Fourth homework assignment announced 24. 5. 2024, due 14. 6. 2024 (prior to the exam as a zip file in Brute)

In this homework assignment, you are expected to collect 20 points, but can actually collect up to 60 points, by working on exercises of your own choice out of the list provided below.

Quantum Fourier Transform

We have learned about quantum Fourier transform. There are many beautiful illustrations of the workings of Fourier transforms:

- `https://www.youtube.com/watch?v=jsuvaibdKg4`
  developed using `https://github.com/MathAnimation1198/ManimTutorial`,
- `https://www.youtube.com/watch?v=h7apO7q16V0`,

but very few for the discrete Fourier transform of quantum Fourier transform.

1. Watching the two tutorials linked above and produce an animated introduction to the quantum Fourier transform for some low N. The demonstration needs to be correct, but the "artistic impression" is more of a bonus level. (10 points)

2. Produce a YouTube video based on the animation. The points will be given for the depth of understanding into the connection, as well as the usability of the illustrations in some future version of the lecture notes. Please ping Jakub, Johannes, or Georgios before you release the video publicly, to check for correctness. (10 points)

Discrete Quantum Walks

3. Demonstrate that the directional bias of a Hadamard walker, as in Figure 6.1, depends on the initial coin state. (3 points)

4. Verify that Fig. 6.3 is indeed correct (make your own plot). Explain what would we expect to see if we measured after each iteration of U. (3 points)

Quantum Walk on a Complete Graph

5. The trap, as defined in Fig. 6.5 makes no sense for a quantum walk (although this is not quite the case for a Szegedy walk). Explain why. (2 points)

6. Analyze the convergence properties of iterated applications of the unitary operator $U^n |x\rangle$ that implements a quantum walk. Under what conditions does $U^n|x\rangle$ converge, and how does the unitary property of preserving distances in the Hilbert space play a role in this convergence? (4 points)

7. Derive the probability of success at step 3 and step 4 for the quantum walk on K_4. (4 points)

8. Simulate the quantum walk on K_4 for a large number of steps and for $N \sim 1000$. Show that the analogue of Fig. 6.6 shows an oscillatory behavior. (4 points)

Szegedy Walks

9. Simulate a Szegedy walk on your favorite graph. Compare the validity of your results against `QuantumWalk.jl`. (6 points)

Continuous-time Quantum Walks

10. Prove that Eq. (6.25) holds. Hint: Use a characteristic property about the columns of L. (4 points)

Quantum Walk on the Hypercube

11. Below Eq. (6.33) we read: "Note that $U(\pi/2)$ flips every bit of the state ... the opposite vertex of the hypercube." Pick your favorite $n > 2$ and demonstrate this. (3 points)

Quantum Amplitude Estimation and Monte Carlo Sampling

12. Derive the intermediate steps between Eqs. (6.57) and (6.58). (4 points)

Quantum Adiabatic Computation

13. Consider $H_{\text{clock init}}$ and assume that the initial clock state is other than $|0\rangle^{\otimes L} := |0^L\rangle^c$. Show that for $L = 4$ and for initial clock state $|0010\rangle$ we get a penalty in the energy. (4 points)

14. Prove that the state $|\gamma_0\rangle$ is an eigenstate of H_{init} with zero eigenvalue. (4 points)

Variational Quantum Algorithms

15. Showcase a small instance (e.g., $n \geq 3$, L = 1, 2), where QAOA produces the global optimum. (4 points)

16. Showcase a small instance (e.g., $n \geq 3$, L = 1, 2), where QAOA produces a particularly bad optimum. (4 points)

17. Summarize the performance guarantees of Brownian rounding for MAXCUT, based on `https://arxiv.org/abs/1812.07769`. (4 points)

18. Explain how the performance guarantees of Brownian rounding for MAXCUT can be extended to performance guarantees of warm-started QAOA of Egger et al. (`https://arxiv.org/abs/2009.10095`). (4 points)

19. The variational quantum factoring (https://arxiv.org/abs/1808.08927) suggests the use of QUBO therein, and explains the QUBO. Showcase an example thereof for factoring 15, incl. the optimizer (i.e., values of all of the variables including the carry bits that attain the best possible objective-function-value). (4 points)

20. The Schnorr factoring paper (https://arxiv.org/pdf/2212.12372.pdf) suggests the use of QUBO therein, but does not explain the QUBO instance used. Formulate the QUBO solved in Schnorr factoring (https://arxiv.org/pdf/2212.12372.pdf). (6 points)

Security and Quantum Error Correction

We have seen that the security applications of quantum computers crucially rely on quantum error correction. Here, we provide some exercises based on the Qiskit Quantum Error Correction, cf. https://github.com/qiskit-community/qiskit-qec/blob/main/docs/tutorials/QEC_Framework_IEEE_2022.ipynb.

21. Showcase the stabilizer corresponding to the gauge group from the tutorial linked above:

```
# Create a Gauge Group
from qiskit_qec.structures.gauge import GaugeGroup

matrix = np.array(
    [
        [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0],
        [0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    ]
)

generators = PauliList(matrix)
gauge_group = GaugeGroup(generators)
print(f"G = {gauge_group.generators}")
```

(5 points)

22. Let us consider the following example from the tutorial. Again, compute the stabilizer and explain how the subsystem surface code displayed by code.draw works in your own words.

```
from qiskit_qec.codes.stabsubsystemcodes import StabSubSystemCode
from qiskit_qec.linear.symplectic import make_isotropic_hyperbolic_form

G = GaugeGroup(PauliList(["X1Y3", "X2X3Y4", "Z1Z5"]))
code = StabSubSystemCode(G)
```

```
cen, x, z = make_isotropic_hyperbolic_form(G.generators.matrix)
print(f"G={PauliList(cen) + PauliList(x) + PauliList(z)}")

import  qiskit_qec.codes.codebuilders.subsystem_surface_code_builder as cb
from cb import SubsystemSurfaceCodeBuilder

code = SubsystemSurfaceCodeBuilder(d=5).build()
code.draw(xcolor="lightcoral", zcolor="skyblue")
```

(5 points)