((ArrayList)lower));
                        combine.addAll(equal);
                        combine.addAll(quicksortAlgo((ArrayList)upper));
                        // System.out.println(combine);
                        return (ArrayList)combine;
                        // return quicksortAlgo((ArrayList)lower), equal,
                         ↪  quicksortAlgo((ArrayList)upper);
                }
        }

}
```

2 Report

To begin I made sure that my program would be able to not only take in an array as an input but also make sure to keep asking if the array was not larger than size one, since this would be sorted already. I then made sure that the numbers in the input would be integers, if not then an error would arise when my array list would try to parseInt() that is not an int. Then I proceeded to get the k value from the user. I created a class Sort, that would run the test case given in the project spec of the extremely large n and finding the median of the entire list. This would be run in the constructor of the class, also all the times I ran my program the number that would be returned from each of the sorting algorithms were extremely close to the actual middle value that I would expect. After an object of class Sort was created, called each of the functions in the class (random selection, deterministic, and quicksort) to sort the inputted array and find the kth element (inputted as well by the user). This would then print out the number with the appropriate format. Inside the constructor of Sort, I tested larger n's and smaller n's and each time random selection would finish up the quickest.

I believe that this algorithm preforms best even when you make the size larger. This may be because it chooses a splitter randomly and because of that we expect the splitter to usually be around the middle of the list making the sort preform quicker.

3 Output from terminal

```
Daniels-MacBook-Pro:303 danielquiroga javac PA2.java
Note: PA2.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Daniels-MacBook-Pro:303 danielquiroga java PA2
Input an array of at least 2 elements:
[7, 2, 4, 6, 9, 11, 2, 6, 10, 6, 15, 6, 14, 2, 7, 5, 13, 9, 12, 15]
What is your k?
10
Testing my sorting alogirthms with an enormous array with random numbers inputted
Select: 49991 in 2 seconds.
Det select: 49991 in 7 seconds.
Quicksort: 49991 in 14 seconds.
Select: 7
Det select: 7
Quicksort: 7
```