1. If theoretical runtime on an algorithm is known, we can make predictions about how long it will take to run for different inputs.
   1. Problem: an input runs in O(f(*n*)) for input
   2. Calculus method
      1. Setup
         1. T(*n*) = actual runtime of the function
         2. c = constant
      3. Example
         1. T(*n*) = 8*n*2– 6*n* + 4
         2. Thus
            1. T(*n*) = O(*n*2)
            2. T(*n*) = O(*n*3)
         3. Because, T(*n*) ≠ O(*n*)
   3. Non-calculus method
      1. If an input runs in O(f(*n*)) time, then the model’s runtime is T(*n*) = cf(*n*).
      2. Assumption
   4. Examples
      1. Algorithm A sorts *n* inputs in O(*n*2) time. It takes 20 ms. to sort 10,000 inputs. How long will it take to sort 40,000 inputs?
         1. T(*n*) = actual runtime = c*n*2
         2. T(10,000) = c(104)2 = 20 ms.
            1. c = 2 / 107 ms.
         3. T(40,000) = (2 / 107)(4 × 108)2 = 320 ms.
      2. A database search with *n* elements takes O(log2*n*) time. If 100,000 searches in a database of size 216 takes 40 ms., how long will 600,000 searches take on a database with 220 elements?
         1. T(*n*) = actual runtime = clog2*n*
         2. 100,000*T*(216) = clog2(216)(100,000) = 40 ms.
            1. c = 40/(16 × 105) ms.
         3. 600,000*T*(220) = (40/(16 x 105))log2(220)(600,000) = 300 ms.
      3. An algorithm processing an *nm* array takes O(*n*2*m*3) time. On an array of size 100 × 200, the algorithm takes 1 second. How long would it take an array sized 400 × 300?
         1. T(*nm*) = actual runtime = c*n*2*m*3
         2. T(100 × 200) = c (1002 × 2003) = 1 s.
            1. c = 1 / (8 × 1010) s.
         3. T(400 × 300) = (1 / (8 × 1010))(4002 × 3003) = 54 s.
2. Sidenote
   1. int overflows after ~2.1 billion
   2. long long: overflows after 4 x 108
      1. %lld
      2. Casting required
3. Base Conversion
   1. COP 3223C
      1. Converting base-2 to decimal
         * 1. Low bit:
           2. High bit: *i*
      2. Converting a byte to decimals
         1. Range = **[0, 255]**
      3. Converting a word to decimals
         1. Same process as a byte, but up to
         2. Range = **[0, 65,535]**
      4. Converting a decimal to binary
         1. Divide by 2
         2. Use carry as the binary value
   2. COP 3502C
      1. In general, in base-10, if we have a number…
         1. where *di* is the *i*th digit.
      2. Example: base-10
         1. Code
            1. int val = 0;
            2. for (i = 0; i < num\_digits; i++)

val = 8 \* val + digits[i];

* + 1. Valid symbols
       1. In base *b*, valid symbols are 0 to *b* – 1
       2. If *b* > 10, start using
    2. Bases
       1. Hexadecimal: *b* = 16
       2. Binary: *b* = 2
       3. Octal: *b* = 8
       4. Decimal: *b* = 10
    3. Base-10 to base *b*
       1. 21510 🡪 digits
          1. Keep dividing 215 by *b*
          2. Read remainders off backwards
       2. Reasoning
          1. 167 % 7 = *d*0
          2. 167 / 7 =
       3. Code
          1. int val = 0;
          2. for (i = 0; i < n; i++)

val = base \* val + value(num[i]);

* + 1. Base-b1 to base b2: base-b1 to base-10 to base-b2
    2. In computer science, base-2, base-10, and base-16…
       1. AC716 🡪 101010001112
       2. Each hex character (0 to F) can be matched to a 4-bit integer in binary.
          1. See notes
          2. Group by *n* in 2*n*