TAKE-HOME CLASS QUIZ SOLUTIONS: DUE MONDAY OCTOBER 28: MATRIX MULTIPLICATION AND INVERSION AS COMPUTATIONAL PROBLEMS

MATH 196, SECTION 57 (VIPUL NAIK)

1. Performance review

26 people took this 16-question quiz. The score distribution is as follows:

- Score of 3: 1 person
- Score of 6: 2 people
- Score of 7: 1 person
- Score of 8: 9 people
- Score of 9: 3 people
- Score of 10: 5 people
- Score of 11: 3 people
- Score of 12: 1 person
- Score of 15: 1 person

The question-wise answers and performance review are below:

(1) Option (A): 25 people
(2) Option (C): 23 people
(3) Option (B): 12 people
(4) Option (A): 15 people
(5) Option (C): 15 people
(6) Option (E): 20 people
(7) Option (B): 9 people
(8) Option (B): 11 people
(9) Option (A): 8 people
(10) Option (B): 5 people
(11) Option (B): 21 people
(12) Option (A): 24 people
(13) Option (D): 10 people
(14) Option (E): 1 person
(15) Option (D): 18 people
(16) Option (A): 14 people

2. Solutions

This quiz tests for a strong *conceptualization* (i.e., a metacognition) of the processes used for matrix multiplication and inversion. It is based on part of the `Matrix multiplication and inversion` notes and is related to Sections 2.3 and 2.4. It does not, however, test all aspects of that material.

PLEASE FEEL FREE TO DISCUSS *ALL* QUESTIONS.

(1) How many arithmetic operations are needed for naive matrix multiplication of a $m \times n$ matrix and a $n \times p$ matrix?
   (A) $O(mnp)$ additions and $O(mnp)$ multiplications
   (B) $O(m+n+p)$ additions and $O(mnp)$ multiplications
   (C) $O(mn)$ additions and $O(np)$ multiplications
   (D) $O(mn+mp)$ additions and $O(mnp)$ multiplications
   (E) $O(m+n+p)$ additions and $O(m+n+p)$ multiplications

Answer: Option (A)

Explanation: The actual number of multiplications is mnp and the actual number of additions is $m(n-1)p$, both of which are $O(mnp)$. In fact, it is $\Theta(mnp)$. (The O notation means an upper bound on order, the Θ notation means upper and lower bounds on order).

Performance review: 25 out of 26 got this. 1 person chose (D).

Historical note (last time): 21 out of 26 got this. 5 chose (D).

(2) What is the arithmetic complexity (in terms of total number of arithmetic operations needed) for naive matrix multiplication of two generic $n \times n$ matrices?

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

Answer: Option (C)

Explanation: Based on Question 1, where m, n, p are all equal to n.

Performance review: 23 out of 26 got this. 3 people chose (D).

Historical note (last time): 23 out of 26 got this. 2 chose (B), 1 chose (D).

(3) Which of the following is the tightest "obvious" lower bound on the possible arithmetic complexity of any generic algorithm for multiplying two $n \times n$ matrices? We use Ω to denote *at least that order*.

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

Answer: Option (B)

Explanation: The product matrix has n^2 entries, all of which could in principle be different and require at least some nonzero computation. In other words, just filling in the output matrix takes n^2 steps. Thus, the obvious lower bound for matrix multiplication is $\Omega(n^2)$.

Performnance review: 12 out of 26 got this. 8 chose (A), 6 chose (C).

Historical note (last time): 12 out of 26 got this. 10 chose (C), 4 chose (A).

(4) What is the minimum number of arithmetic operations needed to compute the product of two generic diagonal $n \times n$ matrices?

(A) n
(B) $n+1$
(C) $2n-1$
(D) $2n$
(E) n^2

Answer: Option (A)

Explanation: All the off-diagonal entries are 0. The diagonal entries are obtained by entry-wise multiplication. Explicitly, if $AB = C$, then:

$$c_{ii} = a_{ii}b_{ii}$$

Thus, each of the n diagonal entries of the matrix requires one multiplication to compute, so the total number of operations necessary is n.

Performance review: 15 out of 26 got this. 8 chose (E), 2 chose (C), 1 chose (D).

Historical note (last time): 20 out of 26 got this. 4 chose (E), 2 chose (C).

(5) What is the minimum number of arithmetic operations needed to compute the product of a generic $1 \times n$ matrix and a generic $n \times 1$ matrix?

(A) n
(B) $n+1$
(C) $2n-1$
(D) $2n$
(E) n^2

Answer: Option (C)

Explanation: This is a dot product computation. It involves n multiplications and $n-1$ additions, so a total of $2n-1$ operations. Alternatively, recall that in general, for multiplying a $m \times n$ matrix and a $n \times p$ matrix, we require mnp multiplications and $m(n-1)p$ additions. Here, $m = p = 1$.

Performance review: 15 out of 26 got this. 8 chose (A), 2 chose (B), 1 chose (E).

Historical note (last time): 16 out of 26 got this. 9 chose (A), 1 chose (D).

(6) What is the minimum number of arithmetic operations needed to compute the product of a generic $n \times 1$ matrix and a generic $1 \times n$ matrix?

(A) n
(B) $n+1$
(C) $2n-1$
(D) $2n$
(E) n^2

Answer: Option (E)

Explanation: This is the Hadamard product or outer product. The product is a $n \times n$ matrix and each entry is simply one product. Thus, there are n^2 multiplications and 0 additions, so a total of n^2 operations.

We can think of this in terms of the general setting of multiplication of a $m \times n$ matrix and a $n \times p$ matrix. The n in our current situation equals both the m and the p of the generic setup and the n of our generic setup equals 1. The number of multiplications is $mnp = n(1)n = n^2$ and the the number of additions is $n(1-1)n = 0$.

Performance review: 20 out of 26 got this. 4 chose (A), 2 chose (C).

Historical note (last time): 14 out of 26 got this. 6 chose (A), 3 chose (C), 2 chose (D), 1 chose (B).

(7) What is the minimum number of arithmetic operations needed to compute the product of a generic $n \times n$ diagonal matrix and a generic $n \times n$ upper triangular matrix? The upper triangular matrix has zero entries below the diagonal. The entries on or above the diagonal may be nonzero (and generically, they will be nonzero).

(A) $n(n-1)/2$
(B) $n(n+1)/2$
(C) $n(n-1)$
(D) n^2
(E) $n(n+1)$

Answer: Option (B)

Explanation: To compute the $(ij)^{th}$ entry of the product, we need to take the dot product of the i^{th} row of the diagonal matrix and the j^{th} column of the other matrix. There is only one nonzero term in the corresponding summation if $i \leq j$, and no nonzero term if $i > j$. So, we have to do 0 additions, and the number of multiplications is the number of entries in the upper triangular part.

So, we need to calculate the number of entries in the upper triangle including the diagonal. There are n positions on the diagonal. There are thus $n^2 - n$ off-diagonal entries, with half of them above the diagonal and half of them below the diagonal. Thus, there are $(n^2-n)/2 = n(n-1)/2$ entries above the diagonal, so a total of $n(n-1)/2 + n = n(n+1)/2$ entries on and above the diagonal.

Performance review: 9 out of 26 got this. 7 each chose (A) and (C), 2 chose (D), 1 left the question blank.

Historical note (last time): 19 out of 26 got this. 5 chose (D), 2 chose (B).

Adding n numbers to each other requires $n-1$ addition operations. In a non-parallel setting, there is no way of improving this.

However, using the associativity of addition, we can write a faster parallelizable algorithm. A simple parallelization is to split the list being added into two sublists of length about $n/2$ each. Delegate the task of adding up within each sublist to different processors running in parallel. Then, add up the numbers obtained. This takes about half the time, with a little overhead (of collecting and adding up). This type of strategy is called a *divide and conquer* strategy. Using a divide and conquer strategy repeatedly, we can demonstrate that the parallelized arithmetic complexity of this approach is $\Theta(\log_2 n)$.

(8) Suppose A is a $1 \times n$ matrix and B is a $n \times 1$ matrix. Assume an unlimited number of processors that all have free read access to both A and B, free write access to the product matrix, and a shared workspace where they can store intermediate results. What is the arithmetic complexity in this context (i.e., the parallelized arithmetic complexity) for computing AB? What we mean here is: what is the smallest depth of a computational tree to compute AB?
(A) $\Theta(1)$
(B) $\Theta(\log_2 n)$
(C) $\Theta(n \log_2 n)$
(D) $\Theta(n^2)$
(E) $\Theta(n^2 \log_2 n)$

Answer: Option (B)

Explanation: Computing the dot product requires computing n individual products, and then adding them up. The computation of the individual products can be done in parallel, so that takes time $\Theta(1)$. The adding up involves the addition of n numbers, which takes then $\Theta(\log_2 n)$. The total time taken is thus $\Theta(\log_2 n)$.

Performance review: 11 out of 26 got this. 11 chose (C), 1 chose (D), 3 chose (A).

Historical note (last time): 22 out of 26 got this. 2 chose (C), 1 each chose (A) and (D).

(9) Suppose A is a $n \times 1$ matrix and B is a $1 \times n$ matrix. Assume an unlimited number of processors that all have free read access to both A and B, free write access to the product matrix, and a shared workspace where they can store intermediate results. What is the arithmetic complexity in this context (i.e., the parallelized arithmetic complexity) for computing AB? What we mean here is: what is the smallest depth of a computational tree to compute AB?
(A) $\Theta(1)$
(B) $\Theta(\log_2 n)$
(C) $\Theta(n \log_2 n)$
(D) $\Theta(n^2)$
(E) $\Theta(n^2 \log_2 n)$

Answer: Option (A)

Explanation: Since the number of columns in A = the number of rows in B is 1, we do not need to perform any additions. Rather, we need to do n^2 multiplications. All these multiplications can be performed in parallel, so the time taken is $\Theta(1)$.

Performance review: 8 out of 26 got this. 7 chose (B), 6 chose (D), 4 chose (C), 1 chose (E).

Historical note (last time): 10 out of 26 got this. 10 chose (C), 4 chose (B), 2 chose (D).

(10) Suppose A and B are two $n \times n$ matrices. Assume an unlimited number of processors that all have free read access to both A and B, free write access to the product matrix, and a shared workspace where they can store intermediate results. What is the arithmetic complexity in this context (i.e., the parallelized arithmetic complexity) for computing AB? What we mean here is: what is the smallest depth of a computational tree to compute AB? Use naive matrix multiplication and speed it up using the parallelized processes discused here.
(A) $\Theta(1)$
(B) $\Theta(\log_2 n)$
(C) $\Theta(n \log_2 n)$
(D) $\Theta(n^2)$
(E) $\Theta(n^2 \log_2 n)$

Answer: Option (B)

Explanation: We need to calculate n^2 matrix entries in the product, but all the entries can be computed in parallel, and the time taken to compute a matrix entry is the time taken to perform a single dot product. As we saw earlier, this is $\Theta(\log_2 n)$.

We are given a $n \times n$ matrix A and we want to use *repeated squaring* to calculate powers of A. For instance, to calculate A^4, we can simply calculate $(A^2)^2$, which requires two multiplications. To calculate A^5, we calculate $(A^2)^2 A$, which requires three multiplications. Assume that we can store any number of intermediate matrices, i.e., storage space is not a constraint.

Performance review: 5 out of 26 got this. 19 chose (E), 2 chose (C).

Historical note (last time): 13 out of 26 got this. 9 chose (E), 2 chose (D), 1 each chose (A) and (C).

(11) What is the smallest number of matrix multiplications needed to calculate A^7 using repeated squaring?

(A) 3
(B) 4
(C) 5
(D) 6
(E) 7

Answer: Option (B)

Explanation: We compute A^2 (first operation), then compute $(A^2)^2 = A^4$ (second operation), then multiply them to get A^6 (third operation), then multiply that by A to get A^7 (fourth operation).

Performance review: 21 out of 26 got this. 3 chose (A), 2 chose (C).

Historical note (last time): 21 out of 26 got this. 3 chose (C), 2 chose (A).

(12) What is the smallest number of matrix multiplications needed to calculate A^8 using repeated squaring?

(A) 3
(B) 4
(C) 5
(D) 6
(E) 7

Answer: Option (A)

Explanation: Three squarings will do the trick.

Performance review: 24 out of 26 got this. 2 chose (B).

Historical note (last time): 23 out of 26 got this. 3 chose (C).

(13) What is the smallest number of matrix multiplications needed to calculate A^{21} using repeated squaring?

(A) 3
(B) 4
(C) 5
(D) 6
(E) 7

Answer: Option (D)

Explanation: Square A four times to get to A^{16}. In the process, we have also found A^2, A^4, and A^8. Multiply A^{16} by A^4 (fifth operation) and then multiply by A (sixth operation).

Performance review: 10 out of 26 got this. 15 chose (E), 1 left the question blank.

Historical note (last time): 12 out of 26 got this. 8 chose (E), 5 chose (C), 1 chose (A).

Suppose A is an *invertible* $n \times n$ matrix. It is possible to invert A using $\Theta(n^3)$ (worst-case) arithmetic operations via Gauss-Jordan elimination. We can thus add computation of the inverse to our toolkit when calculating powers. It is helpful even when calculating positive powers.

Count each matrix multiplication and each matrix inversion as one "matrix operation."

(14) What is the smallest positive r where we can achieve a saving on the total number of matrix operations to calculate A^r by also computing A^{-1}, rather than just using repeated squaring?

(A) 3
(B) 7
(C) 15
(D) 23
(E) 31

Answer: Option (E)

Explanation: The case where we are likely to get the best saving is where $r = 2^s - 1$. In this case, doing the calculation without finding the inverse takes $2s - 2$ steps and doing the calculation using the inverse takes $s + 2$ steps: s to calculate A^{2^s}, 1 to calculate A^{-1}, and 1 more to multiply them and get A^{2^s-1}.. For $s+2 < 2s-2$, we need $s \geq 5$, so the smallest number that works is $2^5 - 1 = 31$.

Performance review: 1 out of 26 got this. 16 chose (C), 4 chose (D), 3 chose (B), 2 chose (A), and 1 left the question blank.

Historical note (last time): 8 out of 26 got this. 9 chose (C), 5 chose (B), 2 chose (D), 1 chose (A).

(15) *Strassen's algorithm* is a *fast matrix multiplication* algorithm that can multiply two $n \times n$ matrices using $O(n^{\log_2 7})$ arithmetic operations. In practice, however, a lot of existing computer code for matrix multiplication, written long after Strassen's algorithm was discovered, uses naive matrix multiplication. Which of the following reasons explain this? Please see Options (D) and (E) before answering.

(A) Strassen's algorithm becomes faster than naive matrix multiplication only for very large matrix sizes.
(B) Strassen's algorithm is more complicated to code.
(C) Strassen's algorithm is not as easily parallelizable as naive matrix multiplication.
(D) All of the above.
(E) None of the above.

Answer: Option (D)

Explanation: Strassen's algorithm for the 2×2 case requires 7 multiplications and 14 additions. The number of additions is more. The saving on multiplication does dominate for large enough matrix sizes, but this does not occur immediately.

Strassen's algorithm is definitely more complicated to code. If it were easy to code, it would also be easier to explain, and we would have discussed it in class.

Parallelization is harder because on the "conquer" part of the divide and conquer strategy. With the naive approach, each entry can be computed totally separately. With Strassen's, there are many different types of computations that need to be done and then pieced together.

Performance review: 18 out of 26 got this. 5 chose (C), 3 chose (A).

Historical note (last time): 20 out of 26 got this. 3 chose (C), 2 chose (E), 1 chose (A).

There exist even faster algorithms for matrix multiplication than Strassen's algorithm. The best known algorithm currently is the *Coppersmith-Winograd algorithm*, which can multiply two $n \times n$ matrices in time $O(n^{2.3727})$. However, the Coppersmith-Winograd algorithm is even more rarely implemented than Strassen's for practical matrix multiplication code (according to some sources, Coppersmith-Winograd has *never* been implemented). The same reasons as those cited above for

the reluctance to use Strassen's algorithm apply. There are some additional obstacles to practical implementations of the Coppersmith-Winograd algorithm that make it even more difficult to use.

(16) Suppose A and B are $n \times n$ matrices. What is the minimum number of matrix multiplications needed generically to compute the product $ABABABABA$?

(A) 4
(B) 5
(C) 6
(D) 7
(E) 8

Answer: Option (A)

Explanation: We calculate AB (first), then square it to get $ABAB$ (second), then square again to get $ABABABAB$ (third), then multiply by A to get $ABABABA$ (fourth).

Performance review: 14 out of 26 got this. 7 chose (B), 3 chose (E), 1 each chose (C) and (D).

Historical note (last time): 13 out of 26 got this. 8 chose (B), 2 chose (C), 2 chose (E), 1 chose (D).