FRANCESCA SPEZZANO

CS 321 Data Structures (Fall 2016)

Homework #1 (80 points), Due Date: 9/27/2016 (Tuesday)

- **Q1(12 points): Asymptotic Notations**

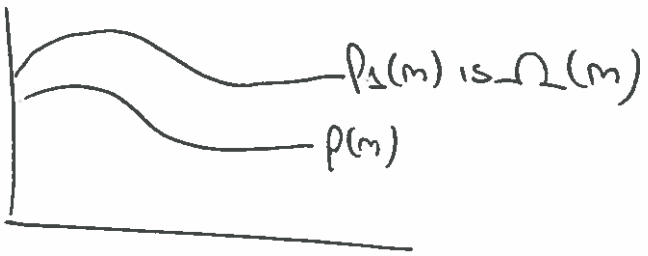
  (a)(4 points) Which one of the following is wrong?

  1. $\Theta(n) + O(n) = \Omega(n)$
  2. $O(n) + \Omega(n) = \Theta(n)$
  3. $\Theta(n) + O(n) = O(n)$
  4. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$

  (b)(4 points) Which one of the following sorting algorithms will have the best best-case running time?

  1. Selection sort $\Theta(n^2)$
  2. Insertion sort $\Theta(n)$
  3. Heap sort $n \log n$
  4. Quick sort $n \log n$

  (c)(4 points) Explain why the statement, "The running time of an algorithm is at most $\Omega(n)$," is meaningless.



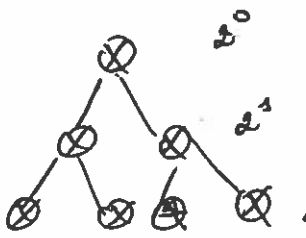
- **Q2(18 points): Running Time and Growth of Functions**

(a)(10 points) Assume evaluating a function $f(n)$ in the pseudocode below takes $\Theta(n)$ time.

```
i = 1;
sum = 0;
while (i <= n)
      do if (f(i) > k)
            then sum += f(i);
         i = 2*i;
```

What is the running time (use an asymptotic notation) of the above code? Justify your answer.

1 f(1)
2 f(2)
4 f(4)
8 f(8)
⋮
n f(n)



$2^h = n$ $\quad h = \log_2 n$

$O(2^{h+1} - 1)$

a tree with n leaves has

$O(2 \cdot 2^{\log_2 n} - 1) = O(n)$ because

$a^{\log_a b} = b$ from log properties

both pay 5c

[end of ch 3]

(b)(8 points) For the following functions, please list them again but in the order of their asymptotic growth rates, from the least to the greatest. For those functions with the same asymptotic growth rate, please underline them together to indicate that.

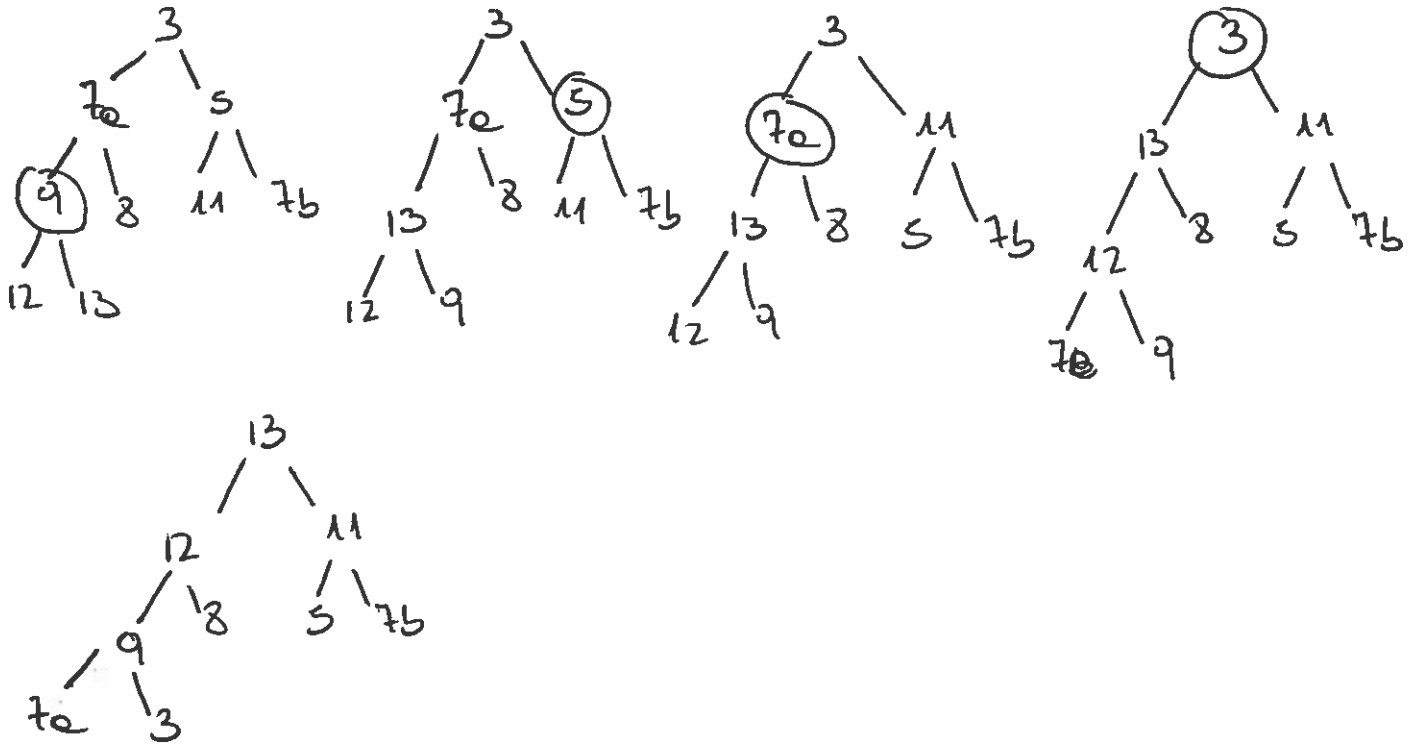
$$(\log_2 n)^n, (\log_2 n^n), n^2, (\log_2 n^2), n^{1/2}, 2^n, \sqrt{2}^2, \log_{10} n$$

$n \log n$ $\quad 2 \log n$

$\sqrt{2}^2$, $\log_{10} n$, $\log_2 n$, $n^{1/2}$, $n \log n$, n^2, 2^n, $(\log_2 n)^n$

- **Q3(28 points): Sorting**

(a)(7 points) For a given input array $A : < 3, 7_a, 5, 9, 8, 11, 7_b, 12, 13 >$, what is the sequence of numbers in A after calling Build-Max-Heap(A) ? (please show the intermediate trees).



(b)(7 points) For a given input array $A : < \dot{2}_a, 9, \dot{3}, \dot{1}, 8, 5, \dot{2}_b, 7, \boxed{4} >$, what is the sequence of numbers in A after the first partition (by calling `Partition(A, 1, 9)`)? Note that 1 and 9 in `Partition(A, 1, 9)` function call are array indexes.

2a 9 3 1 8 5 2b 7 4

2a 3 9 1 8 5 2b 7 4

2a 3 1 9 8 5 2b 7 4

~~2a 3 1 5 8 9 2b 7 4~~

2a 3 1 2b 8 5 9 7 4

2a 3 1 2b 4 5 9 7 8

(c)(8 points) By using the Max-Heap data structure to implement a priority queue, some applications may need to change the data (priority) of a specific node i. That is, given an index i, change the priority of node i to a new priority t. Please write a pseudocode for this procedure.

```
Max-Heap-update(A, i, t)
{
  if ( i<1 or i>heapsize(A) )return error;
  old = A[i]
  A[i] = t
  if ( A[i] > old)
    heapIncreaseKey(A,i,t);
  if ( A[i] < old)
      max-heapify(A,i);
}
```

(d)(6 points) Please describe how to use a priority queue to implement a queue.

Set arrived time as the priority and use a min-heap

- **Q4(22 points): Linear Time Sorting**

(a)(6 points) Please describe the reason(s) why we choose the `counting sort` algorithm to sort each digit in the `Radix Sort`?

Because it is stable and runs in $O(n)$ and it ensures correctness of radix-sort

(b)(6 points) What is the best running time to sort n integers in the range $[0, n^3 - 1]$, and How?

The best running time is $\Theta(n)$

How? use radix sort in radix n

$d = \log_n n^3 = 3 \rightarrow$ constant

radix-n means digits are in $[0, n-1] \Rightarrow k$ is $O($

Time $d * \Theta(n+k) = \Theta(n)$

(c)(10 points) Given an input array A with n integers in $[0, k]$, we can use the array C in the counting sort to find out how many integers in A are in a range $[a, b]$. Write a pseudocode for this query. Assume $C[i]$ already contains the number of input integers $\leq i$.

```
FindNumIn(a, b) // both a and b are integers
```

```
if (a>b or b<0)
    return error;
if (b>k)
    b=k
if (a ≤0)
   return c[b]
else
    return c[b] - c[a-1]
```