CS333 - Programming Assignment: Quantum vs Classical Walks

Motivation
Quantum algorithms are often very different from classical algorithms for similar problems. In this assignment, you'll investigate what happens if you create a quantum version of a classical algorithm.

Guidelines
Please read and abide by the honor code guidelines in the syllabus.

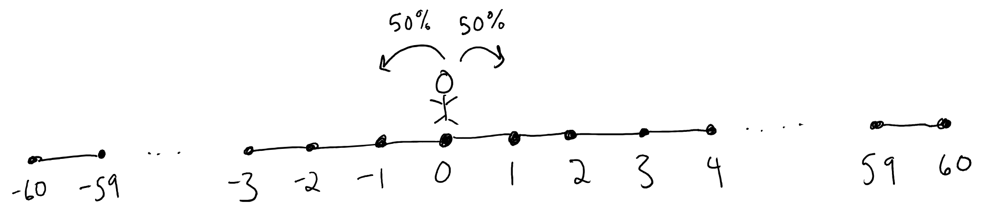
Please read the rubric so you know how you will be graded. For example, turning in a program that compiles and runs without errors but does nothing will earn you more points than a program that is close to working but does not compile or contains errors on running.

You may write this assignment using any classical programming language you would like, but I would suggest using python with the numpy package. (See e.g. `numpy.kron` for tensor product, `numpy.dot` for matrix product, `numpy.conj` for conjugating vectors, etc.)

Please submit your assignment as a zip file. Include in the zip file a .txt file containing the following information:

- Your name
- The name of anyone you worked with and the nature of your collaboration
- The amount of time (approximately) that you spent on this assignment
- The language you wrote the assignment in.
- Your answers to the questions below.

Assignment
Consider a random walker that starts in the middle of a line of points, and with equal probability either moves to the point to its left or the point to its right. If we let this walker walk for a certain time, where it is likely to end up? How does this question differ if the walker is quantum or classical?



For the classical walk, assume the walker starts at position 0 on a line that extends to ± 60, and with 1/2 probability moves to the left and with 1/2 probability moves to the right. (We can imagine that the walker flips a coin at each step, and if the coin is heads, moves to the right, and if tails, to the left.) After k steps, let $Pr(p)$ be the probability that the walker is at position p. Calculate the root mean squared displacement (a measure of how far the walker has moved from the origin):

$$\sqrt{\sum_{p=-60}^{60} Pr(p) \times p^2}. \tag{1}$$

for each value of k from 1 to 50. (We don't want the walker to hit the end of the line, so we stop at 50.)

There are several ways to simulate the classical walk. If you are familiar with it, you can create the probability transition matrix and simulate the walk exactly. Or, you could approximate the probability distribution for each k by letting 1000 individual walkers perform the walk, and then taking the number of walkers at each position divided by 1000 as an estimate of the probability of ending up at that position. Or, you can write the quantum walk first, and then modify so that the state vector instead is a vector containing probabilities and the unitaries are instead transition matrices.

For the quantum walk, we consider two quantum systems, a coin, which is a qubit, and the walker, whose states are the positions on the line: $\{|-60\rangle, |-59\rangle, \ldots, |-1\rangle, |0\rangle, |1\rangle, \ldots, |59\rangle, |60\rangle\}$. The standard basis states of the total system are tensor products of the coin and the walker, and look like $|c\rangle_C |p\rangle_P$ where $c = 0$ or 1, and $p \in \mathbb{Z}$, $-60 \leq p \leq 60$.

The walk consists of two unitaries. The step unitary, U_s, looks at the value of the coin, and if the coin is $|0\rangle$, moves the walker to the right, and if the coin is $|1\rangle$ it moves the walker to the left:

$$\begin{aligned} U_s\left(|0\rangle_C |p\rangle_P\right) &= |0\rangle_C |p+1\rangle_P \\ U_s\left(|1\rangle_C |p\rangle_P\right) &= |1\rangle_C |p-1\rangle_P \end{aligned} \tag{2}$$

The flip unitary U_f, applies a Hadamard to the coin (it is like flipping the coin):

$$U_f\left(|c\rangle_C |p\rangle_P\right) = \left(H|c\rangle_C\right)|p\rangle_P \tag{3}$$

Start with the system initially in the state $|\psi_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)_C |0\rangle_P$. Then apply U_s and U_f in alternation k times. After k steps, measure in the standard basis. Let $Pr(p)$ be the sum of the probabilities of finding the walker at position $|0\rangle_C |p\rangle_P$ and $|1\rangle_C |p\rangle_P$. Calculate the root mean squared displacement (a measure of how far the walker has moved from the origin):

$$\sqrt{\sum_{p=-60}^{60} Pr(p) \times p^2}. \tag{4}$$

Repeat this for all values of k from 1 to 50.

Plot the root mean squared displacement of the quantum and classical walks as a function of number of steps on the same plot, and turn in as part of the zip file you submit. Then answer the following questions in the .txt file:

- Which walker moved faster/farther?
- Why do you think this is? Think about what we discussed in class in terms of how quantum algorithms work.

Extra Credit For extra credit, after implementing the quantum walk in a standard classical programming language, re-implement it in one of the new quantum frameworks or programming languages being developed. You are on your own for learning how to use these languages, as I am not particularly familiar with them. Some options: IBM's qiskit, Microsoft's Quantum Development Kit, Rigetti's Forest. (After dropping your two lowest quiz scores, as per usual, I will replace your lowest quiz score with a higher quiz score, up to 100% for that quiz, depending on your success.)