HOW CAN WE EXPLAIN GROVER’S QUANTUM ALGORITHM?

Grover’s quantum algorithm: the task. In many practical situations, we need to look for an element with the desired property in an unsorted list of n elements.

In the traditional (non-quantum) computing, to find the desired element, we need, in some cases, to check all n elements – because if we miss even one element, we may thus miss the desired element. So, in some cases, we need to make at least n computational steps to find the desired element.

The quantum algorithm proposed by Lov Grover enables us to find the desired element much faster – in time proportional to square root of n.

Formulation of the problem. Grover’s algorithm works – but it is very technical. It is desirable to come up with a natural explanation of this algorithm, so that we can explain, based on the task, why we need to perform the corresponding steps.

As of now, the only available explanation is that this algorithm works, but it would be nice to have a more natural explanation.

Quantum computing: in brief. In quantum physics, in addition to traditional (non-quantum) states s (which are denoted by $|s\rangle$), we can also have superpositions of these states, i.e., states of the type $c_s|s\rangle + c_s'|s'\rangle + \ldots$, where $c_s$, … are complex-valued coefficients for which $|c_s|^2 + |c_s'|^2 + \ldots = 1$. (In quantum computing, usually, only real-valued coefficients are used.)

Each value $|c_s|^2$ represents the probability that we get s when we measure the superposition state. These probabilities should add up to 1, which explains the above condition on the coefficients $c_s$.

Each such state can be represented by a vector $(c_s, c_s', \ldots)$. All changes are described by linear transformations that transform orthogonal vectors into orthogonal ones – this requirement is motivated by the need to preserve the property that the sum of probabilities is equal to 1.

Grover’s algorithm: in brief. In the non-quantum case, to find the desired element in a list of n elements, we input the index $i = 1, 2, \ldots, n$, and we check whether the i-th element has the desired property.

In quantum physics, according to the above general description, in addition to integer values of i, we can also have superpositions:
\[ c_1 |1> + c_2 |2> + \ldots + c_n |n> \]

The quantum analog of checking is the following transformation:

\[ (c_1, \ldots, c_{d-1}, c_d, c_{d+1}, \ldots, c_n) \rightarrow (c_1, \ldots, c_{d-1}, -c_d, c_{d+1}, \ldots, c_n), \]

where \( d \) is the index of the element that has the desired property.

Grover’s algorithm starts with the state \( c \cdot (1, 1, \ldots, 1) \) for an appropriate value \( c \), and then interchangingly applies the above checking transformation and a special transformation known as Grover’s transformation.

Questions: Why should we select this particular starting state? Why should we use Grover’s transformation and not some other formula?

In this paper, we provide answers to these two questions.

Our explanation. The choice of the starting state is the easiest to explain. A priori, we have no reason to assume that some elements of the list are more probable to have the desired property and some are less probable – we have the same ignorance about all the elements. In mathematical terms, our knowledge does not change if we perform an arbitrary permutation of the indices. It is therefore reasonable to require that the starting state should also be invariant with respect to such permutations. Since can always swap \( i \)-th and \( j \)-th elements, this implies that the coefficients at every two elements must be equal – which explains the starting state of Grover’s algorithm.

What about Grover’s transformation? In general, this transformation must be linear: \( c_i \rightarrow a_{ii} \cdot c_i + a_{ij} \cdot c_j + \ldots \) for some coefficients \( a_{ij} \). This transformation should not change if we apply any permutation. If we swap two indices \( i \) and \( j \), we change \( a_{ii} \) to \( a_{jj} \), so we should have \( a_{ii} = a_{jj} \) for all \( i \) and \( j \). Let us denote the common value of these diagonal terms by \( d \).

Every two pairs \((i,j)\) of different indices can also be transformed into each other by some permutation, so all non-diagonal terms should also be equal to each other; let us denote the common value by \( a \).

The states \((1,0,0,\ldots)\) and \((0,1,0,\ldots)\) are orthogonal, so the states \((d,a,a,\ldots)\) and \((a,d,a,\ldots)\) obtained after this transformation should also be orthogonal, i.e., their dot product should be 0: \( 2a \cdot d + (n - 2) \cdot a^2 = 0 \). This is equivalent to \( a \cdot (2d + (n - 2) \cdot a) = 0 \).

If \( a = 0 \), then the state practically does not change, so for a non-trivial transformation, we have \( d = -(n - 2) \cdot (a / 2) \). Substituting this value into the probability-1 requirement \( |d|^2 + |a|^2 + |a|^2 + \ldots = 1 \), we get exactly Grover’s transformation (plus the same transformation followed by multiplication by \(-1\), which, in quantum physics, does not change the state).

Conclusion. Thus, Grover’s algorithm can indeed be explained – by natural invariance with respect to all possible permutations.