

Choosing the number of iterations

We have established that the state vector of the register \mathbf{Q} in Grover's algorithm remains in the two-dimensional subspace spanned by $|A_0\rangle$ and $|A_1\rangle$ once the initialization step has been performed.

The goal is to find an element $x \in A_1$, and this goal will be accomplished if we can obtain the state $|A_1\rangle$ — for if we measure this state, we're guaranteed to get a measurement outcome $x \in A_1$. Given that the state of \mathbf{Q} after t iterations in step 2 is

$$G^t|u\rangle = \cos((2t+1)\theta)|A_0\rangle + \sin((2t+1)\theta)|A_1\rangle,$$

we should choose t so that

$$\langle A_1|G^t|u\rangle = \sin((2t+1)\theta)$$

is as close to 1 as possible in absolute value, to maximize the probability to obtain $x \in A_1$ from the measurement. For any angle $\theta \in (0, 2\pi)$, the value $\sin((2t+1)\theta)$ *oscillates* as t increases, though it is not necessarily periodic — there's no guarantee that we'll ever get the same value twice.

Naturally, in addition to making the probability of obtaining an element $x \in A_1$ from the measurement large, we would also like to choose t to be as small as possible, because t applications of the operation G requires t queries to the function f . Because we're aiming to make $\sin((2t+1)\theta)$ close to 1 in absolute value, a natural way to do this is to choose t so that

$$(2t+1)\theta \approx \frac{\pi}{2}.$$

Solving for t yields

$$t \approx \frac{\pi}{4\theta} - \frac{1}{2}.$$

Of course, t must be an integer, so we won't necessarily be able to hit this value exactly — but what we can do is to take the closest integer to this value, which is

$$t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor.$$

This is the recommended number of iterations for Grover's algorithm. As we proceed with the analysis, we'll see that the closeness of this integer to the target value naturally affects the performance of the algorithm.

(As an aside, if the target value $\pi/(4\theta) - 1/2$ happens to be exactly half-way between two integers, this expression of t is what we get by rounding up. We could alternatively round down, which makes sense to do because it means one fewer query — but this is secondary and unimportant for the sake of the lesson.)

Recalling that the value of the angle θ is given by the formula

$$\theta = \sin^{-1} \left(\sqrt{\frac{|A_1|}{N}} \right),$$

we see that the recommended number of iterations t depends on the number of strings in A_1 . This presents a challenge if we don't know how many solutions we have, as we'll discuss later.

Unique search

First, let's focus on the situation in which there's a single string x such that $f(x) = 1$. Another way to say this is that we're considering an instance of the *Unique search* problem. In this case we have

$$\theta = \sin^{-1} \left(\sqrt{\frac{1}{N}} \right),$$

which can be conveniently approximated as

$$\theta = \sin^{-1} \left(\sqrt{\frac{1}{N}} \right) \approx \sqrt{\frac{1}{N}}$$

when N gets large. If we substitute $\theta = 1/\sqrt{N}$ into the expression

$$t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor$$

we obtain

$$t = \left\lfloor \frac{\pi}{4} \sqrt{N} \right\rfloor.$$

Recalling that t is not only the number of times the operation G is performed, but also the number of queries to the function f required by the algorithm, we see that we're on track to obtaining an algorithm that requires $O(\sqrt{N})$ queries.

Now we'll investigate how well the recommended choice of t works. The probability that the final measurement results in the unique solution can be expressed explicitly as

$$p(N, 1) = \sin^2((2t + 1)\theta).$$

The first argument, N , refers to the number of items we're searching over, and the second argument, which is 1 in this case, refers to the number of solutions. A bit later we'll use the same notation more generally, where there are multiple solutions.

Here's a table of the probabilities of success for increasing values of $N = 2^n$.

N	$p(N, 1)$
2	0.5000000000
4	1.0000000000
8	0.9453125000
16	0.9613189697
32	0.9991823155
64	0.9965856808
128	0.9956198657
256	0.9999470421
512	0.9994480262
1024	0.9994612447
2048	0.9999968478
4096	0.9999453461
8192	0.9999157752
16384	0.9999997811
32768	0.9999868295
65536	0.9999882596

Notice that these probabilities are not strictly increasing. In particular, we have an interesting anomaly when $N = 4$, where we get a solution with certainty. It can, however, be proved in general that

$$p(N, 1) \geq 1 - \frac{1}{N}$$

for all N , so the probability of success goes to 1 in the limit as N becomes large, as the values above seem to suggest. This is good!

But notice, however, that even a weak bound such as $p(N, 1) \geq 1/2$ establishes the utility of Grover's algorithm. For whatever measurement outcome x we obtain from running the procedure, we can always check to see if $f(x) = 1$ using a single query to f . And if we fail to obtain the unique string x for which $f(x) = 1$ with probability at most $1/2$ by running the procedure once, then after m independent runs of the procedure we will have failed to obtain this unique string x with probability at most 2^{-m} . That is, using $O(m\sqrt{N})$ queries to f , we'll obtain the unique solution x with probability at least $1 - 2^{-m}$. Using the better bound $p(N, 1) \geq 1 - 1/N$ reveals that the probability to find $x \in A_1$ using this method is actually at least $1 - N^{-m}$.

Multiple solutions

As the number of elements in A_1 varies, so too does the angle θ , which can have a significant effect on the algorithm's probability of success. For the sake of brevity, let's write $s = |A_1|$ to denote the number of solutions, and as before we'll assume that $s \geq 1$.

As a motivating example, let's imagine that we have $s = 4$ solutions rather than a single solution, as we considered above. This means that

$$\theta = \sin^{-1}\left(\sqrt{\frac{4}{N}}\right),$$

which is approximately double the angle we had in the $|A_1| = 1$ case when N is large. Suppose that we didn't know any better, and selected the same value of t as in the unique solution setting:

$$t = \left\lfloor \frac{\pi}{4 \sin^{-1}(1/\sqrt{N})} \right\rfloor.$$

The effect will be catastrophic as the following table of probabilities reveals.

<i>N</i>	Success probability
4	1.0000000000
8	0.5000000000
16	0.2500000000
32	0.0122070313
64	0.0203807689
128	0.0144530758
256	0.0000705058
512	0.0019310741
1024	0.0023009083
2048	0.0000077506
4096	0.0002301502
8192	0.0003439882
16384	0.0000007053
32768	0.0000533810
65536	0.0000472907

This time the probability of success goes to 0 as N goes to infinity. This happens because we're effectively rotating twice as fast as we did when there was a unique solution, so we end up zooming past the target $|A_1\rangle$ and landing near $-|A_0\rangle$.

However, if instead we use the recommended choice of t , which is

$$t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor$$

for

$$\theta = \sin^{-1}\left(\sqrt{\frac{s}{N}}\right),$$

then the performance will be better. To be more precise, using this choice of t leads to success with high probability.

N	$p(N, 4)$
4	1.0000000000
8	0.5000000000
16	1.0000000000
32	0.9453125000
64	0.9613189697
128	0.9991823155
256	0.9965856808
512	0.9956198657
1024	0.9999470421
2048	0.9994480262
4096	0.9994612447
8192	0.9999968478
16384	0.9999453461
32768	0.9999157752
65536	0.999997811

Generalizing what was claimed earlier, it can be proved that

$$p(N, s) \geq 1 - \frac{s}{N},$$

where we're using the notation suggested earlier: $p(N, s)$ denotes the probability that Grover's algorithm run for t iterations reveals a solution when there are s solutions in total out of N possibilities.

This lower bound of $1 - s/N$ on the probability of success is slightly peculiar in that more solutions implies a worse lower bound — but under the assumption that s is significantly smaller than N , we nevertheless conclude that the probability of success is reasonably high. As before, the mere fact that $p(N, s)$ is reasonably large implies the algorithm's usefulness.

It also happens to be the case that

$$p(N, s) \geq \frac{s}{N}.$$

This lower bound describes the probability that a string $x \in \Sigma^n$ selected uniformly at random is a solution — so Grover's algorithm always does at least as well as random guessing. (In fact, when $t = 0$, Grover's algorithm *is* random guessing.)

Now let's take a look at the number of iterations (and hence the number of queries)

$$t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor,$$

for

$$\theta = \sin^{-1} \left(\sqrt{\frac{s}{N}} \right).$$

For every $\alpha \in [0, 1]$, it is the case that $\sin^{-1}(\alpha) \geq \alpha$, and so

$$\theta = \sin^{-1} \left(\sqrt{\frac{s}{N}} \right) \geq \sqrt{\frac{s}{N}}.$$

This implies that

$$t \leq \frac{\pi}{4\theta} \leq \frac{\pi}{4} \sqrt{\frac{N}{s}}.$$

This translates to a savings in the number of queries as s grows. In particular, the number of queries required is

$$O \left(\sqrt{\frac{N}{s}} \right).$$

Unknown number of solutions

If the number of solutions $s = |A_1|$ is *unknown*, then a different approach is required, for in this situation we have no knowledge of s to inform our choice of t . There are, in fact, multiple approaches.

One simple approach is to choose

$$t \in \left\{ 1, \dots, \lfloor \pi\sqrt{N}/4 \rfloor \right\}$$

uniformly at random. Selecting t in this way always finds a solution (assuming one exists) with probability greater than 40%, though this is not obvious and requires an analysis that will not be included here. It does make sense, however, particularly when we think about the geometric picture: rotating the state of \mathbf{Q} a random number of times like this is not unlike choosing a random unit vector in the space spanned by $|A_0\rangle$ and $|A_1\rangle$, for which it is likely that the coefficient of $|A_1\rangle$ is reasonably large. By repeating this procedure and checking the outcome in the same way as described before, the probability to find a solution can be made very close to 1.

There is a refined method that finds a solution when one exists using $O(\sqrt{N/s})$ queries, even when the number of solutions s is not known, and requires $O(\sqrt{N})$ queries to determine that there are no solutions when $s = 0$.

The basic idea is to choose t uniformly at random from the set $\{1, \dots, T\}$ iteratively, for increasing values of T . In particular, we can start with $T = 1$ and increase it exponentially, always terminating the process as soon as a solution is found and capping T so as not to waste queries when there isn't a solution. The process takes advantage of the fact that fewer queries are required when more solutions exist. Some care is required, however, to balance the rate of growth of T with the probability of success for each iteration. (Taking $T \leftarrow \lceil \frac{5}{4}T \rceil$ works, for instance, as an analysis reveals. Doubling T , however, does not — this turns out to be too fast of an increase.)

The trivial cases

Throughout the analysis we've just gone through, we've assumed that the number of solutions is non-zero. Indeed, by referring to the vectors

$$\begin{aligned}|A_0\rangle &= \frac{1}{\sqrt{|A_0|}} \sum_{x \in A_0} |x\rangle \\ |A_1\rangle &= \frac{1}{\sqrt{|A_1|}} \sum_{x \in A_1} |x\rangle\end{aligned}$$

we have implicitly assumed that A_0 and A_1 are both nonempty. Here we will briefly consider what happens when one of these sets is empty.

Before we bother with an analysis, let's observe the obvious: if every string $x \in \Sigma^n$ is a solution, then we'll see a solution when we measure; and when there aren't any solutions, we won't see one. In some sense there's no need to go deeper than this.

We can, however, quickly verify the mathematics for these trivial cases. The situation where one of A_0 and A_1 is empty happens when f is constant; A_1 is empty when $f(x) = 0$ for every $x \in \Sigma^n$, and A_0 is empty when $f(x) = 1$ for every $x \in \Sigma^n$. This means that

$$Z_f|u\rangle = \pm|u\rangle,$$

and therefore

$$\begin{aligned} G|u\rangle &= (2|u\rangle\langle u| - \mathbb{I})Z_f|u\rangle \\ &= \pm(2|u\rangle\langle u| - \mathbb{I})|u\rangle \\ &= \pm|u\rangle. \end{aligned}$$

So, irrespective of the number of iterations t we perform in these cases, the measurements will always reveal a uniform random string $x \in \Sigma^n$.

Was this page helpful?

Yes



No



Report a bug, typo, or request content on GitHub ↗.

Previous page

Next page