Quantum Algorithms 3/2/2023

Lecture 12: Quantum Walks

Instructor: Dieter van Melkebeek

Following amplitude amplification, we discuss another quantum paradigm, quantum walks. A quantum walk is the quantum version of classical random walk, which has found a lot of algorithmic applications. We begin by discussing classical random walks and study concepts fundamental to their operation, such as Markov chains. Then we develop their quantum counterpart, quantum walks, and discuss how they try to mimic random walks in a way that takes advantage of their quantum features for speedup. This lecture will be more of a survey of the general concepts and an application of quantum walks, with detailed analysis left for next lecture.

1 Solution to Exercise #9

Recall the statement: Consider the algorithm for oblivious amplitude amplification but evaluate the success indicator each time the frequency domain is reached, and only continue in case of no success.

(a) Determine the probability of no success within the first k iterations as a function of p.

(b) Determine the expected number of iterations until the first success as a function of p.

We can interpret the modified algorithm in the two-domain view as in Figure 1, where we consider the situation of failure when we evaluate the success indicator the first time we switch to the frequency domain.

Let us go through Figure 1 step by step. We start from state $|0^\ell\rangle|\phi\rangle$. The state after the first switch to the frequency domain is $A|0^\ell\rangle|\phi\rangle$. The probability of failure upon evaluating the success indicator is $1-p$. In this case the state becomes $|B(\phi)\rangle$. Now the reflection R_{bad} is trivial so we switch back to the time domain. Now we reflect over $|0^\ell *\rangle$ in the time domain and switch back to the frequency domain. At this point, unlike the initial state, the probability of failure when evaluating the success indicator is $\cos^2(2\theta_0)$. Recalling that $\sin(\theta_0) = \sqrt{p}$ and applying the double angle formula $\cos(2\theta) = 1 - 2\sin^2(\theta)$, we can reduce this to $(1-2p)^2$.

Now we can answer the questions in the exercise.

(a) The probability that we fail initially, that is before the first iteration, was $1-p$. Then after each subsequent iteration, the probability of failure is $(1-2p)^2$. Thus the probability of no successes after k iterations is $(1-p)(1-2p)^{2k}$.

(b) The probability of initial success is p. In particular, this means that if $p = 1$ the expected number of iterations is 0. If $p < 1$, then we are running a Bernoulli experiment by repeating the process until the first success. For any trial the probability of success is $q = 1-(1-2p)^2 = 4p(1-p)$. In general, the expected number of trials until the first success for a Bernoulli experiment with probability of success q is $1/q$. So, assuming we don't have an initial success, we get an expected number of trials of $p \cdot 0 + \frac{1-p}{q} = \frac{1}{4p}$. Note that in the case where $p = 1$, this formula would give us $\frac{1}{4}$ which is incorrect. This is because the first measurement is different and must be treated separately.

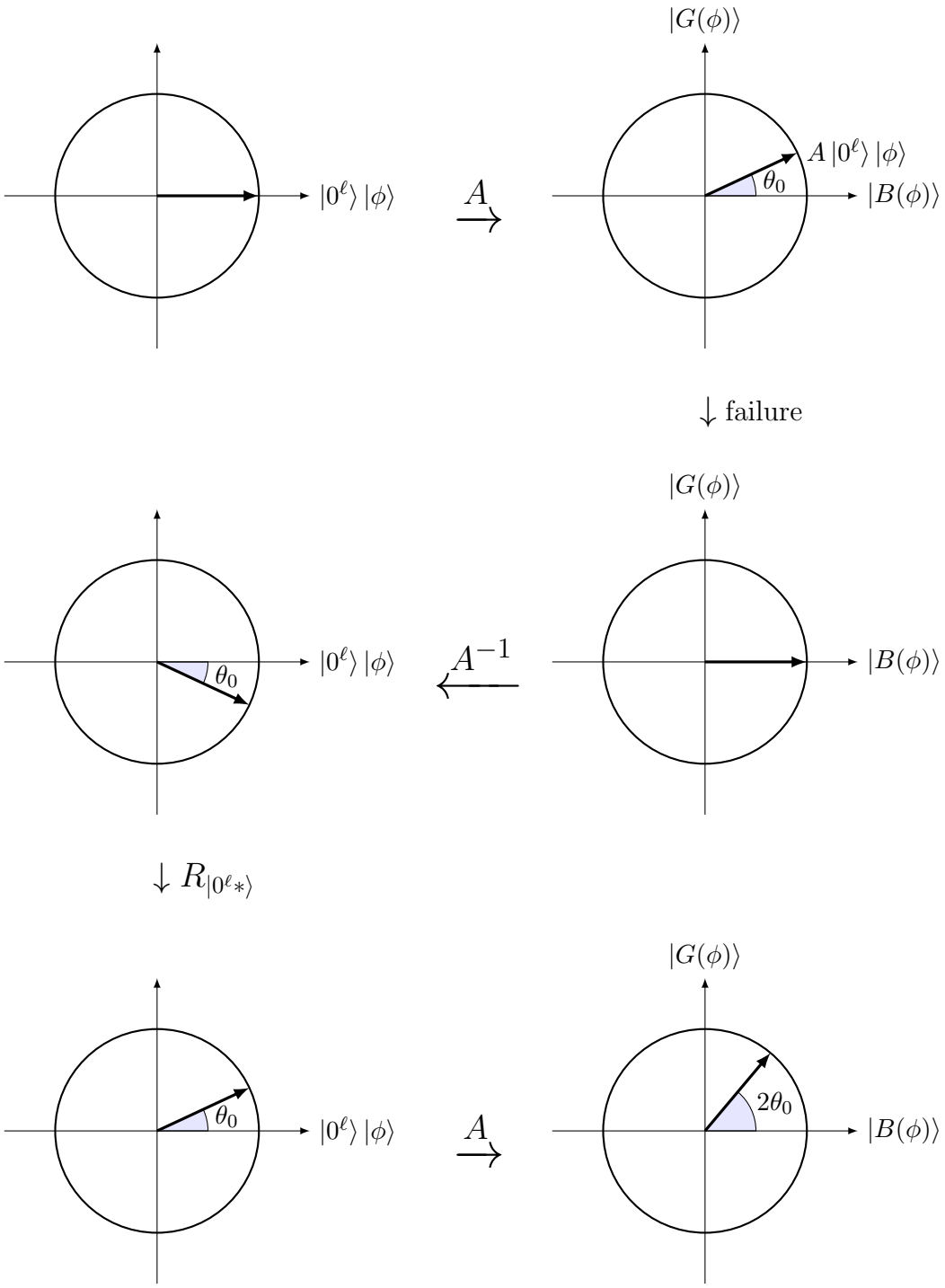


Figure 1: Exercise #9 in the two-domain view

2 Random Walk

Setup. A *graph* is a set of points called *vertices* connected by lines called *edges*. Each edge (u, v) has a *weight* w, which is proportional to the probability of moving to vertex v when currently at vertex u. We only consider finite undirected graphs $G = (V, E)$ where V denotes the set of $N \doteq |V|$ vertices, and E denotes the set of edges.

We can represent the weighted graph as a symmetric function $w : V \times V \to [0, \infty)$, where each possible edge that can exist between two vertices gets mapped to a weight ranging from 0 to infinity. The edges that are in $V \times V$ but not in E have weights equal to zero. The undirected nature of G is reflected in the symmetry of w with respect to the two vertices that it connects: $w(u, v) = w(v, u)$.

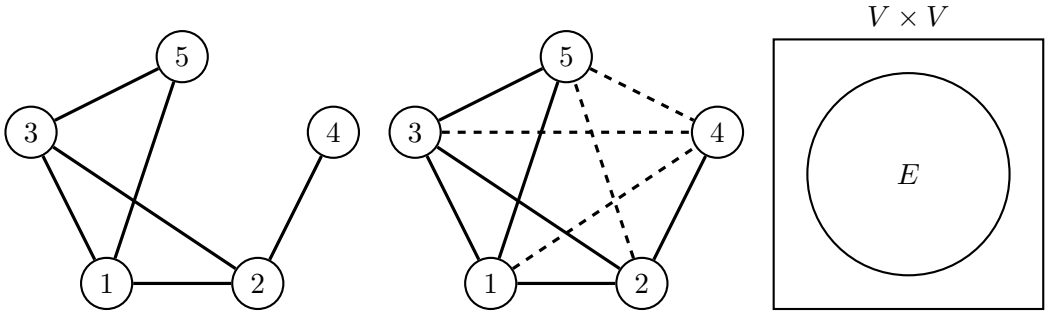


Figure 2: The leftmost graph shows the actual edge set E, the next graph shows the set $V \times V$ where the dotted lines indicate the edges not in E. The rightmost is a Venn-diagram of the edge set.

The weight of a vertex $u \in V$ is defined as the sum of the weights of all incident edges on u:

$$w(u) \doteq \sum_{v \in V} w(u, v)$$

We assume that none of the vertices in G is isolated. In the unweighed setting, this means that every vertex is connected to at least one other vertex. In general, this means that $w(u) > 0$ for each $u \in V$.

Process. A *random walk* is a process that starts by picking a starting vertex from an initial probability distribution σ over V. In each step it moves from the current vertex $u \in V$ to a neighboring vertex $v \in V$ with a probability proportional to $w(u, v)$. In the special case where the graph is unweighted, the subsequent vertex that the walk moves to is picked uniformly at random among the neighbors of the current vertex. This process is repeated for a number of steps. We are interested in how the probability distribution evolves as the number of steps increases. This process can be modeled with Markov chains, which we cover next.

3 Markov chain

One way to model and analyze the behavior of a random walk is by viewing it as a *Markov chain*. This is a memoryless stochastic process, where the probability distribution for the next step only

depends on the current state, and retains no history of prior steps (other than the current state). In the case of random walks on graphs the state is captured by the vertex.

Transition matrix. The probability distribution obtained from a Markov chain can be described by a *transition matrix* T, which is an $N \times N$ matrix with components defined as follows:

$$T_{vu} \doteq \Pr[\text{move to } v \mid \text{at } u] = \frac{w(u,v)}{w(u)}$$

The transition matrix T can be used to describe the probability distribution after t steps of the random walk as $T^t\sigma$, i.e., as the product of the t-th power of T with a column vector representing the initial probability distribution σ. The column vector has one component for every vertex v, equal to the probability of being at v. The transition matrix T is symmetric ($T_{uv} = T_{vu}$ for all $u, v \in V$) iff $w(u) = w(v)$ for all $u, v \in V$, i.e., iff the weight $w(u)$ of every vertex u is the same. For unweighted graphs, T is symmetric iff the degree of every vertex is the same, i.e., the graph G is regular.

Stationary distribution. If a Markov chain converges to some distribution, that distribution needs to be stationary, i.e., invariant under T. There may be multiple stationary distributions, but the following distribution π always is one:

$$\pi(v) \doteq \frac{w(v)}{W} \quad \text{where} \quad W \doteq \sum_v w(v)$$

Proof. When the current distribution is π, the weight that is sent from u to v in one step of the random walk equals $T_{vu} \cdot \pi(u)$. Rewriting this expression, we get the following:

$$T_{vu} \cdot \pi(u) = \frac{w(u,v)}{w(u)} \cdot \frac{w(u)}{W} = \frac{w(u,v)}{W}$$

Since w is symmetric in u and v (i.e., $w(u,v) = w(v,u)$), the weight sent from u to v equals the weight sent from v to u, which gives us the following result:

$$\frac{w(u,v)}{W} = \frac{w(v,u)}{W} = T_{uv} \cdot \pi(u) \qquad \square$$

Note that π is uniform iff $w(u)$ is independent of u iff T is symmetric. In particular, in the unweighted case, π is uniform iff the graph G is regular.

As a side note, a Markov chain that has a stationary distribution σ with the property that $T_{vu} \cdot \sigma(u) = T_{vu} \cdot \sigma(v)$, is called reversible; random walks on undirected graphs are equivalent to reversible Markov chains.

4 Convergence analysis

We restrict to the important case where T is symmetric; if that is the case, we can do a spectral analysis of T to analyze the convergence of the Markov chain to a stationary distribution π. In the general case, one uses the discriminant matrix D defined as:

$$D_{uv} \doteq \sqrt{T_{vu} \cdot T_{uv}}$$

Note that D always is symmetric, and that $D = T$ iff T is symmetric.

Spectrum of T. Since T is a symmetric real matrix, all its eigenvalues are real. Moreover, the eigenvalues have absolute values at most 1.

Proof. Since T is stochastic, for any vector x

$$\|Tx\|_1 \doteq \sum_v |\sum_u T_{vu} x_u| \leq \sum_u |x_u| \sum_v T_{vu} = \sum_u |x_u| \doteq \|x\|_1 \tag{1}$$

If λ is an eigenvalue of T with corresponding eigenvector x, then $|\lambda| \|x\|_1 = \|\lambda x\|_1 = \|Tx\|_1$, which by (1) implies that $|\lambda| \leq 1$. □

The eigenvalue 1 of T represents invariance, so there always is an eigenvalue of 1, namely corresponding to the invariant distribution π. The multiplicity of the eigenvalue 1 equals the number of connected components of G. This means that there is a unique stationary distribution iff G is connected. In addition, T has an eigenvalue -1 iff G is bipartite. Thus, with the exception of the stationary distribution π, all eigenvectors of T have eigenvalues strictly less than 1 in absolute value iff G is connected and non-biparite. We are going to use these facts in our convergence analysis of the stationary distribution π.

Convergence analysis for symmetric T. As a symmetric matrix, T has an orthonormal basis of real eigenvectors with real eigenvalue: $|\phi_i\rangle$ for $i \in [N]$ where $T|\phi\rangle = \lambda_i |\phi\rangle$ and $\lambda_i \in \mathbb{R}$. Since π is invariant under T, we can pick $|\phi_1\rangle$ to be π normalized, i.e., $|\phi_1\rangle = \pi/\|\pi\|_2$ and $\lambda_1 = 1$. Every vector representing the distribution at any point in time throughout the random walk, can be described as a linear combination of these eigenvectors. Note that the component of any distribution σ along π equals π. This is because the inner product

$$(\sigma, \pi) = \frac{1}{N} \sum_{u \in V} \sigma(u) = \frac{1}{N}$$

is independent of σ. It follows that $(\sigma - \pi, \pi) = 0$, so that $\sigma - \pi$ has no component along $|\phi_1\rangle$, so we can write

$$\sigma - \pi = \sum_{i=2}^{N} c_i |\phi_i\rangle \tag{2}$$

for some $c_i \in \mathbb{R}$.

To check for convergence to π, we want to bound the difference between the distribution after t steps of the random walk, and the stationary distribution π. We first bound the two-norm of the difference. Defining

$$\lambda_{\max} \doteq \max\{|\lambda| : \lambda \text{ is eigenvalue of eigenvector } \perp \pi\} = \max_{i=2}^{N}(|\lambda_i|), \tag{3}$$

we have that

$$\begin{aligned}
\|T^t\sigma - \pi\|_2^2 &= \|T^t(\sigma - \pi)\|_2^2 && \text{(invariance of } \pi\text{)} \\
&= \|T^t \sum_{i=2}^{N} c_i \,|\phi_i\rangle\, \|_2^2 && \text{(decomposition (2))} \\
&= \| \sum_{i=2}^{N} c_i \lambda_i^t \,|\phi_i\rangle\, \|_2^2 && \text{(linearity and eigenvector property)} \\
&= \sum_{i=2}^{N} c_i^2 \lambda_i^{2t} && \text{(orthonormality and Pythagorean theorem)} \\
&\leq \lambda_{\max}^{2t} \sum_{i=2}^{N} c_i^2 && \text{(definition (3))} \\
&= \lambda_{\max}^{2t} \| \sum_{i=2}^{N} c_i \,|\phi_i\rangle\, \|_2^2 && \text{(orthonormality and Pythagorean theorem)} \\
&= \lambda_{\max}^{2t} \|\sigma - \pi\|_2^2 && \text{(decomposition (2))}
\end{aligned}$$

Moreover, applying orthogonality and the Pythagorean theorem one more time, we know that

$$\|\sigma - \pi\|_2^2 + \|\pi\|_2^2 = \|\sigma\|_2^2 \leq 1,$$

where the last inequality holds for any probability distribution. Putting everything together, we conclude that

$$\|T^t\sigma - \pi\|_2 \leq \lambda_{\max}^t \|\sigma - \pi\|_2 \leq \lambda_{\max}^t. \tag{4}$$

The last inequality in (4) gives us an upper bound on how far the distribution after t steps of the Markov chain is from the stationary distribution π. We can express the upper bound in terms of the spectral gap δ, where $\delta \doteq 1 - \lambda_{\max}$, as follows:

$$\lambda_{\max}^t = (1 - \delta)^t \leq \exp(-\delta)^t = \exp(-\delta t),$$

where we used the fact that $1 + x \leq \exp(x)$ for $x = -\delta$, which follows from the convexity of the exponential function as $1 + x$ is the tangent to $\exp(x)$ at $x = 0$.

In order to bound the statistical distance between the two distributions, we use the Cauchy-Schwartz inequality to upper bound the 1-norm as a function of the 2-norm:

$$\|x\|_1 = \sum_i |x_i| = (x, \operatorname{sgn}(x)) \leq \|x\|_2 \| \operatorname{sgn}(x)\|_2 = \|x\|_2 \sqrt{N},$$

where $\operatorname{sgn}(x)$ denotes the vector of the same dimension as x whose i-th component is the sign of x_i, i.e., 1 if $x_1 > 0$, -1 if $x_i < 0$, and 0 otherwise. Hence,

$$\|T^t\sigma - \pi\|_1 \leq \sqrt{N} \|T^t\sigma - \pi\|_2 \leq \sqrt{N} \lambda_{max}^t \leq \sqrt{N} \exp(-\delta t).$$

Thus, we can guarantee that $\|T^t\sigma - \pi\|_1 \leq \eta$ by setting $t \geq \frac{1}{\delta} \ln(\sqrt{N}/\eta)$.

Convergence to π is guaranteed as long as $\delta > 0$, which is the case iff G is connected and non-bipartite. The number of steps required to approach π scales as $1/\delta$. If we want to guarantee fast convergence, we need a large spectral gap δ. On the other hand, if δ is close to zero, convergence may be slow.

5 Applications of random walks

Random walks can be used to model real world phenomena; for instance, in physics they are used as simple models of Brownian motion. They also can be used to model gambling. In this section we review two of the most common applications in computer science: sampling and search. We focus more on search as the applications of quantum walks that we will discuss deal with search.

5.1 Sampling

This is the most common use of classical random walks. If we want to sample from a complicated distribution π, we can set up a random walk with stationary distribution π, start from a simple distribution σ (e.g., a point distribution), and run the random walk for a number of steps. If the spectral gap δ is large, then a small number of steps suffice to bring us close to π. This is known as rapid mixing. If the underlying graph is simple, the process yields an efficient way to (approximately) sample from π.

The process may also be of interest in case π is simple, say uniform, but one cannot obtain a uniform sample as a primitive. In such settings expander graphs are often helpful. These are infinite families of graphs that have bounded (i.e., constant) degree and a spectral gap δ that is at least some positive constant. By the above analysis, regular expanders exhibit rapid mixing to the uniform distribution, so a small number of steps of a random walk starting from some fixed vertex suffices to obtain an almost-uniform sample.

5.2 Search

Another application that is perhaps less common classically, but is very important in the quantum setting, is search. Assume we have a set V of vertices, some of which are good and others bad, and our goal is to find a good vertex. As before, we represent goodness as a Boolean function $f : V \to \{0, 1\}$, where $f(v) = 1$ indicates that v is good. One way to search for a good vertex is to pick one uniformly at random, and check whether f evaluates to one. If so, we are done; if not, we retry. If the fraction ϵ of good vertices is large, we only a few trials are needed until the first success, namely $1/\epsilon$ in expectation. However, similar to the above sampling setting, obtaining a uniform vertex may be expensive. Instead of picking a fresh uniform sample for each trial, it may be cheaper to move to a neighboring vertex in an underlying graph G where each edge represents a simple operation, and use that neighbor for our next trial. In such settings, searching for a good vertex using a random walk makes sense, even when using the stationary distribution π as the start distribution σ. We distinguish between three contributions to the cost of this search process: setup, update, and check.

- Setup cost s: The cost of picking a sample from the start distribution σ, which may or may not be the stationary distribution π. We incur this cost each time we start the random walk.
- Update cost u: The cost to execute one step of the random walk. We incur this cost t times when we perform a random walk of length t.
- Check cost c: The cost to determine whether the current vertex v is good, i.e., whether $f(v) = 1$. We may or may not want to do this for every step of the random walk.

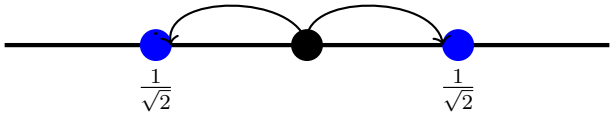
6 Quantum Walk

We like to setup an iterative process just like the classical walk that allows us to mimic a classical random walk, but in such a way that we move faster. Quantum walks take advantage of the quantum phenomenon of interference to speed up the convergence.

Model. For ease of exposition we consider the unweighted symmetric case. Now, if we are at vertex v, we would like to have a unitary transformation (quantum operations are unitary) that maps a vertex v to a uniform superposition of the neighbors of v.

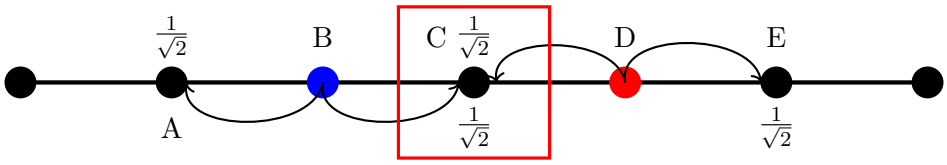
$$|v\rangle \mapsto |N_v\rangle \doteq \frac{1}{\sqrt{|N_v|}} \sum_{u \in N_v} |u\rangle$$

If, however, we apply this operation at every vertex, this is not a unitary operation. Consider a random walk on a line. The above quantum version applied to a basis state $|v\rangle$ results in the following superposition after one step:



As desired, the probability of being at the left neighbor of v equals $1/2$, as is the probability of being at the right neighbor of v. This is realized by transition amplitudes of $\frac{1}{\sqrt{2}}$ to both points.

But if we apply this operation at two vertices that have one common neighbor, the resulting transformation is not unitary, as the following figure illustrates.



Consider the basis states corresponding to the blue and the red vertices. As distinct basis states, the two states are orthogonal. After performing one step of the quantum walk on the blue state, the resulting superposition has an amplitude of $\frac{1}{\sqrt{2}}$ for both A and C:

$$|B\rangle \mapsto \frac{1}{\sqrt{2}}(|A\rangle + |C\rangle)$$

Similarly, one step of the quantum walk on the red state leads to a superposition of C and E, both with amplitude $\frac{1}{\sqrt{2}}$:

$$|D\rangle \mapsto \frac{1}{\sqrt{2}}(|C\rangle + |E\rangle)$$

These two superpositions are no longer orthogonal as they overlap in exactly one component (as marked by the red box in the graphic). A unitary transformation maps orthogonal states to orthogonal states. Therefore, this operation is not unitary.

To fix this we can keep track of more of the history to prevent interference from happening. It suffices to remember the previous vertex to get this idea to work, i.e., the state consists of the current vertex v as well as the vertex u where we came from (or will be moving to next): $|u, v\rangle$.

Two different phases of a quantum walk. Each step of a quantum walk consists of two phases: a coin flip phase and a swap phase.

- *Coin flip phase:* We apply a unitary C_v on $|u\rangle$ that is controlled by $|v\rangle$ and replaces $|u\rangle$ by a superposition that corresponds to a random neighbor u' of v. We call this the *coin flip operator* C:
$$|u,v\rangle \mapsto C\,|u,v\rangle = C_v \otimes I\,|u,v\rangle = C_v\,|u\rangle\,|v\rangle$$
Intuitively, in this step we decide where to move next.

- *Swap phase:* In the second phase we perform the actual move by swapping the two components:
$$|u,v\rangle \mapsto S\,|u,v\rangle \doteq |v,u\rangle$$
Now u is the current vertex, and v the previous one. This phase is deterministic.

7 Choices for coin flip

There are multiple choices for the coin flip operation. The quest is for choices that lead to improvements over the classical setting. Let us again consider the unweighted symmetric case first. Here are two natural choices for the coin flip.

- *Hadamard type:* For this type the amplitude of every neighbor $u' \in N_v$ has the same absolute value. This is a true quantum realization of selecting a neighbor uniformly at random. In the case of a walk on a line, the Hadamard transform is an obvious choice. It leads to interesting results in which the walk after t steps seems to spread more or less uniformly up to a distance of $\Theta(t)$ from the start vertex, whereas a classical walk remains concentrated within a distance $\Omega(\sqrt{t})$ of the start vertex. For higher degrees, one can use generalizations of the Hadamard transform like the Fourier transform over larger groups, which involves complex amplitudes. However, that line of research has not been very fruitful to date.

- *Grover type:* Given that we know which vertex we came from, we can consider two different (real) transition amplitudes: one amplitude a for going back to the current component u (the vertex we came from) and another amplitude b for every other vertex $u' \in N_v$. In fact, treating going back differently sometimes also makes sense in classical random walks, but is nonstandard and requires extending the state space (which is the standard for quantum walks). As we will see, this choice of coin flip is closely related to the Grover iterate, and has led to several interesting speedup results for general random walks. This is the coin flip operator we consider.

Analysis of the Grover coin flip. Let us denote the Grover type coin flip transformation as C_v. The defining operation of C_v when it acts on the state $|u,v\rangle$, where the current vertex v has the neighbouring vertex set N_v can be expressed by the following equation:
$$C_v \otimes I\,|u,v\rangle = a\,|u,v\rangle + b \sum_{u' \in N_n \setminus \{u\}} |u',v\rangle$$

The operator C_v can be described by the following $N \times N$ matrix acting on the neighboring nodes N_v of v, where $N \doteq |N_v|$:

$$C_v \doteq \begin{bmatrix} a & b & b & \cdots & b \\ b & a & b & \cdots & b \\ & & & \cdots & \\ b & b & b & \cdots & a \end{bmatrix}$$

The requirement that C_v be unitary is equivalent to the following two conditions:

$$a^2 + (N-1)b^2 = 1 \quad \text{and} \quad 2ab + (N-2)b^2 = 0.$$

The first condition expresses that every column of the matrix needs to have a two-norm equal to 1. The second condition expresses that the inner product of two distinct columns of the matrix is zero. The system of equations has the following two solutions, one of them being

$$(a, b) = \pm(1, 0)$$

which is just the diagonal matrix and uninteresting, and

$$(a, b) = \pm\left(\frac{2}{N} - 1, \frac{2}{N}\right)$$

We analyze the latter solution further, specifically the eigenstructure of the resulting matrix C_v.

- The uniform superposition $|N_v\rangle = \sum_{u \in N_v} \frac{1}{\sqrt{N}} |u\rangle$ over N_v is an eigenvector with a eigenvalue $a + (N-1)b = 1$. This follows because,

$$\begin{bmatrix} a & b & b & \cdots & b \\ b & a & b & \cdots & b \\ & & & \cdots & \\ b & b & b & \cdots & a \end{bmatrix} \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} a+b+b+b+\ldots \\ b+a+b+b+\ldots \\ b+b+a+b+\ldots \\ \vdots \end{bmatrix} = \frac{1}{\sqrt{N}}(a+(N-1)b) \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix}$$

- As $C_v - (a-b)I$ has rank 1, $a - b = -1$ is an eigenvalue with multiplicity $N - 1$.

Thus, C_v has one eigenvalue 1, and $N - 1$ eigenvalues -1. This means that C_v is a reflection over the eigenvector with eigenvalue 1, which is $|N_v\rangle$. For this reason this coin flip operator is sometimes referred to as the reflection coin flip.

As $|N_v\rangle$ is uniform over N_v, the coin flip operator C_v has a close resemblance to the second of the two unitaries in one step of Grover, namely a reflection about the uniform superposition. This is why it is sometimes also referred to as the Grover coin flip. Our analysis of the resulting quantum walk in the next lecture will give us yet another view on the quadratic speedup that Grover provides.

The above analysis was for the unweighed symmetric case. It generalizes to a reflection over $|N_v\rangle \doteq \sum_u \sqrt{T_{uv}} |u\rangle$ for the symmetric weighted case.

8 Random Walk Search Complexity Analysis

We focus on applications of random walks and their quantum variants for search. To simplify the derivations, we assume that the stationary distribution π is uniform, so the transition matrix T is symmetric and we do not need to refer to the discriminant matrix D. Nonetheless, the results we obtain generalize to the setting of arbitrary T, albeit involving D in the analysis.

Hitting time. The hitting time describes how many steps of the random walk are needed to hit a good vertex, starting from a worst-case distribution σ. We formally define it as maximum over all possible start distributions σ of the expected number of steps until the first hit. Another commonly used definition is as the maximum over all possible start distributions σ of the smallest number of steps t such that the probability of hitting a good vertex within the first t steps is at least 50%. In both definitions the worst-case distributions are point distributions, i.e., one only needs to consider such distributions σ in the definition.

Exercise: Show that the two definition are within a factor of 2 of each other.

As the hitting time depends on both the underlying walk (captured by the transition matrix T) as well as which vertices are good (captured by the predicate f), we write the hitting time as $H(T, f)$. The hitting time can be related in the following manner to the spectral gap δ of T and the weight $\epsilon \doteq \Pr_{v\sim\pi}[f(v) = 1]$ of the good vertices:

$$H(T, f) = O\left(\frac{1}{\delta\epsilon}\right) \tag{5}$$

Note that, if one were to take a fresh sample from π in every step, the hitting time would be $1/\epsilon$. This is because we then run a Bernoulli experiment with success probability ϵ, which takes an expected $1/\epsilon$ steps until the first success. The intuition for the upper bound in (5) is that after an unsuccessful attempt, the vertex is distributed according to current distribution (π if we started from π) restricted to the bad vertices, and $\Omega(1/\delta)$ steps of the random walk are enough to bring that distribution close to π. Thus, an expected $1/\epsilon$ phases of $O(1/\delta)$ steps each suffice to hit a good vertex, which is $O(1/(\delta\epsilon))$ steps in total.

Expected cost of various classical approaches. We consider three approaches and express their expected costs as a function of the three cost components defined in section 4.2.

- *Without a walk:* Every time we pick a new vertex from π. Since we are only setting up one sample and checking one vertex per iteration, the cost per iteration is $s + c$. We repeat this an expected $1/\epsilon$ times to get to the first good vertex. Thus, the cost is,

$$O\left(\frac{1}{\epsilon}(s + c)\right)$$

- *Checking each step:* Performing a random walk and checking at each step whether the vertex is good. The setup stage occurs once in the beginning, then we update and check repeatedly until we arrive at a good vertex. The expected number of iterations till the first success is given by the hitting time $H(T, f)$. Therefore, the total cost is:

$$O(s + H(T, f)(u + c))$$

  This can be cheaper if s is large compared to $u + c$.

- *Checking every τ-th step:* Performing a random walk but only checking at every τ-th step whether the current vertex is good. As above the setup cost is incurred only once in the

beginning and then we check after precisely τ updates, so the cost functions looks like $s + (\tau \cdot u + c) + (\tau \cdot u + c) + \ldots$. By a similar argument as in the second scenario, the expected number of trials till the first success is $H(T^\tau, f)$ as the hitting time for τ steps of the random walk equals $H(T^\tau, f)$. The resulting cost is

$$O(s + H(T^\tau, f)(\tau \cdot u + c)).$$

This may be cheaper in case the checking cost c is high compared to the update cost u.

A particular setting of interest is $\tau = \Theta(1/\delta)$. Based on the intuition behind the upper bound $H(T, f) = O(1/(\delta\epsilon)$ that $\Omega(1/\delta)$ steps of the random walk after an unsuccessful trial bring us (close to) π again, we have that $H(T^\tau, f) = O(1/\epsilon)$ for this choice of τ, so the expected cost is

$$O\left(s + \frac{1}{\epsilon}\left(\frac{1}{\delta}u + c\right)\right).$$

Quantum walk search speed up. We end this section by stating the speedups that can be realized for search by the quantum walks based on the reflection / Grover coin flip and assuming that the initial distribution σ is the stationary distribution π. For comparison, we include the costs for search based on a classical random walk:

Runtime	Classical random walk	Quantum random walk
Checking each step	$O(s + H(T, f)(u + c))$	$\tilde{O}(s + \sqrt{H(T, f)}(u + c))$
Checking each τ-th step	$O(s + H(T^\tau, f)(\tau \cdot u + c))$	$\tilde{O}(s + \sqrt{H(T^\tau, f)}(\sqrt{\tau} \cdot u + c))$
Checking each τ-th step for $\tau = \Theta(1/\delta)$	$O(s + \frac{1}{\epsilon}(\frac{1}{\delta}u + c)$	$\tilde{O}(s + \frac{1}{\sqrt{\epsilon}}(\frac{1}{\sqrt{\delta}}u + c))$

When checking each step, whereas classically the expected number of iterations is $H(T, f)$, the same approach only requires an expected $O\sqrt{H(T, f))}$ steps using a quantum walk. Similarly, when checking each τ-th step, in each iteration, classically we need τ stages to end up at the τ-th stage vertex; however, using the Grover coin flip for quantum walk we only meed $O(\sqrt{\tau})$ updates for roughly the same outcome. This arises from the fast forwarding lemma, which will be covered in the next lecture. Note the square roots on the hitting time, δ, and ϵ, resulting in the potential for a square-root speedup. Two caveats:

- The costs s, u, and c refer to somewhat different processes in the classical vs. the quantum setting.
- The costs in the quantum setting have additional polylog factors, represented by the use of the symbol $\tilde{O}$ instead of O. The additional polylog factors are not needed when the goal is to merely detect the existence of a good vertex.

9 Collision Problem

The collision problem takes as input a black-box for a function $h : \{0, 1\}^n \to R$ for some range R and the goal is to find two distinct inputs x_1, x_2 such that $h(x_1) = h(x_2)$ or report that no such pair

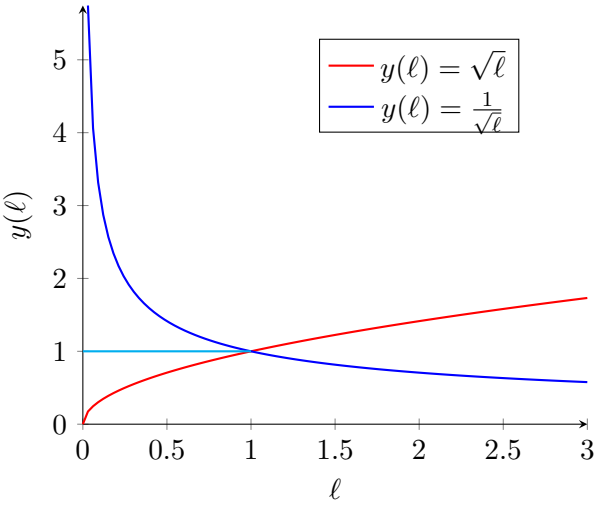


Figure 3: Intersection point yields optimum up to factor of 2.

of inputs exists. The decision version of this problem is known as *element distinctness*. Quantum walk search yields an optimal query complexity up to constant or poly-log factors (depending on the version). The collision problem was one of the main drives behind research into quantum walk search.

First attempt. The first idea to tackle such a problem would be to use the Grover search algorithm. We would pick a subset of some ℓ inputs uniformly at random and check if there is a collision among them, where ℓ is a parameter. If no collisions are found among the initial group, we use the Grover search algorithm to find a collision with any of those ℓ inputs. Assuming there is a collision, in the worst case there is only a single collision. For that collision to be found, at least one of the two inputs involved in the collision must be among the ℓ inputs chosen. The probability of that happening is $\Theta(\binom{N-2}{\ell-1}/\binom{N}{\ell})$, which is at best $\Theta(\frac{\ell}{N})$. We repeat the process $\Omega(\frac{N}{\ell})$ times to get a high confidence. As the number of queries for one iteration is $\ell + \sqrt{N}$, the overall number of queries is $\Omega\left(\frac{N}{\ell}(\ell + \sqrt{N})\right)$, which isn't any better than $\Theta(N)$.

Second attempt. For the second attempt we use the same algorithm as above except we apply amplitude amplification to boost the success of finding a collision between one of the ℓ inputs and the other inputs. Using this method we are able to decrease the number of repetitions from $\Theta(\frac{N}{\ell})$ to $\Theta(\sqrt{\frac{N}{\ell}})$. This reduces the overall number of queries to $\Theta\left(\sqrt{\frac{N}{\ell}}(\ell + \sqrt{N})\right)$. As the function $g(l) = \sqrt{\frac{N}{\ell}}(\ell + \sqrt{N})$ goes to ∞ as $\ell \to 0$ or $\ell \to \infty$, we can optimize the function for ℓ. Whenever a function is the sum of a decreasing and increasing term (as is the case here), finding the intersection point gives the optimum value up to a factor of two, which is good enough as we don't care about constants. The factor of two follows because before the intersection point, the decreasing term is larger than the value μ at the intersection point, and after the intersection point the increasing term is larger than μ, so the minimum cannot be smaller than μ, whereas the value at the intersection point of the sum of the two terms equals 2μ. See Figure 3 for an illustration.

In this case, equating both terms yields the requirement

$$\sqrt{N\ell} = \frac{N}{\sqrt{\ell}},$$

which is equivalent to $\ell = \sqrt{N}$, yielding a total of $\Theta(N^{3/4})$ queries. While this is an improvement, we can do better.

Using quantum walk search. For quantum walk search, we begin the same way as the first two attempts by randomly selecting a subset of ℓ inputs and checking if there is a collision among them. The difference now is that if no collision has been found among the ℓ inputs, we simply swap one of the ℓ inputs for an input not among the ℓ inputs and check if the new input collides with any of the other $\ell - 1$ inputs. This can be viewed as a random walk on a graph for which subsets of size ℓ among the N values serve as the vertices and edges connecting subsets that are only one swap apart. This type of graph is known as the Johnson graph $J(N, \ell)$.

The Johnson graph $J(N, \ell)$ is a regular graph of a degree of $\ell(N-\ell)$. As the graph is undirected and regular, the stationary distribution π is uniform. It can be shown that $J(N, \ell)$ has a spectral gap of $\delta = \frac{N}{\ell(N-\ell)}$, which is $\Theta(\frac{1}{\ell})$ for $\ell \ll N$. Assuming that there is at least one collision, within this graph there must be at least $\binom{N-2}{\ell-2}$ good subsets (ones in which a collision is found). Using this we get:

$$\epsilon \geq \binom{N-2}{\ell-2} \Big/ \binom{N}{\ell} = \Theta\left(\frac{\ell^2}{N^2}\right)$$
$$H = O\left(\frac{1}{\delta\epsilon}\right) = O\left(\frac{N^2}{\ell}\right)$$

The query cost for the setup is $s = \ell$ as we need to query f for each of the selected ℓ values. The query cost for the update is $u = 2$ as we need to clear the original value in a reversible way (that is, query f and XOR the result with the stored value) and query f for the new value. Finally, the query cost for the check is $c = 0$ as we already have the outputs of f for the ℓ values. This results in a total decision cost of $O(s + \sqrt{H}(u + c)) = O(\ell + \frac{N}{\sqrt{\ell}})$ queries. This cost is optimized for $\ell = N^{2/3}$. To see this, we use the same strategy as before where we set the increasing term equal to the decreasing term.

$$\ell = \frac{N}{\sqrt{\ell}}$$
$$\ell^{3/2} = N$$
$$\ell = N^{2/3}$$

Using this optimal value for ℓ (up to constant factors), we get a tight upper bound of $O(N^{2/3})$ on the number of queries, which turns out to be tight (up to constant factors). For search, rather than decision, an additional polylog cost factor is incurred.