Homework (Week 1)

1. For each function f (n) and time t in the following table, determine the largest size n of a problem P that can be solved in time t if the algorithm for solving P takes f (n) **microseconds**, i.e. 10-6 seconds. Two entries are already completed.
2. Observe how the numbers are growing (fast or slow) vertically from algorithm complexity *log* n to 2n, and horizontally from calculation time 1 Second to 1 Century.
3. Write down your own comments about these numbers in the table, such as which algorithm is the most efficient one in the table and which one is the worst, and any other thoughts as much as you can.

Hint: You can do all rows except for n *log* n just by setting the function equal to the value, i.e. f(N) = T, where T is 1 second (i.e. 106 microseconds), 1 Hours (i.e. 60x60x106 microseconds), etc. and solving for N. The *log* is 2 based, i.e. *log*2. For the n*log*(n), the easiest technique is unfortunately to simply use trial-and-error. You can also write a simple Python program to help you complete this assignment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 Second | 1 Hour | 1 Month | 1 Century |
| *log*(n) | ≈ 10300000 | ≈ 101.08#10^9 | ≈ 107.77\*10^11 | ≈ 109.46\*10^14 |
| n | 106 | 3.6\*109 | 2.59\*1012 | 3.15\*1015 |
| n*log*(n) |  |  |  |  |
| n2 | 103 | 6.00\*104 | 1.61\*106 | 5.62\*107 |
| n3 | 102 | 1.53\*103 | 1.37\*104 | 1.47\*105 |
| 2n | 20 | 32 | 41 | 51 |

Observation:

Log(n) is the fastest algorism. 2n is the slowest algorism. The rest are arranged from faster to slower