COS 485 — Homework 6

Samuel Barton Benjamin Montgomery

April 18th, 2017

Problem 1

In this problem we are asked to make a decision tree argument about the lower limit on the worst case number of comparisons needed to find a key in a sorted list.

Let us assume our list has N items. We assume that these items are in sorted order. There are N possible solutions, namely keys, from which we must choose one. Thus the height of the decision tree must be large enough to cover the search space of N items. Since we have sorted the items we have a binary decision, either an item is greater than its predecessor or it is less than its predecessor. Thus we have a binary tree where each node can have at most two children, the height of the tree can be calculated by solving the following equation.

$$N \leq 2^h \implies \lceil \lg(N) \rceil \leq h \implies h \geq \lceil \lg(N) \rceil$$

Since the height of the decision tree is equal to the number of needed comparisions, we have that there can be a minimum of $\lg N$ comparisions in the worst case to find a particular key in a sorted list of N elements.

Problem 2

In this problem we consider the case of one miserly king who has demanded a weighing of N coins with the understanding that one coin is counterfeit and lighter than the others.

The algorithm to efficiently solve this is as follows:

1. Divide coins into three parts A, B, and C where the size each is $\left\lfloor \frac{n}{3} \right\rfloor$. If N is not divisible by 3, then we will put the 1 or two additional coins in D and set them aside.
2. Weigh two of A, B, or C.

   If $A = B$, then throw out A and B and keep C

   If $A < B$, then throw out B and C and keep A

   If $B < A$, then throw out A and C and keep B
3. Repeat until only one of A, B, and C remain
4. If N is divisible by 3, then we are done. Otherwise, we have either 1 or 2 elements in D to consider. Compare whichever of A, B, or C remains with 1 of the elements in D using the above logic.

In the worst case, N is not divisible by 3 and has a remainder of 2. Thus there are 2 elements in D and we do $\lceil \log_3(n) \rceil$ comparisons to end up with 1 element from A, B, or C and two elements from D to consider. We make one additional comparision with 1 element from D and the remaining element from the prior comparisons, and can then determine which coin is the counterfeit. Thus in the worst case our algorithm will have a time complexity equal to the number of comparisons which is

$$T_n \;=\; \lfloor \log_3(n) \rfloor + 1 \;=\; \lceil \log_3(n) \rceil$$

Problem 3

In this problem we are asked to make a decision tree argument to determine the absolute lower bound for any algorithm to solve the *Miserly King Coin Counterfeit Checking Problem.*

In the problem we are given N coins, and so there are N possible solutions to the problem. Each decision we make has three possible outcomes, so we have a ternary tree. Thus in order to determine the minimum possible height of a decision tree with N leaves we must solve the following equation:

$$N \leq 3^h \implies \log_3 N \leq \log_3 3^h \implies \log_3 N \leq h$$

Thus we see that there must be at least $\log_3 N$ decisions for any algorithm to solve the *Miserly King Coin Counterfeit Checking Problem.*

Problem 4

In order to have a worst-case linear algorithm for finding the mode in an array, consider the following algorithm:

1. Create an empty HashMap of the form K, V where K is a unique SAT score, and V is the number of occurences of that score.
2. Iterate over the array. For each element, check to see if it is a key in our HashMap.

    If it is not a key, add it to the HashMap, and set the value to 1.

    If it is a key, increment the associated value.
3. Iterate over the keys in the HashMap, and find the key with the maximum value. This will be the mode.

Our analysis is as follows:

1. Creation of the **empty** HashMap is $O(1)$.
2. Our iteration through the array is $T_n = n$. At each element, we perform a *get* operation in the average case, generally considered to be $O(1)$. We then perform another (average case) $O(1)$ insertion, whether it be an entry that gets inserted, or an existing value that gets updated. This is a total of $T_n = 3n$
3. Our linear pass across the set of keys in the HashMap is $T_n = k$ where k is the number of unique SAT scores. In the worst case, there are n keys, so we have $k = n$ and $T_n = n$.

Therefore, we can express our time complexity as the following:

$$T_n = O(1) + 3n + n \implies T_n \in \Theta(n)$$

Completely Optional Sidenote:
If we were given the scores as fixed width integers, we could radix sort the values (Cost $\Theta(kn)$, but k is a constant due to the construction of this problem, making it $\Theta(n)$) Then, we could iterate through the values once, using the following rules.

1. Let there be variables used to store the current best mode found, the number of occurrences it has, and an external variable used to keep track of how good the current run is.

2. Iterate element by element through the sorted array. As you go through each element, keep track of how many times it occurs. If this number is greater than our best so far, update the variables representing our best mode and it's number of occurrences. This is $\Theta(n)$.

Thus, we expect this solution to be $\Theta(n)$.

Problem 5

To solve this problem, we may reuse our solution to problem 4 verbatim. To reiterate:

1. Create an empty HashMap of the form K, V where K is a unique SAT score, and V is the number of occurences of that score.

2. Iterate over the array. For each element, check to see if it is a key in our HashMap.

   If it is not a key, add it to the HashMap, and set the value to 1.

   If it is a key, increment the associated value.

3. Iterate over the keys in the HashMap, and find the key with the maximum value. This will be the mode.

As noted above, this is

$$T_n = O(1) + 3n + n \implies T_n \in \Theta(n)$$

Problem 6

In this continuation of the *Miserly King* problem we are asked to determine the maximum number of coins we may weigh in three weighings if we add the condition that the counterfeit may be lighter or heavier than the real coin. Also, we have access to an arbitrarily large number of good coins to use as reference. We'll call the good pile G.

The algorithm to solve this is as follows:

1. Divide the coins up into three partitions A, B, and C as before
2. weigh A and B
3. If $weight(A) = weight(B)$ then the counterfeit is in C
4. If $weight(A) < weight(B)$ or $weight(A) > weight(B)$, then weigh A against G and note whether A was heavier or lighter than B
5. if $weight(A) < weight(G)$, then the counterfeit coin is in A, and it is lighter than the good coins
6. if $weight(A) > weight(G)$, then the counterfeit coin is in A, and it is heavier than the good coins
7. if $weight(A) = weight(G)$, then the counterfeit coin is in B and we know its weight as we noted whether it was heavier or lighter than B
8. Now that we know whether the counterfeit is heavier or lighter than the good coins we can continue as we did in the original *Miserly King* problem.

Thus, while before we could weigh N coins in $\lceil \log_3 N \rceil$ steps, now that we have to do one more weighing to determine the weight difference between the counterfeit and a good coin, it will take $\lceil \log_3 N \rceil + 1$ weighings to weigh N coins.

Solving this equation for N with 3 weighings gives us the following:

$$\lceil \log_3 N \rceil + 1 = 3 \implies \lceil \log_3 N \rceil = 2 \implies N = 3^2 = 9$$