Name, SID, Date tim allec

In Class Assignment 10: Heapsort

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.

2 Assignment Description

2.1 Big Picture

Heapsort combines the time complexity of Merge Sort with the space complexity of Insertion Sort.

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.

```
HEAPSORT(A)
1  BUILD-MAX-HEAP(A)
2  for $i = A.length$ downto 2
3      exchange $A[1]$ with $A[i]$
4      $A.heap\text{-}size = A.heap\text{-}size - 1$
5      MAX-HEAPIFY(A, 1)

BUILD-MAX-HEAP(A)
1  $A.heap\text{-}size = A.length$
2  for $i = \lfloor A.length/2 \rfloor$ downto 1
3      MAX-HEAPIFY(A, i)

MAX-HEAPIFY(A, i)
1  $l = $ LEFT(i)
2  $r = $ RIGHT(i)
3  if $l \leq A.heap\text{-}size$ and $A[l] > A[i]$
4      $largest = l$
5  else $largest = i$
6  if $r \leq A.heap\text{-}size$ and $A[r] > A[largest]$
7      $largest = r$
8  if $largest \neq i$
9      exchange $A[i]$ with $A[largest]$
10     MAX-HEAPIFY(A, largest)
```

```
public class Heapsort{
    public static void BuildMaxHeap(int[] A){
        int heapSize = A.length;
        for(int i = heapSize/2; i >= 0; i--){
            MaxHeapify(A, i, heapSize);
        }
    }
    public static void MaxHeapify(int[] A, int i, int heapSize){
        int l = 2*i+1;
        int r = 2*i+2;
        int largest;
        if(l < heapSize && A[l] > A[i]){
            largest = l;
        }else{
            largest = i;
        }
        if(r < heapSize && A[r] > A[largest]){
            largest = r;
        }
        if(largest != i){
            int temp = A[i];
            A[i] = A[largest];
            A[largest] = temp;
            MaxHeapify(A, largest, heapSize);
        }
    }
    public static void Heapsort(int[] A){
        int heapSize = A.length;
        BuildMaxHeap(A);
        for(int i = A.length-1; i >= 1; i--){
            int temp = A[0];
            A[0] = A[i];
            A[i] = temp;
            heapSize--;
            MaxHeapify(A, 0, heapSize);
        }
    }
    public static void main(String[] args){
        int[] A = {4, 1, 3, 2, 16, 9, 10, 14, 8, 7};
        Heapsort(A);
        for(int i = 0; i < A.length; i++){
            System.out.print(A[i] + " ");
        }
    }
}
```

```
PARENT($i$)
1  return $\lfloor i/2 \rfloor$
```

```
LEFT($i$)
1  return $2i$
```

```
RIGHT($i$)
1  return $2i + 1$
```

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)

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(nlogn)
- 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.

Game Development: Sorting algorithms are used in various aspects of game development, such as rendering, pathfinding, and resource management. The optimized Heapsort can be beneficial in scenarios where memory usage needs to be minimized without sacrificing sorting 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.

```
public class Heapsort {
    public static void BuildMaxHeap(int[] A) {
        int heapSize = A.length;
        for (int i = heapSize / 2 - 1; i >= 0; i--) { // Iterate down from the last non-leaf node
            MaxHeapify(A, i, heapSize);
        }
    }
    public static void MaxHeapify(int[] A, int i, int heapSize) {
        int largest = i;
        int l = 2 * i + 1;
        int r = 2 * i + 2;

        if (l < heapSize && A[l] > A[largest]) {
            largest = l;
        }
        if (r < heapSize && A[r] > A[largest]) {
            largest = r;
        }
        if (largest != i) {
            int temp = A[i];
            A[i] = A[largest];
            A[largest] = temp;
            MaxHeapify(A, largest, heapSize);
        }
    }
    public static void Heapsort(int[] A) {
        int heapSize = A.length;
        BuildMaxHeap(A);
        for (int i = A.length - 1; i >= 1; i--) {
            int temp = A[0];
            A[0] = A[i];
            A[i] = temp;
            heapSize--;
            MaxHeapify(A, 0, heapSize);
        }
    }
    public static void main(String[] args) {
        int[] A = {4, 1, 3, 2, 16, 9, 10, 14, 8, 7};
        Heapsort(A);
        for (int i = 0; i < A.length; i++) {
            System.out.print(A[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.