Homework 1: Complexity and ADTs

Alexander Shah

EN.605.202.81 Section 84

1.

For positive integers, we can represent ternary values similarly to base 2. e.g. 3^3+3^2+3^1+3^0, where we sum up powers of 3 instead of powers of 2. This works well but isn't more efficient for positive integers than binary. However, other considerations may make trits a worse choice. For example boolean logic using trits would effectively work like binary with duplicates. 1 AND 2 would be no different than 1 AND 1. Currently it's easier to differentiate between two voltage levels than 3 in analog applications which may make ternary signals difficult to accurately interpret. Existing signaling and logic relies on binary properties at a hardware level such as logic gates. While some systems could continue to use binary and add ternary logic, we would then need to differentiate base 2 and 3 encodings and generally keep track of a mixed base environment which may be complex.

2.

We start at an arbitrary address of 100. Each value is 4 bits long, and each row is 20 values long. So we would need to multiply the number of rows down we are by 20 plus the number of elements into the row we are, all multiplied by 4 to account for the size plus our offset.

For RM[5][3],

100+(5*20+3)*4 = 512

and for RM[9][19],

100+(9*20+19)*4 = 896

3.

Since we consider the diagonal, the max nonzero elements must be (n(n+1))/2. In a 3x3 matrix we would expect 6 elements to be nonzero, and (3*4)/2 = 6.

A matrix can be stored sequentially as a one dimensional array such as a list, for example in row order. K is a formula to find the index position where a nonzero value of a lower triangular matrix can be stored in a 1d array. To find this, use n(n+1)/2 based on the row index i to calculate previous triangles' elements, and add j units. k = (i * (i + 1) / 2) + j

$$\begin{array}{c|cccc} & 0 & 1 & 2 & j \\ \hline 0 & 1 & 0 & 0 & \\ 1 & 2 & 3 & 0 & \\ 2 & 4 & 5 & 6 & \\ i & & & & \end{array}$$

$(0,0) = 0*1/2 +0 = 0$

$(1,0) = 1*2/2 +0 = 1$

$(1,1) = 1*2/2 +1 = 2$

$(2,0) = 2*3/2 +0 = 3$

$(2,1) = 2*3/2 +1 = 4$

$(2,2) = 2*3/2 +2 = 5$

4.

The tridiagonal matrix has elements in its diagonal, and the diagonals alongside the diagonal, which are one element shorter than n, the diagonal. So in total there should be n + (n-1) + (n-1) elements, or 3n-2. A 3x3 matrix should have 7 nonzero elements, and 3*3-2 = 7. These elements can be stored sequentially in a one dimensional array like a list in row order. For nonzero elements in this array, we know that elements can be found above and below any given diagonal where i=j, so i=j+-1 and vice versa. So a good k could be k= 2i+j. We double the row value and add j units. This ensures no duplicates when assigning indices.

$$\begin{array}{c|cccc} & 0 & 1 & 2 & j \\ \hline 0 & 1 & 2 & 0 & \\ 1 & 3 & 4 & 5 & \\ 2 & 0 & 6 & 7 & \\ i & & & & \end{array}$$

$(0,0) = 2*0+0 = 0$

(0,1) = 2*0+1 = 1

(1,0) = 2*1+0 = 2

(1,1) = 2*1+1 = 3

(1,2) = 2*1+2 = 4

(2,1) = 2*2+1 = 5

(2,2) = 2*2+2 = 6

5.

The two functions always intersect at the origin, n=0, and can cross up to 2 more times depending on the a,b values. When a>b they intersect at n=0 only; when b<a they intersect at n=0, and up to 2 more times; and when a=b they intersect at n=0 only. The a,b values can cause the functions to drastically change their slope, enough to make n*log(n) appear worse than n^2 with the right values.

6.

1 hour in microseconds = 3,600,000,000

lg(n) = 3,600,000,000

n= 2^3,600,000,000

7.

1 hour in microseconds = 3,600,000,000

n^3 = 3,600,000,000

Take the cube root to find approximately n=1532.61886479

or n = 1532