

This problem set is due *Wednesday, September 9th, at 11:59 PM ET*. It will be graded for completion: for every problem, give as much of a solution as you can. If you get stuck, explain where you got stuck, what you are confused about, and what you tried to get around it.

See the AI policy and the collaboration statement in the syllabus. In short, do not use AI. Feel free to collaborate with others, but your write-up must be your own, and you must disclose your collaborators and any outside sources.

**Problem 1** (Complex vector spaces). Quantum mechanics uses vectors of complex numbers; this requires some getting used to. Let  $a = \begin{bmatrix} 1 \\ e^{i\pi/2} \end{bmatrix}$  and  $b = \begin{bmatrix} e^{i\pi/4} \\ e^{i(3\pi/4)} \end{bmatrix}$ . What is  $a^\dagger b$ ? Here,  $i = \sqrt{-1}$ , and  $a^\dagger$  denotes the *conjugate transpose*, e.g.

$$\begin{bmatrix} x_0 \\ x_1 \end{bmatrix}^\dagger = \begin{bmatrix} \overline{x_0} & \overline{x_1} \end{bmatrix}.$$

Solve this two ways:

- (a) Compute  $a^\dagger b$  using the formula  $e^{it} = \cos(t) + i \sin(t)$ .
- (b) Compute  $a^\dagger b$  *without* using that formula or expanding out the  $e^{it}$  expressions; instead, use the laws of exponents and complex conjugation.

**Problem 2** (Randomness in complex space). Let  $p \geq 2$  be an even integer. Consider rolling two  $p$ -sided dice to get two uniformly random numbers  $A$  and  $B$  between 1 and  $p$ ; then, let  $Z = A \cdot e^{iB\pi/p}$  be a random point on the complex plane, corresponding to those dice rolls.

- (a) Compute the mean,  $E[Z]$ , and the variance,  $E[|Z - E[Z]|^2]$ , of  $Z$ . You may need the identity  $1 + x + x^2 + \dots + x^k = \frac{1-x^{k+1}}{1-x}$ .
- (b) Now, consider:
  - (i) the distribution of  $Z$ , conditioned on the event that  $A > p/2$ .
  - (ii) the distribution of  $Z$ , conditioned on the event that  $B > p/2$ .

Where is the conditional mean relative to  $E[Z]$ ? Did the conditional variance get larger or smaller compared to that of  $Z$ ? Exact formulas are not required, but justify each comparison.

**Problem 3** (Tensor products). Quantum computing uses tensor products everywhere, but they are usually taught very late into a mathematics curriculum, at a high level of abstraction. Here is a crash course.

Consider two vectors of length two,

$$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}; \quad y = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}.$$

Here, the entries of these vectors can be complex or real numbers. Then their tensor product is a vector of length  $2 \times 2 = 4$ , defined to be

$$x \otimes y = \begin{bmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{bmatrix}.$$

If  $x$  and  $y$  represent the states of two different systems, the tensor product of the two typically represents the joint state of the systems. We will see this later, where the systems are qubits.

Tensor products satisfy many of the properties of multiplication: for vectors  $u, v$ , and  $w$ , and a complex number  $c$ ,  $(u+v) \otimes w = u \otimes w + v \otimes w$ , and  $(cu) \otimes w = c(u \otimes w)$ . However, *watch out*, because it is very much not like multiplication in some ways:  $u \otimes v = v \otimes u$  is not always true, and  $u \otimes v + w$  doesn't make sense as an expression if, for example, all the vectors have length 2.

- (a) There are some vectors of length 4 which cannot be written as a tensor product  $x \otimes y$ , for any choice of  $x, y \in \mathbb{C}^2$ . Find one and prove that it satisfies this property.
- (b) Is the following true: any vector  $v$  of length 4 can be written as the sum of tensor product vectors,  $v = x^{(1)} \otimes y^{(1)} + x^{(2)} \otimes y^{(2)} + \dots + x^{(k)} \otimes y^{(k)}$ ? If true, find the smallest  $k$  such that every  $v \in \mathbb{C}^4$  is a sum of at most  $k$  product vectors, and explain why. If false, find such a  $v$  and explain why.
- (c) Show that  $(u \otimes v)^\dagger (x \otimes y) = (u^\dagger x)(v^\dagger y)$ , where  $u, v, x, y \in \mathbb{C}^2$ .

**Problem 4** (Linear algebra algorithms). The most important matrix in quantum computing is the Hadamard matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}.$$

This is the  $2^1 \times 2^1$  Hadamard matrix, also called  $H_1$ , but there are also  $2^n \times 2^n$  Hadamard matrices  $H_n$  which are defined recursively:

$$H_{n+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} +H_n & +H_n \\ +H_n & -H_n \end{bmatrix}.$$

(We will see that, equivalently,  $H_{n+1} = H_1 \otimes H_n = H_1 \otimes H_1 \otimes \dots \otimes H_1$ .)

- (a) Let  $A$  be an arbitrary  $2^n \times 2^n$  matrix, and let  $u$  and  $v$  be arbitrary vectors of length  $2^n$ . How many arithmetic operations (e.g. addition, multiplication, conjugation) are needed to compute  $u^\dagger A v$ ? Show that  $100 \cdot (4^n)$  operations suffice. Argue that this algorithm cannot be significantly improved, i.e. any algorithm for computing  $u^\dagger A v$  for any  $A, u$ , and  $v$  must sometimes use  $\geq 1/100 \cdot (4^n)$  operations.<sup>1</sup>
- (b) Now, suppose  $A = H_n$  and  $u$  and  $v$  have only 5 non-zero entries each. Assume  $u$  and  $v$  are given in sparse form: for each vector, you receive a list containing the index and value of every non-zero entry, e.g.  $u$  has value 1 at index 5, value -1 at index 109, .... Give me an algorithm which computes  $u^\dagger A v$  in a number of operations which is polynomial in  $n$ . This is much faster than the general algorithm!<sup>2</sup>

Hint: What is the formula for an entry of the Hadamard matrix? It will be useful to think of indexing into the Hadamard matrix, not with a number  $j$  from 0 to  $2^n - 1$  (indexing from 0), but with a string of 0/1 bits  $b_1 \dots b_n$  of length  $n$  representing  $j$ 's representation in binary. You will need to use the recursive structure of the Hadamard matrix from above.

<sup>1</sup>In computer science, we would use big O asymptotic notation for this. The above statements together prove a  $\Theta(4^n)$  bound on computing  $u^\dagger A v$ :  $O(4^n)$  operations suffice and  $\Omega(4^n)$  operations are necessary.

<sup>2</sup>If you want an additional challenge: show how to compute  $u^\dagger H_n v$  in  $O(n2^n)$  operations, for arbitrary, not necessarily sparse vectors  $u$  and  $v$ . This is much faster than the naive algorithm of  $O(4^n)$ , and is one of the gems of the theory of algorithms.