Intorduction to Quantum Computing: Homework #6

Due on May 29th 2020

Jakub Filipek

Problem 1

Part (a)

Firstly, let us assume that we have 1 ancilla qubit used for the encoding of values x_j. Hence, we can encode negative values, and we can think of a first qubit as a sign qubit fo simplicity. Hence, whenever, a value sinks below 0, we will get $|1\rangle$ in the first register.

Algorithm 1: Find Minimum Element Algorithm

```
Result: Index of the minimal element with probability p = 1/2
Let n = log_2 N
Let O_f |ψ_{0,1,...,n-1}⟩ = |ψ_0⟩
Let mid = -N/2;
for i = 0 to n do
    Let success = -1;
    for j = 0 to k do
        Prepare State Σ_j |j⟩|0⟩
        Apply O_mem. (Σ_j |j⟩|x_j⟩)
        Add mid to the second register (Σ_j |j⟩|x_j + mid⟩)
        Let r be a result of a random Grover search using O_f (i.e. either 0, or 1)
        if r == 1 then
            success = 1;
            break;
        end
    end
    mid = mid + success · N/2^{i+1};
end
```

From the above algorithm we can see that there are $O(k \log N \log N)$ queries to O_{mem}, where first $\log N$ is due to outer loop, and second due to the Random Grover Search which actually goes more like $\log \sqrt{N}$. We have to tune value of k such that the overall probability of returning a correct answer is $\frac{1}{2}$.
Let x be a probability of success correct answer of single iteration of outer for loop. Then:

$$\frac{1}{2} = x^{\log_2 N}$$

$$\log_2 \frac{1}{2} = \log_2 x \log_2 N$$

$$-\frac{1}{\log_2 N} = \log_2 x$$

Since, as mentioned in class random grover search has success probability of $\frac{3}{4}$. Then:

$$x = 1 - (\frac{3}{4})^k$$
$$\log_2 x = \log_2(1 - \frac{3}{4}^k)$$

Combining both equations we get:

$$\log_2(1 - (\frac{3}{4})^k) = -\frac{1}{\log_2 N}$$
$$1 - (\frac{3}{4})^k = 2^{-\frac{1}{\log_2 N}}$$
$$(\frac{3}{4})^k = 1 - 2^{-\frac{1}{\log_2 N}}$$
$$k \log_2(\frac{3}{4}) = \log_2(1 - 2^{-\frac{1}{\log_2 N}})$$
$$k = \frac{\log_2(1 - 2^{-\frac{1}{\log_2 N}})}{\log_2(\frac{3}{4})}$$

We see that $k(N)$ is subpolynomial (as shown on Fig. 1).

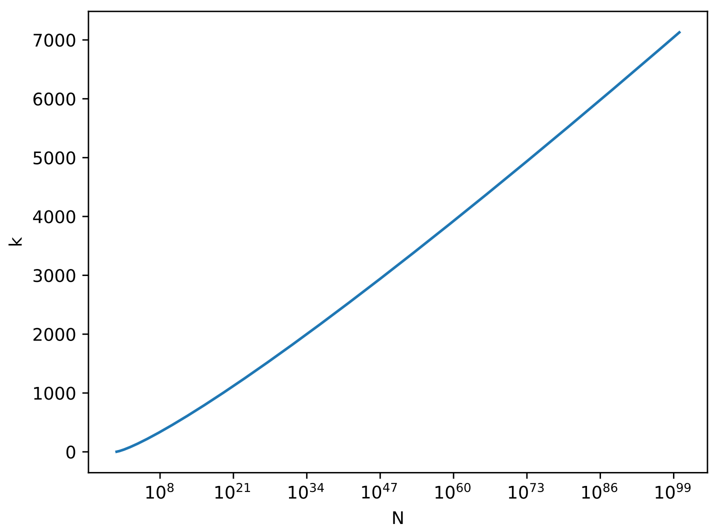


Figure 1: Plot of k with respect to N

Since logarithms raised to arbitrary (constant) power are faster (in asymptotic notation) than polynomials, we get: $\log^N k \in O(\sqrt{N})$.
Hence overall number of queries to $O_{mem} \in O(k \log N \log N) \in O(\sqrt{N} \log N)$.

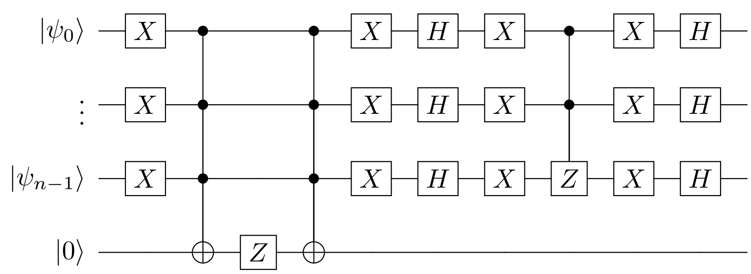
Note: The overall runtime of the algorithm is still actually worse than $O(\sqrt{N} \log N)$, since Random Grover algorithm requires close to $O(\sqrt{N})$ queries to O_f.

Part (b)

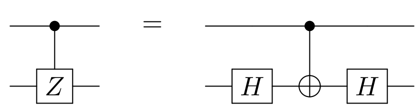
Let us prove that by contradiction. Assume that there exists an algorithm, which can perform a search of smallest element faster than $\Omega(\sqrt{N})$.
Let us imagine that a specific problem is to find the smallest element of an array of all 1's, but a single 0 at index m. This way the lowest value is at the index m. Then, by applying an X gate to every register, we can frame it as a marked item search problem (with 1 at index m). Thus, by finding index m in the original problem, we would be able to find m in the new problem.
However, as mentioned in the class amplitude amplification is an optimal algorithm for the marked item problem and it is $\Omega(\sqrt{N})$. Hence existence of an algorithm that can find the smallest element faster than $\Omega(\sqrt{N})$ is contradictory. Thus, by contradiction, *the number of queries needed to find the smallest element [...] is in* $\Omega(\sqrt{N})$.

Problem 2

Part (a)



Where the multi-controlled gates, can be decomposed into Toffoli/CNOT gates by stacking them into a V shape circuit, or using techinque by Craig Gidney described in question 2 of problem set 2.
Lastly controlled-Z gate, can be decomposed into:

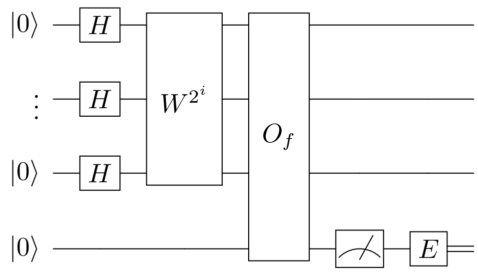


Part (b)

I will use a method described by Figure 2 in Okamoto and Watanabe.
In there the additional qubit is used to determine whether the Grover algorithm found the answer. Hence, the answer is deterministic. To make sure that the average runtime is $O(\sqrt{N})$, we do similar trick to Random Grover Search, where we exponentially increase the number of rotations for iteration.

Part (c)

First let us introduce a classical operation E, which will end the whole algorithm if the input is 1, and continue otherwise.
Let us also use W defined in part a. Let us also have the oracle $O_f|x\rangle|c\rangle = |x\rangle|c \oplus f(x)\rangle$, according to the one defined in the problem (i.e. only for $|0\rangle^{\otimes n}$ f returns 1).
Then for iteration i we have circuit:



And such circuit is repeated for $i = 0, 1, 2, 3, ...$ until E terminates it.

Part (d)

I will refer to the equations 1 and 4 in Okamoto and Watanabe, since they provide a proof for average runtime of the above algorithm to have $\frac{8\pi}{3}\sqrt{\frac{N}{t}}$.
In case of this problem the number of solutions (t) is 1, since only all-zero string returns a correct answer. Hence the problem we have to solve is:

$$\frac{8\pi}{3}\sqrt{2^n} = 2^n - 1$$
$$n = \lceil 6.17 \rceil = 7$$

Hence if we are supposed to use 7 or more qubits the quantum algorithm outperforms the classical (assuming similarities in all other aspects).