Problem Set 1

Justin Ely

615.202.81.FA15 Data Structures

08 September, 2015

1)

Base 10	Base 2	Base 3
Decimal	Binary	Ternary
0	0000	000
1	0001	001
2	0010	002
3	0011	010
4	0100	011
5	0101	012
6	0110	020
7	0111	021
8	1000	022

This implementation would be a problem in practice as computer switches, at their simplest level, have only 2 states: on/off. This means that any ternary implementation would still need to be converted to binary at the lowest level in order to execute.

2)

The address of a particular element of an array is given by:

$$address = offset + (row * 20 + column)(bytes/element). \quad (1)$$

For the two cases given in the problem description, the addresses are as follows, with RM being a 10x20 row-major array where each element is 4 bytes.

1. RM[5][3]: address = 100 + (5*20 +3)*4 = 512
2. RM[9][19]: address = 100 + (9*20 + 19)*4 = 896

3)

The maximum number of non-zero elements is given by $N_{nonzero} = \sum_{k=n}^{0} k$. This can be mapped to a sequential array with the function $f(i,j) = \frac{i(i-1)}{2} + j - 1$ (1-indexed).

4)

The maximum number of non-zero elements is given by the main diagonal elements plus the diagonals immediately above and below, which in a square array is $N_{nonzero} = n+2(n-1)$. To store the nonzero elements sequentially in another array, one formula to map is given by the function: $f(i,j) = magic$ (i.e. I didn't figure this one out).

5)

With $a = 5, b = 100$, the two functions are equal at approximately $n = 30$. Below this point, the n^2 function is preferable, while afterwards the $nlogn$ function is better.

6)

a]

$$1hr = 3600s = log(N)(10^{-6}s) \tag{2}$$
$$N = 10^{3600/10^{-6}} = infinite \tag{3}$$

Using Python to evaluate, an overflow error was encountered trying to evaluate the expression. This does not mean an algorithm of this type could solve an infinitely sized problem, rather that the value is sufficiently high to be unrepresentable with the built-in precision. Clearly log(n) complexity allows for solving large-scale problems.

b]

$$1hr = 3600s = N^3(10^{-6}s) \tag{4}$$
$$N = (3600/10^{-6})^{1/3} = 1532 \tag{5}$$

With an n^3 complexity and the speed specified, a problem of size 1532 (arbitrary units) could be solved in an hour. As an example, performing some analysis/calculation of n^3 complexity on an array of 1532 elements would take an hour. Considering this is not a large array size for many applications, n^3 complexity would be a very limiting factor.

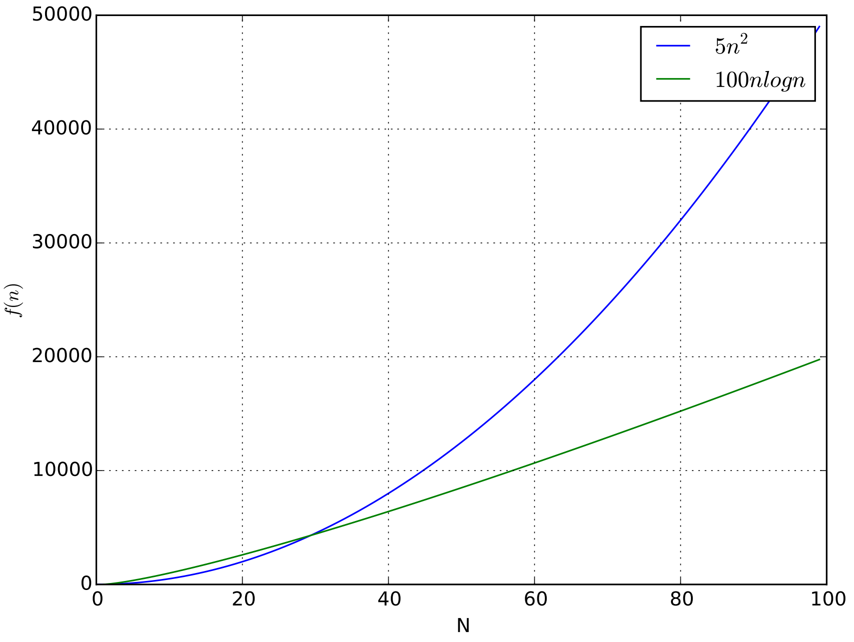

Figure 1: Growth of n^2 and $nlogn$ algorithms plotted as a function of problem size. The two work functions are equivalent at a problem size of roughly 30.