Homework Assignment 1 - Complexity and ADTs

Q1 - Q4 are related to the ADT material. In the lecture, we talked about "outside the black box" and "inside the black box". The related discussion question will give you some practice with working "outside the black box". The assignment is intended to help you think about "inside the black box", by having you look at alternative representations. In Q1, you look at an alternative to binary. In Q2, you look at where stuff might be stored as a consequence of a selected implementation. In Q3 and Q4, you look at an alternative data structures to leverage the fact that many locations in a standard array might always be empty. In Q6 and Q7 you look at how the time is affected by the underlying cost of the algorithm. Question 4 is extra credit. The score adjustment will be made after the Homework is graded.

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:

Number	Representation
0	0000
1	0001
2	0002
3	0010
4	0011
5	0012
6	0020
7	0021
8	0022
9	0100
…	…

I believe the reason that we do not do things this way is the hardware of a standard computer. You can store information in computers as 1 and 0 with transistors. I do not believe that tertiary transistors are nearly as cheap as binary transistors, or as easy to work with, so we use binary to store information. I believe quantum computers can store data in multiple states (rather than just 1 and 0), but that tests the limits of my understanding, and is likely beyond the scope of this course.

In summary, we store information in bits because that is how transistors can store the data.

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:
The Array RM looks like the following:

```
RM = [[  0,  1,  2,  3,  4,  5,  6,  7,  8,  … , 19],
      [20, 21, 22, 23, … ],
      [40, 41, 42, … ],
      …
     ]
```

RM[5][3] is the sixth row (5+1) and fourth column (3+1). The value of RM at this location is 103. Multiply this by 4 bytes per element and you get the value 412. Adding the base address of 100, and taking into account the element length of 4 bytes, the address for RM[5][3] should be [512, 513, 514, 515].

RM[9][19] is the tenth row (9+1) and twentieth column (19+1). The value of RM at this location is 199. Multiply this by 4 bytes per element and you get the value 796. Adding the base address of 100, and taking into account the element length of 4 bytes, the address for RM[5][3] should be [896, 897, 898, 899].

In questions 3 and 4, we suggest indexing the array from one, since matrices are normally indexed from (1,1). After you find your formula, then you can adjust it to index the array from zero.

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:
One example of such a matrix is as follows: (The highlighted index is used for demonstration purposes for the second portion of the quesiton)

a = [[1, 0, 0, 0],
[1, 1, 0, 0],
[1, 1, 1, 0],
[1, 1, 1, 1]]

Because in the first row, there can be up to 1 non-zero element, and in the second row there may be up to 2 non-zero elements, and in the third row there may be up to three non-zero elements, the max number of non-zero elements = $n!$

With the first row at i=0 and the first column at j=0, the index k in a matrix where you can store a non-zero element can be given as:

$$k = i! + j$$

If the first index in k is 0, the highlighted value in matrix a should be stored at index 8. For the equation, i=3, j=2.

$$3! + 2 = 8$$

It works!

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:

One example of such a matrix is as follows: (The highlighted index is used for demonstration purposes for the second portion of the quesiton)

```
a  = [[1, 1, 0, 0, 0, 0],
      [1, 1, 1, 0, 0, 0],
      [0, 1, 1, 1, 0, 0],
      [0, 0, 1, 1, 1, 0],
      [0, 0, 0, 1, 1, 1],
      [0, 0, 0, 0, 1, 1]]
```

In the first row you can store up to 2 non-zero elements. The same goes for the last row. In all the middle rows you can store up to 3 non-zero elements. Therefore a function for the max number of non-zero elements can be given as:

$$non - zero\ elems = 4 + 3(n - 2)$$
$$simplifies\ to: 3n - 2$$

A formula for storing these elements can be given as a piecewise equation

$$i > j \rightarrow 3(i)$$
$$i = j \rightarrow 3(i) + 1$$
$$i < j \rightarrow 3(i) + 2$$

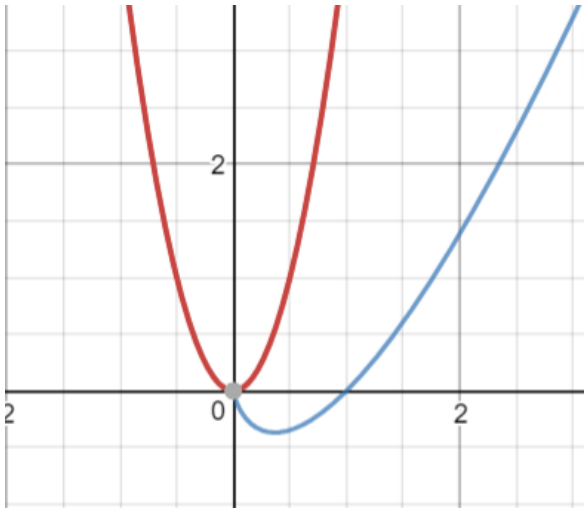
Using the highlighted element, for i=2 and j=3 we get

$$3(2) + 2 = 8$$

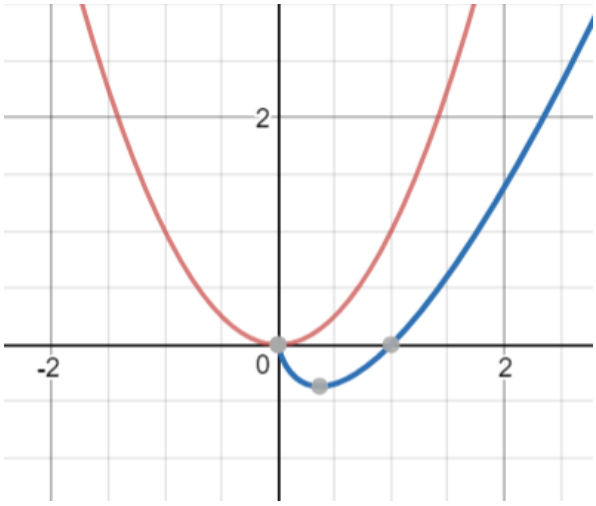
It works!

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.

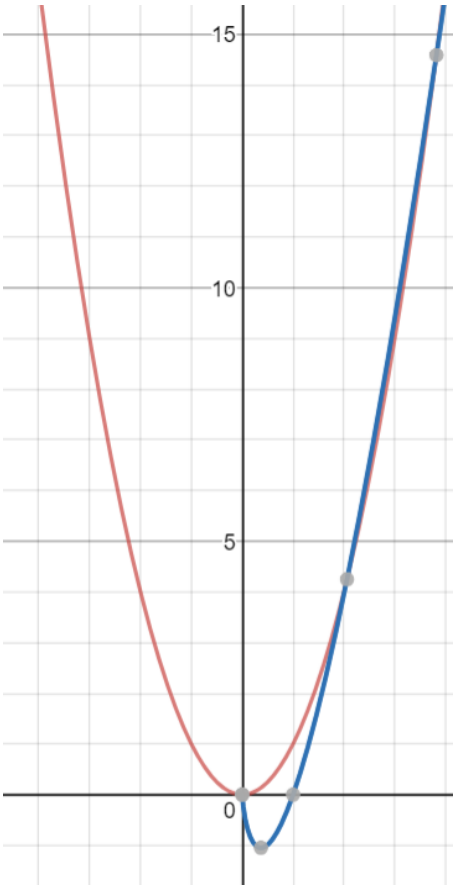
Answer:
For b > a, the lines never intersect, f(n) is always greater. See chart below (a=4, b=1):



For b = a, the lines still never intersect, f(n) is always greater. See chart below:



For b > a, the lines may intersect twice. I used values of b=2.85 and a=1. Intersections happened at (2.061, 4.247) and (3.819, 14.584). See chart below



Charts plotted using: https://www.desmos.com/calculator

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?

$$1\ hour\ =\ 60\ minutes\ =\ 3600\ seconds\ =\ 3{,}600{,}000{,}000\ microseconds$$

Solving for n, we get

$$n = 2^{3{,}600{,}000{,}000} = 10^{10^{9.035}}$$

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

$$n^3 = 3{,}600{,}000{,}000$$

Rounding down we get:

$$n = 1532$$