Jeffrey Wan/ Spring 2020

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.

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?***

I think we don't do trit representation because of the difficulty in internal representation? That's my guess. Bits can be represented as powers of 2 being added or not added. For example, 0101 is 2^0 + 2^2 = 5 and that's easy to do.

However, trits are more difficult. If there was a physical way do represent 0,1,2 even… it's not as easy to do binary or trinary math. 0211 would be 18 + 3 + 1 = 22 which is more complicated than simple binary math. I'm sure what the equation is but it involves powers of 3 as we move further left and mutiplication by two so that's more computation steps and therefore more T(N) to do calculations. I think that's why we we stick with binary representation.

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]?**

The array RM drawn out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
81 …
…

RM[5][3] = 100 + (5 * 20 * 4) + (3*4) = 512
RM[9][19] = 100 + (9 * 20 * 4) + (19*4) = 720 + 76 = 100 + 796 = 896

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.**

Say n = 5 and the top left is 1,1 and the one to the right of it is 1,2

x0000
xx000
xxx00
xxxx0
xxxxx

There are 10 zeroes

Say n = 4

x000
xx00
xxx0
xxxx

for every row added, you have N more elements. So for a N=3 matrix, you have 1 + 2 + 3 non-zero elements which is 6. This is a famous proof where the summation is the sum is N(N+1)/2.

What does it mean to be stored sequentially? You can store the non-zero elements in an array?

i-values can be: 0,0,1,0,1,2,0,1,2,3,0,1,2,3,4
Those i values are obtained moving from left to right, top down. In the first row, there is 1 x so the i value is 0. On the second row, there are 2 xs so the i values are 0 and 1.

j values correspond to the x values so can be stored like this:
j: 0, 0,1,0,1,2,0,1,2,3,0,1,2,3,4

k values can be:
k: 0,1,2,3,4,5,6,7,8,9,10…

Some values we want to store are 0, 5, 6, 10, 11, 12, etc… a formula could be:
??? Unsure of what the formula could be.

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.**

So I had to look up what a tridiagonal matrix was. From wolfram:

> Tridiagonal Matrix
> A square matrix with nonzero elements only on the diagonal and slots horizontally or vertically adjacent the diagonal (i.e., along the subdiagonal and superdiagonal),

So for N = 5,

xx000
xxx00
0xxx0
00xxx
000xx

That has 12 0s and 13 non zeroes so maybe the equation for non-zeroes is 3n-2 which would be 15-2 = 13

There are 2 non-zero values on the first row, 3 on the second row, 3 on the third, 3 on the 4th, 2 on the last. So i can be stored as:

i : 0 0 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 …

j values. The first j value has a column value of 0, the value to the right of that one has a j value of 1. ON the second row, the j values are 0,1,2 so in sequence, "j" can be stored as:
j : 0 1 0 1 2 1 2 3 2 3 4 3 4 5 4 5 6 ...

k can be something like this:
k: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 …

The equation for k values that are stored that are non-zero could be something like:

k = 3 * (i – 1) + 3 + j – i = 2i + j
For example, for i = 2, j = 1, 5 should be stored. 2i+j = 2*2+1 = 5 so it checks out!

5. **Consider two functions f(n)=an², 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? an² 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.**

Say a = 10, and b = 100. so f(n) = 10n^2 and g(n) == 100nlgn. They intersect at 0 and 1.371,18.804

I observe the g(n) gets very negative for small values of n and begins at 0 and is only valid for positive values of 0. f(n) is larger than g(n) for large values of n and the domain includes negative n. They look similar for larger and larger values of n and so I think they are similar algorithms but f(n) has an n^2 in the equation so grows faster than g(n) which explains why it is above g(n) for every value of n after a sufficiently large n.

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?**

microsecond = 1 millionth of a second
1 hour has 3600 seconds
microsecond is 10^(-6)

3600 = lg(N)*10^(6)
e^3600 = 10^(6)*N
e^3600/(10^(6)) = N

7.**What is the maximum size of a problem that can be solved in one hour if the algorithm takes n³ microseconds?**

3600 = n^3*10^(6)
3600/10^6 = n^3
the positive cube root of (3600/10^6) = N