timothy allec

Name, SID, Date .....

In Class Assignment 8: Insertion Sort

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.LinkedList;

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

2 Assignment Description

2.1 Big Picture

This algorithm is commonly used to sort playing cards in a players' hand at a poker table.

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.

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.

Output: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

INSERTION-SORT(A)

```
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

```
public class InsertionSort {
    public static int[] insert(int[] a) {
        for (int j = 1; j < a.length; j++) {
            int key = a[j];
            int i = j - 1;
            while (i >= 0 && a[i] > key) {
                a[i + 1] = a[i];
                i = i - 1;
            }
            a[i + 1] = key;
        }
        return a;
    }

    public static void main(String[] args) {
        int[] a = {5, 2, 4, 6, 1, 3};
        int[] b = insert(a);
        for (int i = 0; i < b.length; i++) {
            System.out.println(b[i]);
        }
    }
}
```

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.

array: O(n^3)

.....

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.

array: 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: linked list: O(n^2)
- Space Complexity: linked list: 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.

After the development of dynamic, a linked list optimization would be better for game development such as the game Asteroids

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 optimization utilizes a LinkedList instead of an Array for InsertionSort

import java.util.LinkedList;

public class InsertionSortOptimization{
    public static void insertion_sort(LinkedList<Integer> A){
        for(int j = 1; j < A.size(); j++){
            int key = A.get(j);
            int i = j-1;
            while(i >= 0 && A.get(i) > key){
                A.set(i+1, A.get(i));
                i = i-1;
            }
            A.set(i+1, key);
        }
    }

    public static void main(String[] args){
        LinkedList<Integer> A = new LinkedList<Integer>();
        A.add(5);
        A.add(2);
        A.add(4);
        A.add(6);
        A.add(1);
        A.add(3);
        insertion_sort(A);
        for(int i = 0; i < A.size(); i++){
            System.out.println(A.get(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.