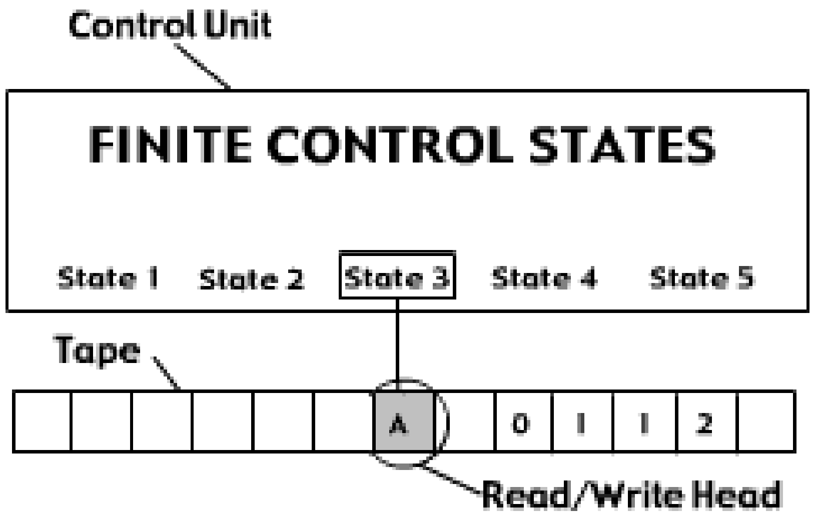
計算機科学の視点からの
量子アルゴリズムの応用例

Shigeru YAMASHITA†, † Ritsumeikan University

Today's Talk

- Introduction
  - Shigeru Yamashita
  - Topics for C. S. people (Yamashita's Perspective)
    - NP, Query Complexity
- Amplitude Amplification and Its Applications
- Quantum Walk and Its Applications

Turing's Machine



- Alphabet Σ, state space K
- $f: K \times \Sigma \to K \times \Sigma \times \{\leftarrow, \rightarrow, ?\} \times \{\text{Halt}, \text{Yes}, \text{No}\}$
- Language: L µ Σ^* is decided by M_L

Shigeru Yamashita ger@cs.ritsumei.ac.jp

P vs. NP

Polynomial Time (*PTIME*)

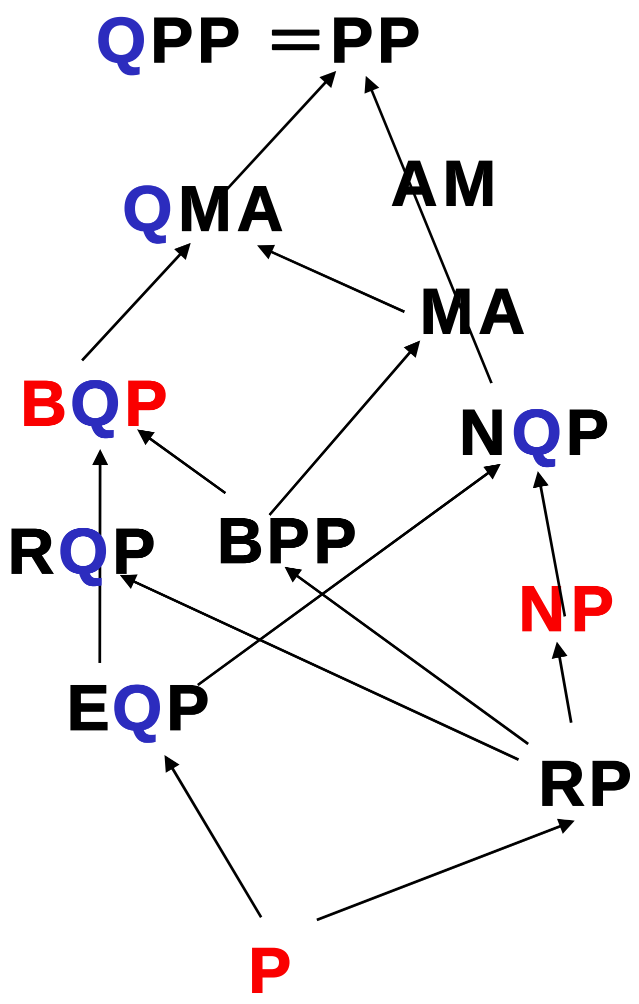
L is in P if there exists a Turing Machine M which for every x, decides if x is in L in polynomially many steps.

Non-Deterministic Polynomial Time

L is in NP if there exists a Turing Machine M s.t. for every x

- *If x is in L then there **exists w** s.t. M(x,w)→"Yes" in PTIME.*
- *If x is **not** in L then there is **no** such w.*

Computational Complexity Class



Most People in C. S. believe

Polynomial Time Turing Hypothesis:
Any physical computing device can be simulated by a randomizing Turing machine that takes a number of steps that grows as at most some fixed polynomial in the quantity T+S+E where T, A and E are the time, space and energy used by the computing device.

How about Quantum T. M. …..?

NP-completeness

*A problem P is NP-hard if **every** problem in NP has a polynomial-time reduction to P.*

Moral: At least as hard as any other problem in NP

---

*If P is in NP **and** NP-hard then P is NP-complete.*

Cook's Theorem
Satisfiability (SAT) is NP-Complete

What is SAT?

For n variable Boolean Function

$f(X) = (x_1 + x'_2 + x_3)(x'_4 + x_5 + x_6) \ldots (x_n + x'_{n-2} + x_1)$

Find an variable assignment $X = (x_1, x_2, \ldots, x_n)$

s. t. $f(X) = 1$

- There are obviously 2^n possible assignments to the n variables, so exhaustive search takes time $O(2^n)$
- It's NP-complete
  - if we can solve SAT quickly, we can solve anything in NP quickly (Cook's theorem, 1971)
- Many and varied applications in itself:
  - theorem proving
  - hardware design
  - machine vision

Shigeru Yamashita ger@cs.ritsumei.ac.jp

k-SAT

- If the maximum number of variables in each clause is *k*, we call the problem *k*-SAT
- 1-SAT is simple: $f(X) = (x_1)\,(x'_2)\,(x_N)$
  - and can be solved in time $O(n)$
- 2-SAT is also straightforward
  - can be solved in time $O(n^2)$ using a simple random walk algorithm, which we will see later
- 3-SAT is NP-complete

Very Hard to Analyze Computational Complexity; Consider "Query Complexity"

Unstructured Search for SAT

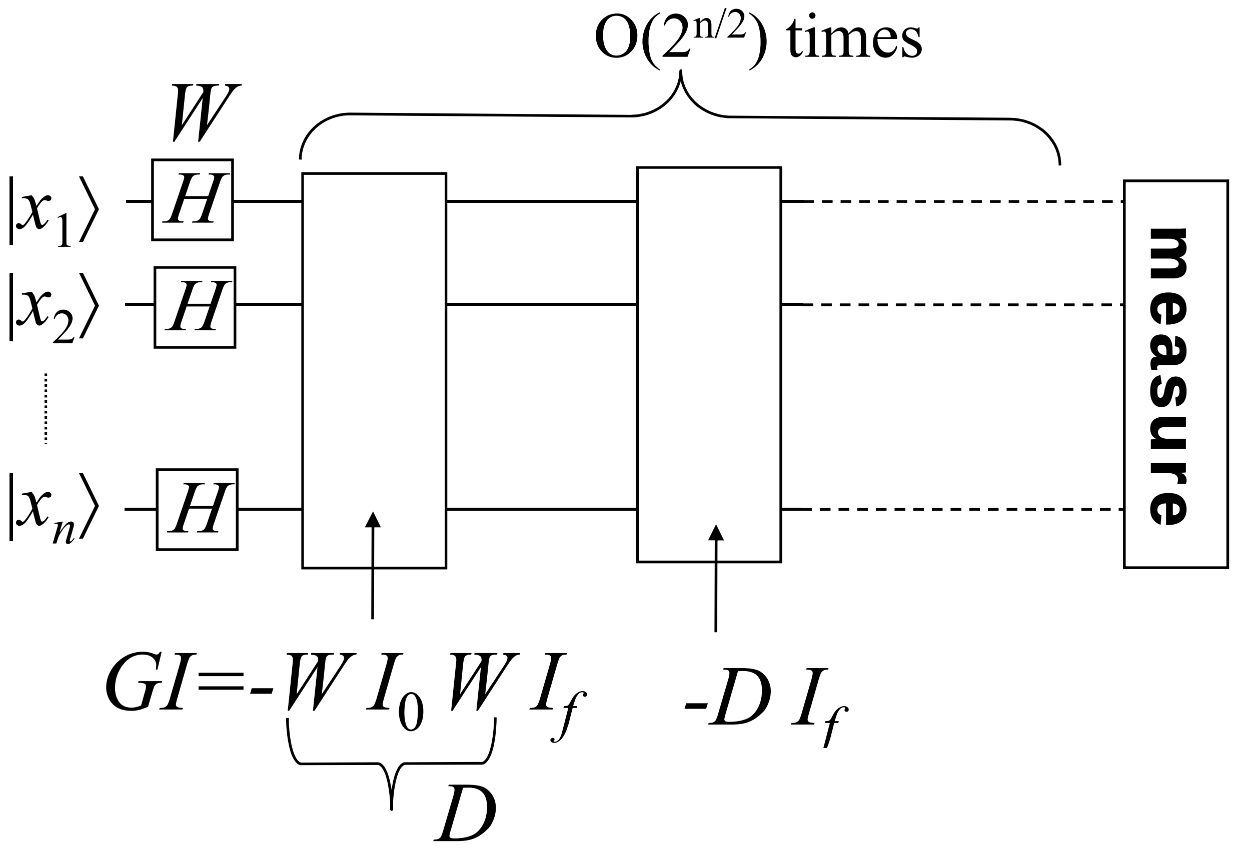
- Don't use any knowledge of the problem's structure; just pass in an assignment and ask "does this satisfy the expression?"
- What we can do is only to evaluate f
- # of such evaluations is called query complexity
- (Usually) Computational Complexity =
  # of queries * (query cost)

Today's Talk

- Introduction
  - –Shigeru Yamashita
  - –Topics for C. S. people (Yamashita's Perspective)
    - NP, Query Complexity
- Amplitude Amplification and Its Applications
  - –Generalization of G. S. to A. A.
  - –How to utilize A. A.
- Quantum Walk and Its Applications

Grover's Algorithm Revisited

Find a variable assignment s. t. a function becomes 1 (among 2^n possible assignments)



The computational cost = query comp = $O(2^{n/2})$

Solving SAT based on Sampling

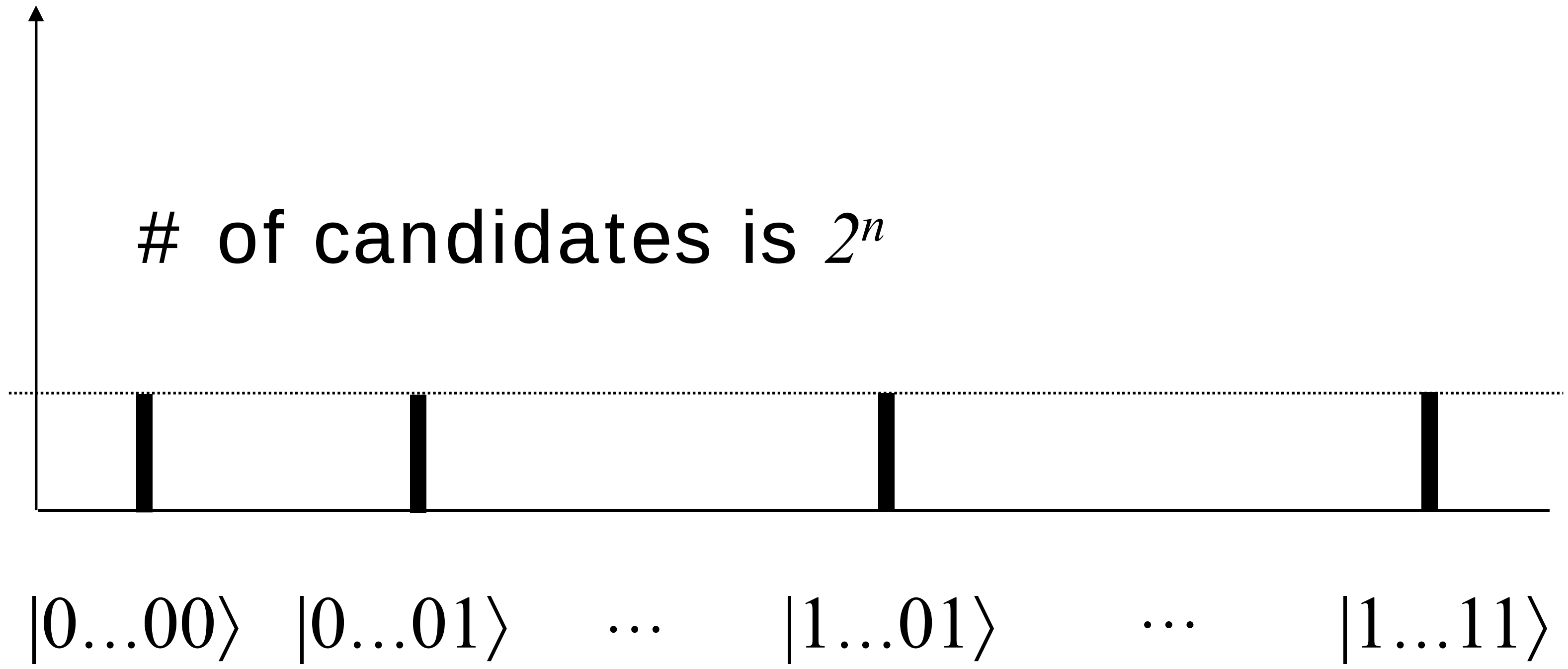
For N variable Boolean Function
$f(X) = (x_1 + x'_2 + x_3)\,(x'_4 + x_5 + x_6)\ \ldots\ (x_N + x'_{N-2} + x_1)$
Find a variable assignment $X = (x_1, x_2, \ldots, x_N)$
s. t. $f(X) = 1$

(1) Select a variable assignment, x, from 2^n assignments randomly
(2) Check whether $f(x) = 1$

⇨ Succ. Prob. = $1/2^n$

⇨ Succ. Prob. = const. by $O(2^n)$ samplings
(Quantumly, $O(2^{n/2})$ by G. S.)

Sampling from all the possible candidates



Succ. Prob. of one sampling is $1/2^n$

Classical Sampling : $O(2^n)$

Grover Search: $O(2^{n/2})$

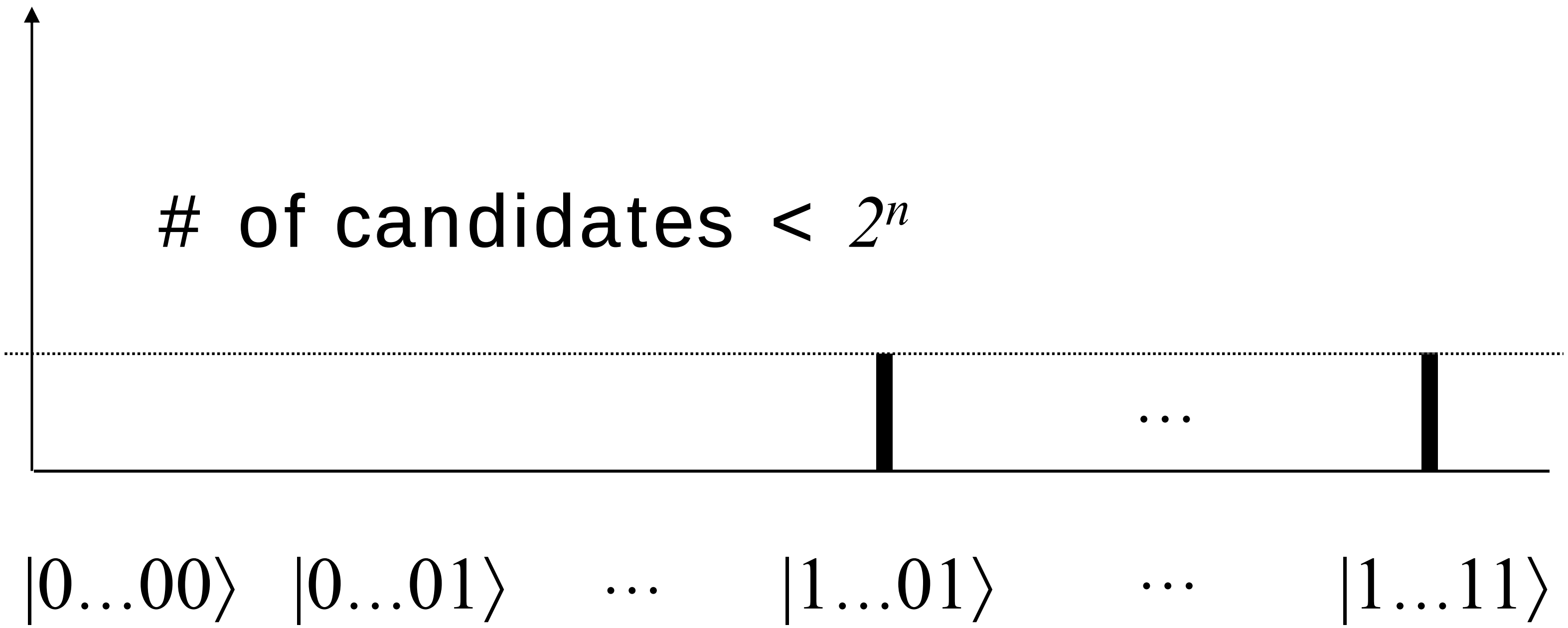
(Classical) Sampling from narrowed set of candidates

- Solution candidates (range) is known
- Check an answer is easy

Then, we can do the following:

If the succ. prob. of a sampling is a, we can get an answer with constant prob. by repeating the sampling $O(1/a)$ times.

Quantum Sampling from narrowed set of candidates



If the succ. prob. of a sampling is a, we can get an answer with constant prob. by repeating the sampling $O(1/a)$ times.

If we have a quantum algorithm, A, to create the above superposition, $O(\sqrt{1/a}\,)$ repetition is enough

(Quantum) Amplitude Amplification

A : (Quantum) Algorithm without measurement

a : The prob. of getting the correct answer by measuring the result of A

To boost the succ. prob. to constant,

By classical amplification

Repeat A $O(1/a)$ times

By quantum amplification

Repeat $Q = -\mathrm{A}I_0\mathrm{A}^{-1}I_f$ $O\left(\sqrt{1/a}\right)$ times

Relation to G. S.

Repeat $Q = -\mathrm{A} I_0 \mathrm{A}^{-1} I_f$ $\quad O\left(\sqrt{1/a}\right)$ times

Generalizations of G. S.

$$GI = -W I_0 W I_f$$

- Without prior knowledge
  $\rightarrow W$: Generating uniform superposition
- With some knowledge concerning solutions
  $\rightarrow \mathrm{A}$: Generating narrowed superposition

Analysis of A. A. (1/2)

$|\psi\rangle = A|0\rangle = |\psi_1\rangle + |\psi_0\rangle \rightarrow Q = -AI_0A^{-1}I_f$

a: Succ. Prob. of A $\quad (a = \langle\psi_1|\psi_1\rangle)$

$$Q|\psi_1\rangle = (1-2a)|\psi_1\rangle \quad -2a|\psi_0\rangle$$

$$Q|\psi_0\rangle = 2(1-a)|\psi_1\rangle + (1-2a)|\psi_0\rangle$$

$$\alpha_1|\psi_1\rangle + \alpha_0|\psi_0\rangle \rightarrow \beta_1|\psi_1\rangle + \beta_0|\psi_0\rangle$$

$$\begin{pmatrix}\beta_1\\ \beta_0\end{pmatrix} = Q\begin{pmatrix}\alpha_1\\ \alpha_0\end{pmatrix} = \begin{pmatrix}1-2a & -2a\\ 2(1-a) & 1-2a\end{pmatrix}\begin{pmatrix}\alpha_1\\ \alpha_0\end{pmatrix}$$

Analysis of A. A. (2/2)

$$\begin{pmatrix} \beta_1 \\ \beta_0 \end{pmatrix} = Q \begin{pmatrix} \alpha_1 \\ \alpha_0 \end{pmatrix} = \begin{pmatrix} 1-2a & -2a \\ 2(1-a) & 1-2a \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_0 \end{pmatrix}$$

$$Q^j \left|\psi\right\rangle = \frac{1}{\sqrt{a}} \sin((2j+1)\theta_a) \left|\psi_1\right\rangle + \frac{1}{\sqrt{1-a}} \cos((2j+1)\theta_a) \left|\psi_0\right\rangle$$

$$(\sin^2 \theta_a = a)$$

Set $j = \lfloor \pi / 4\theta_a \rfloor = O(1/\sqrt{a})$

Then, $|\psi_1\rangle$ is measured w.p. (at least) *1-a*

Summary of A. A.

- A. A. is a generalization of G. S. algorithm
- It can boost the success probability of randomized algorithm quadratically faster than classically if we have a way to verify the solution
- Usually, we consider query complexity, and how to reduce it by using A. A. is one of research topics.

Today's Talk

- Introduction
  - Shigeru Yamashita
  - Topics for C. S. people (Yamashita's Perspective)
    - NP, Query Complexity
- Amplitude Amplification and Its Applications
  - Generalization of G. S. to A. A.
  - How to utilize A. A.
- Quantum Walk and Its Applications

How to use A. A.

- Generally speaking, we can do quadratically faster than the classical corresponding search based on sampling

Example 1: How to use prior knowledge

Example 2: How to use A. A.

Example 1: How to use prior knowledge (1/3)

For n variable Boolean Function

$$f(X) = (x_1 + x'_2 + x_3)(x'_4 + x_5 + x_6) \ldots (x_n + x'_{n-2} + x_1)$$

Find an variable assignment $X = (x_1, x_2, \ldots, x_n)$

s. t. $f(X) = 1$

Prior knowledge: The variable assignment should have exactly k 1's

(Classical) sampling using the knowledge

1. Select a variable assignment that has exactly k 1's 2n assignments randomly
2. Check whether $f(x) = 1$

$$\text{Succ. Prob.} = 1 / {}_nC_k \longrightarrow O\left({}_nC_k\right) = O\left(\sqrt{2^n}\right)$$

Shigeru Yamashita ger@cs.ritsumei.ac.jp

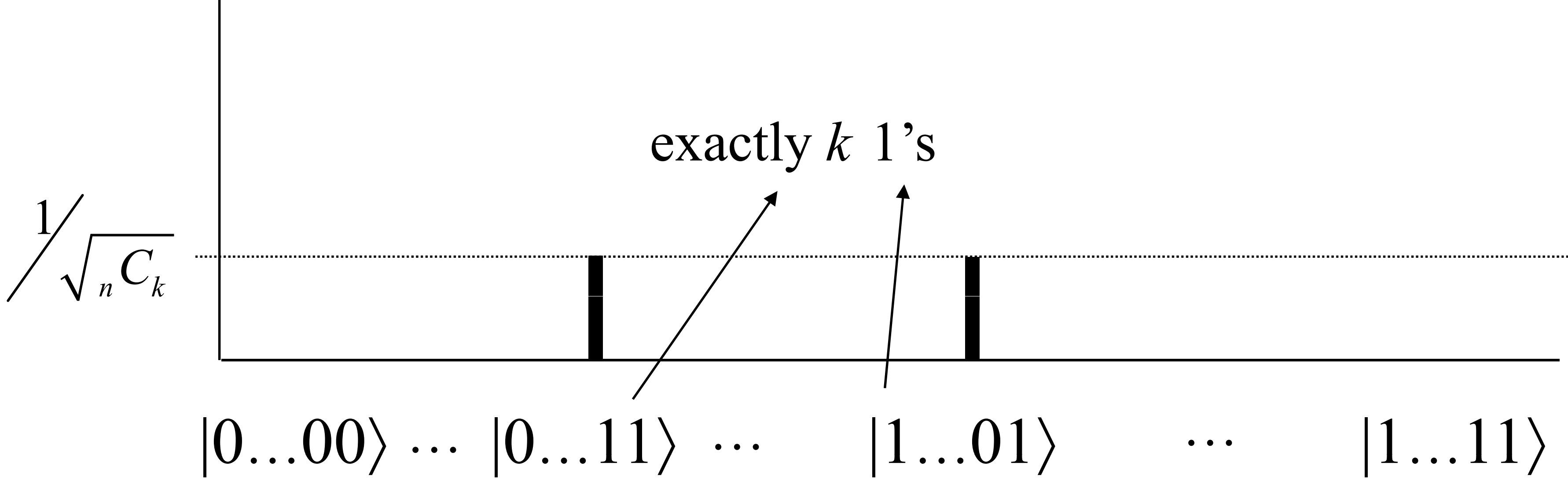
Example 1: How to use prior knowledge (2/3)

(Classical) sampling using the knowledge

1. Select a variable assignment that has exactly k 1's
   2n assignments randomly
2. Check whether $f(x) = 1$

We need to construct a quantum algorithm A that

generates the following states from $|0\ldots00\rangle$



Example 1: How to use prior knowledge (3/3)

$$A = \otimes \begin{pmatrix} \sqrt{1-\frac{k}{n}} & \sqrt{\frac{k}{n}} \\ \sqrt{\frac{k}{n}} & -\sqrt{1-\frac{k}{n}} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

$$|0\rangle \rightarrow |0\rangle \ \sqrt{1-k/n}$$
$$|0\rangle \rightarrow |1\rangle \ \sqrt{k/n}$$
$$|1\rangle \rightarrow |0\rangle \ \sqrt{k/n}$$
$$|1\rangle \rightarrow |1\rangle \ -\sqrt{1-k/n}$$

$$|0010010110\rangle$$
$$\uparrow \quad \uparrow \ \uparrow \ \uparrow$$
$$|0000000000\rangle$$

$$\left(1-\frac{k}{n}\right)^{\frac{n-k}{2}} \left(\frac{k}{n}\right)^{\frac{k}{2}} \approx O\left(1 \Big/ \sqrt{{}_nC_k}\right)$$

$$Q = -I_0 A^{-1} I_f A$$

$$\Rightarrow \quad O\left(\sqrt{{}_nC_k}\right)$$

Example 2: How to use A. A. (1/3)

Element distinctness

7	9	2	···	1
x_1	x_2	x_3		x_N

- Numbers $x_1, x_2, \ldots, x_N$.
- Determine if two of them are equal.
- Classically: N queries are needed

Example 2: How to use A. A. (2/3)

Randomized Algorithm for E. D.

1. Choose $\sqrt{N}$ numbers $i_1, \cdots, i_{\sqrt{N}} \in \{1, 2, \cdots, N\}$.
   If any two of $f(i_1), \cdots, f(i_{\sqrt{N}})$ are equal, $\boxed{O(\sqrt{N})\text{queries}}$
   output the two equal elements.
2. Find k (by G. S.) s. t. $f(k) = f(i_j)$ where k is $\boxed{O(\sqrt{N})\text{queries}}$
   chosen from the remaining $N - \sqrt{N}$ indecies.

* We do not need to query $f(i_j)$ at step 2.

If there is a pair i and j s. t. $f(i) = f(j)$, the algorithm succeds when i is chosen at step 1. Thus, the success probability $\geq \dfrac{1}{\sqrt{N}}$.

Example 2: How to use A. A. (3 / 3)

Boost Succ. Prob. by A. A.

Recall that
if the succ. Prob. = a
we can get the constant error algorithm by repeating the algorithm $O\left(\sqrt{1/a}\right)$ times

Thus, in this case, we need to repeat

the algorithm $O\left(\sqrt{\frac{1}{1/\sqrt{N}}}\right) = O\left(N^{1/4}\right)$ times

The total query complexity is

$$O\left(N^{1/4}\sqrt{N}\right) = O\left(N^{3/4}\right)$$

Summary of A. A.

- A. A. is a generalization of G. S. algorithm
- It can boost the success probability of randomized algorithm quadratically faster than classically **if we have a way to verify the solution**
- Usually, we consider **query complexity**, and how to reduce it by using A. A. is one of research topics.

Today's Talk

- Introduction
  - Shigeru Yamashita
  - Topics for C. S. people (Yamashita's Perspective)
    - NP, Query Complexity
- Amplitude Amplification and Its Applications
  - Generalization of G. S. to A. A.
  - How to utilize A. A.
  - Additional Topic
- Quantum Walk and Its Applications

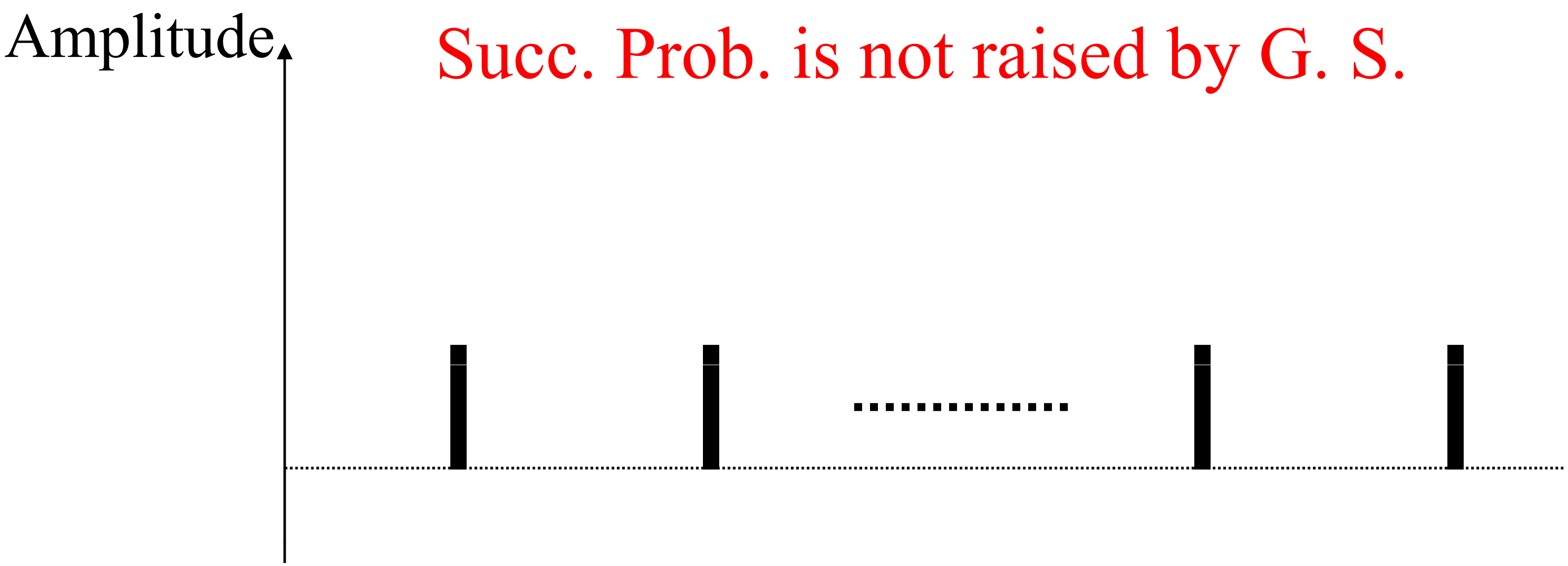
When the succ. prob. is very high

Grover's Search

Repeat $GI = -W\, I_0\, W\, I_f$ $\quad O\left(1/\sqrt{a}\right)$ times

When a=1/2

Succ. Prob. is not raised by G. S.



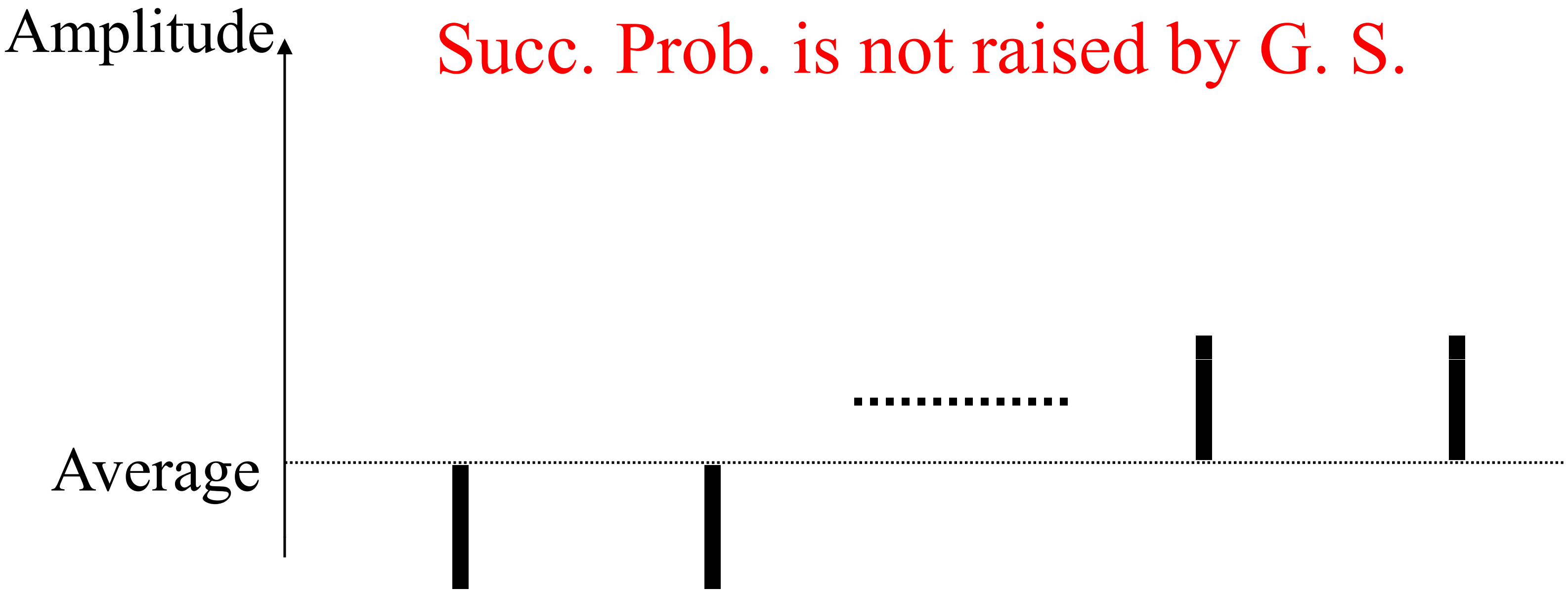
When the succ. prob. is very high

Grover's Search

Repeat $GI = -W\, I_0\, W\, I_f$ $\quad O\left(1/\sqrt{a}\right)$ times

When a=1/2

Succ. Prob. is not raised by G. S.



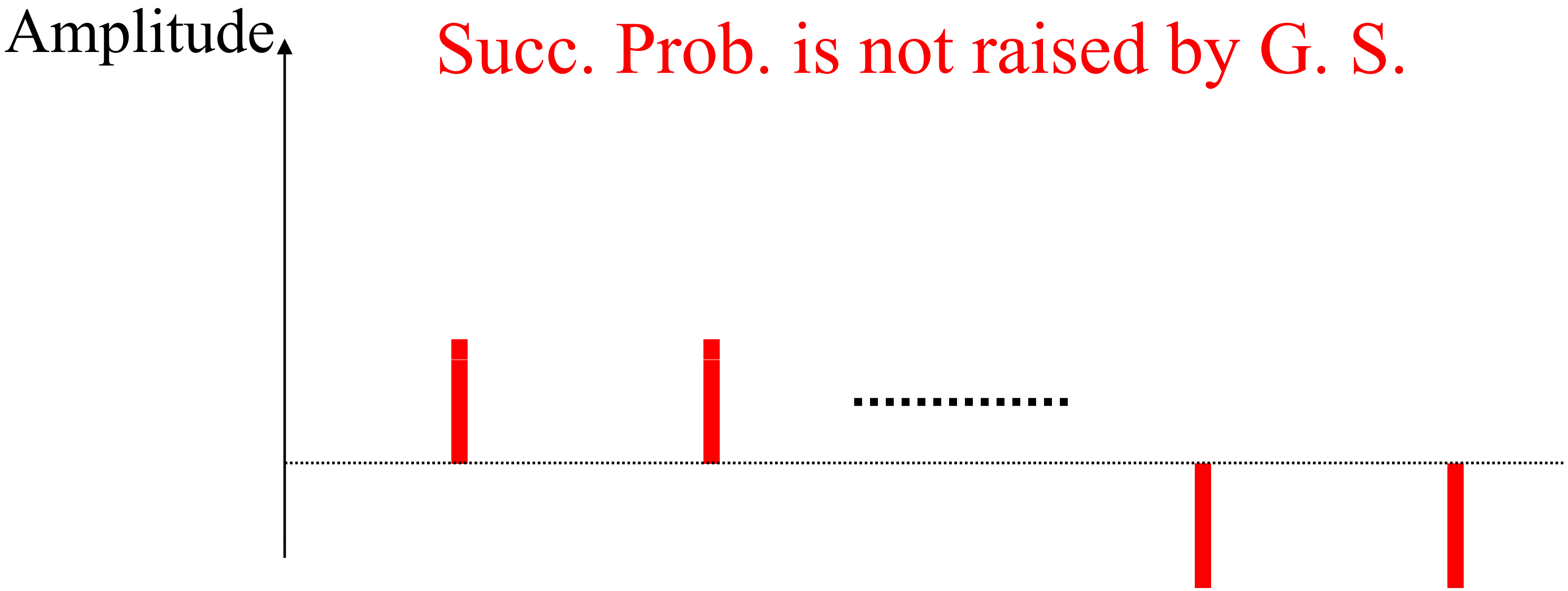
When the succ. prob. is high

Grover's Search

Repeat $GI = -W\, I_0\, W\, I_f$ $\quad O\left(1/\sqrt{a}\right)$ times

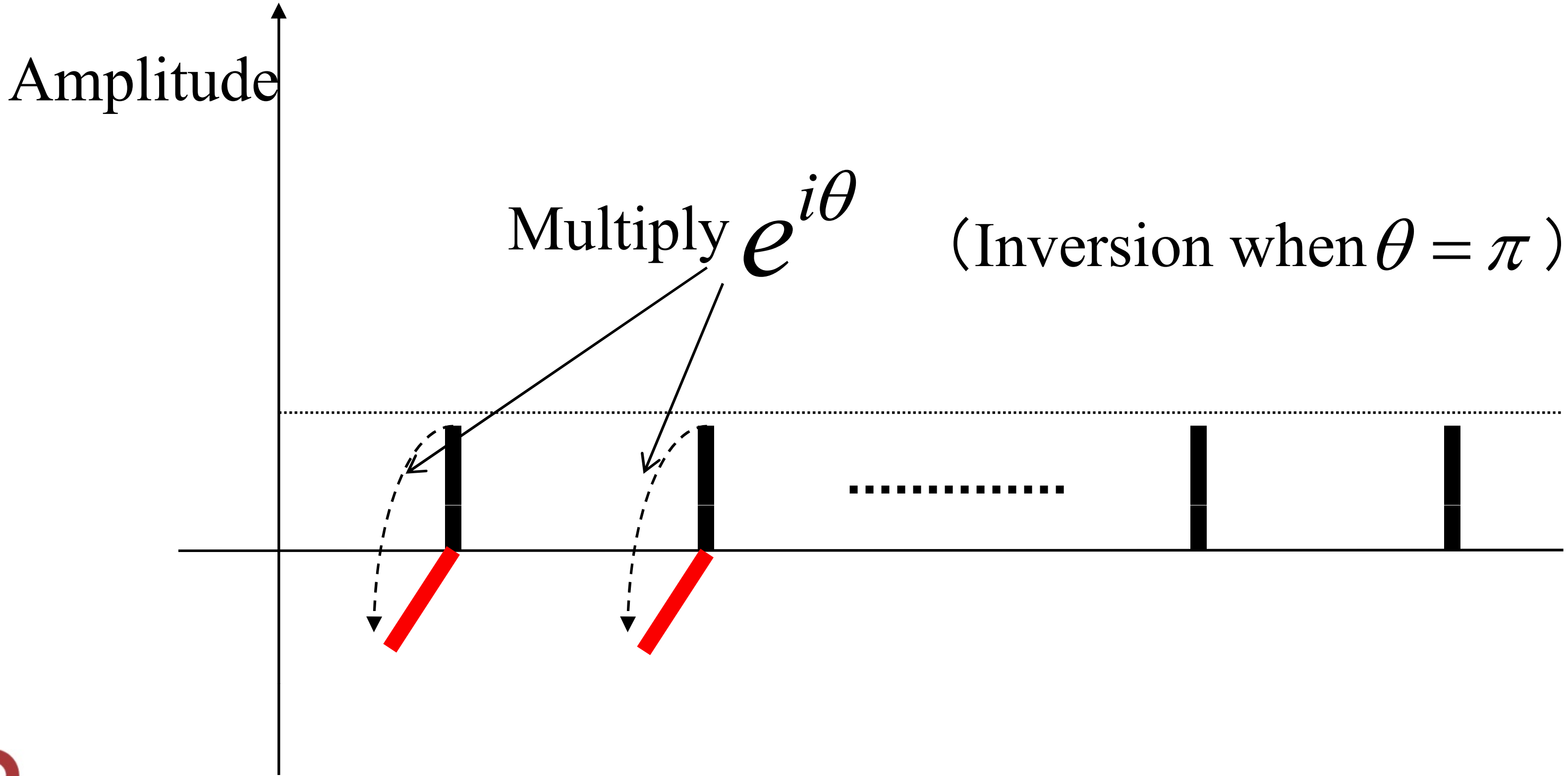
When a=1/2

Succ. Prob. is not raised by G. S.



Generalization of I_0 and I_f (1/4)

When the succ. prob. is high, A. A. does not help
$\longrightarrow$ decrease the succ. prob. purposely



Generalization of I_0 and I_f (2 / 4)

Repeat $Q = -WI_0WI_f$ $O\left(1/\sqrt{a}\right)$ times

I_0 …Invert $|0\rangle$
I_f… Invert $|i\rangle$ (s.t. $f(i)=1$)

S_0 …Multiply $e^{i\theta}$ to $|0\rangle$
S_f… Multiply $e^{i\theta}$ to $|i\rangle$ (s.t. $f(i)=1$)

$Q' = -WS_0WS_f$ **Just apply once!**

Generalization of I_0 and I_f (3 / 4)

$$W|0\rangle = |\psi_1\rangle + |\psi_0\rangle \quad \begin{cases} |\psi_1\rangle = 1/\sqrt{N} \sum_{f(i)=1} |i\rangle \\ |\psi_0\rangle = 1/\sqrt{N} \sum_{f(i)\neq 1} |i\rangle \\ (a = \langle \psi_1 | \psi_1 \rangle) \end{cases}$$

$$\Big\downarrow Q'$$

$$(-2e^{i\theta} + 1 - (e^{i\theta} - 1)^2 a)|\psi_1\rangle + (-e^{i\theta} - (e^{i\theta} - 1)^2 a)|\psi_0\rangle$$

The Prob. of getting $|\psi_0\rangle$ (= error prob.)

$$= \ (1 - 2\cos\theta - 2(1 - \cos\theta)(1-a)^2)(1-a)$$

Generalization of I_0 and I_f (4 / 4)

$$\boxed{\text{error prob.} = (1-2\cos\theta - 2(1-\cos\theta)(1-a)^2)(1-a)}$$

$$\cos\theta = \frac{2a-1}{2a} \rightarrow \text{error prob.} = 0$$

$$\cos\theta = \frac{1}{2} \rightarrow \text{error prob.} = (1-a)^3$$

(Classically, error prob. $=(1-a)$)

Outline

1. What is query complexity？

2. Amplitude Amplification and Its Algorithmic Applications

3. Quantum Walk and Its Algorithmic Applications
    1. How important for computation
    2. Intuitive difference between random and quantum walks
    3. Algorithmic Applications
        1. Spatial Search
        2. Element Distinctness

4. （Some of my research topics) if time permits

What are (quantum / random) walks?

- A random walk is the simulation of the random movement of a particle around a graph
- A quantum walk is the same – but with a quantum particle
  - not the same as running a normal random walk algorithm on a quantum computer
- Random walks are a useful model for developing classical algorithms; quantum walks provide a new way of developing quantum algorithms
  - which is particularly important because producing new quantum algorithms is so hard

Example: Random walk for 2-SAT (1/3)

Input: Boolean formula (conjunction of clauses of 2 variables)

$$f(X) = (x_1 \vee x'_2) \wedge (x'_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x'_1 \vee x'_3)$$

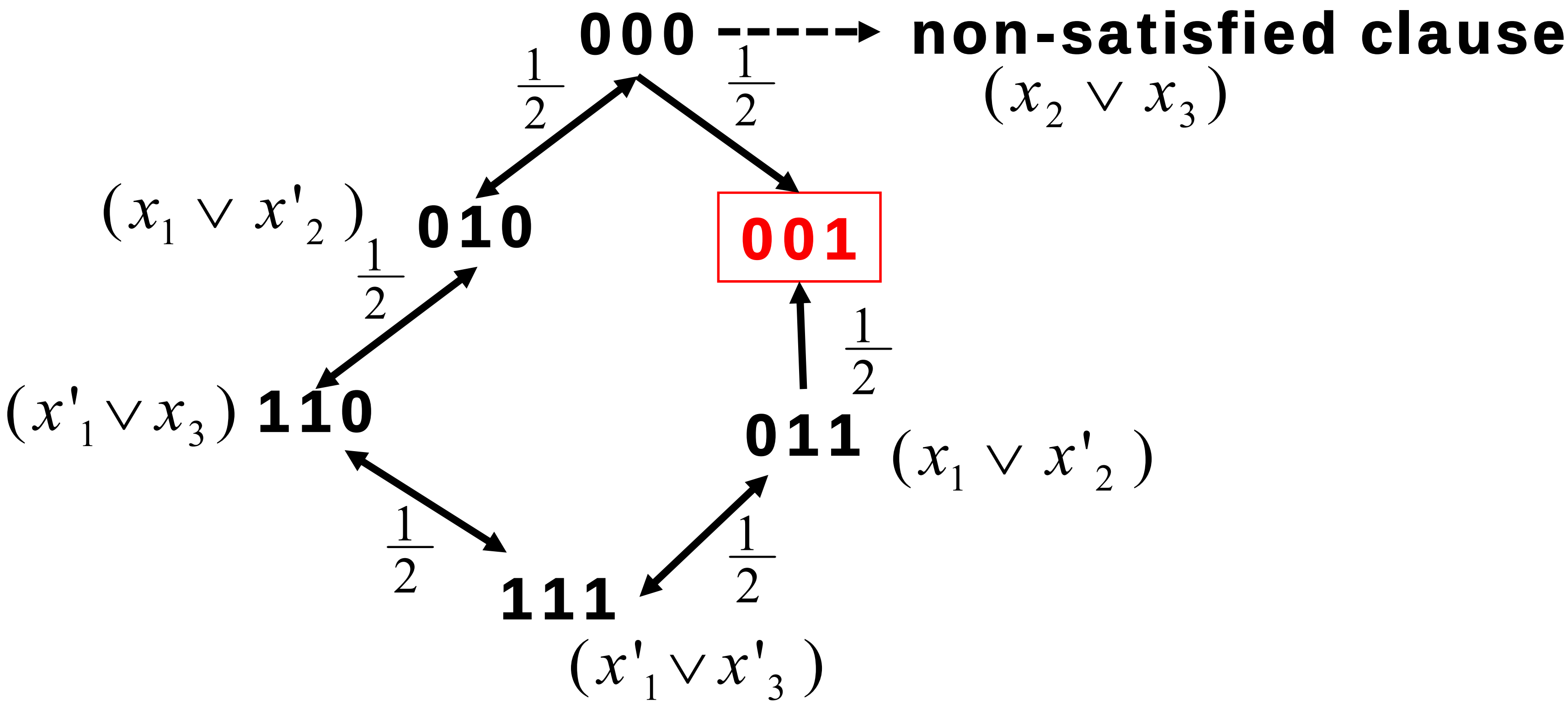
Question: Is the formula satisfaisable?
(ex. YES, 001 is satisfying assignment)

Algorithm:

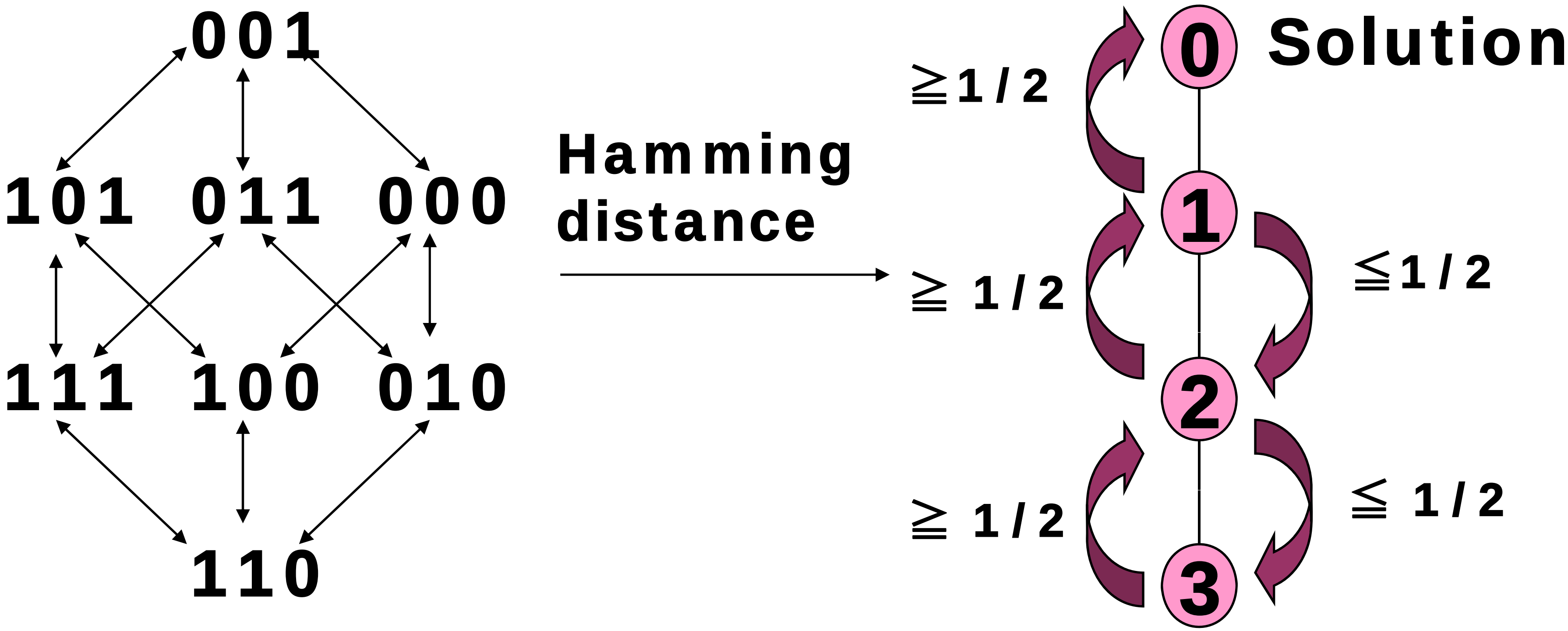
1) initialise the variables u.a. random (ex. 000)
2) if all clauses satisfied – STOP, otherwise:
3) chose a non-satisfied clause, chose one of its two variables and flip its value; return to 2)

Example: Random walk for 2-SAT (2/3)

$$f(X) = (x_1 \vee x'_2) \wedge (x'_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x'_1 \vee x'_3)$$



Example: Random walk for 2-SAT (3/3)



Random walk on a line with n+1 vertices !

After $t=2n^2$ repetitions, the succes probability is >1/2 (if the formula is satisfiable).

Random walk algorithm for 3-SAT

- Schöning developed (1999) a simple randomised algorithm for 3-SAT:
  - start with a random assignment to all variables
  - find which clauses are not satisfied by the assignment
  - flip one of the variables which appears in that clause
  - repeat until satisfying assignment found (or 3n steps have elapsed)

- This simple algorithm has worst-case time complexity of $O(1.34^n)$
  - Still exponential, but better than G. S.?

Outline

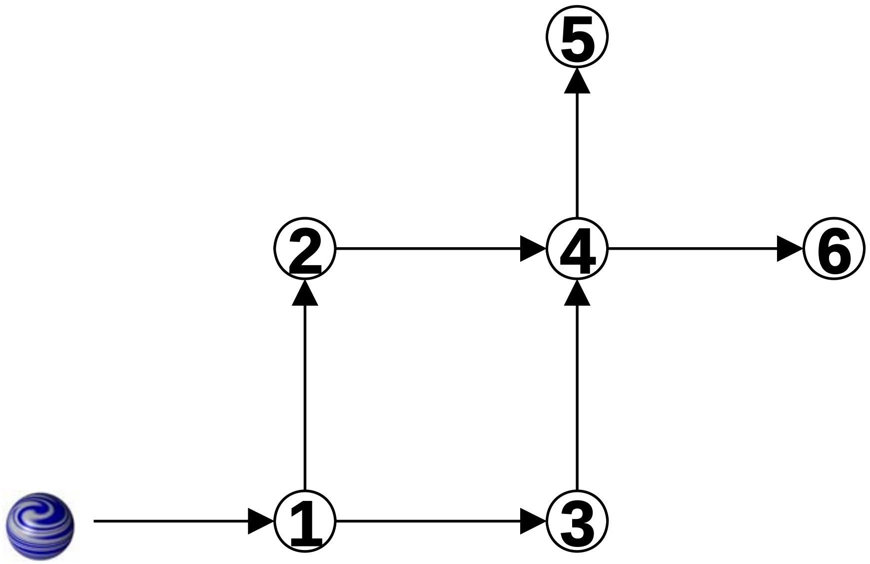
1. What is query complexity？

2. Amplitude Amplification and Its Algorithmic Applications

3. Quantum Walk and Its Algorithmic Applications
    1. How important for computation
    2. Intuitive difference between random and quantum walks
    3. Algorithmic Applications
        1. Spatial Search
        2. Element Distinctness

4. （Some of my research topics) if time permits

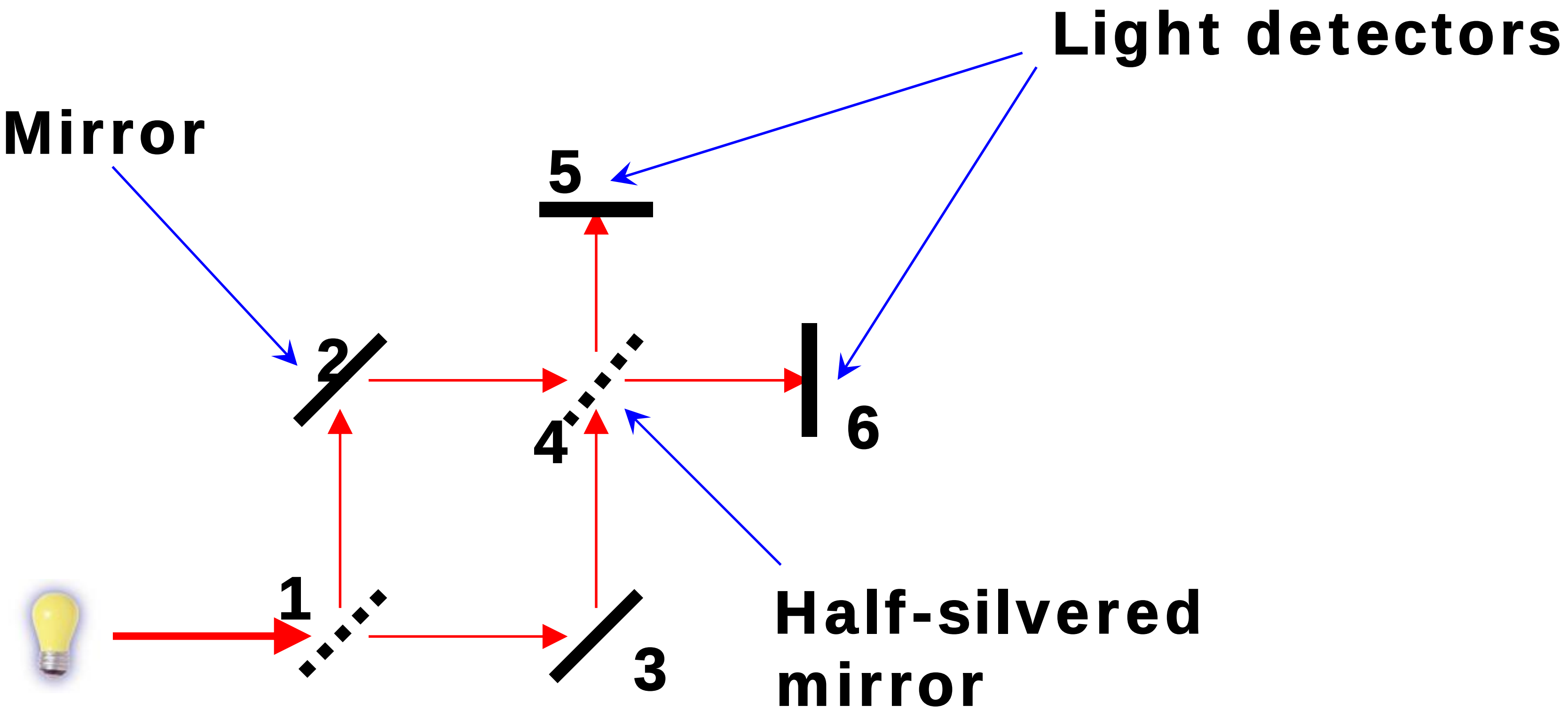
Physical intuition behind a classical random walk on a graph



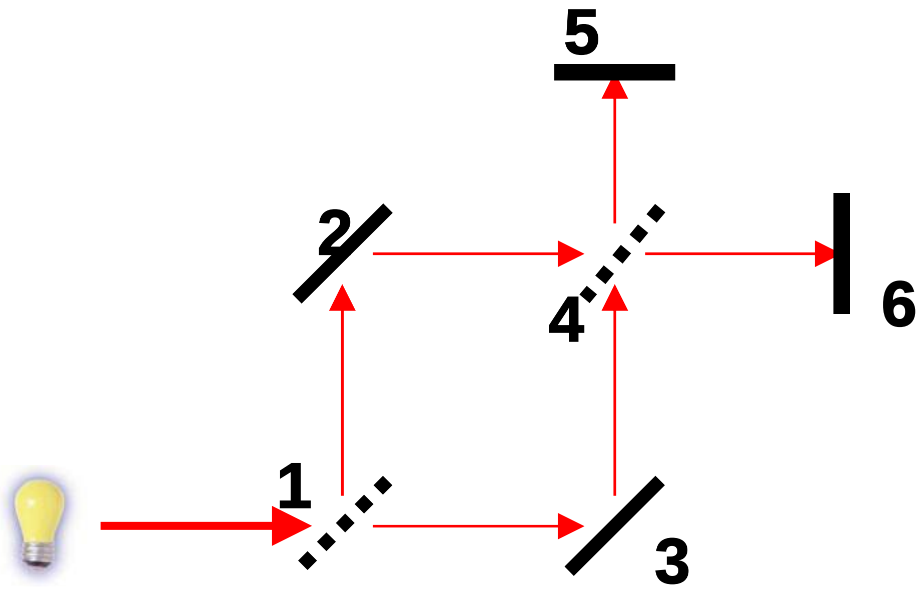
Time	Probability at vertex					
	1	2	3	4	5	6
0	1					
1		½	½			
2						
3				**1**	½	½

- **After 3 steps we are in position "5" or "6" with equal probability.**

Physical intuition behind a quantum walk on a graph



Physical intuition behind a quantum walk on a graph

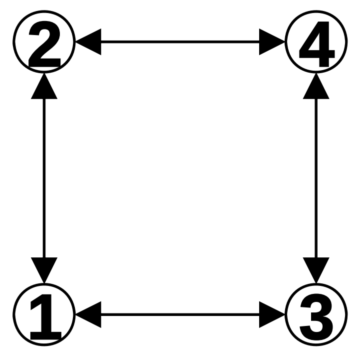


Time	Amplitude at point					
	1	2	3	4	5	6
0	1					
1		$\frac{i}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$			
2				$\frac{i+1}{\sqrt{2}}$		
3						**1**

- **After 3 steps we are guaranteed to be in detector "6" – this is caused by *quantum interference.***

Mathematical definition of a random walk

- Express a classical random walk as a matrix W of transition probabilities
  - where the entries in each column sum to 1
- Express a position as a column vector ***v***
- Performing a step of the walk corresponds to pre-multiplying ***v*** by W
- Performing n steps of the walk corresponds to pre-multiplying ***v*** by W^n



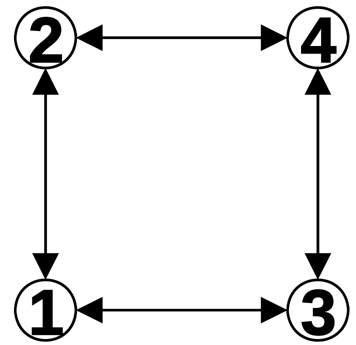
$$= \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Shigeru Yamashita ger@cs.ritsumei.ac.jp

Mathematical definition of a quantum walk

- Very similar, but:
  - probabilities combine differently (sum of the amplitudes squared must be 1)
  - the transition matrix must be *unitary* (ie. send unit vectors to unit vectors)

- This will not in general be the case, so we may need to modify the structure of the graph – for example, by adding a *coin space*

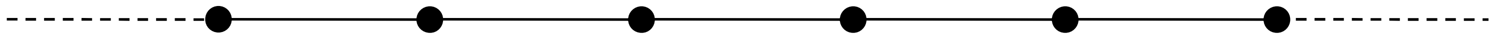


= (e.g.)

$$\begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{pmatrix}$$

Classical random walk on the line

- **Consider a walk on the following simple infinite graph:**



- Useful models for many random processes

- When the walker has equal probability to move left or right, average distance from the start position after time n is $\sqrt{n}$

- This is not reversible, so we cannot simply do this in quantumly.

Quantum walk on the line

- We have two quantum registers:
  - a coin register holding $|L\rangle$ or $|R\rangle$
  - a position register $|p\rangle$

- One step of the walk
  - coin flip: $|L\rangle \to |L\rangle + \mathbf{i}|R\rangle$, $|R\rangle \to \mathbf{i}|L\rangle + |R\rangle$

$$\boxed{\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}}$$

  - shift: $|L\rangle|p\rangle \to |L\rangle|p-1\rangle$, $|R\rangle|p\rangle \to |R\rangle|p+1\rangle$

unitary

- When the walker has equal probability to move left or right, average distance from the start position after time n is n

Any unitary operator can be used. E. g., $\boxed{\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}}$

-

Quantum walk on the line

$$|L\rangle \to |L\rangle + \mathbf{i}|R\rangle$$
$$|R\rangle \to \mathbf{i}|L\rangle + |R\rangle$$

0. start $\to |R\rangle|0\rangle$
1. coin $\to (\mathbf{i}|L\rangle + |R\rangle)|0\rangle$
   shift $\to \mathbf{i}|L\rangle|-1\rangle + |R\rangle|1\rangle$
2. coin $\to (\mathbf{i}|L\rangle - |R\rangle)|-1\rangle + (\mathbf{i}|L\rangle + |R\rangle)|1\rangle$
   shift $\to \mathbf{i}|L\rangle|-2\rangle - |R\rangle|0\rangle + \mathbf{i}|L\rangle|0\rangle + |R\rangle|2\rangle$
3. coin $\to (\mathbf{i}|L\rangle - |R\rangle)|-2\rangle - |R\rangle|0\rangle + (\mathbf{i}|L\rangle + |R\rangle)|2\rangle$
   shift $\to \mathbf{i}|L\rangle|-3\rangle - |R\rangle|-1\rangle - |R\rangle|1\rangle + \mathbf{i}|L\rangle|1\rangle + |R\rangle|3\rangle$

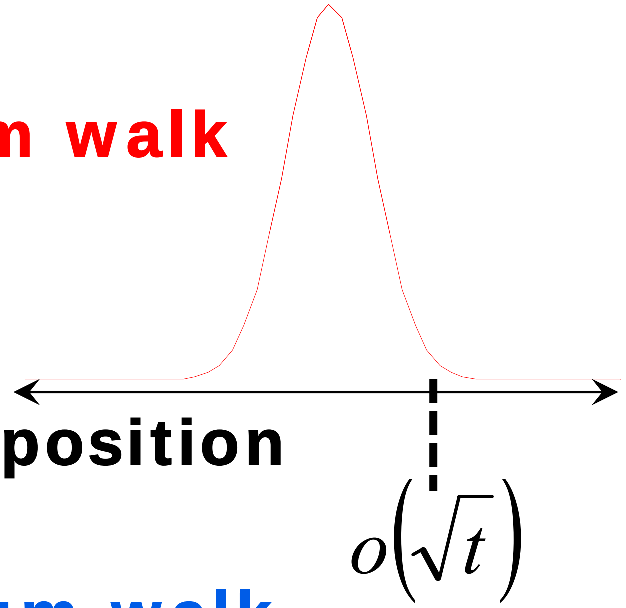
Time	Probability at vertex						
	-3	-2	-1	0	1	2	3
0				1			
1			$\frac{1}{2}$		$\frac{1}{2}$		
2		$\frac{1}{4}$		$\frac{1}{2}$		$\frac{1}{4}$	
3	$\frac{1}{8}$		$\frac{1}{8}$		$\frac{5}{8}$		$\frac{1}{8}$

Random walk on the line

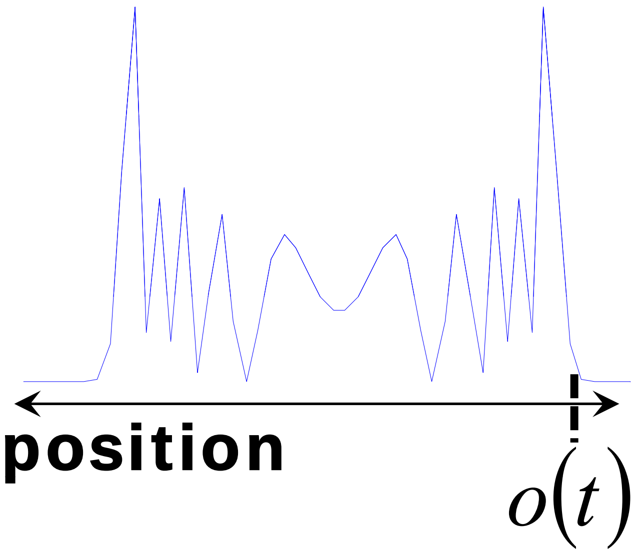
Time	Probability at vertex						
	-3	-2	-1	0	1	2	3
0				1			
1			$\frac{1}{2}$		$\frac{1}{2}$		
2		$\frac{1}{4}$		$\frac{1}{2}$		$\frac{1}{4}$	
3	$\frac{1}{8}$		$\frac{3}{8}$		$\frac{3}{8}$		$\frac{1}{8}$

Random vs. quantum walk on the line

Probability distribution by random walk after t steps



Probability distribution by quantum walk after t steps



The peaks and troughs in this graph are caused by quantum interference.

Outline

1. What is query complexity？

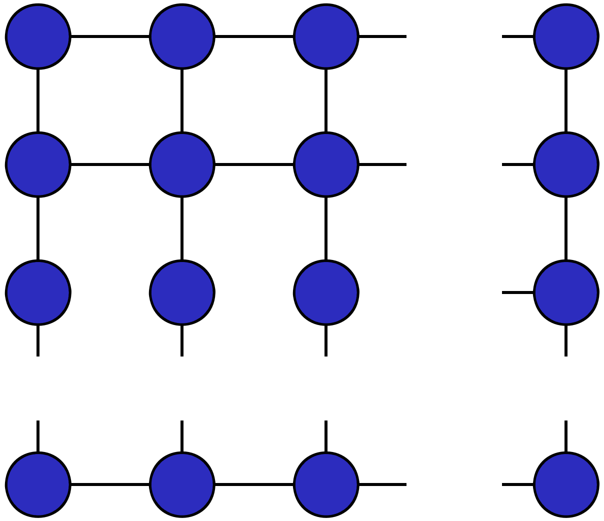
2. Amplitude Amplification and Its Algorithmic Applications

3. Quantum Walk and Its Algorithmic Applications
   1. How important for computation
   2. Intuitive difference between random and quantum walks
   3. Algorithmic Applications
      1. Spatial Search
      2. Element Distinctness

4. （Some of my research topics) if time permits

Quantum search on grids

$$\sqrt{N}\times\sqrt{N}=N$$ **nodes**



- Find a marked node
- Grover's algorithm takes $O(\sqrt{N})\times O(\sqrt{N})=O(N)$ steps.

No quantum speedup.

Shigeru Yamashita ger@cs.ritsumei.ac.jp

Quantum walk on grid

- Basis states

  $|x,y,\leftarrow\rangle$, $|x,y,\rightarrow\rangle$, $|x,y,\uparrow\rangle$, $|x,y,\downarrow\rangle$.

- Coin flip on direction:

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

- Shift:
  - $|x,y,\leftarrow\rangle \Rightarrow |x-1,y,\rightarrow\rangle$
  - $|x,y,\rightarrow\rangle \Rightarrow |x+1,y,\leftarrow\rangle$
  - $|x,y,\uparrow\rangle \Rightarrow |x,y-1,\downarrow\rangle$
  - $|x,y,\downarrow\rangle \Rightarrow |x,y+1,\uparrow\rangle$

Search by quantum walk

- Perform a quantum walk with “coin flip”:
  - C in unmarked locations;
  - -I in marked locations.
- After $O\left(\sqrt{N \log N}\right)$ steps, measure the state.
- Gives marked $|x, y, d\rangle$ with prob. 1/log N

- By using A. A., total cost becomes $O\left(\sqrt{N}\log N\right)$

Outline

1. What is query complexity？

2. Amplitude Amplification and Its Algorithmic Applications

3. Quantum Walk and Its Algorithmic Applications
    1. How important for computation
    2. Intuitive difference between random and quantum walks
    3. Algorithmic Applications
        1. Spatial Search
        2. Element Distinctness

4. （Some of my research topics) if time permits

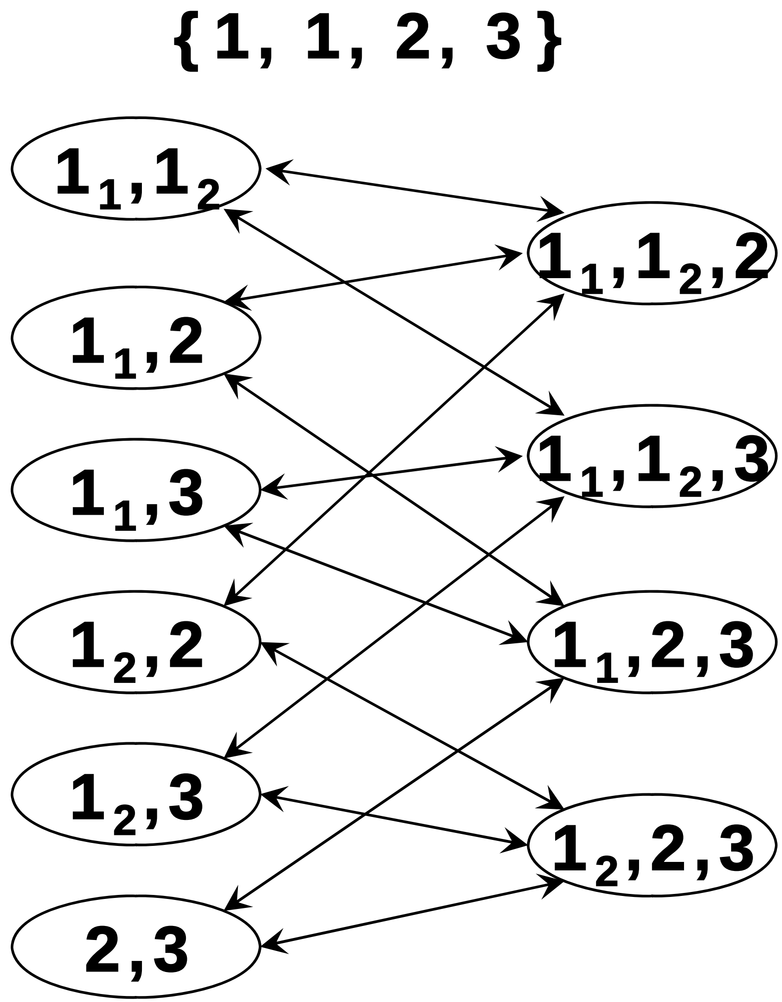
Element distinctness

7	**9**	**2**	**…**	**1**
$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$		$\mathbf{x_N}$

- Numbers x_1, x_2, ..., $x_{N.}$
- Determine if two of them are equal.
- Well studied problem in classical CS.
- Classically: N steps.
- Quantumly, $O(N^{2/3})$ steps.

Quantum walk algorithm for E. D. (1 / 2)

- We use a quantum walk on a graph where the vertices are subsets of **S** containing either M or M + 1 elements for some M < N
- Two vertices are connected if they differ in exactly one element
- The graph on the right encodes the set {1, 1, 2, 3} for M = 2



Quantum walk algorithm for E. D. (2/2)

- Basic walk algorithm:
  1. start with some subset **S'** $\subseteq$ **S** (where |**S'**| = M)
  2. check whether **S'** contains any duplicates (needs O(M) queries)
  3. if not, change to a different subset **S''** that differs in exactly one element
  4. check **S''** for duplicates (needs 1 query)
  5. repeat steps 3 and 4 until a duplicate is found

- Because this is a quantum walk, we can start with a superposition of all M-subsets

Analysis of the quantum walk

- In total, we need (M + r) queries, where
  - M is the number of elements in the initial subset
  - r is the number of steps of the quantum walk

- When $M = N^{2/3}$ and $r = N^{1/3}$, a solution can be found with high probability.
  Thus the query comp. = $O(N^{2/3})$

Summary

- Introduction
  - –Shigeru Yamashita
  - –Topics for C. S. people (Yamashita's Perspective)
    - NP, Query Complexity
- Amplitude Amplification and Its Applications
- Quantum Walk and Its Applications

***You can just Google to find papers concerning these topics.**