Adam Rich
EN.605.202.87.SP18 Data Structures
Module 1, Homework Assignment 1
February 6, 2018

Homework Assignment 1 - Complexity and ADTs

1. Show how nonnegative numbers can be represented in an imaginary ternary computer using trits (0,1,2) instead of bits (0,1). Why don't we do things this way?

ANSWER:
A trit could take the values 0, 1 or 2. So, if we use 4-trits to store our numbers they would be represented like these examples

$0\ 0\ 0\ 0 = 0 * 3^3 + 0 * 3^2 + 0 * 3^1 + 0 * 3^0 = 0$
$0\ 2\ 2\ 1 = 0 * 3^3 + 2 * 3^2 + 2 * 3^1 + 1 * 3^0 = 25$
$2\ 2\ 2\ 2 = 2 * 3^3 + 2 * 3^2 + 2 * 3^1 + 2 * 3^0 = 80$

If we used a 8-trit (a tryte, maybe?) then there are $3^8 = 6561$ possible unsigned integers or 0 to 6560. Similarly, if we use *n*-trits we can get 0 to $3^n - 1$.

The way I understand it, computers use bits because the two states correspond to "on" and "off", or the "presence of charge" and the "absence of charge". Having a third state would require being able to measure a weaker charge or a half-charge, something that is not really reliable.

2. If each element of an array RM,with 10 rows and 20 columns, stored in row major order, takes four bytes of space, where the first element of RM starts at 100, what is the address of RM[5][3] and RM[9][19]?

ANSWER:
Assuming that "100" is decimal and that each byte increments the address by 1 in decimal. Also assuming that this matrix is indexed from (1, 1). So, the address of RM[1][1] is 100 and the address of RM[1][2] is 104, etc.

There are 20 columns in row-major order so,
address of RM[1][1] is 100
address of RM[2][1] is 104
address of RM[3][1] is 108
address of RM[n][1] is 100 + (n – 1) * 4
address of RM[20][1] is 176

So, the address of RM[2][1] (the next element) is going to be 180 or 100 + (2 – 1) * 20 * 4. The address of RM[r][c] is

$$(r - 1) * 80 + (c - 1) * 4 + 100$$

The address of RM[5][3] will be

$$(5 - 1) * 80 + (3 - 1) * 4 + 100 = 428$$

And the address of RM[9][19] is

$$(9 - 1) * 80 + (19 - 1) * 4 + 100 = 812$$

3. A lower triangular matrix is an nxn array in which has a[i][j] = = 0 if i<j. What is the maximum number of non zero elements? How can they be stored in memory sequentially? Find a formula k= f(i,j) to store location a[i][j] in k (you only want to store the nonzero elements). Do not write code. Code would work if you were manually converting the matrix from one form to the other all at once, but you need the formula to convert otherwise.

ANSWER:

An *n* by *n* lower diagonal matrix will have a maximum number of non-zero elements if all the entries in the lower diagonal are non-zero. The first column doesn't have to have any zeros, the second must have 1 (or n – 1 non-zeros max), etc. The last column will have exactly one non-zero entry, as a max. So, the max number of non-zero entries is

$$1 + 2 + \cdots + (n-1) + n = \frac{n(n+1)}{2}$$

The matrix

$$\begin{bmatrix} e_{1,1} & 0 & 0 & 0 \\ e_{2,1} & e_{2,2} & 0 & 0 \\ e_{3,1} & e_{3,2} & e_{3,3} & 0 \\ e_{4,1} & e_{4,2} & e_{4,3} & e_{4,4} \end{bmatrix}$$

can be indexed as follows

$$\begin{bmatrix} x_1 & 0 & 0 & 0 \\ x_2 & x_3 & 0 & 0 \\ x_4 & x_5 & x_6 & 0 \\ x_7 & x_8 & x_9 & x_{10} \end{bmatrix}$$

The subscript of x, call it k, can calculated as a function of the indexes of the *e*'s, or *i* and *j*. The first element in row *i* has $\frac{1}{2}(i-1) * i$ elements before it, using the equation above. So,

$$k = j + \frac{i(i-1)}{2}$$

where $x_k = e_{i,j}$ and $i \leq j$.

4. A tridiagonal matrix is an nxn array in which has a[i][j] = = 0 if |i-j| > 1. What is the maximum number of non zero elements? How can they be stored in memory sequentially? Find a formula k= f(i,j) to store location a[i][j] in k, when |i-j| <=1 (you only want to store the nonzero elements). Do not write code. Code would work if you were manually converting the matrix from one form to the other all at once, but you need the formula to convert otherwise.

ANSWER:
A tri-diagonal matrix looks something like this

$$\begin{bmatrix} e_{1,1} & e_{1,2} & 0 & 0 & 0 \\ e_{2,1} & e_{2,2} & e_{2,3} & 0 & 0 \\ 0 & e_{3,2} & e_{3,3} & e_{3,4} & 0 \\ 0 & 0 & e_{4,3} & e_{4,4} & e_{4,5} \\ 0 & 0 & 0 & e_{5,4} & e_{5,5} \end{bmatrix} \text{ or } \begin{bmatrix} x_1 & x_2 & 0 & 0 & 0 \\ x_3 & x_4 & x_5 & 0 & 0 \\ 0 & x_6 & x_7 & x_8 & 0 \\ 0 & 0 & x_9 & x_{10} & x_{11} \\ 0 & 0 & 0 & x_{12} & x_{13} \end{bmatrix}$$

The maximum number of non-zero elements will always be $n + 2(n-1) = 3n - 2$.

To convert from *(i, j)* to *k,* notice that the element indexed by *(m, m)* is always going to have

$$k = 3m - 2$$

because of the equation above. Then we just need to add one or subtract one to get the adjacent elements of the matrix. So, generally,

$$k = 3i - 2 + (j - i)$$
$$k = 2i - 2 + j$$

5. Consider two functions **f(n)=an^2**, and **g(n)=bnlgn**. For what value of **n** do they intersect, and at which value(s) of the constants **a** and **b**? Pick some values of **a** and **b** and then find **n**. Experiment with different sets of values for **a and b**. What trends do you observe? **an^2** and **bnlgn** are terms that might come from an expression representing the amount of work done by a piece of code. Looking for where they are equal will help you decide which part is dominant. *We suggest solving this problem empirically, rather than mathematically, i.e. draw graph of the functions.* Remember that **a** and **b** are constants, but may not necessarily be integer.

ANSWER:
n has to be greater than 0.

$$f(n) = g(n)$$
$$a * n^2 = b * n * \log_2 n$$
$$a * n = b * \log_2 n$$
$$c * n = \log_2 n$$
$$c = \frac{\log_2 n}{n}$$

for $c = a/b$. The right hand side is always going to less than 1.0, so for f and g to meet, a has to be less than b. In fact, you can prove the the right hand side has a maximum at Euler's number. Define h(n) as

$$h(n) = \frac{\log_2 n}{n}$$
$$h(n) = \frac{1}{\ln 2} * \frac{\ln n}{n}$$

$$h'(n) = 0 = \frac{d\left(\frac{\ln n}{n}\right)}{dn}$$
$$0 = n * n^{-1} - \ln n$$
$$0 = 1 - \ln n$$
$$\ln n = 1$$
$$n = e$$
$$c_{max} = h(e) \cong 0.5307$$

So for g and f to intersect c will need to be less than or equal to that. When c = c-max they intersect right at e = 2.71828. And this is true regardless of the values of a and b (although bigger a and b will give bigger values of f and g)

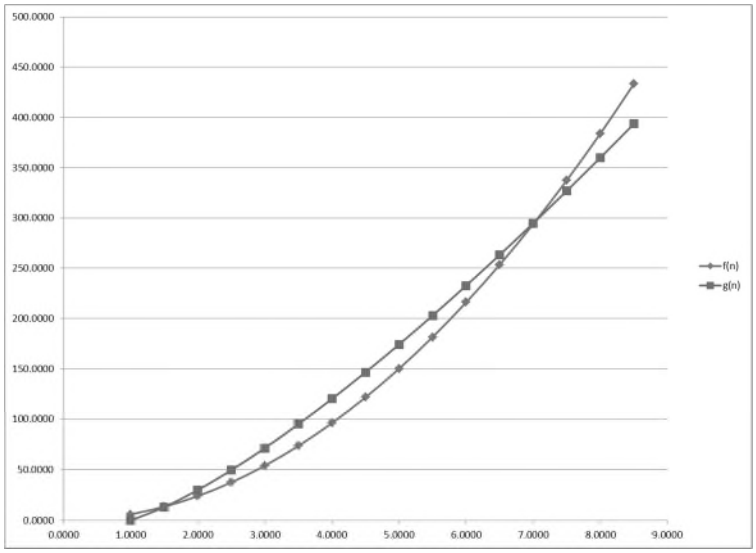
a	b	c	n	f(n)	g(n)
2.6537	5.0000	0.5307	2.7183	19.6083	19.6083
5.3074	10.0000	0.5307	2.7183	39.2165	39.2165
7.9611	15.0000	0.5307	2.7183	58.8248	58.8248
10.6148	20.0000	0.5307	2.7183	78.4330	78.4330
13.2684	25.0000	0.5307	2.7183	98.0413	98.0413
15.9221	30.0000	0.5307	2.7183	117.6496	117.6496

When c is less than c-max there are two intersection points on either side of n = e

a	b	c	n	f(n)	g(n)
2.5000	5.0000	0.5000	2.0000	10.0000	10.0000
2.5000	5.0000	0.5000	2.7183	18.4726	19.6083
2.5000	5.0000	0.5000	4.0000	40.0006	40.0006

a	b	c	n	f(n)	g(n)
6.0000	15.0000	0.4000	1.5272	13.9941	13.9944
6.0000	15.0000	0.4000	2.7183	44.3343	58.8248
6.0000	15.0000	0.4000	7.0378	297.1859	297.1859

Using the same values for a and b gives this graph, where you can see the two intersection points.



If f and g represent the work done by two pieces of code with input size n, g(n) will almost always perform better than f(n). But when a and b are just right there will be a small range of n values around e where g(n) performs better.

For Q6 and Q7 just worry about time, not space. Feel free to leave your response in exponential form.

6. What is the maximum size of a problem that can be solved in one hour if the algorithm takes lg n microseconds?

ANSWER: there are 3.6 billion microseconds per hour. So, the answer is n = 2 ^ (3.6 billion) . . . that is really an astronomically large number.

7. What is the maximum size of a problem that can be solved in one hour if the algorithm takes n^3 microseconds?

ANSWER: there are 3.6 billion microseconds per hour. So, the answer is the cube root of 3.6 billion or n = 1532.62 about.