KEY

Week 1	Exercise 2	The Curious Powers of Logarithms	20 Minutes

Goals:
Practical exposure of powers and logarithms, Powers of Two

Description:
Gene's family has been remarkably consistent in its growth over the past few generations. Every child in each generation has had two children going back to Gene's great-great-grandmother, *i.e.* Gene's mother's mother's mother's mother.

Tasks:

- Model the blood relatives in a family tree. Assume that there is no intermarrying among the family and every coupling has been with an unrelated partner. You do not need to include partners in this tree.
- Identify the total number of blood relatives in the family tree.
- For each generation, identify it with a **generation number** indicating how many generations removed from Gene's great-great-grandmother they are.
- For each generation, identify how many members it includes.

Questions:

1. For the overall family tree, what mathematical function models the overall number of blood relatives?
   assuming power(base, exponent), power(2, generations + 1) - 1 →total blood relatives
2. What mathematical function models the relationship of generation number to number of descendents in that generation?
   power(2, generation_number) →descendents in generation
3. What mathematical function models the number of relatives of that generation to the generation number?
   Assuming logarithm(base, value), logarithm(2, relatives_of_generation) →generation number

Extension:
When Annie is nervous, Annie habitually folds paper repeatedly in half. Annie was on the phone with a parent and started folding paper. After the call, Annie considered how many layers of paper she had and deduced the number of folds from that guesstimate.

- What mathematical function describes the relationship between folds and layers?
  While power is acceptable, this is technically the logarithm because we work the power operation backwards.
- If Annie counts 8 layers, how many folds did Annie make?
  $2^3 = 8$ therefore the answer is 3 folds
- If Annie counts more than 200 layers but less than 500 layers, how many folds did Annie make?
  $2^8 = 256$ therefore the answer is 8 folds

- If Annie assumes that there are definitely more than fifteen thousand layers but there are clearly less than twenty thousand layers, how many folds did Annie make?
  2^{14} = 16384 therefore the answer is 14 folds

Conclusions:

- In computing we primarily focus on logarithms and powers in base 2. Other bases might introduce some novel ideas and techniques, but they will tend to be more complex. The tradeoff between simplicity and complexity will often render alternatives in other bases moot. The simplicity of base 2 makes it easy to remember and generally applicable, so it is a more valuable tool for computer scientists than most other bases and it will allow us to model ideas like doubling and halving and answer questions like: “how many time can I divide this in half until I am left with only one value?”.
- Memorizing powers of 2 allows you to approximate logarithms useful in CS in your head. Base 2 is everywhere in programming, so it is very useful to identify values that align on the powers of 2 table just for context; however, there are many esoteric applications that you will discover later in life if you take the time to learn base 2 powers and logarithmic progressions.
- Powers of two models doubling. If you play close attention to the industry, you will find powers of and double progressions everywhere in computer science and computer engineering. For example, memory sizes have typically followed a doubling principle, as chips reduced in size, manufacturers were able to double the capacity in the next generation without increasing the size of the hardware, so you will find hardware like drives of size 8GB, 16GB, 32GB, 64GB, etc. This has been a consistent progression since the beginning of the PC era where memory modules followed the progression of 256KB, 512KB, 1MB, 2MB, 4MB, etc. all the way up to the average 16GB memory prevalent in off the shelf computers today. This also can be applied to logic, so while we may use hardware examples here, in class we will see that we may define algorithms that are subject to this doubling growth in efficiency which is a bad thing. We really want to avoid designing algorithms that get twice as expensive for an increase of size 1 to input.
- Logarithms are “reverse powers” so they model repeated halving. In a discrete domain, the base 2 logarithm predicts the maximum number of times that repeated halving can be done to a whole number before it cannot be subdivided any more. The halving strategy is very powerful when it models the efficiency of our program. We will look at algorithms that use a “divide and conquer” strategy and these typically involve a logarithmic progression. If a doubling progression for an algorithm is slow and bad, then a logarithmic progression for an algorithm should be faster and better.

Further Information:

Powers of Two : https://en.wikipedia.org/wiki/Power_of_two
Binary Logarithm : https://en.wikipedia.org/wiki/Binary_logarithm

Power : Base 2

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1024$

$2^{11} = 2048$

$2^{12} = 4096$

$2^{13} = 8192$

$2^{14} = 16384$

$2^{15} = 32768$

$2^{16} = 65536$

Logarithm : Base 2

$\log_2 1 = 0$

$\log_2 2 = 1$

$\log_2 4 = 2$

$\log_2 8 = 3$

$\log_2 16 = 4$

$\log_2 32 = 5$

$\log_2 64 = 6$

$\log_2 128 = 7$

$\log_2 256 = 8$

$\log_2 512 = 9$

$\log_2 1024 = 10$

$\log_2 2048 = 11$

$\log_2 4096 = 12$

$\log_2 8192 = 13$

$\log_2 16384 = 14$

$\log_2 32768 = 15$

$\log_2 65536 = 16$