CMPE 598 - Lecture Notes

Gözde Berk

April 10, 2018

Assume that we are given a program which can compute a function $f : \{0,1\}^n \to \{0,1\}$. f can be assumed as a complicated boolean formula. We can find the correct assignment of bits after running the program 2^n times. f has only one n-bit input which makes it true. All other $2^n - 1$ inputs make it false. We want to find that particular input. So, worst-case time is equal to 1 run time x 2^n. On average, it is 1 run time x $2^{\frac{n}{2}}$. *Grover's algorithm* to be introduced below runs in $2^{\frac{n}{2}}$ times. Therefore, it provides a speed up.

1 Grover's algorithm

If somebody gives a classical problem, we can write it in terms of toffoli gates. We will build a quantum circuit which consists of the big combination of gates such that a big quantum transformation is going to be handled. We will use $n + 1 + m$ qubits, where m is large enough so we can compute the transformation

$|x\sigma 0^m\rangle \to |x(\sigma \oplus f(x))0^m\rangle$

x is n-bit, σ is 1-bit and $\sigma \oplus f(x)$ is 1-bit. In the end, n bits which are in their original values, 1 bit for the result and m additional bits in their original values are given as output.

The algorithm starts with initializing everything to 0 and then applying Hadamard operation to first n bits.

$$H = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Firstly, we have n bits with equal probability. (like a uniform number generator)

1.1 Pseudocode of the Grover's algorithm

Initialize everything to 0;
Apply Hadamard operation to the first n bits;
for $i = 1...2^{\frac{n}{2}}$ **do**
  Step 1;
  1.1 Compute $|x\sigma 0^m\rangle \to |x(\sigma \oplus f(x))0^m\rangle$;
  1.2 if *the* $n+1$*st qubit is 1* **then**
    Multiply vector by -1;
  else
    Do nothing;
  end
  1.3 Compute $|x\sigma 0^m\rangle \to |x(\sigma \oplus f(x))0^m\rangle$;
  Step 2;
  2.1 Apply Hadamard to the first n qubits;
  2.2;
  2.2.1 if *the first n qubits are all zero* **then**
    Flip the $n+1$st qubit;
  end
  2.2.2 if *the* $n+1$*st qubit is 1* **then**
    Multiply by -1;
  end
  2.2.3 if *the first n qubits are not all zero* **then**
    Flip the $n+1$st qubit;
  end
  2.3 Apply Hadamard to the first n qubits;
end
Measure the first n qubits, check if the value "a" you read makes $f(a) = 1$

Algorithm 1: Grover's algorithm

1.2 Analysis of the algorithm

Let $u = \frac{1}{2^{\frac{n}{2}}} \sum_{x \epsilon \{0,1\}} |x\rangle$, after applying Hadamard to first n.

Let "a" be the special input that we are looking for.

First n bits can be represented by 2^n dimensions. a corresponds to 1 dimension. It can be represented in 2 dimensions with $|a\rangle$ axis and $|e\rangle$ axis. In $|a\rangle$ axis, seeing one of the $2^n - 1$ inputs other than a is 0. In $|e\rangle$ axis, seeing a is 0. $|e\rangle$ is equally away from all $2^n - 1$ vectors. e corresponds to the equal superposition of all vectors excluding a. $|e\rangle = \sum_{x \neq a} |x\rangle$

Step 1, reflects around $|e\rangle$ as in Figure 1.

Before step 1, $u = \frac{1}{2^{\frac{n}{2}}} |000...0\rangle + \frac{1}{2^{\frac{n}{2}}} |000...1\rangle + ... + \frac{1}{2^{\frac{n}{2}}} |a\rangle + ... + \frac{1}{2^{\frac{n}{2}}} |111...1\rangle$

After step 1, it becomes $\frac{1}{2^{\frac{n}{2}}} |000...0\rangle + \frac{1}{2^{\frac{n}{2}}} |000...1\rangle + ... + \frac{-1}{2^{\frac{n}{2}}} |a\rangle + ... + \frac{1}{2^{\frac{n}{2}}} |111...1\rangle$

The amplitude of $|a\rangle$ becomes $-$ but the probability remains the same.

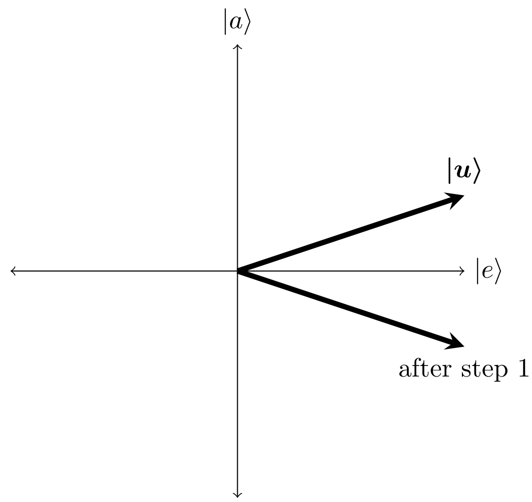

Figure 1: Step 1 - Reflecting around $|e\rangle$

Step 2, reflects around $|u\rangle$ as in Figure 2.

The first Hadamard operation rotates the coordinate axis and the second one goes back to the previous coordinate axis to handle reflection around $|u\rangle$.

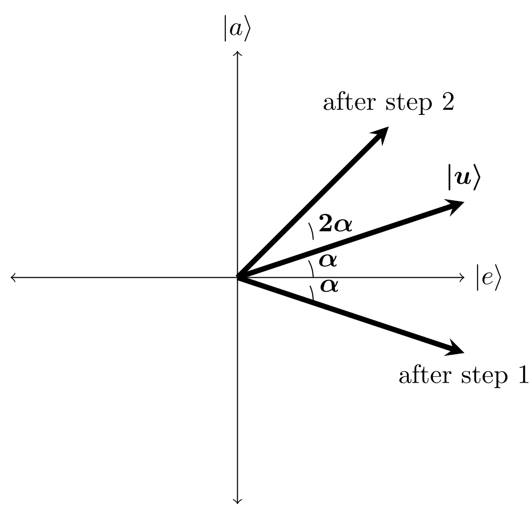

Figure 2: Step 2 - Reflecting around $|u\rangle$

1.3 Angle Analysis

We know α since the projection of $|u\rangle$ on $|a\rangle$ has length $\frac{1}{2^{\frac{n}{2}}}$. If n is big, $\frac{1}{2^{\frac{n}{2}}}$ is small. For small angles, $sin\alpha \simeq \alpha$.

$\frac{\pi}{2} - arcsin(\frac{1}{2^{\frac{n}{2}}})$ is the distance to cover

$2arcsin(\frac{1}{2^{\frac{n}{2}}})$ is the distance covered in each iteration

The moves for each step can be seen in Figure 3. (2α in each iteration)

Since $\alpha \geq sin\alpha$ for $\alpha > 0$, $\alpha = \frac{1}{2^{\frac{n}{2}}}$.

As a result, the number of iterations required is approximately $2^{\frac{n}{2}}$ for large values of n.

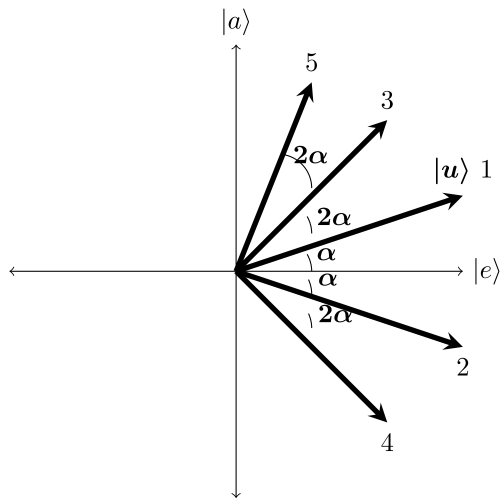

Figure 3: Angle analysis

2 Shor's algorithm for factorization

Given a positive integer, find its factors.

There exists a fast classical algorithm for detecting whether the number is prime. If so, problem is solved.

There exists a fast classical algorithm for detecting whether the number is a power. (i.e. of the form a^b for $b > 1$)

If you can find a factor, you can find all other factors as well using the same method repeatedly.

Shor's algorithm is a quantum algorithm to find prime factors of an integer. In other words, it is for integer factorization.