Problem Set 2
ISE 407 – Computational Methods in Optimization
Due: September 27, 2019
Dr. Ralphs

1. AHU, 1.3-1.4

2. Miller and Boxer, Chater 1, Problem 2.

3. In practice, many matrix computations involve sparse matrices. Data structure for storing sparse vectors and matrices store only the nonzero entries. Although this requires storing not only the values of the nonzeros, but their location within the vector/matrix, it is usually still more memory efficient to use such data structures than to explicitly store all matrix entries. This problem examines some of the details of the efficiency of these data structures. In a future assignment, you will be asked to implement and empirically test a data structure for sparse matrices.

    (a) Describe in detail a data structure for storing a sparse matrix in C. Try to determine at approximately what level of sparsity (percentage) the sparse matrix data structure uses less memory than the explicit representation.

    (b) Compare the efficiency with which such a sparse matrix can be initialized to all zeros with the efficiency with which the same operation can be performed using an explicitly stored matrix.

    (c) Give a pseudocode implementation of a function for multiplying two matrices stored in your sparse representation without taking into account any cache effects.

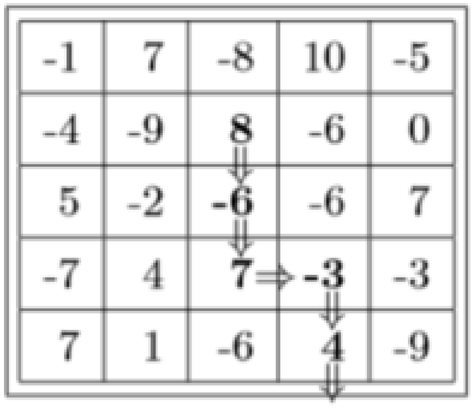
    (d) Analyze the efficiency with which two sparse matrices can be multiplied using your data structure and compare it to the efficiency with which the same operation can be performed on two explicitly stored matrices using a straightforward theoretical analysis that doesn't take into account cache effects.

    (e) In practice, how would you take cache effects into account in implementing sparse matrix multiplication? What are the differences (if any) from the case with explicitly stored matrices?

    (f) Compare your dense matrix multiplication to the sparse matrix multiplication provided by NumPy/SciPy (empirically).

4. Valkin's Mile is a solitaire game played on an $n \times n$ square grid in which each grid square contains an integer (positive or negative). The player starts in any chosen square and the may move either right or down in each turn until some move results in exiting the board. The score for a given sequence is the sum total of the values of all squares in the path. For example, the sequence shown in Figure 1 would result in the score $8 - 6 + 7 - 3 + 4 = 10$ (this is not the best possible score on this board).

    (a) Write pseudo-code for an algorithm to determine the maximum score for a given board as efficiently as possible (hint: use recursion). In part (c), you will need



-1	7	-8	10	-5
-4	-9	**8**	-6	0
5	-2	**-6**	-6	7
-7	4	**7**	**-3**	-3
7	1	-6	**4**	-9

Figure 1: A game of Valkin's Mile

to prove that the running time of your algorithm cannot be made any more efficient so think carefully about whether your algorithm can be improved. You must justify (at least informally) that your algorithm is correct.

(b) Determine the running time of your algorithm.

(c) Explain why your algorithm is asympototically optimal (obviously, this means that your algorithm should actually be asymptotically optimal, which the naive versions are not).