# CMPE 130 Midterm Exam #1 Fall 2015

88

12

## 15:00—16:15 Tuesday September 22, 2015

**Student Name** _____(print)

**Student ID**_____

**(4 points) Problem 1 (A):** Fill into line 6 and line 7 of the pseudo code in the **"while loop"** of INSERSION-SORT below. (The INSERSION-SORT sorts the numbers in the array in ascending order.)

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++
8   A[i+1] = key
```

-2

**(3 points) Problem 1(B):** Apply INSERTION-SORT to array

| | | | |
|---|---|---|---|
| 31 | 41 | 59 | 26 |

and show the intermediate and final result for index j=2, 3 and 4.

J=2

| | | | |
|---|---|---|---|
| 31 | 41 | 59 | 26 |

J=3

| | | | |
|---|---|---|---|
| 31 | 41 | 59 | 26 |

J=4

| | | | |
|---|---|---|---|
| 26 | 31 | 41 | 59 |

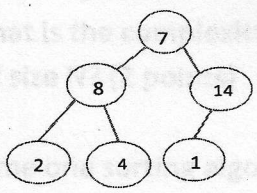
**(4 points) Problem 1( C ):** Create an array out of the elements {41, 26, 59, 31} that has the worst performance under INSERTION-SORT.

Array that leads to the worst performance is

| | | | |
|---|---|---|---|
| 59 | 41 | 31 | 26 |

considering INSERTION-SORT will sort from least to greatest

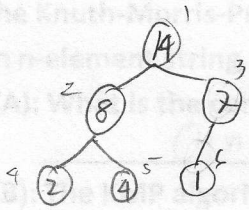
**(2 points) Problem 2(A): Is the data structure below a MAX-HEAP**




Not MAX-HEAP b/c 7 is not the greatest among the elements

✓

**(5 points) Problem 2 (B): If NO, apply MAX-HEAPIFY to convert it into a heap.**
**Write the resulting heap structure below.**




✓

**(5 points) Problem 2 ( C ): create an array corresponding to the result in 2(B)**

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 14 | 8 | 7 | 2 | 4 | 1 |

✓

**(15 points) Problem 3: Using Attachment #1 as a model, illustrate the operation of HEAPSORT on the array A<5, 13, 2, 25, 7, 17, 20, 8, 4>**
**Write your answer in the attached sheet.**

**Problem 4: Many sorting algorithms use "comparing two numbers" as the basic Operation.**

**(4A) What is the complexity of a brute-force comparison-based sorting algorithms for an array of size N? (2 points)** $O(n^2)$ ✓

**(4B) Name one sorting algorithm that does not use "compare". (2 points)**
**What is the complexity of this algorithm? (2 points)**

(-2) counting sort with complexity $O(n \log(n))$

**Problem 5 The Knuth-Morris-Pratt algorithm is widely used in matching an m-element pattern to an n-element string.**

**(2 points) 5 (A): What is the complexity of the brute-force string matching algorithm?** $O(mn)$ ✓

**(2 points) 5 (B): The KMP algorithm reduces the complexity to** $O(m+n)$ ✓ **by a pre-processing.**

**(5 points) 5 ( C ): The pre-processing involves the use of prefix in the pattern. Compute the Prefix function of the KMP algorithm for the pattern below.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | T | C | A | C | A | T | C |

**Fill in the answer prefix function( ) below.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 2 | 3 |

✓

**Problem 6: The basic idea behinds Quicksort is to use a pivot to divide the input array into two sub-arrays, one being smaller than the pivot and the other larger. This process is called PARTITION.**

**(A) Complete the pseudo code for PARTITION below by filling in line 5, 6 and 7. (6 points)**

```
PARTITION (A, p, r)
1  x = A[r]
2  I = p
3  For j = p to r-1
4      If A[j] <= x
5      swap A[j] and A[i]
6      i++
7  swap A[i+1] and A[r]
8  return i+1
```

✓

**(B) Apply (A) to array <1,2,3,4,5,6,7,8> step by step. (3 points)**

**(C) Apply (A) to array <8,7,6,5,4,3,2,1> step by step. (3 points)**

**(D) What is the complexity of Quicksort? (2 points)**

**(E) Is (B) and ( C ) at the worst case complexity? (2 points) Why? (2 points)**

(B) p i j r | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ⇒ p i j r | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

(C) i p j r | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | ⇒ i j r | 7 | 8 | 6 | 5 | 4 | 3 | 2 | 1 | ⇒ i j r | 6 | 8 | 7 | 5 | 4 | 3 | 2 | 1 |

i r | 5 | 8 | 7 | 6 | 4 | 3 | 2 | 1 | ⇒ i r | 4 | 8 | 7 | 6 | 5 | 3 | 2 | 1 | ⇒ i | 3 | 8 | 7 | 6 | 5 | 4 | 2 | 1 |

⇒ | 2 | 8 | 7 | 6 | 5 | 4 | 3 | 1 | ⇒ Finally swap A[i+1] + A[r]

| 1 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |

(D) Quick sort could be $O(n^2)$ at worst $O(n)$ at best

(E) Considering Quick sort will sort from least to greatest B is best case while C is worst case.

**Problem 7: Internet security is of fundamental importance. It uses "keys" to encrypt messages communicated in Internet. There are two types of key generation algorithms: symmetrical key and public key.**

**(7A) Is public key algorithm more secure than the symmetrical key algorithm? (2 points) Why? (2 points)**

Yes public key is more secure than symmetrical key b/c public key has two parts a public and private key and you need both to decrypt; symmetrical uses one key which could easily be brute forced due to its symmetrical nature.

**(7B) RSA is a famous public key algorithm. It chooses the key as illustrated below.**

1 **choose two large prime numbers, p and q**

2 **choose n =pq and z=(p-1)(q-1)**

3 **choose e (e< n) and relatively prime to z**

4 **choose d such that ed-1 is exactly divisible by z**

**What is the public key in the above RSA algorithm? (4 points)**

-4 n = public key

**What is the private key in the above RSA algorithm? (4 points)**

-4 z = private key

**(7C) Does Bob need Alice's private key to decrypt an encrypted message from Alice? (2 point) why? (2 point)**

No, Alice's private key is private to her, the public key is exchanged and from this we can compute the private key needed to decrypt.

heap $O(n \log n)$ Quick sort $O(n^2)$ Insert = $O(n \log n)$

**Problem 8.**

**(8A)What is an algorithm? (2 points)**

An algorithm is a process of steps that when given the same input you will end up at an output that is predictable according to the algorithm.

**(8B) When you design an algorithm, what do you need to consider? (2 points)**

You need to consider the time complexity as well as the space complexity

**(8C) Design an Internet Search algorithm following the steps below.**

**What is your input? (1 point)**

Hello

**What is your output? (1 point)**

Hello World

**Write a pseudo code to explain the search algorithm. (4 points)**

given string
match string with string + other

**What is the complexity of your search algorithm? (2 points)**

$O(mn)$ where m is length of string and n is length of possible string match

**How can you improve (reduce) the complexity? (1 points)**

To reduce the complexity instead of brute force string matching hash the given string, store index of possible hashed string matches and only compare those with the same hash result.

Attachment #1

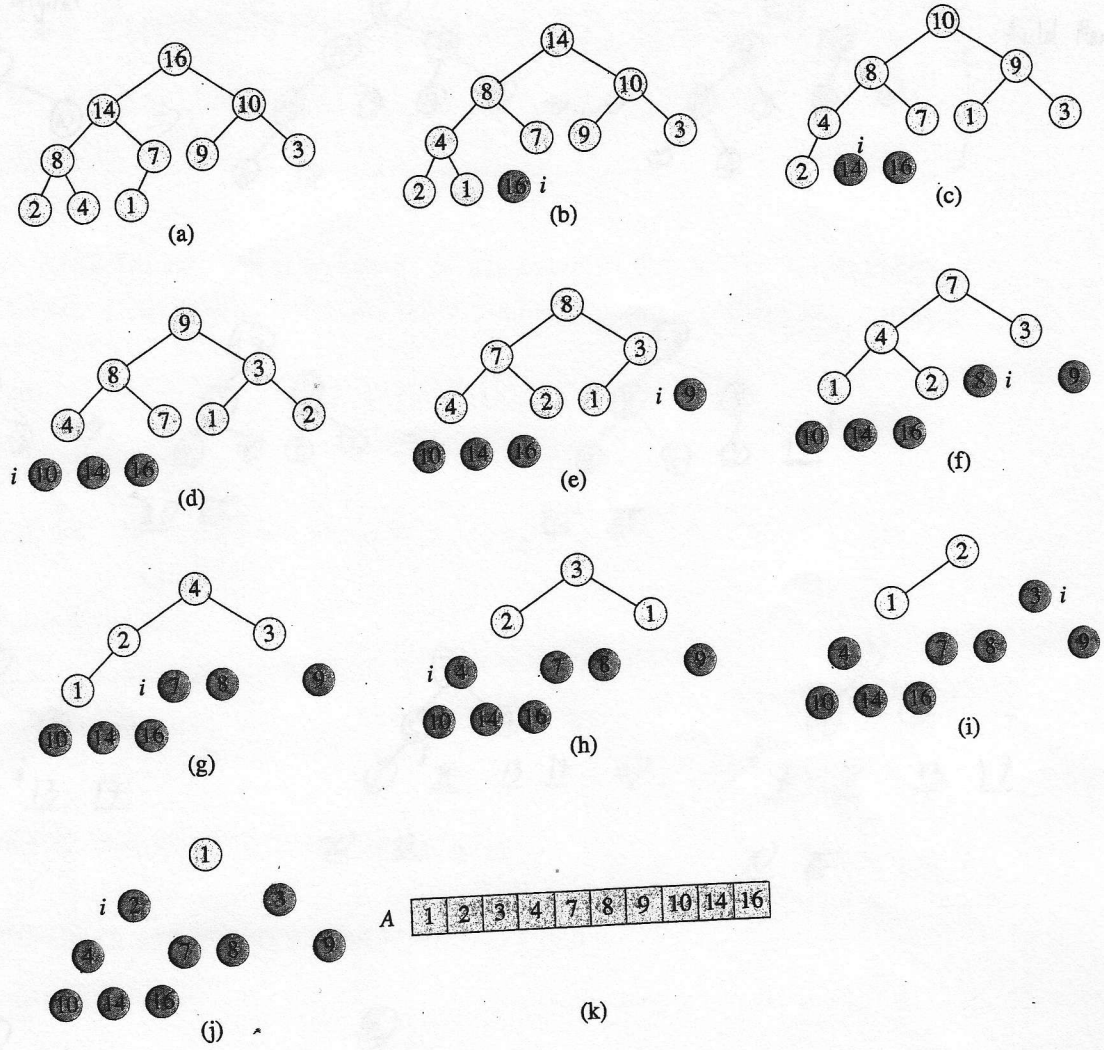
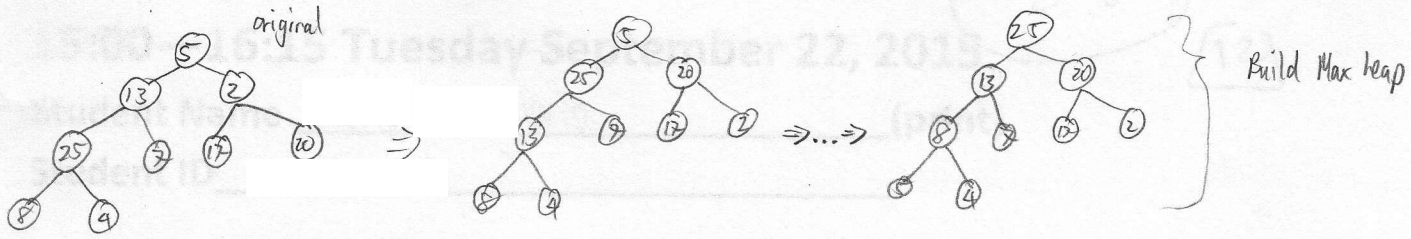
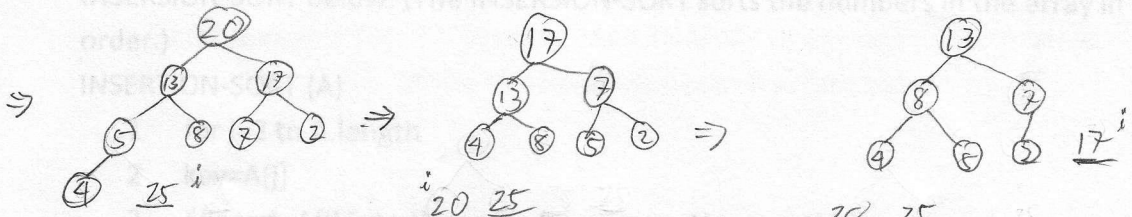
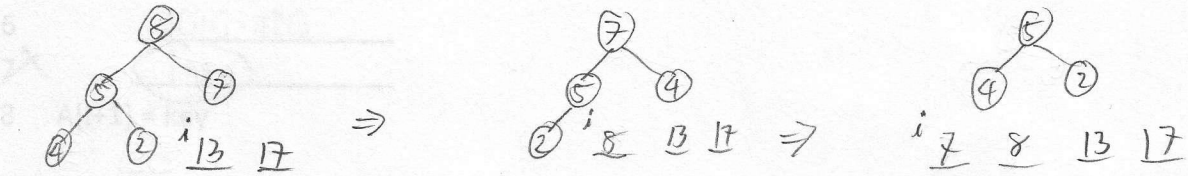


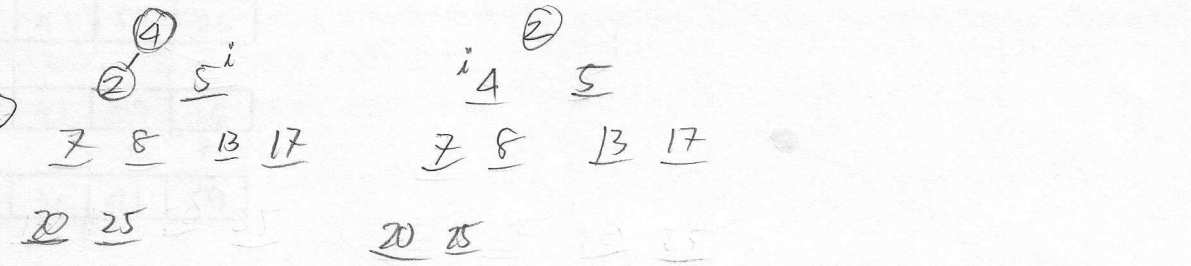

**Figure 6.4** The operation of HEAPSORT. **(a)** The max-heap data structure just after BUILD-MAX-HEAP has built it in line 1. **(b)–(j)** The max-heap just after each call of MAX-HEAPIFY in line 5, showing the value of $i$ at that time. Only lightly shaded nodes remain in the heap. **(k)** The resulting sorted array $A$.

Problem 3: Given: array A< 5, 13, 2, 25, 7, 17, 20, 8, 4 >







A: | 2 | 4 | 5 | 7 | 8 | 13 | 17 | 20 | 25 | sorted