Charles Kolozsvary Period 2 Algorithms 1-23-20
Analysis of Algorithms – Mathematical Models

3 Analysis of Algorithms - Mathematical Models
What does linearithmic mean? Give an example
Describe the bullets in the Scientific method and the Principles paragraphs.
Start on page 21 in the attached pdf.

Q 1.
Linearithmic means that given a size of N samples for performing a calculation, the time that calculation will take is proportional to N * log (N) samples. This means that the ratio between sample size is much closer to 1 (which would be linear) than if the runtime was quadratic. A linearithmic algorithm will be able to manage larger and larger sample sizes of data and produce the same calculation as quadratic in a fraction of the time; it will keep up as the input size grows. One example of this kind of relation between sample size and runtime is found in the FFT algorithms which is used in many modern devices to break down the wave-form of N samples of a signal into periodic components which is the basis for DVD's, JPEG's, MRI's, etc.

Q 2. a.
The Steps in the Scientific method are as follows:
Observe
Hypothesize
Predict
Verify
Validate

Observe – finding a feature of the natural world and taking a closer look into how it works and how to change the manner in which it acts in given situations. Regarding what we are learning now, we are observing the runtime of an algorithm on a computer, and therefore looking to understand how we can manipulate the runtime and how it acts in given situations and programs.

Hypothesize – Creating a model of how events will unfold that is consistent with the observations we make. In this it is creating relationships between a programs characteristics and the runtime the aforementioned program will require to complete its task.

Predict – Use the hypothesis to create predictions about how an event will unfold. Here we will use our models (hypothesis) to make predictions about the runtime of a given program/ algorithm on a different computer or for larger problem sizes.

Verify – Continue to make further observations and check to see if the predictions you made line up with the observations. For this situation that means predicting the runtime of a given algorithm and observing if our prediction is correct.

Validate – Continue to verify observations until the hypothesis (model) works in direct conjunction with any given observation or situation. This is the culmination of verifying our predictions so that our overarching model is directly associated with our observations; Our understanding of how a program's contents relate to runtime are accurate with every observation we make.

Q 2. b.
Principles Explained:

Reproducible – The experiment should be able to be run by others in the same manner and they should have the same results.

Falsifiable – The means by which you conduct your experiment lends itself to allowing your hypothesis to be proven wrong based on the relation between predictions and observations.

Analysis of Algorithms – Mathematical Models
Notes:

When looking at any program it is possible to make an exact mathematical formula to account for all operations and therefore run time, but this gets complicated very quickly, and is often not very useful as sample sizes grow very large.

One way to simplify the process while still obtaining valuable information is to see which action in the serious is most frequent and therefore costly and building the runtime around the frequency of that component of the program.

This is possible by using tilde notation where for a given function f(n) and the most costly/largest/time consuming part of the function g(n) the lim as n→ infinity of f(n)/g(n) = 1
Demonstrating that really what is most important is the most frequent operation rather than all operations

Another way to find complexity is to express the operation and the interval for which it is accessed as an integral.

EX) 1+2+3+ … + N = $\int_{x=1}^{N} x dx$ = 1/2N^2

EX no. 2)

In the example below there are N(N -1) array accesses and therefore the tilde notation is ~N^2 and the runtime is N^2 complexity as dictated by the number of array accesses

```
public void example(int N, int[] a, int[] b)
{
    int count = 0;
    for(int i = 0; i<N; i++)
    {
        for(int j = i+1; j<N; j++)
        {
            if(a[i] == b[j])
            {
                count++;
            }
        }
    }
}
```