DSA Midterm Exam (SET A)

Thursday, 29 July 2021 2:07 PM

Subject: ITX2010 / IT2230 / CSX3003 / CS2201 Data Structures and Algorithms
Date: Friday 6th August 2021
Time: 12:00-14:00 (2 hours)
Lecturer: Chayapol Moemeng

Instructions:

1. This is an open book examination. Students can use any material.
   a. However, students cannot communicate with their classmates or anyone by any means.
   b. Additional inquiries are to be asked to the proctor only.
2. Turn on the camera for Microsoft Teams at all times,
   a. turn on your camera,
   b. share your screen, and
   c. record the video.
3. Read all questions carefully and write all answers on the blank space of paper.
4. Do not answer briefly without details. Answers must include either steps of work or an explanation of how the answers are derived.
5. When students wish to submit their work
   a. Photograph the answers, one photo for one question.
   b. Insert the photo into this file.
   c. Upload only this file to Microsoft Teams before 14:00.
   d. All algorithms can be expressed either in pseudocode or Python; unless the instruction requests specifically.
6. The purpose of this examination is to assess students' learning outcomes. Academic honesty is strictly applied. Do not attempt to copy others' work in this mode of examination.

Marking Scale:

Essay 6 questions, 60 marks
Total 60 marks

Student Name:	Saw Zwe Wai Yan
Student ID:	6318013

1. [10 marks] Show the running time analysis of Insertion sort algorithm on A, where A = [1,2,3,4,5,6,7,8,9,10].

```
INSERTION-SORT(A)
1  for j ← 2 to length[A]
2      do 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             do A[i + 1] ← A[i]
7                i ← i − 1
8         A[i + 1] ← key
```

Insertion Sort
The first line of code for j<--- 2 to the length[A] is assigning the length and index.
What we will be doing is that we will set a marker for the sorted section after the first element.
By the list Given A=[1,2,3,4,5,6,7,8,9,10]

The first element which is "1" would be in the sorted part of the list, the rest [1,2,3,4,5,6,7,8,9,10] in the list would be in the unsorted part. This is all done in one part of the list, there is no another list involved. Note, I will be using different colors in the list to represent the sorted list and unsorted list. Red (sorted), Black(unsorted)

Running time?

Line 2 is where we choose the first unsorted element, which is "2" in this case. Line5-7. The while loop is to shift elements to the right create position for the unsorted array. We compare this element with all the elements in the sorted list part. Currently we only have 1 element. We check whether it's smaller or greater than the elements in the sort. [1,2,3,4,5,6,7,8,9,10]. Line 8, we insert the unsorted element to its correct position.

The process is repeated until there is no more elements in the lost.
[1,2,3,4,5,6,7,8,9,10]

2nd element in the list="2"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

3nd element in the list="3"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

4nd element in the list="4"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

5nd element in the list="5"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

6nd element in the list="6"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

7nd element in the list="7"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

8nd element in the list="8"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

9nd element in the list="9"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

10nd element in the list="10"
Compare with the previous list
Finds the correct position
[1,2,3,4,5,6,7,8,9,10]

There is no elements left in the unsorted list. The loops ends, the program is finished.

2. [10 marks] Show the running time analysis of bubblesort algorithm on A,
where A = [1,2,3,4,5,6,7,8,9,10].

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

Bubble sort is similar to insertion where recursion takes places.
Given the list A=[1,2,3,4,5,6,7,8,9,10]

It seems that, the pesudo code for the sort seems wrong.

running time?

If we are using ascending, the Line 2 should be j=A.length-i-1
And in line 3 A[j]>A[j+1], Line 4 A[j] exchange with A[j+1]

If we are using descending, the Line 2 should be j=A.length-i-1
And in line 3 A[j]>A[j-1]

I will be showing using ascending order.
The first element in the list "1".
When i=0
J varies from 0 to 9.

In Line 3 and 4
The adjacent element are compared. [1,2,3,4,5,6,7,8,9,10]
In this case, 1 and the second element which is "2". If the conditions are met we swap them.

1st run i=0, J varies from 0 to 9.
[1,2,3,4,5,6,7,8,9,10]
1 and 2 comparison 1 is not greater, moves on to the next adjacent element which is 2 and 3
2 and 3 the process goes on until the last element of the loop and when all the adjacent elements are compared which the value of j is reduced afterwards

i=1,J varies from 0 to 8.
[1,2,3,4,5,6,7,8,9,10]

i=2,J varies from 0 to 7.
[1,2,3,4,5,6,7,8,9,10]

Loops continues until 1 varies from 0 to 1.
The ascending sort is finished.

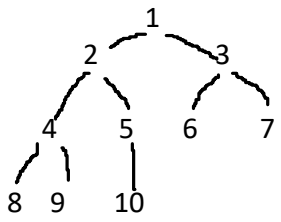
3. [10 marks] Show how to find the number of times MAX-HEAPIFY is called in BUILD-MAX-HEAP(A), where A = [1,2,3,4,5,6,7,8,9,10].

```
MAX-HEAPIFY(A, i)
1  l = LEFT(i)
2  r = RIGHT(i)
```

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

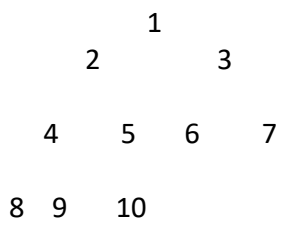
```
BUILD-MAX-HEAP(A)
1  A.heap-size = A.length
2  for i = ⌊A.length/2⌋ downto 1
3       MAX-HEAPIFY(A, i)
```

First we build a heap for the given array



number of time HEAPIFY is called ?

Parent node is always larger than the child node in max heap



4
10

2 is greater than 1 we swap
5 is greater than 1 we swap

10

2 3

4 5 6 7

8 9 1

4. [10 marks] The standard Counting sort algorithm does not support negative values in input array. Modify the Counting sort algorithm to support negative values.

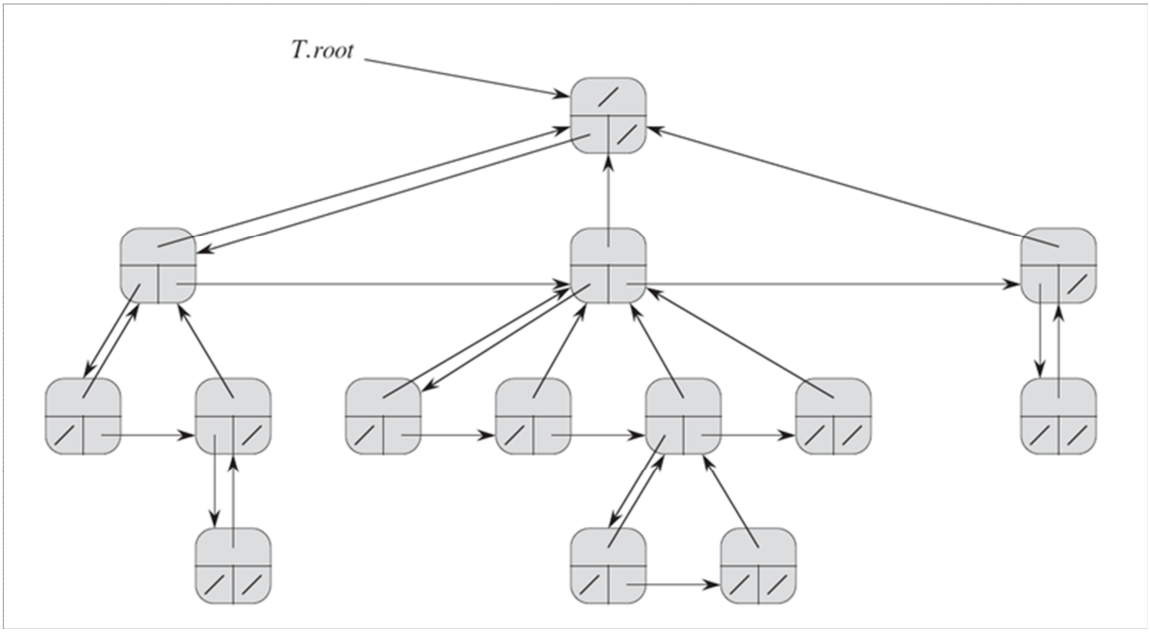
```
COUNTING-SORT(A, B, k)
 1  for i ← 0 to k
 2       do C[i] ← 0
 3  for j ← 1 to length[A]
 4       do C[A[j]] ← C[A[j]] + 1
 5  ▷ C[i] now contains the number of elements equal to i.
 6  for i ← 1 to k
 7       do C[i] ← C[i] + C[i − 1]
 8  ▷ C[i] now contains the number of elements less than or equal to i.
 9  for j ← length[A] downto 1
10       do B[C[A[j]]] ← A[j]
11          C[A[j]] ← C[A[j]] − 1
```

```
Counting-Sort(array):
Max<--max(array)
Min<--min(array)
Range=Max-Min+1

Create a count array to store counts

Count[range]
Output=[length(array)]
For i= 0 to array.length
        Count[array[i]-Min]+=1
For i=1 to len(Count)
        Count[i]+=Count[i-1]
For I length.array-1,-1,-1
        Output[Count[array[i] - min] - 1] = array[i]
For  I =0 to length.array
        Array=output[]
```

6/10

not from the give algo.

5. [10 marks] Write an algorithm for changing parent CHANGE-PARENT(x,newp), where x is the reference to a node and newp is the new parent. The base condition is x must be a leaf and the right-most child only and the new parent will add x as the right-most child. The algorithm can be presented in Python or pseudocode.



DNA

6. [10 marks] Suppose that we are building a web app; it uses phone numbers as the ID of customers. Assume all data are in the memory. Among the four data structures: array, heap, linked list, and hash table; analyse which one is the most suitable to maintain all phone numbers which allows quickest access time with the most efficient memory utilisation. For example, is 0969876543 our customer?

The hash table would be one of the most efficient and suitable one compared to the other data structures. This type of data structure that is used to store key pairs in the care, the customer phone numbers and it corresponding data sets. It uses hash function to find the index where data can be easily searched or inserted. This structure is also suitable for large data as well. As for linked list, it is best suited when the data in the structure is constantly changing. In the care the data of the customers will change less often since we are using the phone number as ID making the data non-volatile.

8/10 should add examples.