CSC265 Fall 2020 Homework Assignment 4

The list of people with whom I discussed this homework assignment: Kunal Chawla

In the solutions to problem 1 and 2, we use the following basic facts from algebra and linear algebra[1]:

- $\mathcal{B}$ is a field with the usual integer addition and multiplication mod 2. It is also a vector space of dimension 1.
- $\mathcal{B}_n$ is a vector space of dimension n over the field $\mathcal{B}$ with the usual operations mod 2 with the standard basis $e_1, \ldots, e_n$, where $e_i \in \mathcal{B}_n$ has 1 in the ith index and 0 everywhere else.
- the kernel of a linear transformation $L : V \to W$ is the set $\ker L = \{v \in V : L(v) = 0\}$ and the range of L is the set $\text{range}\, L = \{w \in W : \exists v \in V, L(v) = w\}$ and $\ker L$ is a subspace of V, $\text{range}\, L$ is a subspace of W.
- Matrix multiplication is a linear transformation.
- Rank-nullity holds for linear transformations between finite dimensional vector spaces: for a linear transformation $L : \mathcal{B}_n \to \mathcal{B}_m$, $n = \dim \ker L + \dim \text{range}\, L$.
- A k dimensional subspace of $\mathcal{B}_n$ has 2^k elements (this is not in the reference, but the justification is that if a subspace $V \subseteq \mathcal{B}_n$ is spanned by linearly independent vectors $v_1, \ldots, v_k$, then V is the set of linear combinations of $v_1, \ldots, v_k$, of which there are 2^k).

1. **Solution:** Since $a \neq b$, we know that there exists a nonzero entry in $a - b$. We also have that $aX = bX$ iff $aX - bX = 0$ iff $(a - b)X = 0$ (by linearity of the inner product, we can combine a and b and write $a - b$). Notice that we can view $(a - b)$ as a 1 by n boolean matrix and now $(a - b)X$ becomes a multiplication of a matrix and vector.

   Now, we can view multiplication by $(a - b)$ as a linear transformation from $\mathcal{B}_n$ to $\mathcal{B}$. Our problem reduces to finding the size of $(a - b)$'s kernel. Since $(a - b)$ has a nonzero entry (suppose that is the jth entry), we know that if we take $X = (0, \ldots, 1, \ldots, 0) = e_j$ (all entries are 0 except for the jth), we have $(a - b)X = 1$. Also, if we take $X = (0, \ldots, 0)$, the product is 0. Thus, we have shown that $\text{range}(a - b) = \mathcal{B}$, which has dimension 1. By rank-nullity, we know $\dim \ker(a - b) = n - 1$. By the last fact listed above, we know $|\ker(a - b)| = 2^{n-1}$.

   Thus, exactly half the vectors in $\mathcal{B}_n$ satisfy $aX = bX$ as X. And since the probability distribution is uniform, we have

   $$\Pr_{X \in \mathcal{B}_n}[aX = bX] = (1/2^n) \sum_{X \in \mathcal{B}_n} \mathbb{I}_{\ker(a-b)}[X] = (1/2^n)2^{n-1} = 1/2,$$

   where $\mathbb{I}_{\ker(a-b)}$ is the indicator function for $\ker(a - b)$.

2. **Solution:** Since C, D are distinct, we know that $C - D$ has a nonzero entry, say, at index (i, j). Thus, if we take $X \in \mathcal{B}_n$ to be e_j (the vector with 1 in the jth index and 0 everywhere else), then we would have $(C - D)X \neq (0, \ldots, 0)$ since the ith index of $(C - D)X$ is $\sum_{k=1}^{n}(C -$

[1] As prof. Ellen requested, the reference for the following facts is Michael Artin's Algebra, pages 81 (for fields), 84-91 (for vector spaces and dimension), and 103 (for rank-nullity).

$D)_{i,k}X_k = (C-D)_{i,j}X_j = 1$. From this, we can conclude that the range of $(C-D)$ has dimension at least 1 (a 0 dimension subspace contains only the zero vector, since range$(C-D)$ includes $(C-D)X \neq (0,\ldots,0)$ and is a subspace, we must have dim range$(C-D) \geq 1$). By rank-nullity applied to $(C-D) : \mathcal{B}_n \to \mathcal{B}_n$, we get $n = \dim\ker(C-D) + \dim \text{range}(C-D)$. Thus, we get $\dim\ker(C-D) \leq n-1$, which means (by the last fact listed at the beginning) $|\ker(C-D)| \leq 2^{n-1}$. Hence,

$$\Pr_{X \in \mathcal{B}_n}[CX = DX] = (1/2^n) \sum_{X \in \mathcal{B}_n} \mathbb{I}_{\ker(C-D)}[X] \leq (1/2^n)2^{n-1} = 1/2,$$

where $\mathbb{I}_{\ker(C-D)}$ is the indicator function for $\ker(C-D)$

3. **Solution:** Let RANDOM(a,b)[2] be a random number generator giving a random number between a and b, inclusive, with uniform distribution. We also reference MATRIX-MULTIPLY(A,B)[3], a subprogram that multiplies a $p \times q$ matrix A and a $q \times r$ matrix B in $O(pqr)$ time (the sizes of A and B can be incompatible, in which case the program throws an error). To use this, we modify MATRIX-MULTIPLY slightly: on line 8, instead of $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$, we have $c_{ij} = (c_{ij} + a_{ik} \cdot b_{kj}) \mod 2$. Note that this change does nothing to the running time of MATRIX-MULTIPLY because mod 2 takes constant time.

   Now, we are ready to state our randomized matrix verification algorithm. In the following pseudocode, X_j means the jth entry of vector X.

```
RANDOM-CHECK(P, Q, R)
 1  let X be a new n × 1 column vector
 2  for i = 1 to k
 3      for j = 1 to n
 4          X_j = RANDOM(0, 1)
 5      PQX = MATRIX-MULTIPLY(P, MATRIX-MULTIPLY(Q, X))
 6      RX = MATRIX-MULTIPLY(R, X)
 7      for j = 1 to n
 8          if PQX_j ≠ RX_j
 9              return FALSE
10  return TRUE
```

   To see this algorithm is correct, there are two parts:

   (a) If $PQ = R$, MATRIX-MULTIPLY returns TRUE: this is evident from the fact that MATRIX-MULTIPLY returns true by "default". The only times the algorithm returns FALSE are when $PQX_j \neq RX_j$ on line 8, which only happens when $PQ \neq R$.

   (b) If $PQ \neq R$, MATRIX-MULTIPLY return FALSE with a probability at least $1-2^{-k}$. To show this, it is enough to show that TRUE is returned (i.e line 10 executes) with probability at most 2^{-k}.

   We know that line 10 executes only when for all k iterations of the for loop on lines 2 to 9, $PQX = RX$ (this is what the for loop on lines 7 to 9 is checking). However, we know

[2] CLRS page 117.
[3] CLRS page 371

from the for loop on lines 3 to 4, X is randomly generated with uniform distribution: the probability of X being equal to any specific n-bit vector J is $1/2^n$ since $X_i = J_i$ with probability $1/2$.

From Q2, we know that the probability that an n-bit vector gives equal results when PQ and R are applied is at most $1/2$. Since at the beginning of each iteration (lines 2 to 9), X is generated with uniform distribution, we can conclude $PQX = RX$ with probability at most $1/2$. Hence, we obtain that $PQX = RX$ on all of the k iterations occurs with probability at most $(1/2)^k = 1/2^{-k}$. The algorithm returns TRUE exactly when this happens, so the algorithm returns false with probability at least $1 - 1/2^{-k}$.

Now for the time complexity. This algorithm runs in $O(kn^2)$ (consider integer arithmetic and comparison and random number generation as complexity measure). First, generating vector X takes n steps, one for each entry on lines 3 to 4. Also, checking if $PQX = RX$ takes n steps, one for each entry on lines 7 to 9. In between, we consider the number of steps for performing matrix multiplication. On line 5, we first multiply Q with X, taking $O(n \cdot n \cdot 1) = O(n^2)$ steps, then multiply P with QX, again taking $O(n^2)$ steps[4]. On line 6, we multiply R with x, taking $O(n^2)$ steps. Therefore, each iteration of the main for loop takes $3O(n^2) + 2n = O(n^2)$ steps. Since the loop runs for k iterations, we obtain that RANDOM-CHECK runs in $O(kn^2)$ time.

[4]the time complexity of matrix multiplication is given above in the first paragraph