Due: March 19th, 2025
Date: March 28, 2025

HW4 (25 points)

Problem 1. *(Points :6)* For this problem, consider the following algorithm to amplify a quantum algorithm A given as a blackbox. Unfortunately, we do not know the exact success probability p^* of Q, however, you know some p_0 such that $p^* \geq p_0 > 0$. Let the corresponding angles be denoted $\theta^* = \sin^{-1}\sqrt{p^*}$ and $\theta_0 = \sin^{-1}\sqrt{p_0}$. [1]

It uses a bound r which is set as the integer such that $\frac{\pi}{4}\frac{1}{3^{r+1}} < \theta_0 \leq \frac{\pi}{4}\frac{1}{3^r}$.

```
c = 0 // iteration counter
r = $\lfloor \log_3 \frac{\pi}{4\theta_0} \rfloor$
for $i = r \to 1$ :
    Let $t_i = \frac{3^i - 1}{2}$
    Amplify $Q$ using $t_i$ calls to the Grover iterator
    Perform a measurement; return if a good solution is observed
return 'none'
```

(a) **(3 points)** Prove that the number of calls to the iterator is upper bounded by $\frac{9\pi}{16}\frac{1}{\theta_0} \approx \frac{9\pi}{16}\frac{1}{\sqrt{p}}$.

(b) **(2 points)** Let j be the integer such that

$$\frac{\pi}{4}\frac{1}{3^j} < \theta^* \leq \frac{\pi}{4}\frac{1}{3^{j-1}}.$$

Prove that the probability that the *j-th* iteration of the algorithm is unable to find a good solution is at most $\frac{1}{2}$.

(c) **(1 points)** Prove that the probability that the algorithm returns "none" is at most $\frac{1}{2}$.

For the next problem, you may use the following unordered search algorithms as subroutines. Here, N denotes the size of the array and m denotes the number of good solutions it has; m need not be known.

Grover If m is known, the **Grover** algorithm can solve the unordered search problem with probability of error less than m/N using $O(\sqrt{N/m})$ queries to A. If $m = 0$, it retuns "none" with no error.

ExactGrover If m is known, then the **ExactGrover** algorithm can solve the unordered search problem with no error using $O(\sqrt{N/m})$ queries to A.

GroverLV The **GroverLV** algorithm can solve the unordered search problem using expected $O(\sqrt{N/m})$ calls to A. It does not require knowledge of m, but if $m = 0$, the algorithm runs forever.

GroverMC The **GroverMC** algorithm can solve the unordered search problem using $O(\sqrt{N/m})$ calls to A. If $m > 0$, it returns a good solution with probability at least $2/3$, and if $m = 0$, it always returns "none".

Problem 2. *(Points :8)* This is a question on unordered search. You are given query access to a binary array A and you have to find the index of any "1" in the array. If A has no "1", it should return "none".

You will first design two separate algorithms for two different cases.

[1]The key idea is to divide the range of angles $[0, 3\pi/4)$ into disjoint intervals

$$(\alpha_0, \alpha_1], (\alpha_1, \alpha_2], (\alpha_2, \alpha_3], (\alpha_3, \alpha_4], \ldots$$

where, we define the boundaries as $\alpha_i = \frac{\pi}{4}\frac{1}{3^{i-1}}$, and then apply the Grover iterator appropriate number of times for each interval. The right θ^* would be in one such interval, and the algorithm hopes to be correct with high probability for that interval.

(a) **(3 points)** Suppose you are told a number s such that A has at most s solutions. Design a quantum algorithm that finds a good solution with probability 1, using $O(\sqrt{sN})$ queries to A.

(b) **(3 points)** Now, suppose you are told a number s such that A has at least $s+1$ solutions. Design a quantum algorithm that finds a good solution with probability at least $1-2^{-s}$, using $O(\sqrt{sN})$ queries to A.

(c) **(2 points)** Finally, you will design an algorithm for the original problem by combining the two earlier algorithms. Let $\epsilon = 1/2^r$ for some $r \in (1, N)$ representing the desired probability of error. Design a quantum algorithm that makes $O(\sqrt{N \log(1/\epsilon)})$ queries to A [2] and solves the search problem with probability at least $1-\epsilon$, i.e., if there is no solution, it should always return "none", and if A has "1", it may still return "none" with error at most ϵ.

For all the questions, write the algorithm, analyse the number of queries, and analyse the probability of error.

Problem 3. *(Points :1+1+2+3+1+3=11)* Consider the following problem of generating an examination time-table. Suppose that the DOAA office has to schedule T exams (denoted $E^1 \dots E^T$) among T slots (denotes $S_1 \dots S_T$). There is a list of registered students for each exam (say, the students taking exam E^i are denoted $s_1^i s_2^1 \dots$), and no student should be assigned to different exams in the same slot. Furthermore, the office wants to ensure that the overall free-time the students get *between* the exams is as large as possible.

One way to model this problem is to use the following decision variables: X_{pa} which is set to 1 (representing True) if exam p is scheduled in the slot a, and 0 (representing False) otherwise. Define *constants* D_{ab} to be the "free time" available between time-slot a and time-slot b (e.g., suppose a is Monday 9:00-11:00am and b is Monday 3:00-5:00pm; then D_{ab} is 4 hours); naturally, D_{ab} is 0 if b is earlier than a.

To compute the free time of a student, simply add the free times between all pairs of exams of the student. For example, suppose a student has to take 3 exams and all of them are scheduled on Monday: from 9:00-10:00am, 12:00-1:00pm, and 3:00-4:00pm. Then, her free time is 2+2+5=9 hours.

(a) Write down an equation that captures the constraint that each exam must be assigned to exactly slot.

(b) Write down an equation that captures the constraint that each slot must be allotted to exactly one exam.

For the next question, suppose a student is appearing for only one exam. Note that such a student imposes absolutely no restriction on the scheduling of that exam, i.e., scheduling that exam in no slot would violate any constraint related to that student.

Now, consider a student who will take only two exams – E^p and E^q. Here is a generic expression that captures the free-time that student will get between these two exams: $\sum_{a=1}^{T} \sum_{b=1, b\neq a}^{T} D_{ab} X_{pa} X_{qb}$ which simply sums across all possibilities of slots for the two exams.

(c) Write a function in terms of the above variables that represents the sum total of all between-exam free-times across all students. You may find these constants handy: n_{pq} denotes the number of students who are taking both E^p and E^q. You can ignore the constrains in (a) and (b) for constructing the function.

(d) Write an objective function for the exam scheduling optimization problem in the form of a QUBO. Include all relevant constraints.

(e) Write down a Hamiltonian H whose ground state energy equals the optimal value (free-time) of the above problem.

For the above questions, do not make any assumption about the number of exams a student may take, or the duration of the exams.

(f) Solve the above optimization problem for the instance given in the attached datasheet using DWave's QUBO solver. You have to submit a PDF of your notebook for this part that, at the end, clearly prints the **optimal schedule for each of the exams** as well as the **optimal free-time**.

---

[2] Most efficient classical approaches have query complexities of the form $O(\dots \log(1/\epsilon))$.