TIMOTHY ALLEC

Name, SID, Date .....

In Class Assignment 4: Randomized Select

Benjamin Sanders, MS November 25, 2020

1 Introduction

You will need to work individually to complete this assignment. Write your name at the top of all pages for this assignment. Turn in all work to Blackboard on or before the deadline to receive credit.

You may use additional libraries and online resources, if you get them approved in writing, over email, from the instructor first. If you have received approval from the instructor, write the approved libraries and any references in the space below.

import java.util.Random;

.....
.....
.....
.....

2 Assignment Description

2.1 Big Picture

This algorithm introduces randomness to achieve a faster time complexity than using a repeated search for the minimum to return the ith smallest element of an array.

2.2 Algorithm Implementation

Implement the following algorithm in Java, using the Vector data structure for any 1-D array, 2-D array, or linear algebra purposes.

RANDOMIZED-SELECT(A, p, r, i)

```
1  if p == r
2      return A[p]
3  q = RANDOMIZED-PARTITION(A, p, r)
4  k = q − p + 1
5  if i == k            // the pivot value is the answer
6      return A[q]
7  elseif i < k
8      return RANDOMIZED-SELECT(A, p, q − 1, i)
9  else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

RANDOMIZED-PARTITION(A, p, r)

```
1  i = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

RANDOMIZED-SELECT returns the ith smallest element of the array $A[p..r]$.

Name, SID, Date .....

The key to the algorithm is the PARTITION procedure, which rearranges the subarray $A[p..r]$ in place.

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

2.3 Time Complexity Analysis

Where n is the number of data points in A, analyze the time complexity of the given algorithm with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n^2)$ in the space below.

o (n log(n))

2.4 Space Complexity Analysis

Where n is the number of data points in A, analyze the space complexity of the given algorithm with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n \cdot log(n))$ in the space below.

o(n)

2.5 Optimize the Algorithm for a Purpose

Choose an optimization, either in time or in space, for the given algorithm. Write your intended big-O notation, i.e. $O(1)$, $O(n)$, etc., in the space below, and write N/A in the other space.

- Time Complexity: N/A
- Space Complexity: O(n) In the Partition method, you can avoid unnecessary swaps when the elements are already in the correct order. This can be done by checking if a swap is necessary before performing it.

What application would benefit from the purpose of the above optimization? Why? Write two sentences to answer these questions in the space below.

applications that would work for this would be for searching through a list of names

Name, SID, Date .....

2.6 New Algorithm Design and Implementation

In the space below, design an algorithm that achieves the same purpose of the given algorithm, but includes the optimization you have specified above. Use pseudocode written in a style similar to the given algorithm, and implement it in Java. You may use as many additional pages as necessary for this purpose.

This algorithm optimizes the Partition method by ensuring that unneccesary swaps do not occur by adding the if(i!=j) statement

```
import java.util.Random;

public class RandomizedSelect {
   // this method performs Randomized select
   public static int Function(int[] A, int p, int r, int i) {
      if (p == r) {
         return A[p];
      }
      int q = RandomizedPartition(A, p, r);
      int k = q - p + 1;
      if (i == k) {
         return A[q];
      } else if (i < k) {
         return Function(A, p, q, i);
      } else {
         return Function(A, q + 1, r, i - k);
      }
   }
 public static int RandomizedPartition(int[] A, int p, int r) {
      // implementation of the method
      Random myRandom = new Random(System.currentTimeMillis());
      int i = p + myRandom.nextInt(r - p + 1);
      swap(A, i, r);
      return Partition(A, p, r);
   }
   public static int Partition(int[] A, int p, int r) {
      int x = A[r];
      int i = p - 1;
      for (int j = p; j <= r - 1; j++) {
         if (A[j] <= x) {
            i = i + 1;
            if (i != j) {
               swap(A, i, j);
            }
         }
      }
      swap(A, i + 1, r);
      return i + 1;
   }

 public static void swap(int[] A, int i, int j) {
      int temp = A[i];
      A[i] = A[j];
      A[j] = temp;
   }

   public static void main(String[] args) {
      int[] newArray = {7, 3, 6, 0, 8, 4};
      int nth = 4;
      int nthSmallest = Function(newArray, 0, newArray.length - 1, nth);
      System.out.println("The " + nth + "th smallest element is: " + nthSmallest);
   }
}
```

3 What to Turn In

Turn in one PDF or Word document on Blackboard, containing the following items.

1. All pages scanned or photographed of the In Class Assignment completed document.
2. Any additional pages you used to complete the assignment.
3. All code created for the assignment, along with test cases.
4. One statement indicating which parts of your implementation(s) are working, and which parts are not.
5. Screenshots demonstrating the code working, if it is working.