**LAB6**

**Jimmy Palma**

**610756**

1. **Show that any comparison-based algorithm to sort 4 elements requires at least 5 comparisons in the worst case.**

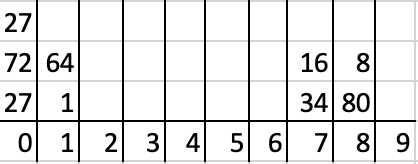
h = height of the tree

h >= log(n!)

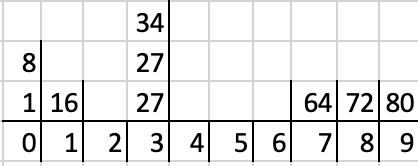
Log(4!) = log(24) = 4,58

1. **Carry out the steps of RadixSort to sort the following {80, 27, 72, 1, 27, 8, 64, 34, 16} – Hint: use 9 for your radix.**

80%9 = 8; 27%9=0; 72%9=0; 1%9=1; 27%9=0; 8%9=8, 64%9=1; 34%9=7; 16%9 = 7



80/9 = 8; 27/9=3; 72/9=8, 1/9=0; 27/9=3; 8/9 = 0, 64/9 = 7; 34/9 = 3; 16/9=1



Sorted:

1. **Describe an O(n) algorithm that does the following: Given an input array of n integers lying in the range 0 .. 3n - 1, the algorithm outputs the first integer that occurs in the array only once. (You may assume that each input array contains at least one number that has no duplicates in the array.) Explain why your algorithm has an O(n) running time.**  
    **Example: If the input array is [1, 2, 4, 9, 3, 2, 1, 4, 5], then the return value is 9 since 9 is the first integer that occurs in the array only once.**

Algorithm bucketSort(A, m)

Input array A of (key, element) items with keys in the range [0, m - 1]

Output array B sorted b increasing keys

bucket <- array of m empty list

bucket1 <- array of m empty list

for i <- 0 to n –1

(k, o) <- A[i]

bucket[k].insertLast ((k, o))

for i <- 0 to n –1

(k, o) <- bucket [i]

bucket1[k].insertLast ((k, o))

for j <- 0 to m - 1

If A[j] and bucket1[j].size().==1  
 return A[j]

1. **The Fibonacci numbers are defined in the following way: F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)** 
   1. **Devise a recursive algorithm for computing the nth Fibonacci number.**

Algorithm Fibonacci recursive

Input: n number

Output: the n finonacci

If n = 0 or n = 1

Return 1

Return fibonacci(n-1) + fibonacci(n-2)

* 1. **Can you use one of the techniques in lesson 2 to compute the running time of your recursive algorithm?**

T(n) = 3 + T(n-1)+ T(n-2)

n+(n-1)+(n-2)…...1 = n(n-1)/2

=> O(n²)

* 1. **Devise an iterative algorithm for computing the nth Fibonacci number.**

Algorithm Fibonacci iterative

Input: n number

Output: the n finonacci

Temp <- temp number

Fib = 1

Prev = 1

If n = 0 or n = 1

Return 1

For i = 2 to n do

Temp = fib

Fib += prev

Prev = temp

* 1. **What is the running time of your iterative algorithm?**

O(n)