TAKE-HOME CLASS QUIZ SOLUTIONS: DUE MONDAY OCTOBER 14: MATRIX COMPUTATIONS

MATH 196, SECTION 57 (VIPUL NAIK)

PLEASE DO *NOT* DISCUSS ANY QUESTIONS EXCEPT THE STARRED OR DOUBLE-STARRED QUESTIONS.

1. Performance review

28 people took this 5-question quiz. The score distribution was as follows:

- Score of 1: 1 person
- Score of 2: 3 people
- Score of 3: 6 people
- Score of 4: 11 people
- Score of 5: 7 people

The question-wise answers and performance review are below:

(1) Option (C): 27 people
(2) Option (B): 25 people
(3) Option (B): 13 people
(4) Option (D): 18 people
(5) Option (B): 21 people

2. Solutions

This quiz has a few questions on the mechanics of the computational execution of Gauss-Jordan elimination, and it has one question on setting up a linear system.

Suppose f is a function on the positive integers that takes positive integer values. Suppose n is a parameter related to the input size of an algorithm. We say that the running time of an algorithm (respectively, the space requirement of the algorithm) is:

- $O(f(n))$ if, for large enough n, it can be bounded from above by a positive constant times $f(n)$.
- $\Omega(f(n))$ if, for large enough n, it can be bounded from below by a positive constant times $f(n)$.
- $\Theta(f(n))$ if it is both $O(f(n))$ and $\Omega(f(n))$.

You can read more at:
`http://en.wikipedia.org/wiki/Big_O_notation`

(1) (*) If you treat each arithmetic operation (addition, subtraction, multiplication, division) of numbers as taking constant time, and all entry rewrites and changes as again taking constant time per entry, what would be the best description of the worst-case running time of the algorithm to convert a $n \times n$ matrix to reduced row-echelon form? (Note that this complexity is termed *arithmetic complexity* and can be distinguished from the *bit complexity* of the algorithm, which could be considerably higher).
(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^3)$
(D) $\Theta(n^4)$
(E) $\Theta(n^5)$
Answer: Option (C)
Explanation: We have $\Theta(n^2)$ row operations, and the row operations all take $\Theta(n)$ time. Overall, we get $\Theta(n^3)$ as the arithmetic complexity.

More can be found in the lecture notes on Gauss-Jordan elimination. You can also learn more about the arithmetic complexity of Gaussian elimination by looking it up online.

Performance review: 27 out of 28 got this. 1 chose (B).

Historical note (last time): 24 out of 27 got tihs. 1 each chose (A), (B), and (E).

(2) (*) If you treat each arithmetic operation (addition, subtraction, multiplication, division) of numbers as taking constant space, and all matrix entries as taking constant space, what would be the best description of the worst-case space requirement of the algorithm to convert a $n \times n$ matrix to reduced row-echelon form? Assume that space is reusable, i.e., it is possible to rewrite over existing space used.

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^3)$
(D) $\Theta(n^4)$
(E) $\Theta(n^5)$

Answer: Option (B)

Explanation: We can reuse the matrix space to keep rewriting over existing entries. We need some additional workspace for working memory, but this is quite small relative to the size of the matrix, and does not affect the order estimate.

More in the lecture notes on Gauss-Jordan elimination.

Performance review: 25 out of 28 got this. 2 chose (D), 1 chose (C).

Historical note (last time): 26 out of 27 got this. 1 chose (A).

(3) (*) Suppose the coefficient matrix of a linear system with n variables and n equations is known in advance, and we can spend as much time processing it as we desire in advance (this time will not count towards the running time of the algorithm). In other words, we can use Gauss-Jordan elimination to row-reduce the coefficient matrix in advance. However, we do not have the output column with us in advance. What is the worst-case running time of the part of the algorithm that runs after the output column is known?

(A) $\Theta(n)$
(B) $\Theta(n^2)$
(C) $\Theta(n^3)$
(D) $\Theta(n^4)$
(E) $\Theta(n^5)$

Answer: Option (B)

Explanation: If we store the sequence of row operations used to convert the coefficient matrix to reduced row-echelon form (there are $\Theta(n^2)$ such operations) we simply need to apply these operations to the output column, then read off the solutions. The arithmetic time complexity of this is $\Theta(n^2)$.

Performance review: 13 out of 28 got this. 11 chose (A), 3 chose (C), 1 chose (D).

Historical note (last time): 17 out of 27 got this. 5 chose (A), 4 chose (C), 1 chose (E).

(4) Which of the following matrices does *not* have the identity matrix as its reduced row-echelon form?

(A)

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

(B)

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 5 \\ 0 & 0 & 7 \end{bmatrix}$$

(C)

$$\begin{bmatrix} 4 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 5 & -6 \end{bmatrix}$$

(D)

$$\begin{bmatrix} 1 & 2 & -3 \\ 4 & -3 & -1 \\ -2 & 1 & 1 \end{bmatrix}$$

(E)

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 7 \\ 3 & 7 & 11 \end{bmatrix}$$

Answer: Option (D)

Explanation: We can work to convert to rref to verify this, but one easy way of seeing that this matrix does not have full rank is to note that each row sum is 0, which indicates that $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ are both solutions to the system of linear equations with this as the coefficient matrix and outputs all zeros. In other words, the solution space is not zero-dimensional, and hence, the matrix does not have full rank.

As for the other options:

- Option (A): This is a diagonal matrix with all entries nonzero. Thus, its rref is the identity matrix. It is easy to see this: just divide each row by its diagonal entry.
- Option (B): This is upper-triangular. We can first convert the diagonal entries to 1 by dividing. Then, we can do row subtractions and clear out everything above the diagonal.
- Option (C): This is lower-triangular, and works for a reason similar to Option (B).
- Option (E): This actually needs to be worked out using row reduction.

Performance review: 18 out of 28 got this. 4 chose (E), 2 each chose (B) and (C), 2 wrote incorrect free-form responses.

Historical note (last time): 24 out of 27 got this. 1 each chose (B), (C), and (E).

(5) A number of different consumer price indices have been constructed. All of them use the market prices for an existing collection of commodities (though not all of them use every commodity in the collection) and take a different "weighted" linear combination of those. For instance, one price index might be 3 times (the price per ton of wheat on the Chicago wheat market) + 4 times (the price of 1 gallon of unleaded gasoline at a particular gas station) + 17 times (the price of Burt's chapstick). Another price index might use 30 times (the price of Transcend's 32 GB flash drive) + 14 times (the price of 1 gallon of gasoline at a particular gas station).

What is a good way of modeling these?

(A) The prices of the various goods are stored in a matrix, the different weightings used in various indices are stored in a vector, and the consumer price indices arise as the output vector of the matrix-vector product.

(B) The different weightings used in various indices are stored in a matrix, the prices of the various goods are stored in a vector, and the consumer price indices arise as the output vector of the matrix-vector product.

(C) The prices of the various goods are stored in a matrix, the consumer price indices are stored as a vector, and the weightings used in the indices arise as the output vector of the matrix-vector product.

(D) The different weightings used in various indices are stored in a matrix, the consumer price indices are stored in a vector, and the prices of the various goods arise as the output vector of the matrix-vector product.

(E) The consumer price indices are stored in a matrix, the prices of the various goods are stored in a vector, and the weightings used in the indices arise as the output vector of the matrix-vector product.

Answer: Option (B)

Explanation: Each row represents the weightings used for a particular index. The input column is the input vector of prices. The output column is the vector of the various index values.

Performance review: 21 out of 28 got this. 7 chose (A).

Historical note (last time): 22 out of 27 got this. 4 chose (A), 1 chose (C).