timothy allec

Name, SID, Date .....

In Class Assignment 12: Randomized Quicksort

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.concurrent.ThreadLocalRandom;

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

2 Assignment Description

2.1 Big Picture

Randomized Quicksort is a variable time-complexity algorithm. Often, Randomized Quicksort beats Quicksort when the likelihood that the unsorted data is distributed in a perfectly randomized way is low.

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-PARTITION(A, p, r)

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

The new quicksort calls RANDOMIZED-PARTITION in place of PARTITION:

RANDOMIZED-QUICKSORT(A, p, r)

1 **if** $p < r$
2 q = RANDOMIZED-PARTITION(A, p, r)
3 RANDOMIZED-QUICKSORT$(A, p, q-1)$
4 RANDOMIZED-QUICKSORT$(A, q+1, r)$

```
public class RandomizedQuicksort {
    public static int[] randomizedPartition(int[] A, int p, int r) {
        int i = (int) (Math.random() * (r - p + 1)) + p;
        int temp = A[i];
        A[i] = A[r];
        A[r] = temp;
        return partition(A, p, r);
    }

    public static void randomizedQuicksort(int[] A, int p, int r) {
        if (p < r) {
            int[] q = randomizedPartition(A, p, r);
            randomizedQuicksort(A, p, q[0] - 1);
            randomizedQuicksort(A, q[1] + 1, 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; j++) {
            if (A[j] <= x) {
                i++;
                int temp = A[i];
                A[i] = A[j];
                A[j] = temp;
            }
        }
        int temp = A[i + 1];
        A[i + 1] = A[r];
        A[r] = temp;
        int[] indices = {i + 1, i + 1}; // Initialize both indices to i + 1
        // Adjust the indices to cover equal elements
        for (int k = i; k >= p; k--) {
            if (A[k] == A[i + 1]) {
                indices[0] = k;
            } else {
                break;
            }
        }
        for (int k = i + 2; k <= r; k++) {
            if (A[k] == A[i + 1]) {
                indices[1] = k;
            } else {
                break;
            }
        }
        return indices;
    }

    public static void main(String[] args) {
        int[] A  = {1,3,10,2,7,5,4,6,9,8};
        randomizedQuicksort(A, 0, A.length - 1);
        for (int i : A) {
            System.out.print(i + " ");
        }
    }
}
```

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^2)

.....

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)

Name, SID, Date .....

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: O(n^2)
- Space Complexity: O(n)

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

Database Systems: Randomized Quicksort can improve the efficiency of sorting operations in database systems, especially for large-scale datasets. It can enhance query processing, indexing, and data retrieval tasks, leading to faster response times and improved overall system performance.

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.

```
import java.util.concurrent.ThreadLocalRandom;

public class optimization {
    public static void randomizedQuicksort(int[] A, int p, int r) {
        if (p < r) {
            int[] q = randomizedPartition(A, p, r);
            randomizedQuicksort(A, p, q[0] - 1);
            randomizedQuicksort(A, q[1] + 1, r);
        }
    }

    public static int[] randomizedPartition(int[] A, int p, int r) {
        int i = ThreadLocalRandom.current().nextInt(p, r + 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;
        int j = p - 1;
        for (int k = p; k <= r - 1; k++) {
            if (A[k] < x) {
                i++;
                j++;
                swap(A, i, j);
            } else if (A[k] == x) {
                j++;
                swap(A, k, j);
            }
        }
        swap(A, i + 1, r);
        return new int[]{i, j};
    }

    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[] A = {1, 3, 10, 2, 7, 5, 4, 6, 9, 8};
        randomizedQuicksort(A, 0, A.length - 1);
        for (int i : A) {
            System.out.print(i + " ");
        }
    }
}
```

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.