Title: Quantum Walks and algorithmic speedup

Date: Nov 24, 2004 02:00 PM

URL: http://pirsa.org/04110038

Abstract:

Quantum Walks and Algorithmic Speedup

Richard Cleve (Waterloo)

Joint work with: Andrew Childs (MIT)
Enrico Deotto (MIT)
Eddie Farhi (MIT)
Sam Gutmann (Northeastern)
Dan Spielman (MIT)

Quantum Walks and Algorithmic Speedup

Richard Cleve (Waterloo)

Joint work with: Andrew Childs (MIT)
Enrico Deotto (MIT)
Eddie Farhi (MIT)
Sam Gutmann (Northeastern)
Dan Spielman (MIT)

Motivation

Find new quantum algorithms

Motivation

Find new quantum algorithms

Shor's algorithms:
based on the ***Quantum Fourier Transform***

[Shor '94]

Motivation

Find new quantum algorithms

Shor's algorithms:
based on the ***Quantum Fourier Transform***

Grover's algorithm:
based on ***Amplitude Amplification***

[Shor '94][Grover '96]

Motivation

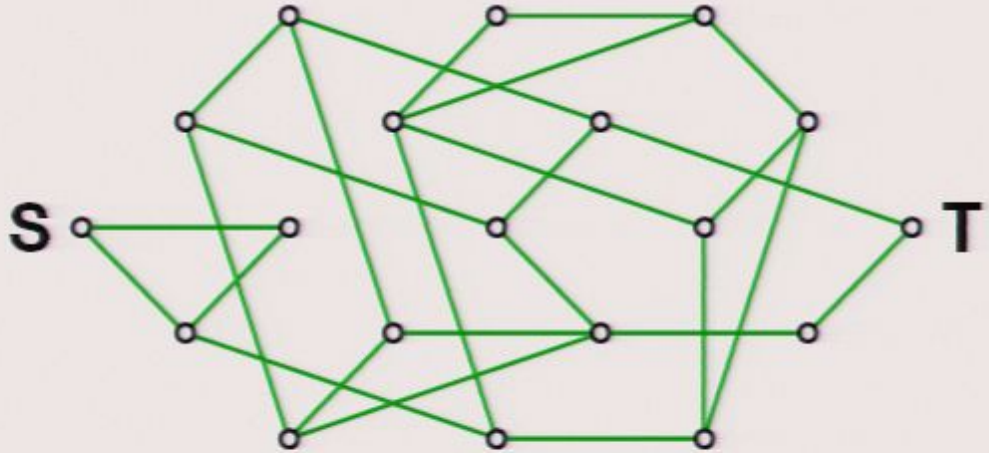
Find new quantum algorithms

Shor's algorithms:
based on the ***Quantum Fourier Transform***

Grover's algorithm:
based on ***Amplitude Amplification***

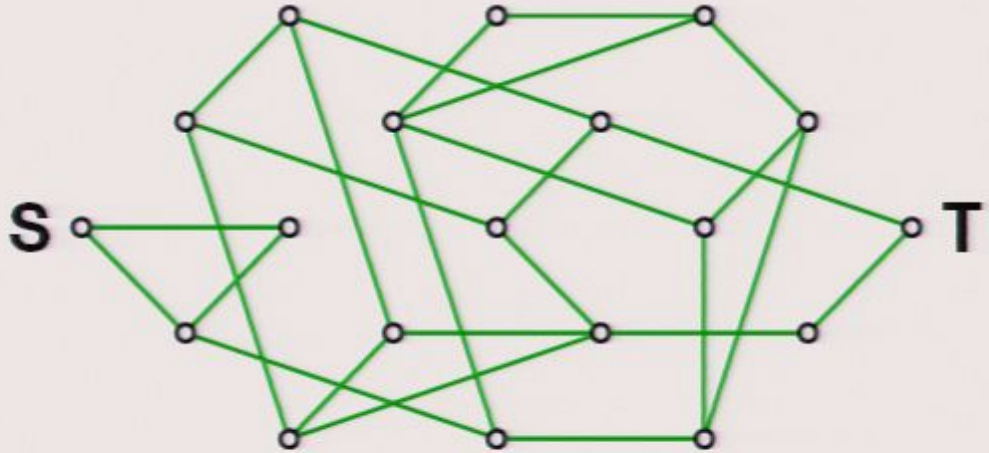
Are there other useful algorithmic techniques?

Classical random walks



Algorithmic tools for searching and sampling

Classical random walks

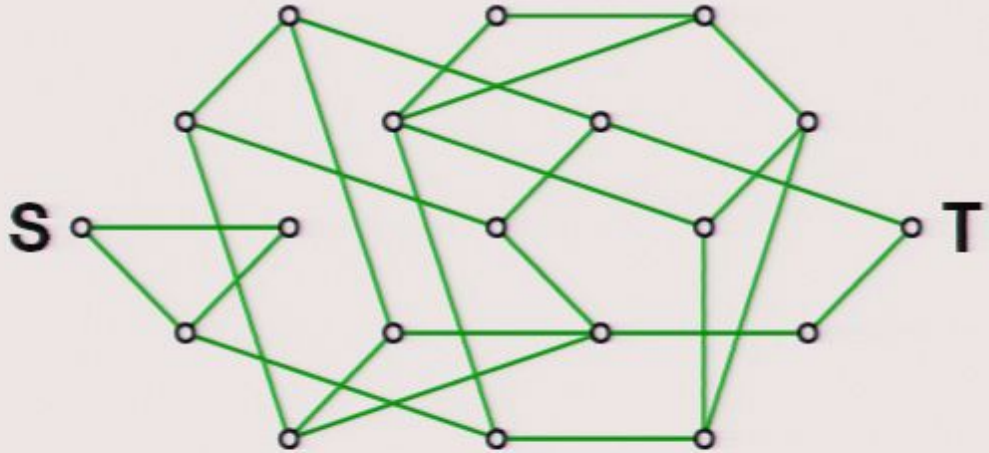


Algorithmic tools for searching and sampling

Example 1: ST-connectivity (undirected graphs)

[Aleliunas, Karp, Lipton, Lovász & Rackoff '79]

Classical random walks



Algorithmic tools for searching and sampling

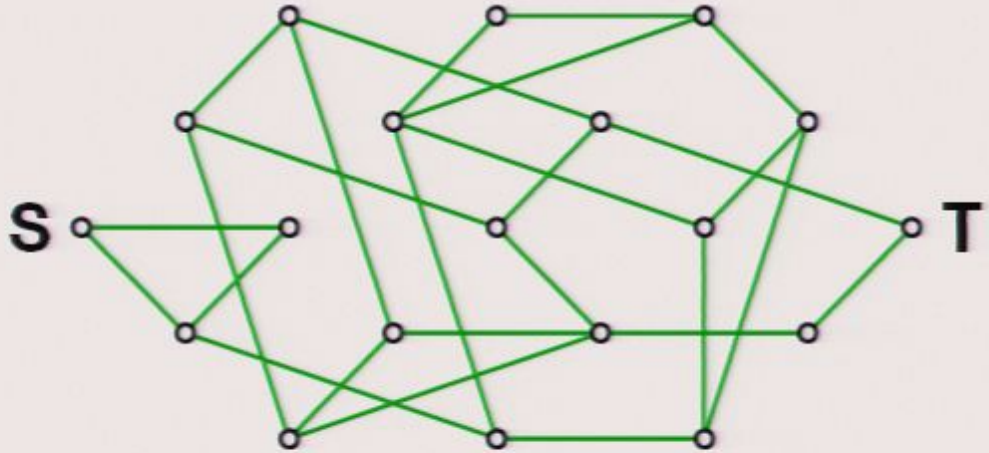
Example 1: ST-connectivity (undirected graphs)

A very space efficient algorithm:

1. perform a random walk starting from **S**
2. if **T** is ever reached then **YES**, else **NO**

[Aleliunas, Karp, Lipton, Lovász & Rackoff '79]

Classical random walks



Algorithmic tools for searching and sampling

Example 1: ST-connectivity (undirected graphs)

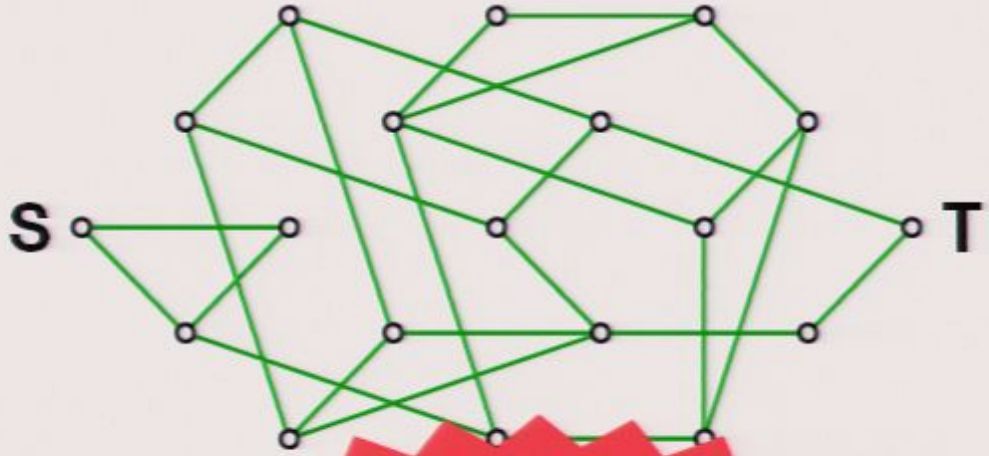
A very space efficient algorithm:

1. perform a random walk starting from **S**
2. if **T** is ever reached then **YES**, else **NO**

Space $O(\log n)$ and time $O(n^3)$ for ***any*** n-vertex graph,

[Aleliunas, Karp, Lipton, Lovász & Rackoff '79] 3

Classical random walks



Algorithmic tools fo

Example 1: ST-

A very space effi

1. perform a rando
2. if **T** is ever reached

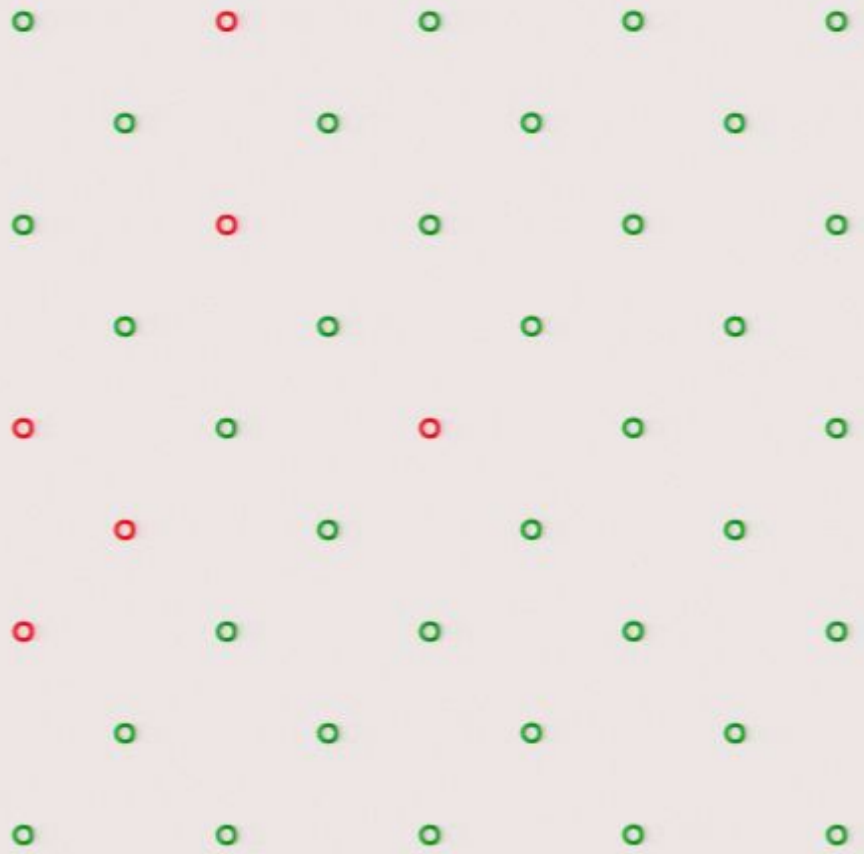
November 8, 2004:
deterministic algorithm
using similar space/time

Space $O(\log n)$ and time $O(n^3)$ for ***any*** n-vertex graph,

[Aleliunas, Karp, Lipton, Lovász & Rackoff '79] [Reingold '04]

Random walks as algorithmic tools

Example 2: uniform samping on a subset

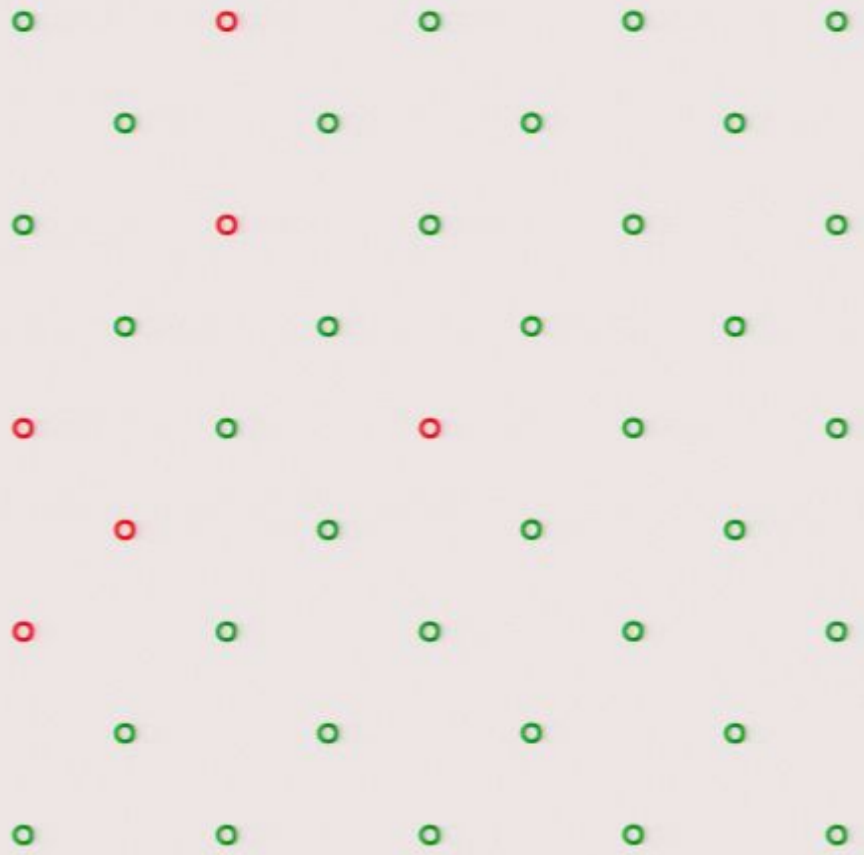


[Broder '86][Jerrum & Sinclair '89][L-S & Vigoda '01] 4

Random walks as algorithmic tools

Example 2: uniform samping on a subset

Execute random walk on a related graph that quickly approaches uniform distribution



[Broder '86][Jerrum & Sinclair '89][L-S & Vigoda '01] 4

Random walks as algorithmic tools

Example 2: uniform samping on a subset

Execute random walk on a related graph that quickly approaches uniform distribution

[Broder '86][Jerrum & Sinclair '89][L. S. & Vigoda '01] 4

Random walks as algorithmic tools

Example 2: uniform samping on a subset

Execute random walk on a related graph that quickly approaches uniform distribution

Basis of some very good approximation algorithms

[Broder '86][Jerrum & Sinclair '89][L-S & Vigoda '01] 4

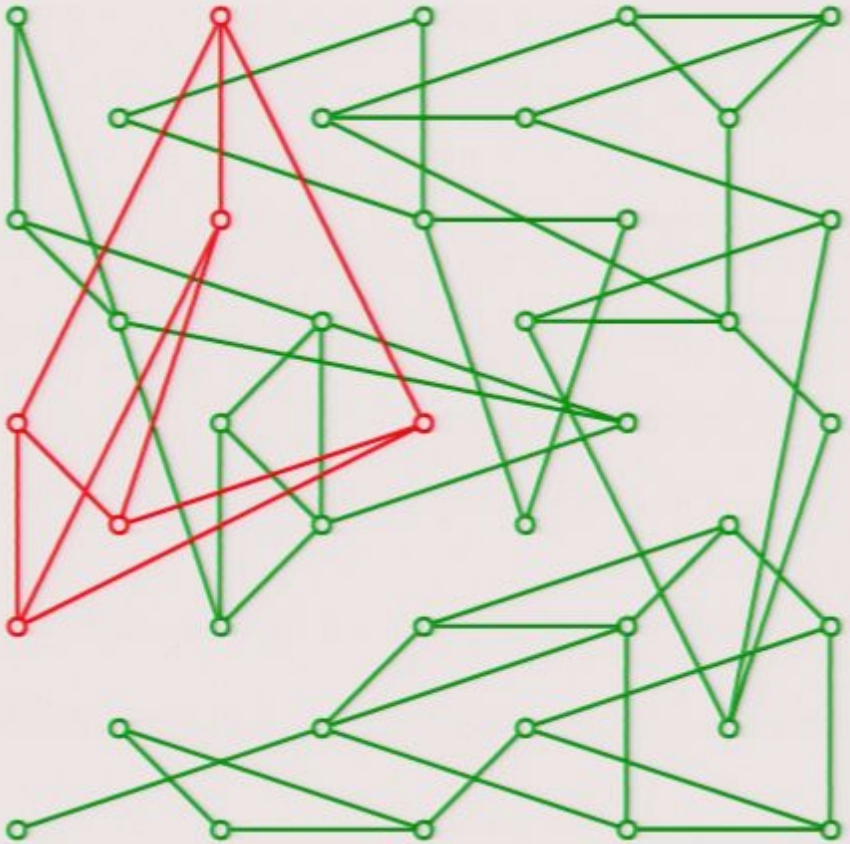
Random walks as algorithmic tools

Example 2: uniform samping on a subset

Execute random walk on a related graph that quickly approaches uniform distribution

Basis of some very good approximation algorithms

E.g., approximating the ***permanent*** of a matrix



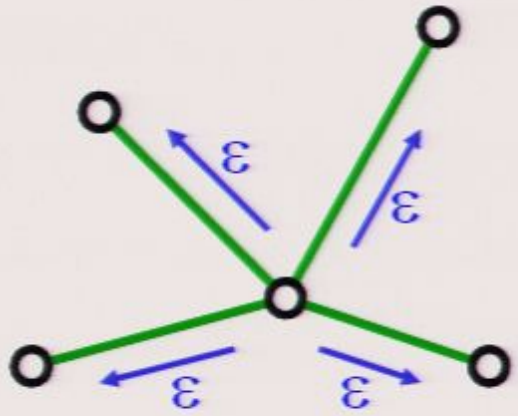
[Broder '86][Jerrum & Sinclair '89][J., S. & Vigoda '01] 4

Quantum walks on graphs

Quantum walks on graphs

Continuous-time *random* walk:

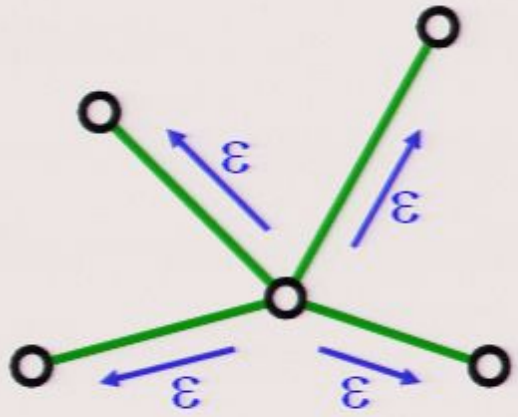
in each small ε time interval, move to each neighbor with probability ε (lim $\varepsilon \to 0$, like a Poisson process)



Quantum walks on graphs

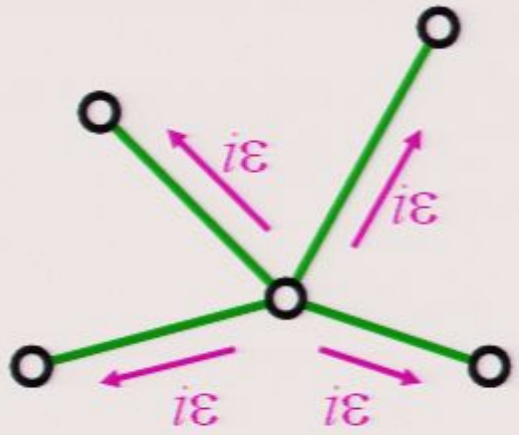
Continuous-time *random* walk:

in each small ε time interval, move to each neighbor with probability ε (lim $\varepsilon \to 0$, like a Poisson process)



Quantum analog:

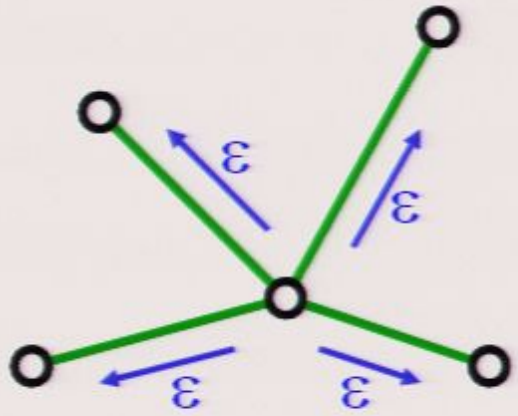
in ε time interval, move to neighbors with amplitude $i\varepsilon$ (lim $\varepsilon \to 0$)



Quantum walks on graphs

Continuous-time *random* walk:

in each small ε time interval, move to each neighbor with probability ε (lim $\varepsilon \to 0$, like a Poisson process)

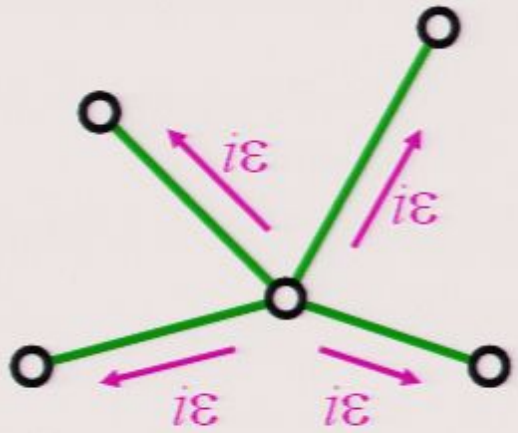


Quantum analog:

in ε time interval, move to neighbors with amplitude $i\varepsilon$ (lim $\varepsilon \to 0$)

Evolving for time t: e^{-iAt} |initial state⟩

where $A_{xy} = \begin{cases} 1 & \text{if } (x,y) \text{ edge} \\ 0 & \text{otherwise} \end{cases}$

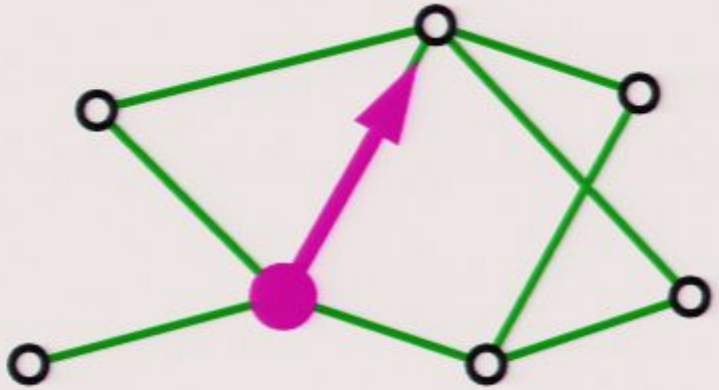


Quantum walks on graphs

***Discrete-time* version:**

Acts on extended Hilbert space

$|\text{vertex}\rangle \otimes |\text{orientation}\rangle$



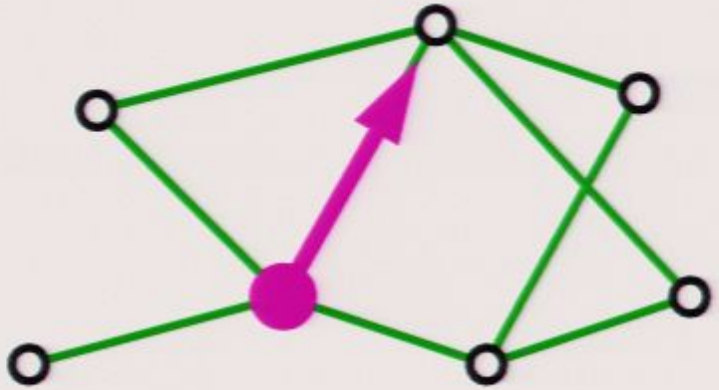
[Aharonov, Ambainis, Kempe & Vazirani '01][A. Bach, Nayak, Vishwanath & Wotters '01] 6

Quantum walks on graphs

***Discrete-time* version:**

Acts on extended Hilbert space

$|\textbf{vertex}\rangle \otimes |\textbf{orientation}\rangle$



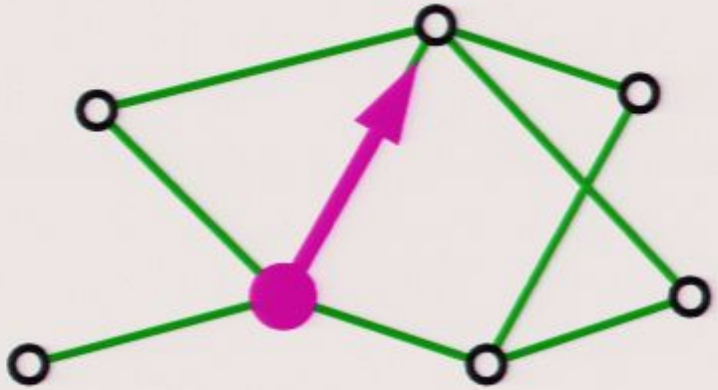
Quantum walk on unbounded line

Quantum walks on graphs

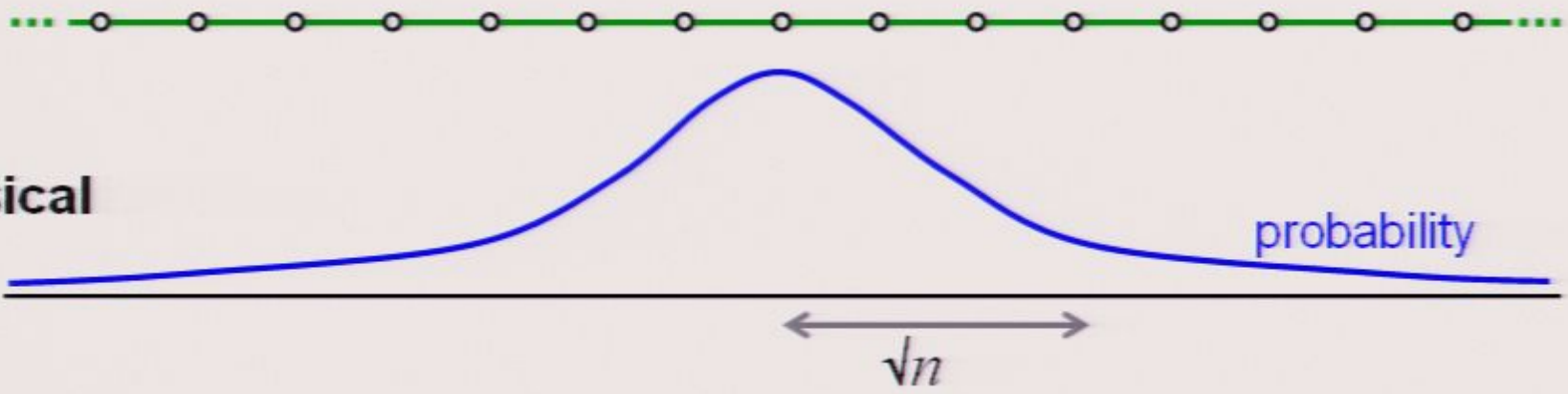
***Discrete-time* version:**

Acts on extended Hilbert space

$|\text{vertex}\rangle \otimes |\text{orientation}\rangle$



Quantum walk on unbounded line

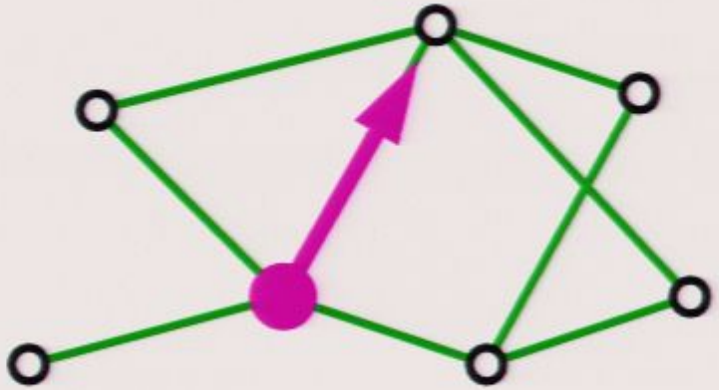


Quantum walks on graphs

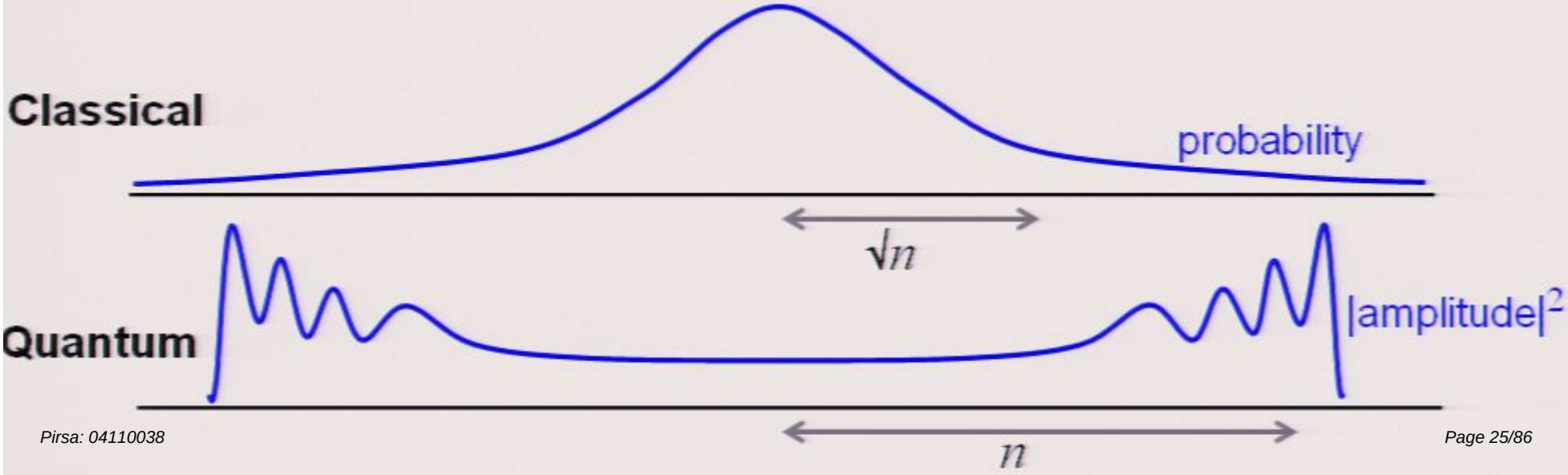
***Discrete-time* version:**

Acts on extended Hilbert space

$|\text{vertex}\rangle \otimes |\text{orientation}\rangle$

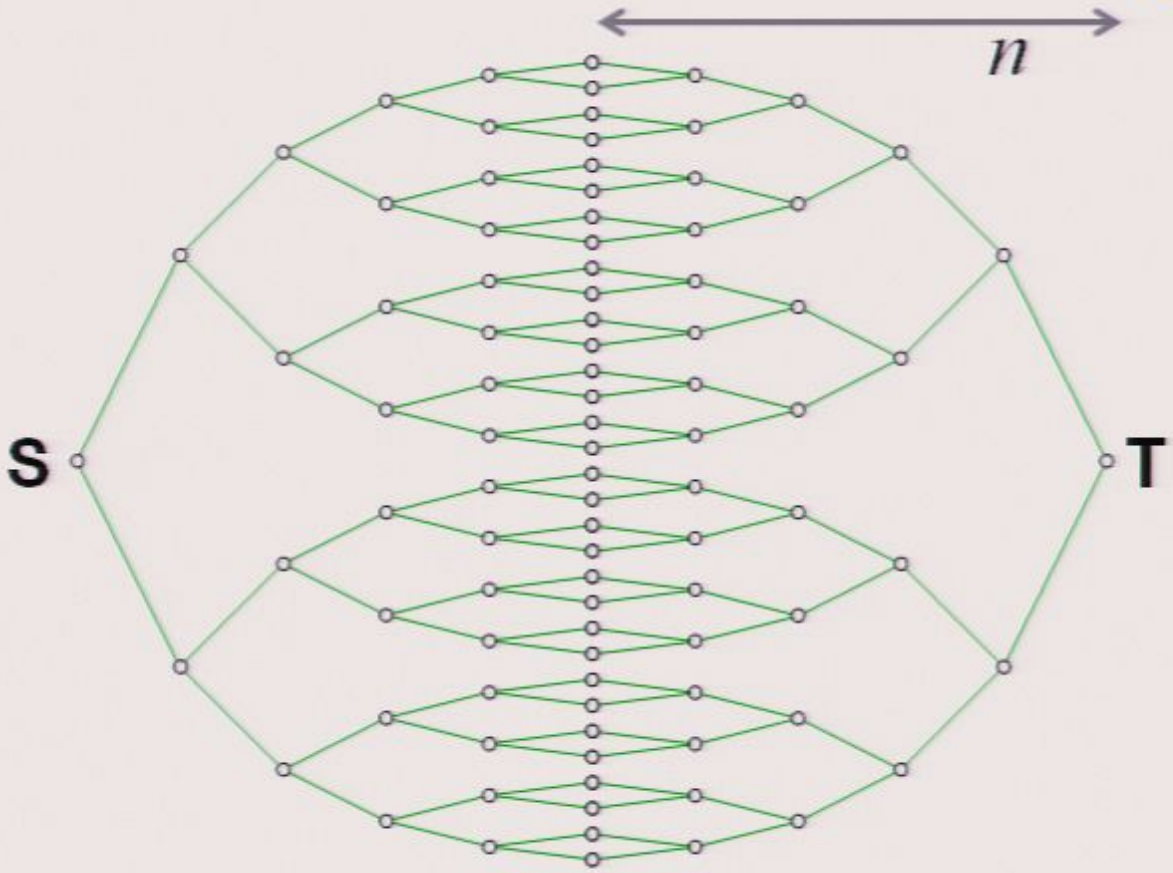


Quantum walk on unbounded line



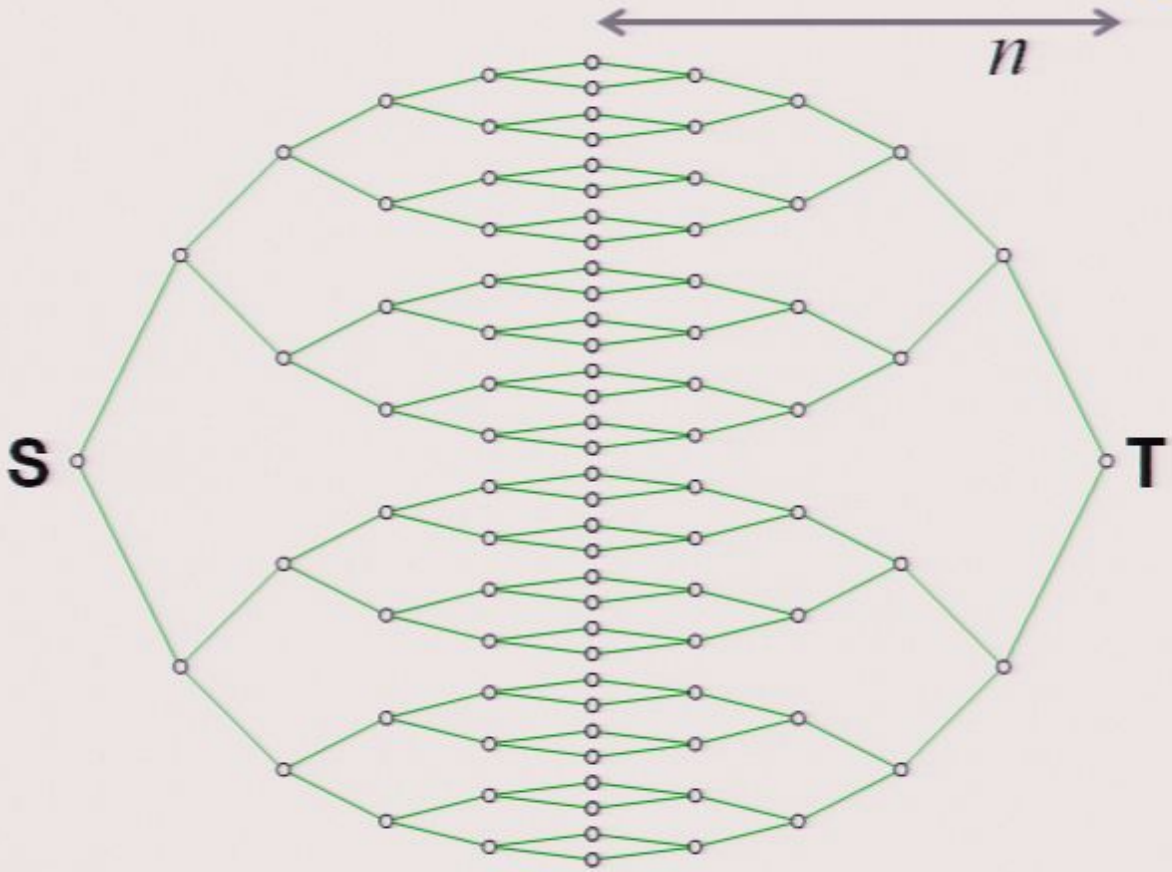
[Aharonov, Ambainis, Kempe & Vazirani '01][A. Bach, Nayak, Vishwanath & Wehner '01] 6

Quantum vs classical hitting times



[Childs, Farhi & Gutmann '01]

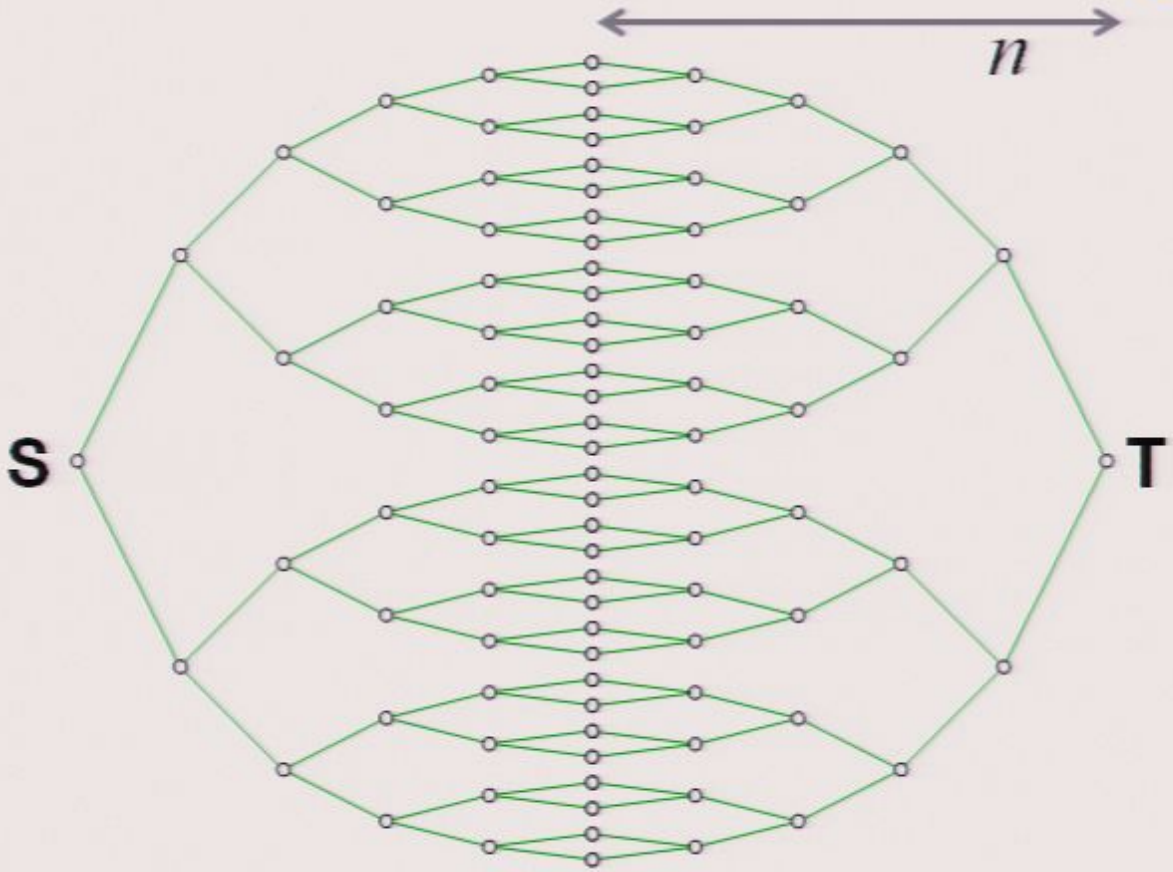
Quantum vs classical hitting times



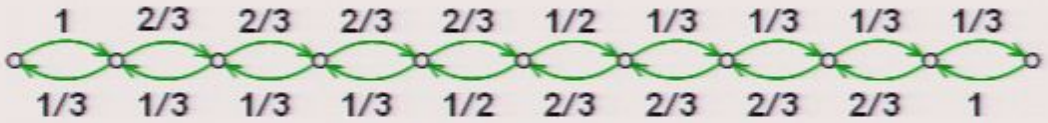
A ***random*** walk takes ***exponential*** time to get from **S** to **T**

[Childs, Farhi & Gutmann '01]

Quantum vs classical hitting times

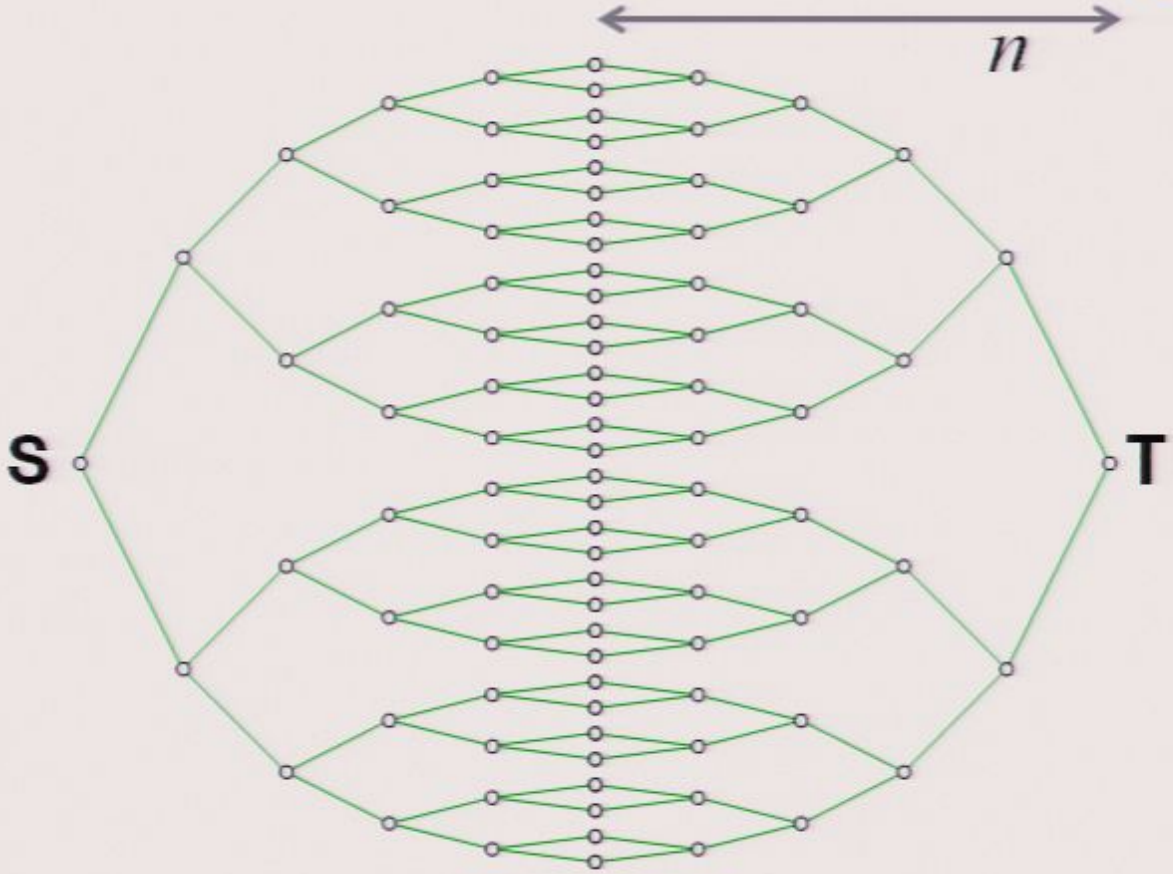


A ***random*** walk takes ***exponential*** time to get from **S** to **T**



[Childs, Farhi & Gutmann '01]

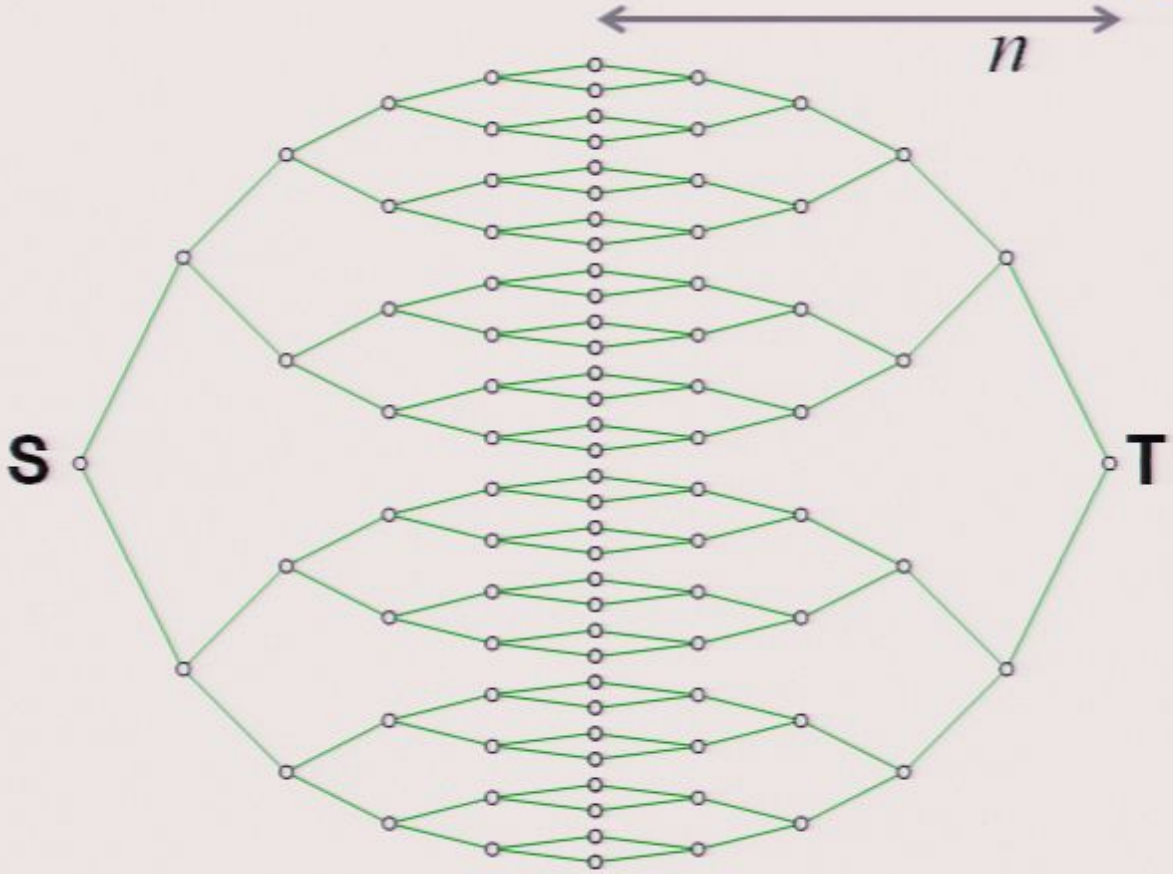
Quantum vs classical hitting times



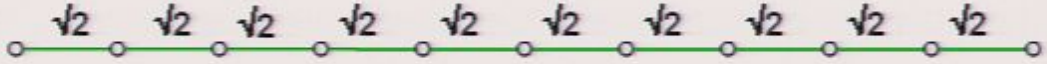
A ***quantum*** walk takes ***polynomial*** time to get from **s** to **t**

[Childs, Farhi & Gutmann '01]

Quantum vs classical hitting times

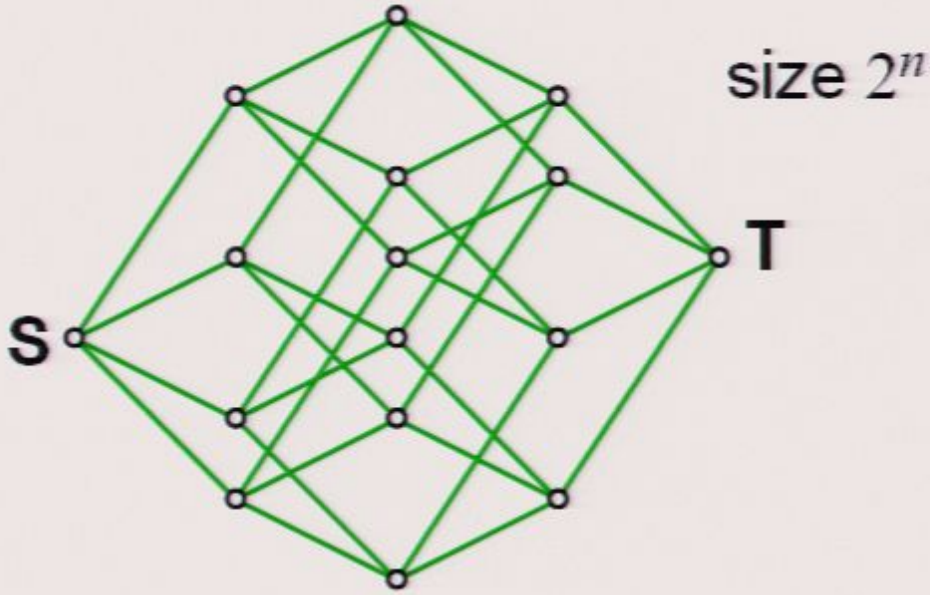


A ***quantum*** walk takes ***polynomial*** time to get from **S** to **T**

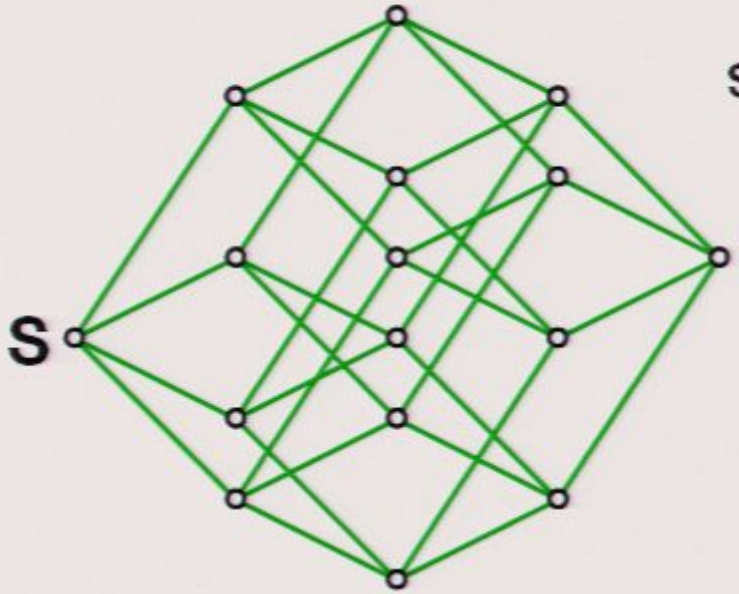


[Childs, Farhi & Gutmann '01]

n-dimensional hypercube

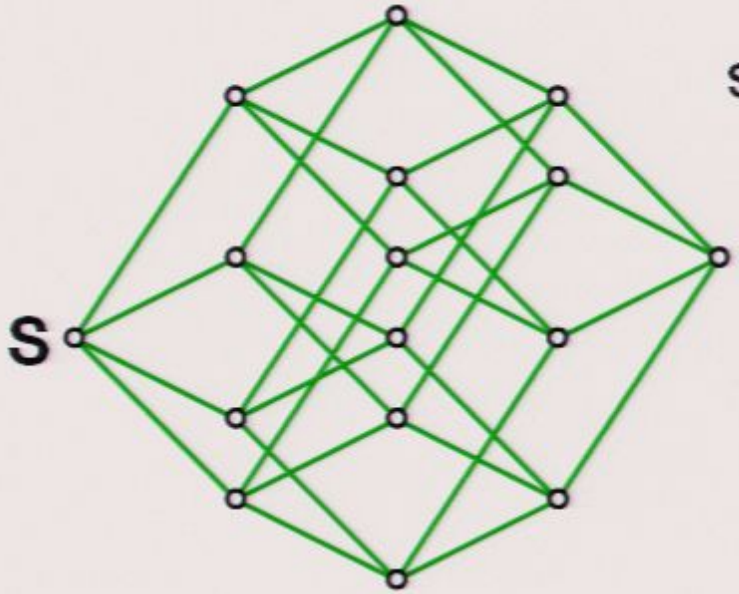


n-dimensional hypercube



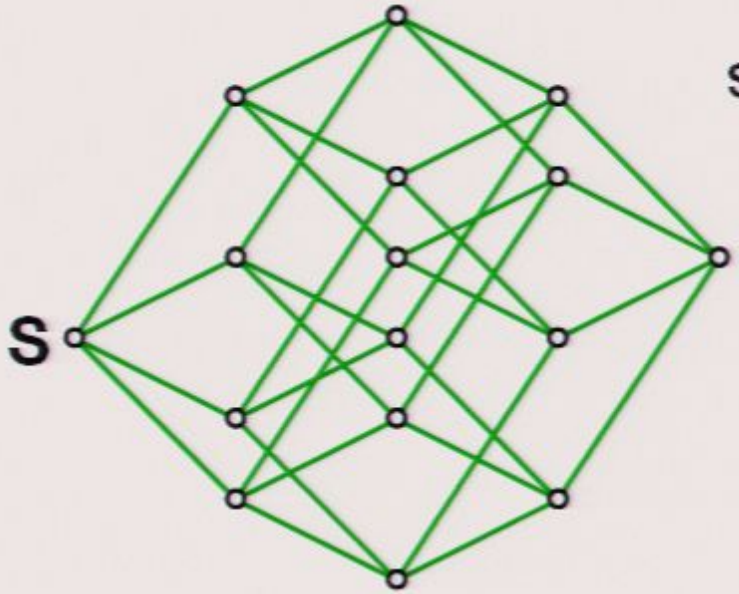
- Random walk takes ***exponential*** time to get from **S** to **T**

n-dimensional hypercube



- Random walk takes ***exponential*** time to get from **S** to **T**
- Quantum walk takes ***polynomial*** time to get from **S** to **T**

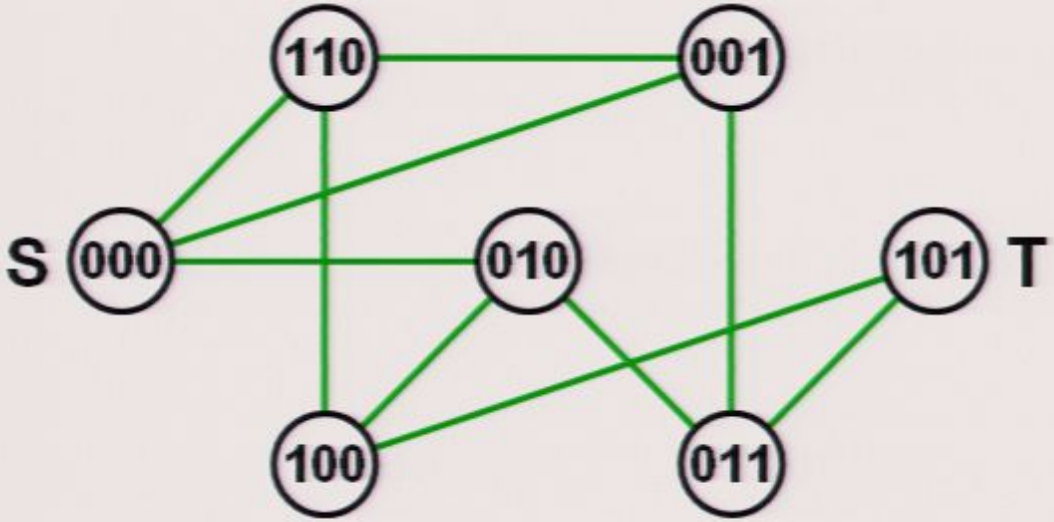
n-dimensional hypercube



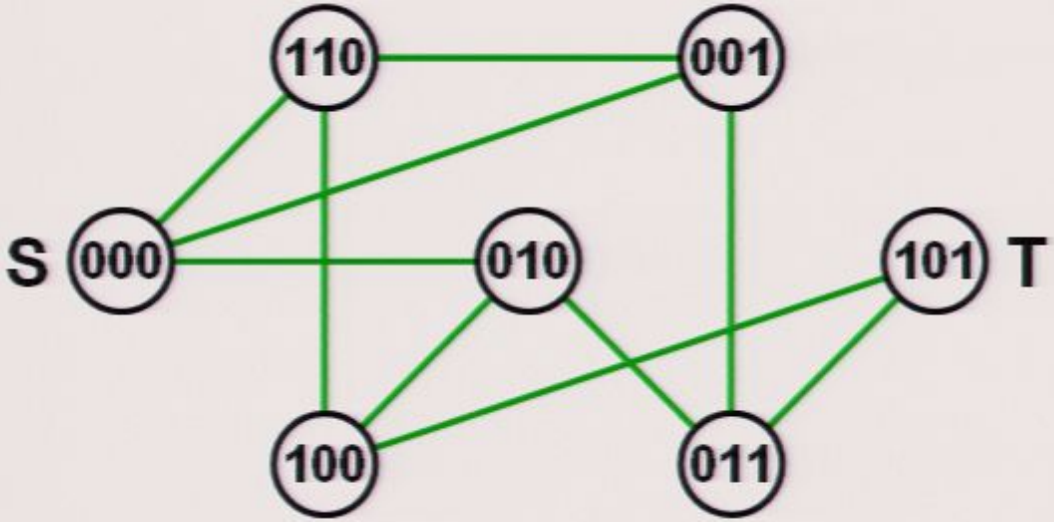
- Random walk takes ***exponential*** time to get from **S** to **T**
- Quantum walk takes ***polynomial*** time to get from **S** to **T**

ST-traversal problem: starting at node **S**, "find" node **T**

Representing graphs

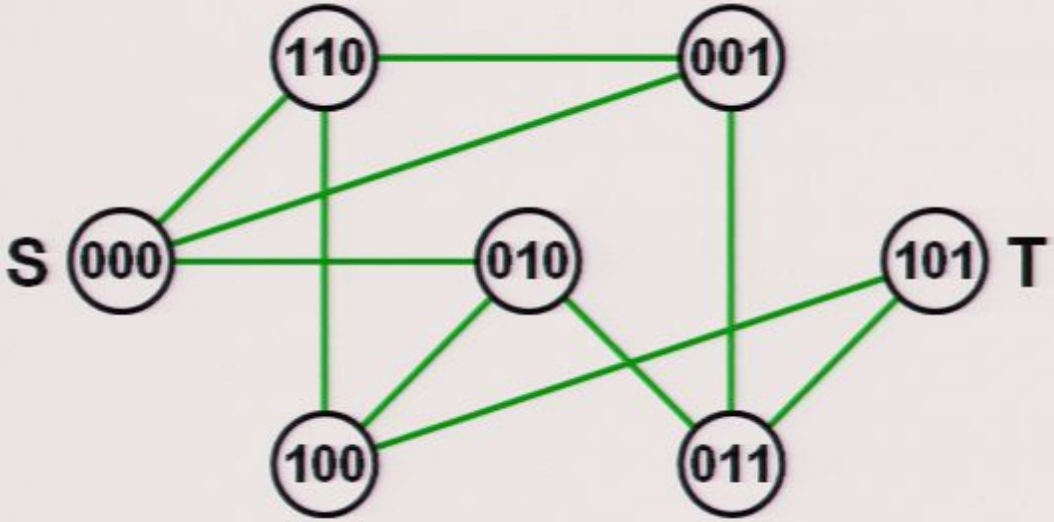


Representing graphs

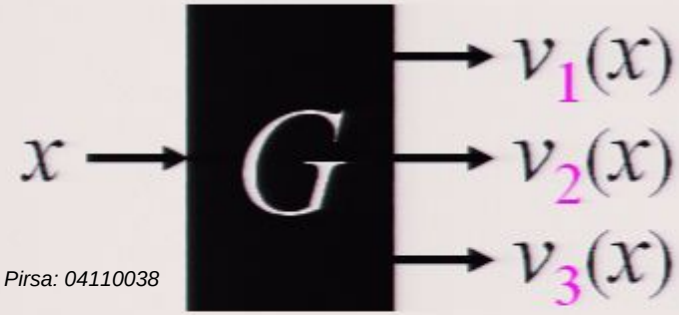


node	neighbors		
000	110	001	100
001	000	011	110
010	000	011	100
011	010	101	010
100	110	010	101
101	011	100	111
110	100	000	001
111	111	111	111

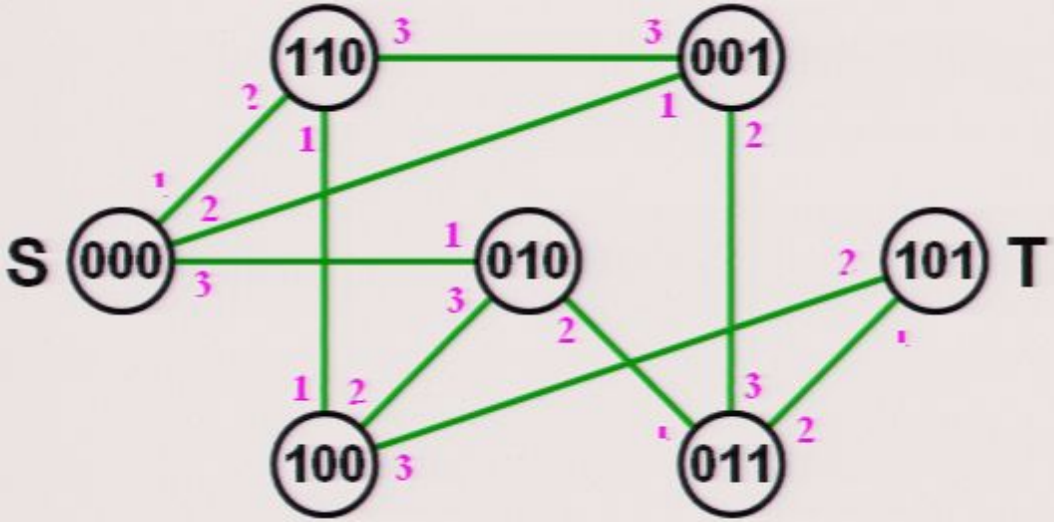
Representing graphs



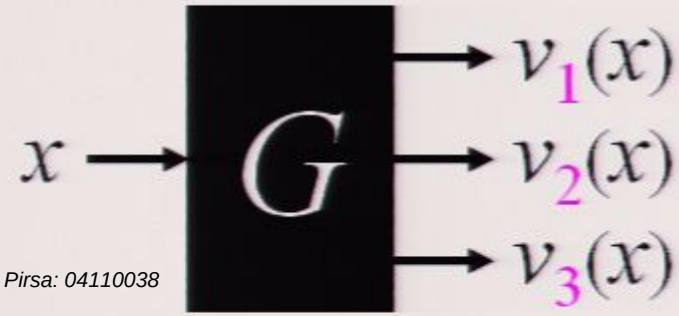
node	neighbors		
000	110	001	100
001	000	011	110
010	000	011	100
011	010	101	010
100	110	010	101
101	011	100	111
110	100	000	001
111	111	111	111



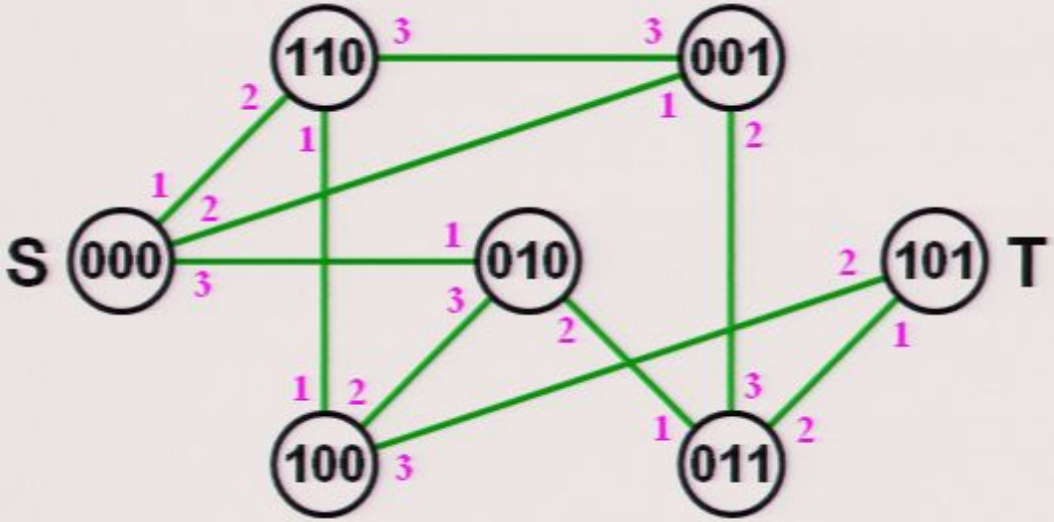
Representing graphs



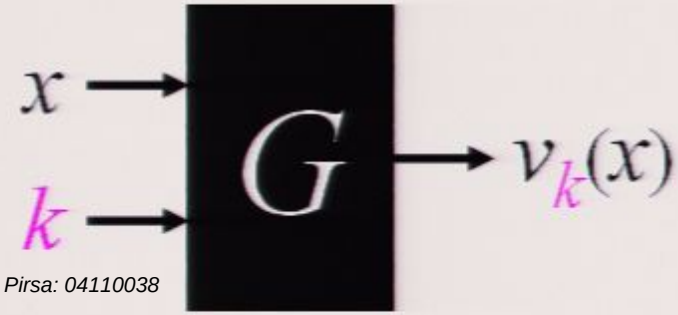
node	neighbors		
000	110	001	100
001	000	011	110
010	000	011	100
011	010	101	010
100	110	010	101
101	011	100	111
110	100	000	001
111	111	111	111



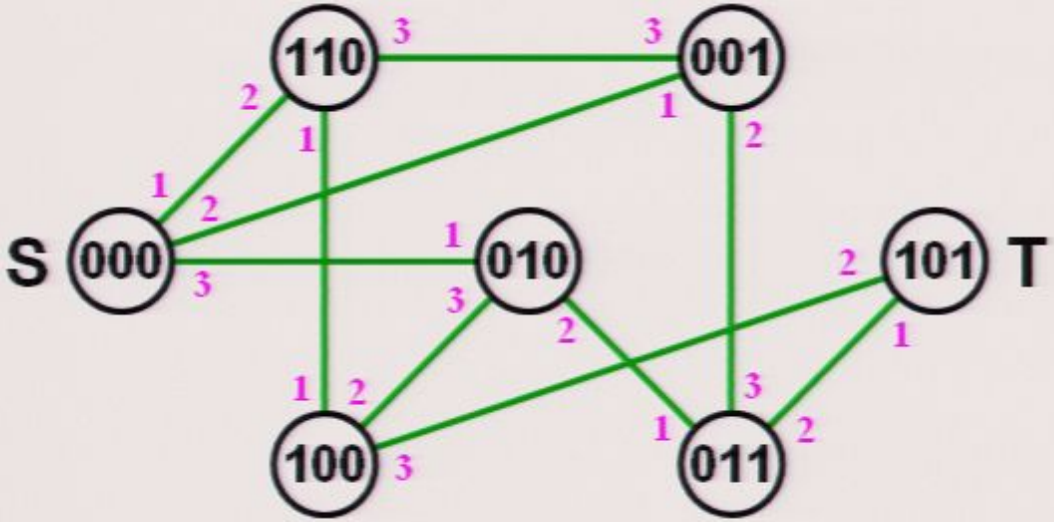
Representing graphs



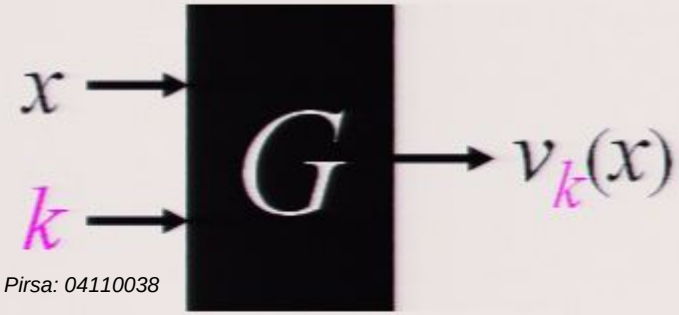
node	neighbors		
000	110	001	100
001	000	011	110
010	000	011	100
011	010	101	010
100	110	010	101
101	011	100	111
110	100	000	001
111	111	111	111



Representing graphs



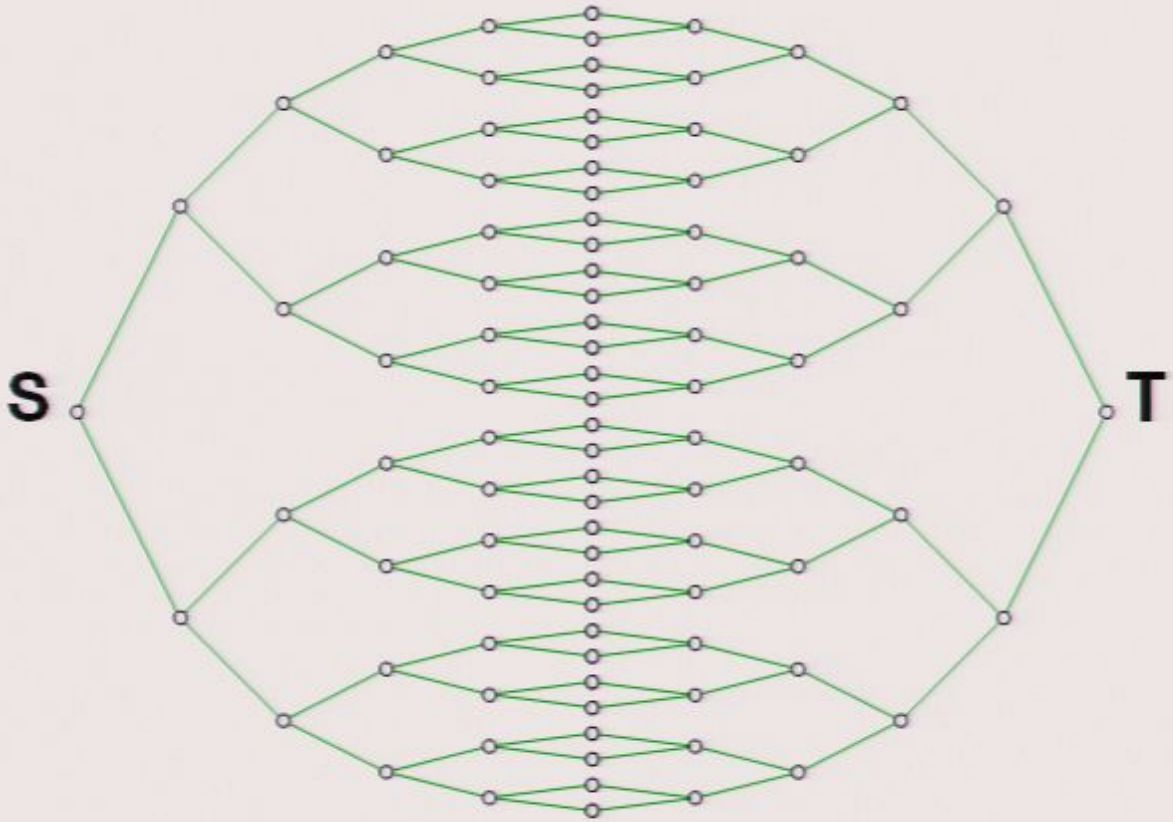
node	neighbors		
000	110	001	100
001	000	011	110
010	000	011	100
011	010	101	010
100	110	010	101
101	011	100	111
110	100	000	001
111	111	111	111



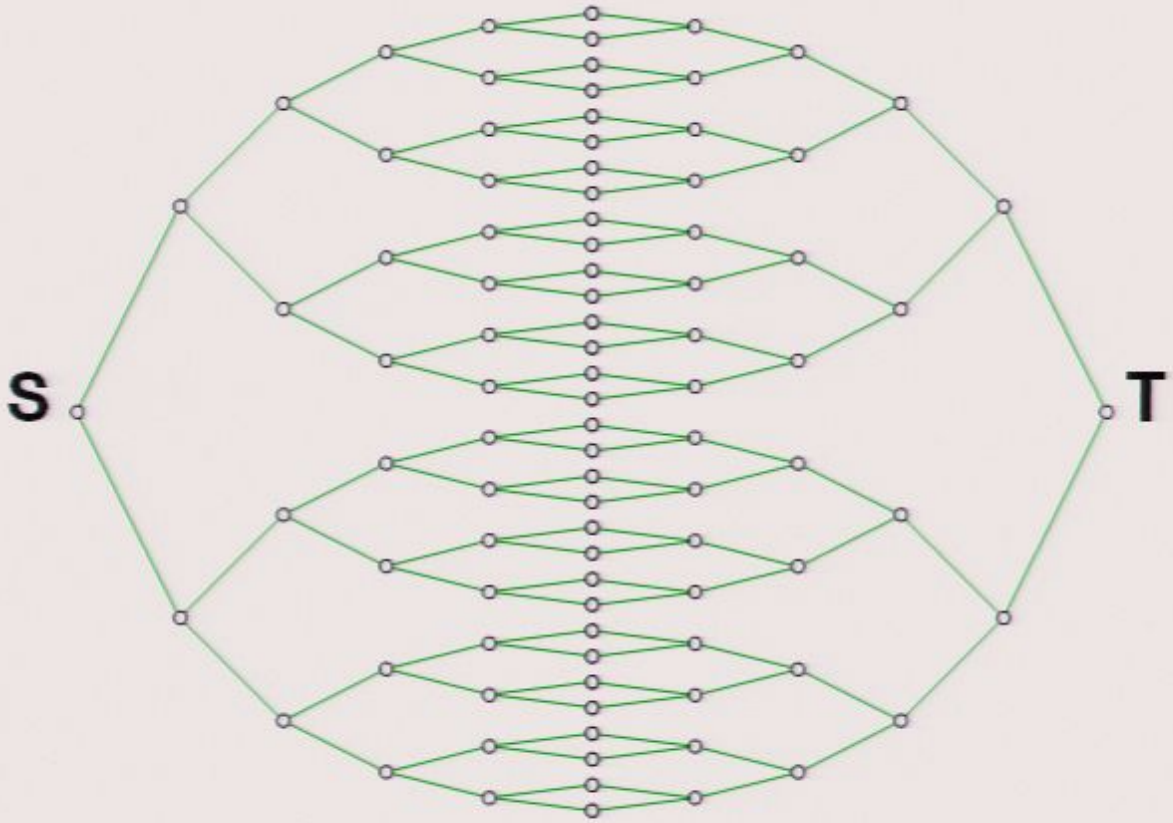
ST-traversal problem: given a black box for G and the name of node **S**, find the ***name*** of node **T**

"virtual labyrinth"

ST-traversal on GFG-graph

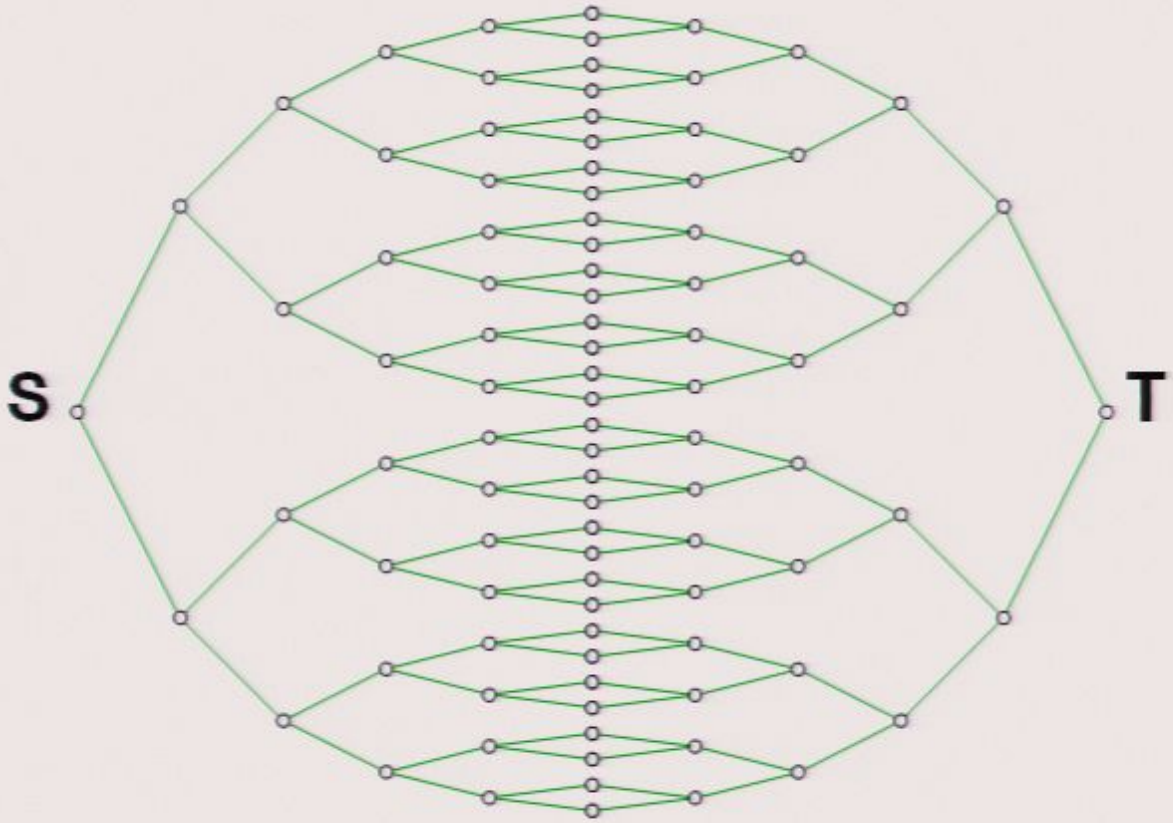


ST-traversal on GFG-graph



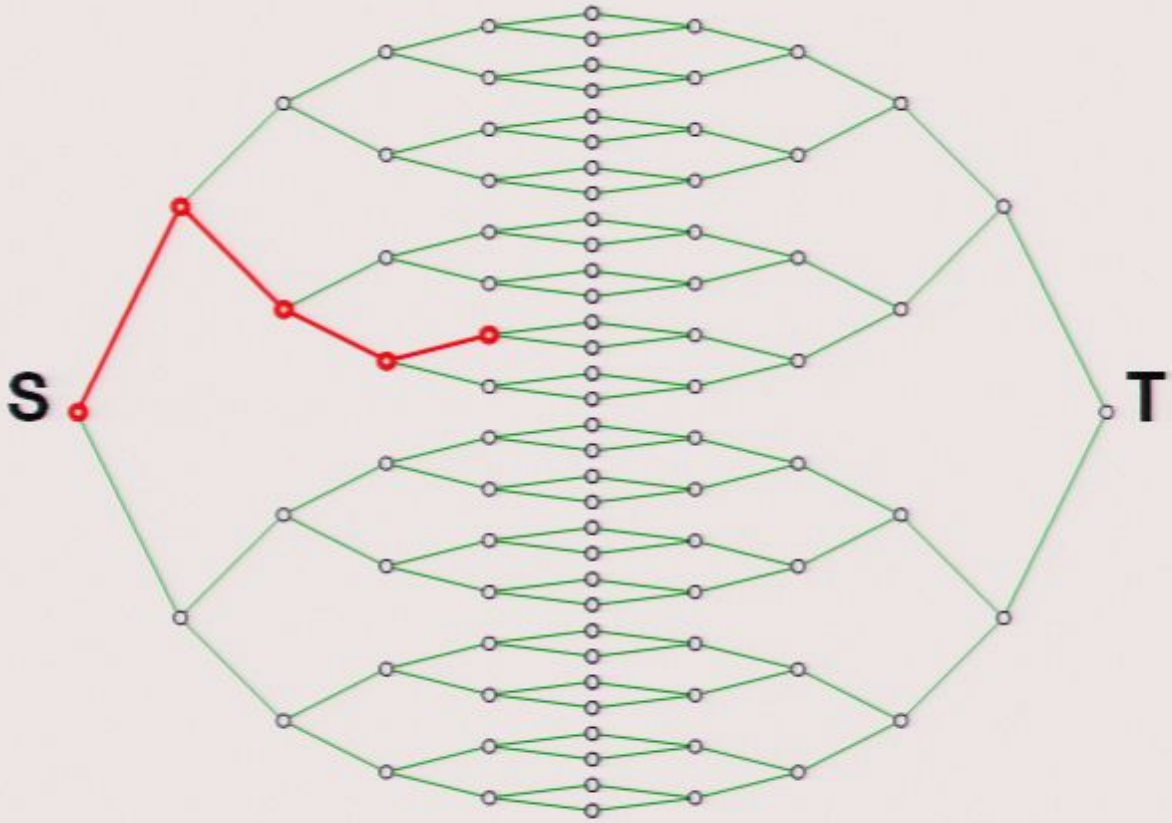
- Random walk costs $2^{O(n)}$

ST-traversal on GFG-graph



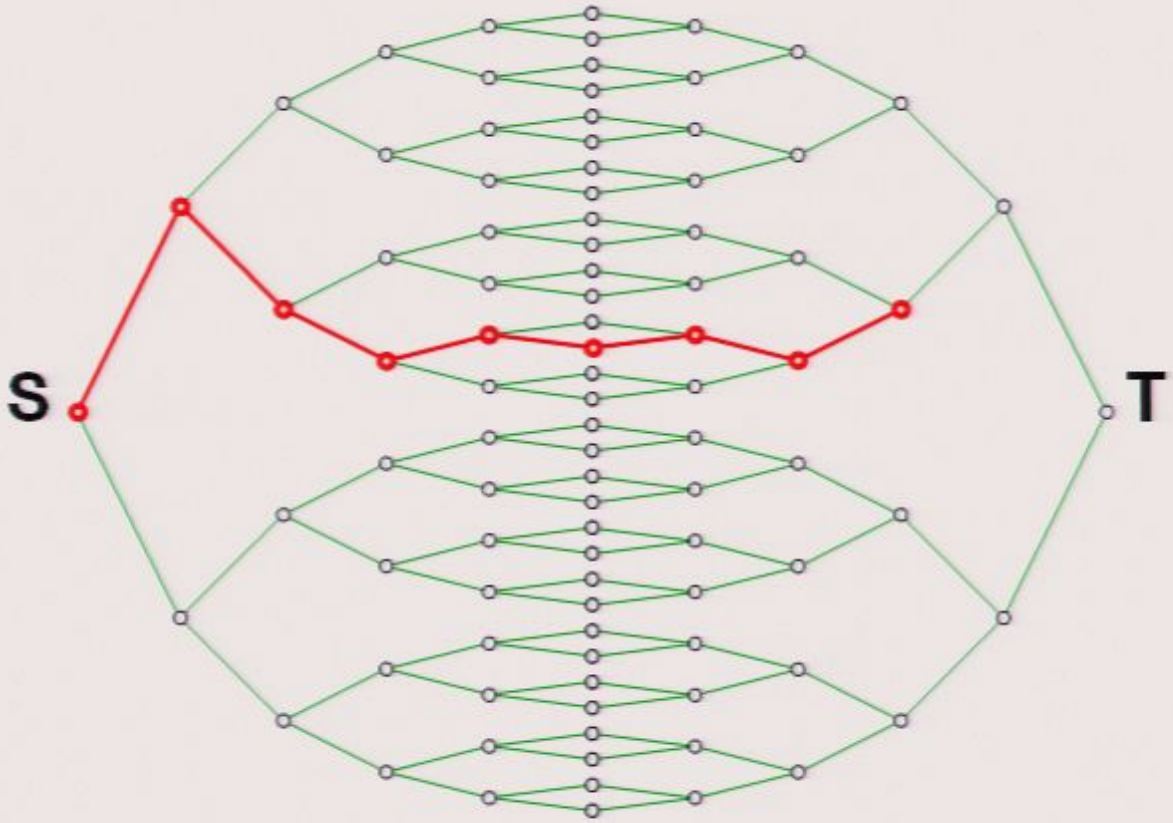
- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$

ST-traversal on GFG-graph



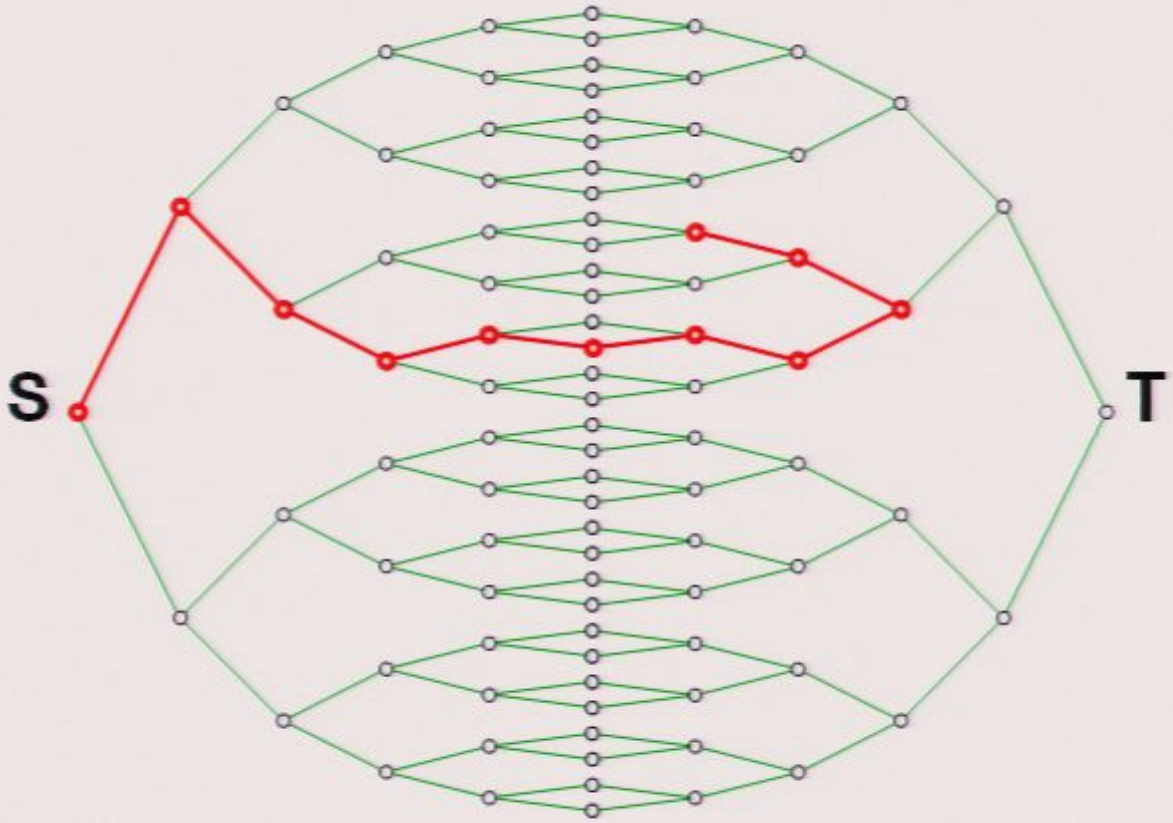
- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$

ST-traversal on GFG-graph



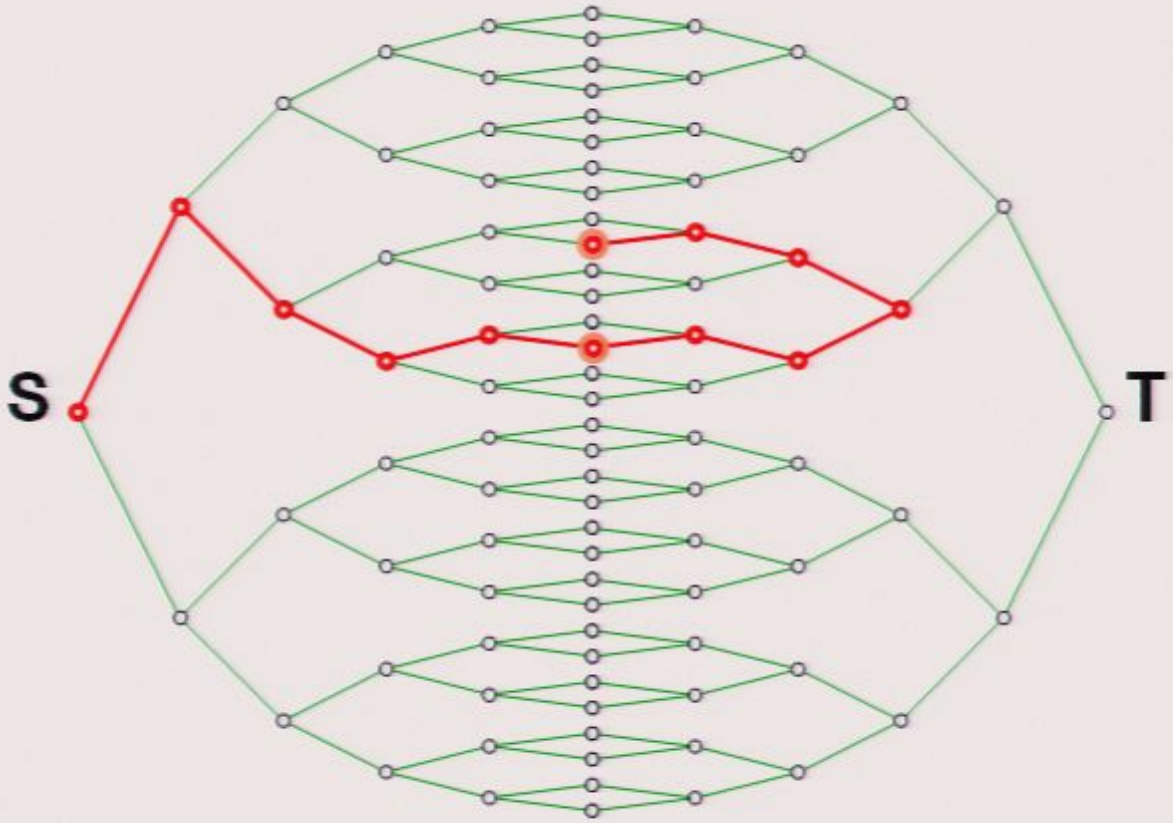
- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$

ST-traversal on GFG-graph



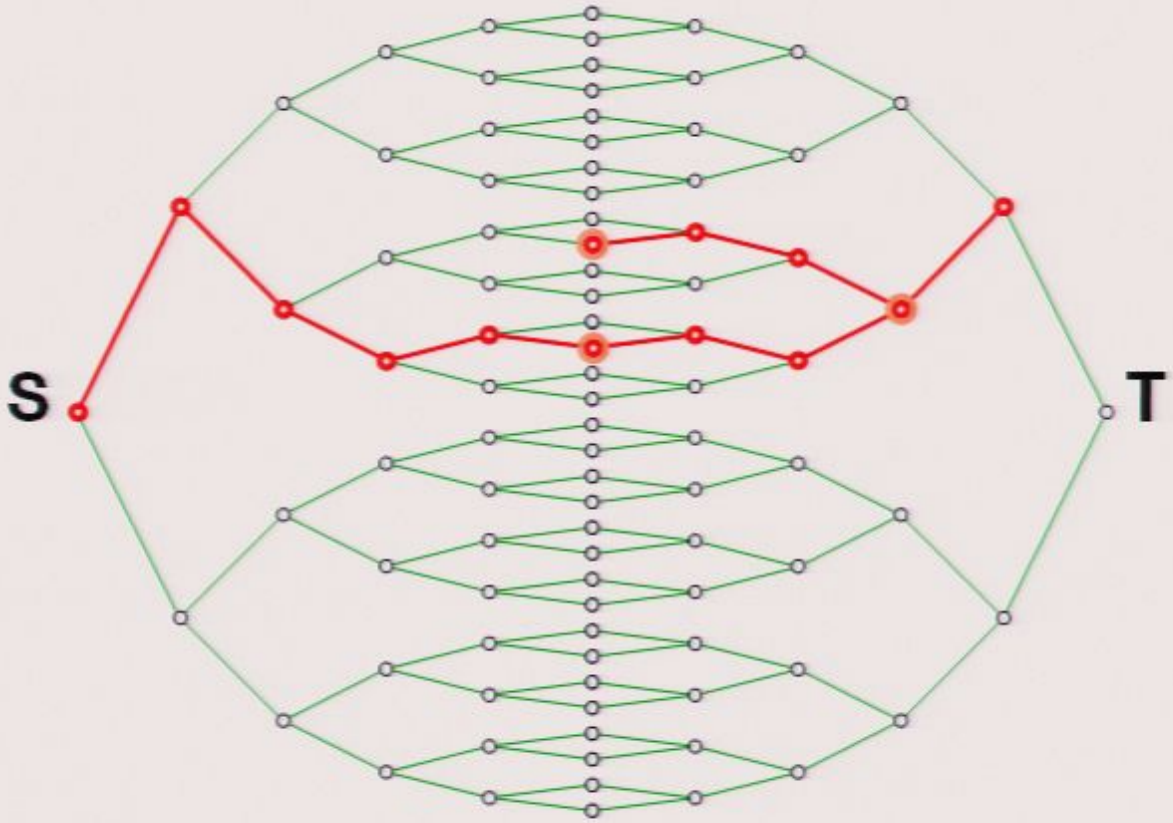
- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$

ST-traversal on GFG-graph



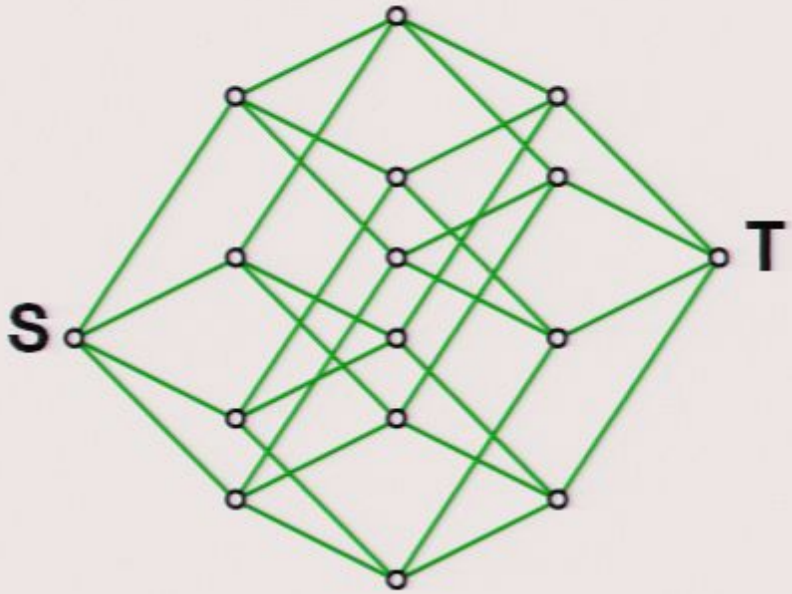
- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$

ST-traversal on GFG-graph

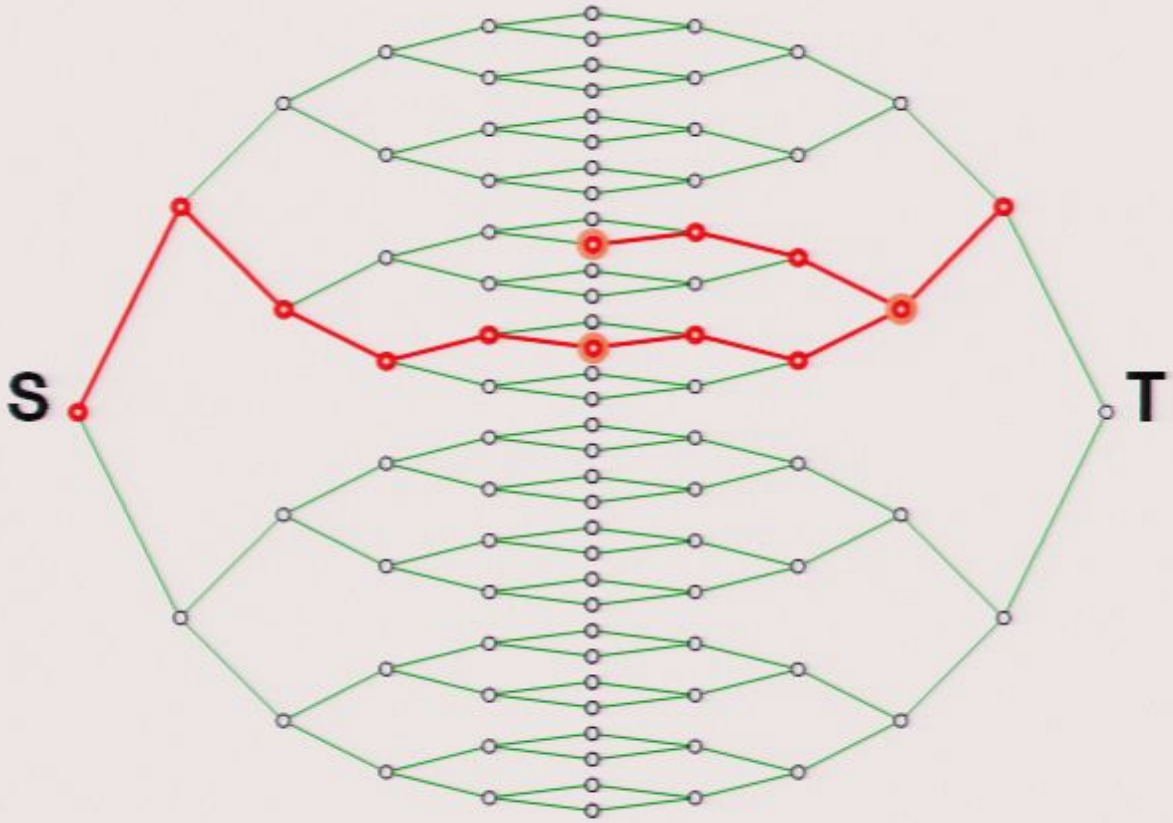


- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$
- Classical algorithm based on backtracking costs $O(n^2)$

ST-traversal on hypercube

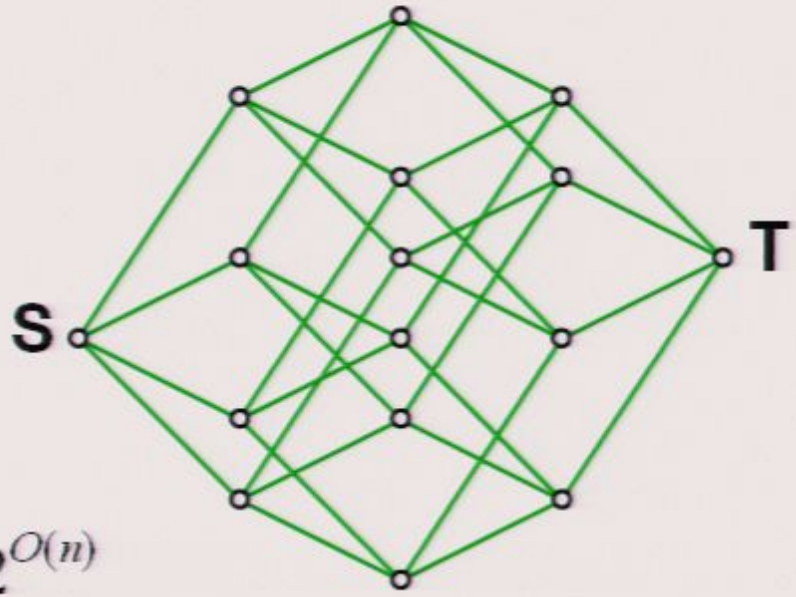


ST-traversal on GFG-graph



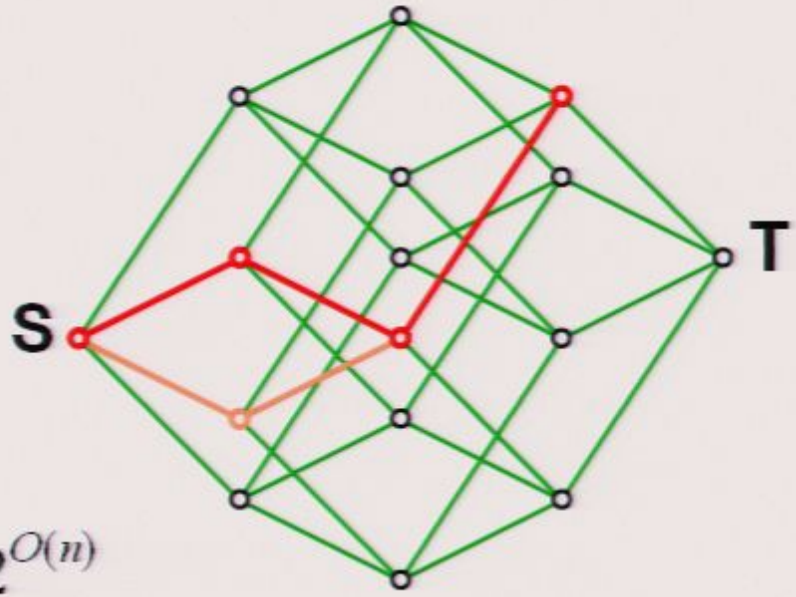
- Random walk costs $2^{O(n)}$
- Quantum walk with success probability $1/n$ costs $O(n^4)$
- Classical algorithm based on backtracking costs $O(n^2)$

ST-traversal on hypercube



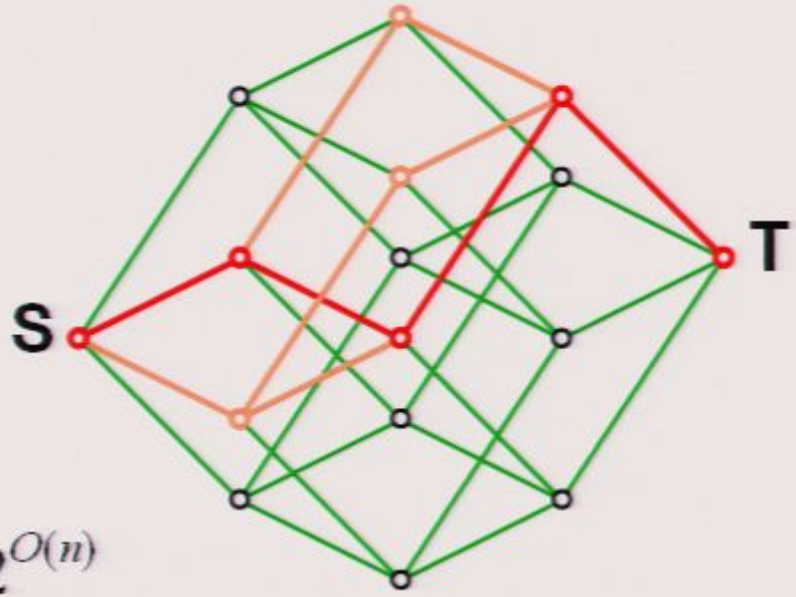
- Random walk $2^{O(n)}$
- Quantum walk $O(n)$

ST-traversal on hypercube



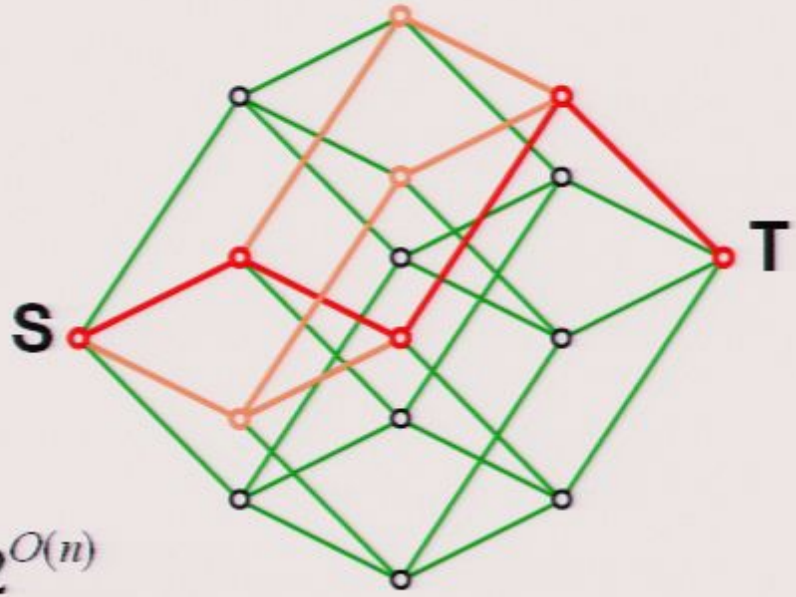
- Random walk $2^{O(n)}$
- Quantum walk $O(n)$

ST-traversal on hypercube



- Random walk $2^{O(n)}$
- Quantum walk $O(n)$
- Classical algorithm based on "dynamic programming" $O(n^3)$

ST-traversal on hypercube



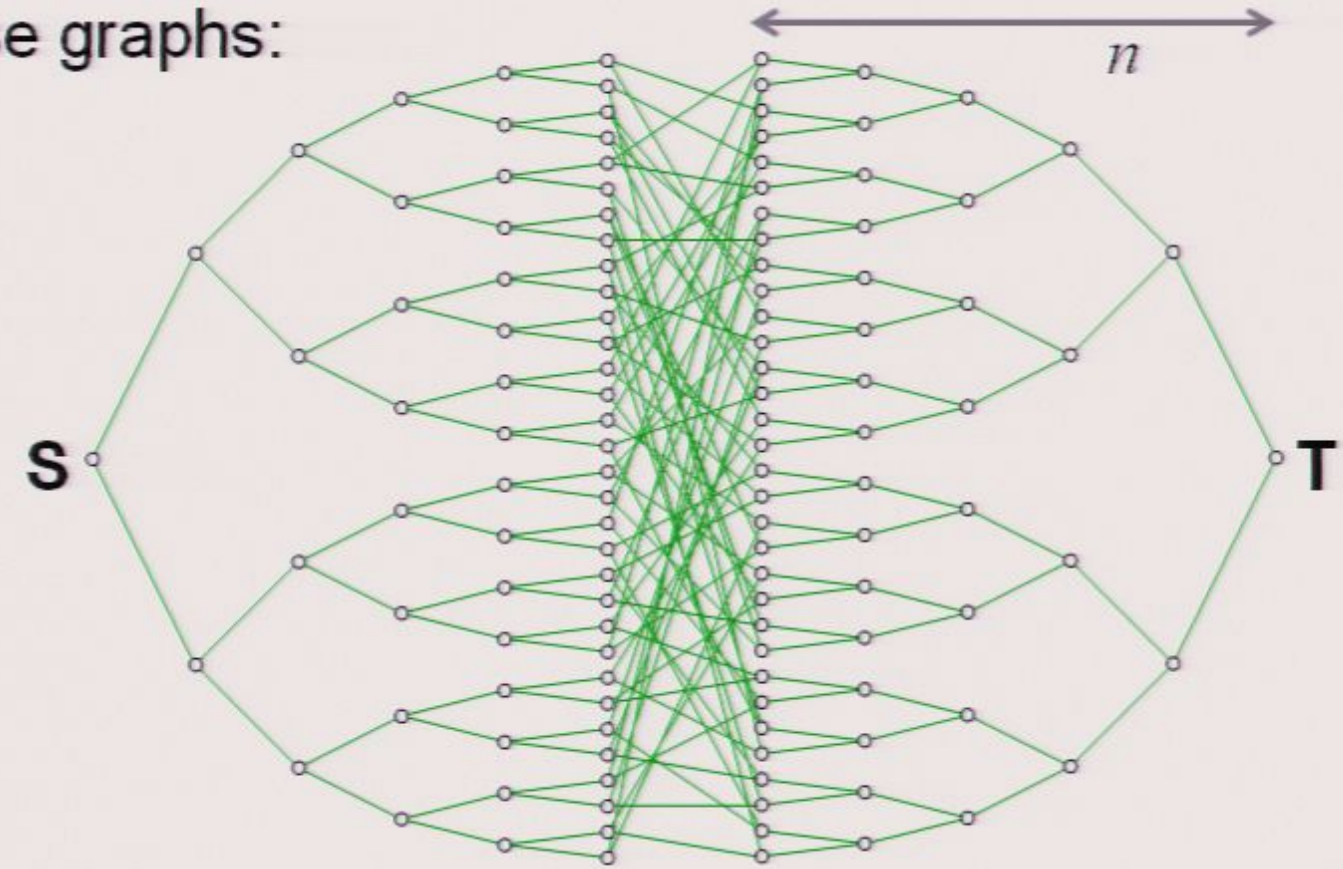
- Random walk $2^{O(n)}$
- Quantum walk $O(n)$
- Classical algorithm based on "dynamic programming" $O(n^3)$

Question: is there a graph for which a quantum walk finds **T** exponentially faster than ***any*** classical algorithm?

Answer: yes

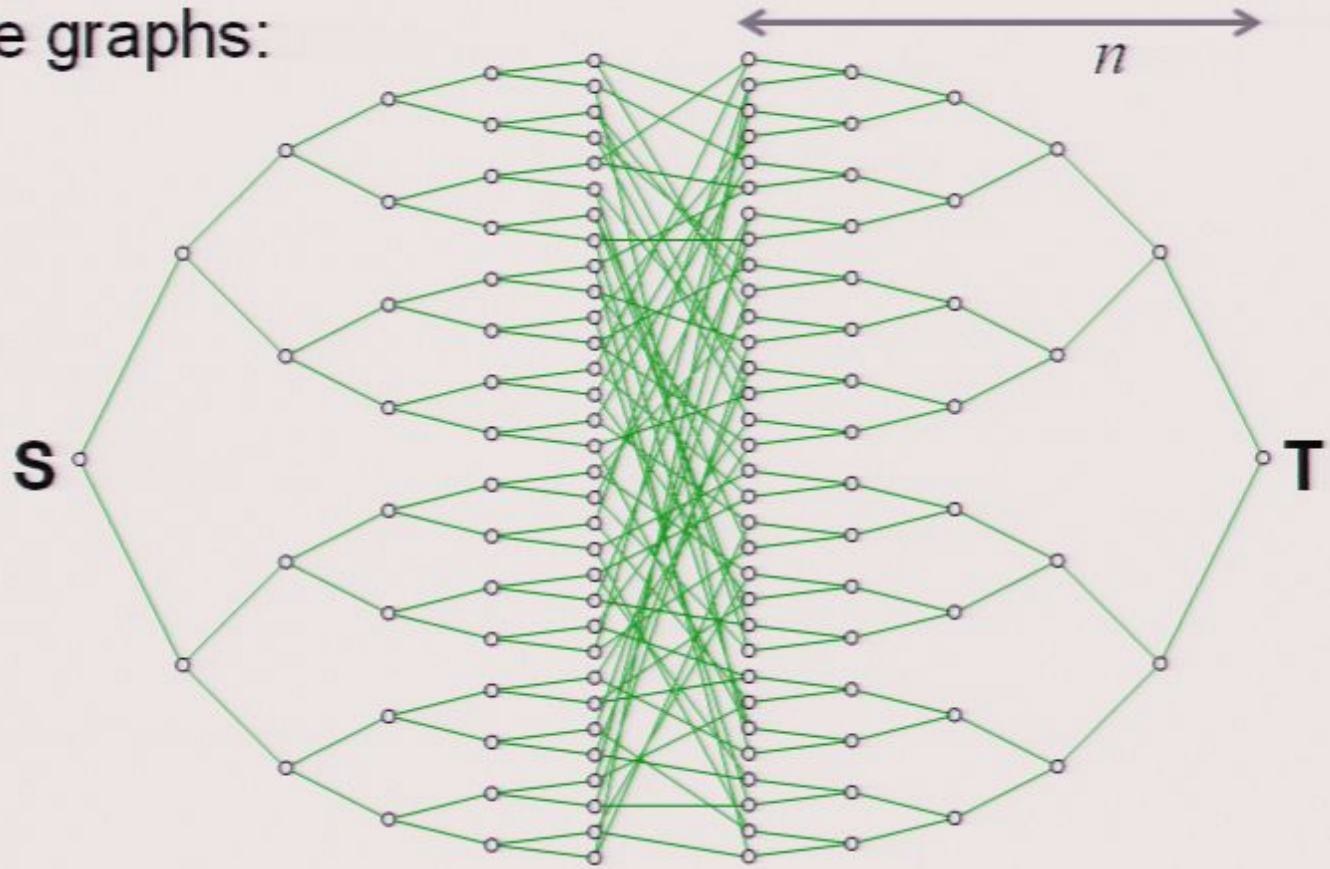
Answer: yes

For these graphs:



Answer: yes

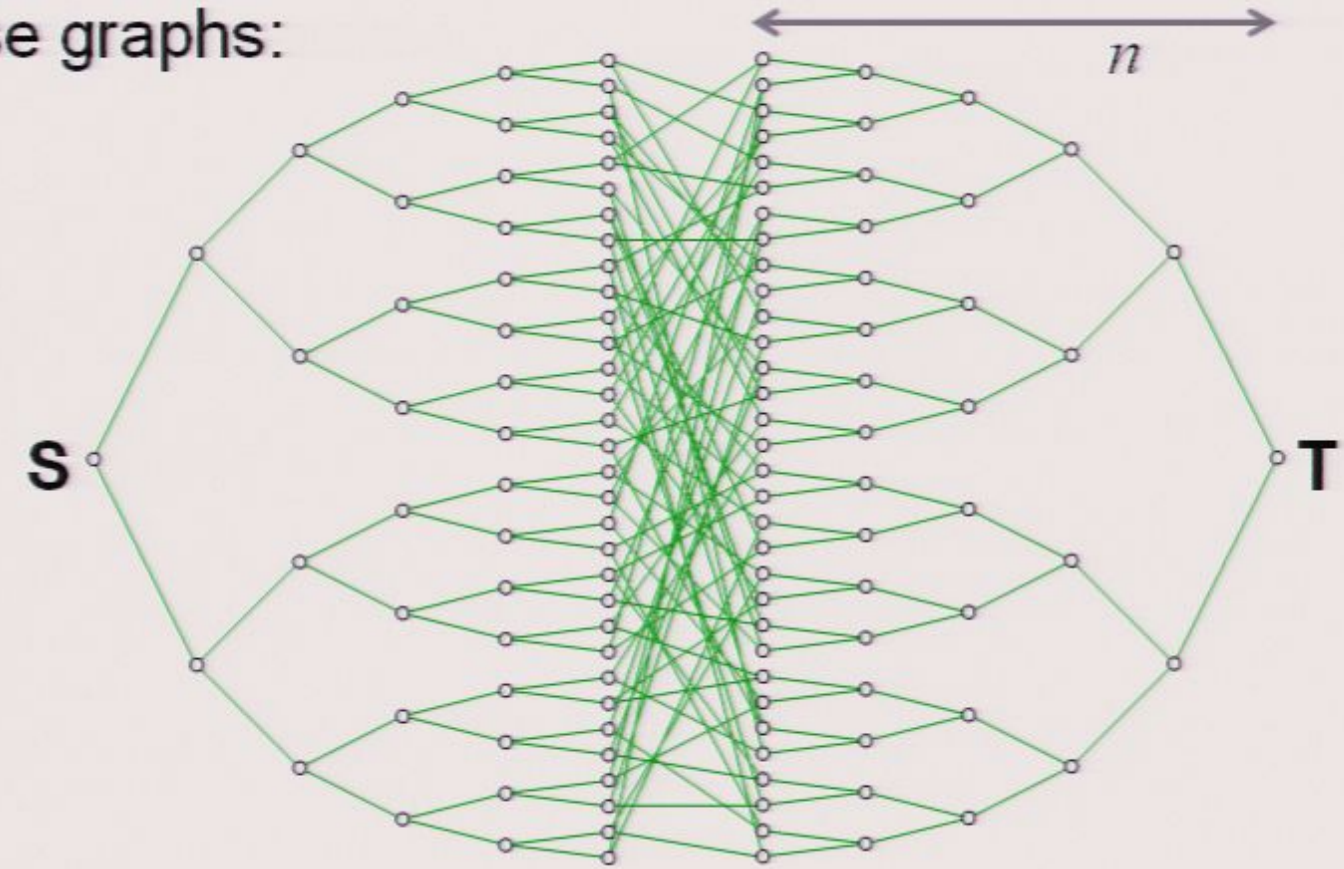
For these graphs:



- ***Any*** classical **ST**-traversal algorithm takes exponential time

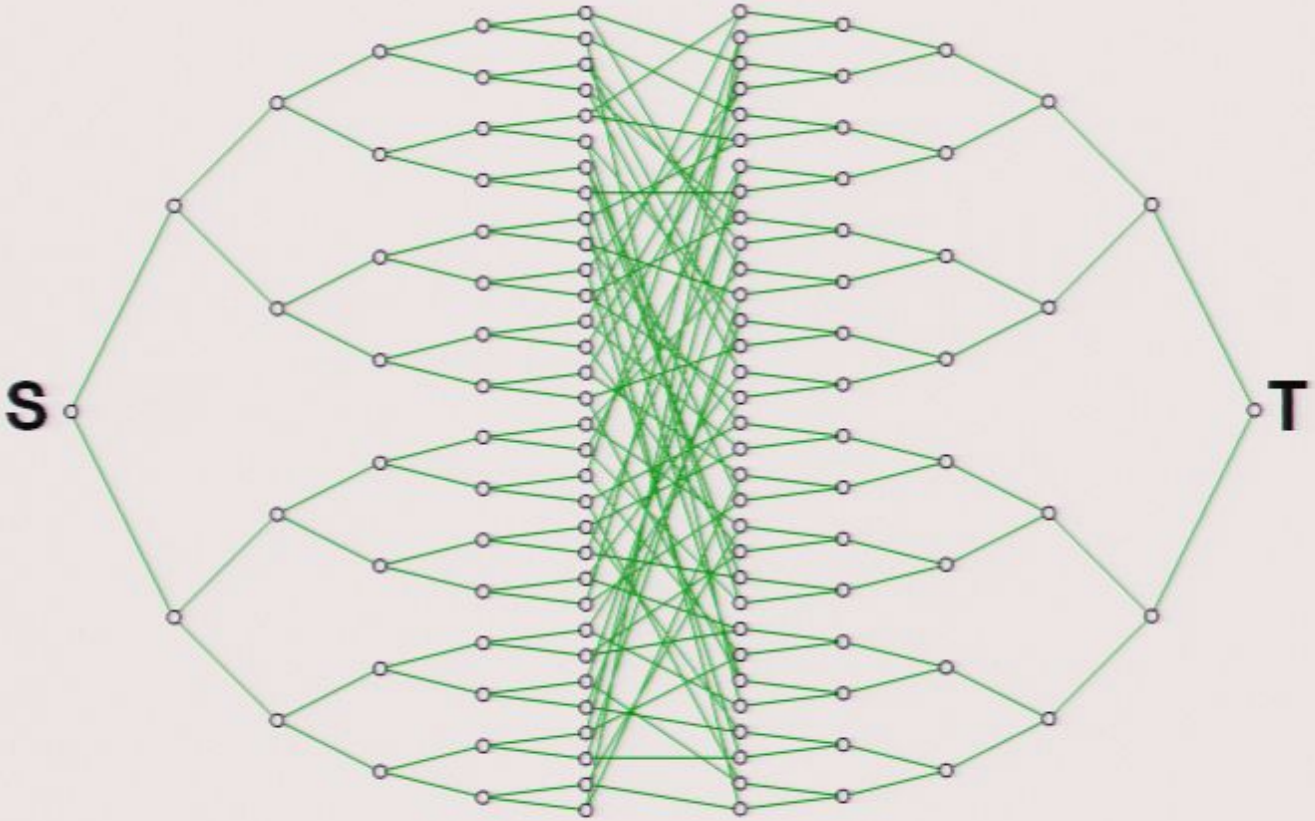
Answer: yes

For these graphs:

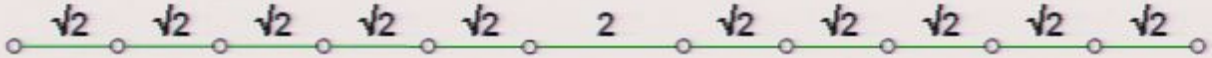


- ***Any*** classical **ST**-traversal algorithm takes exponential time
- Algorithm based on ***quantum walks*** takes polynomial time

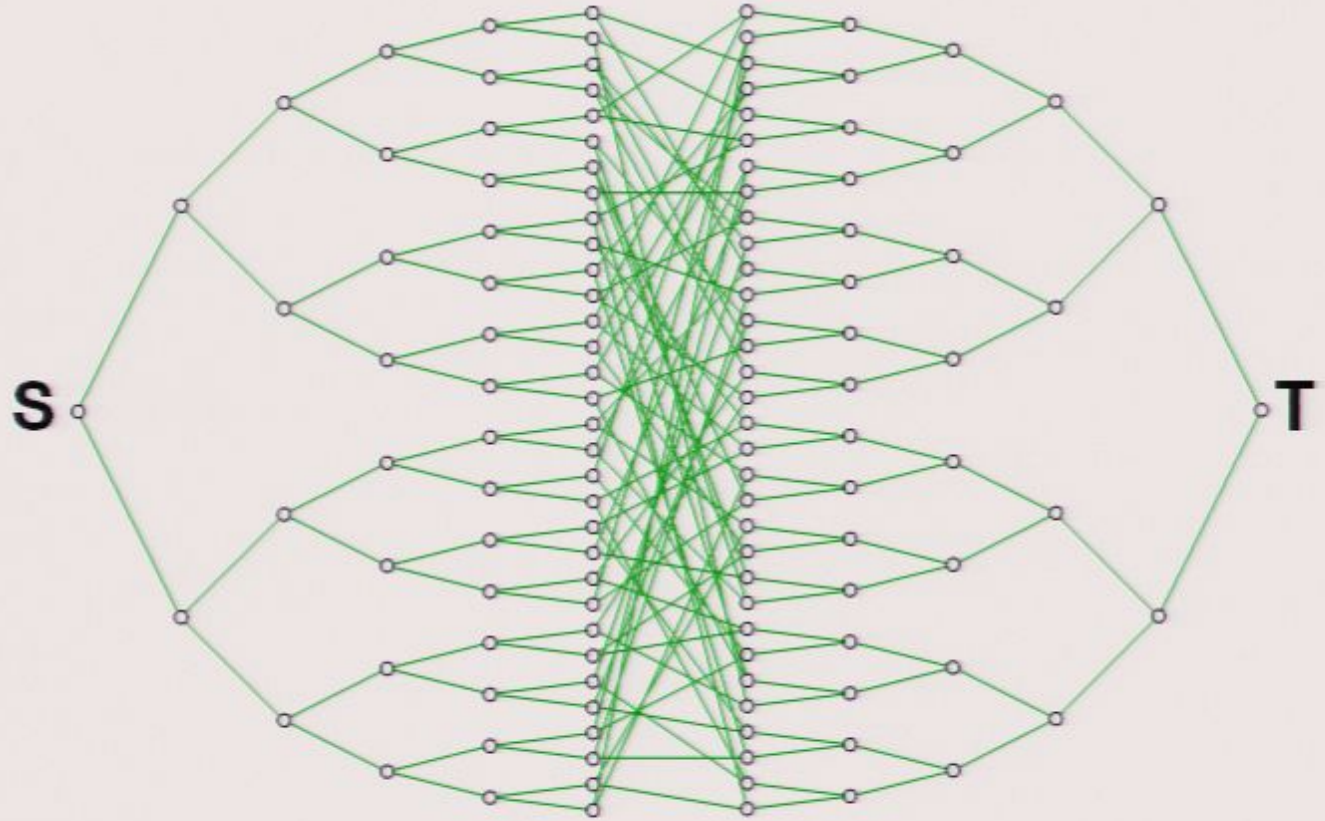
Analysis of continuous-time QW



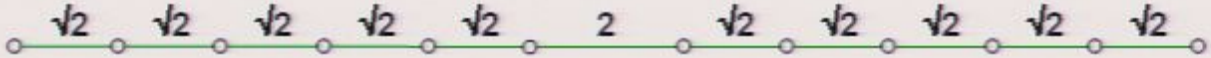
Column subspace:



Analysis of continuous-time QW

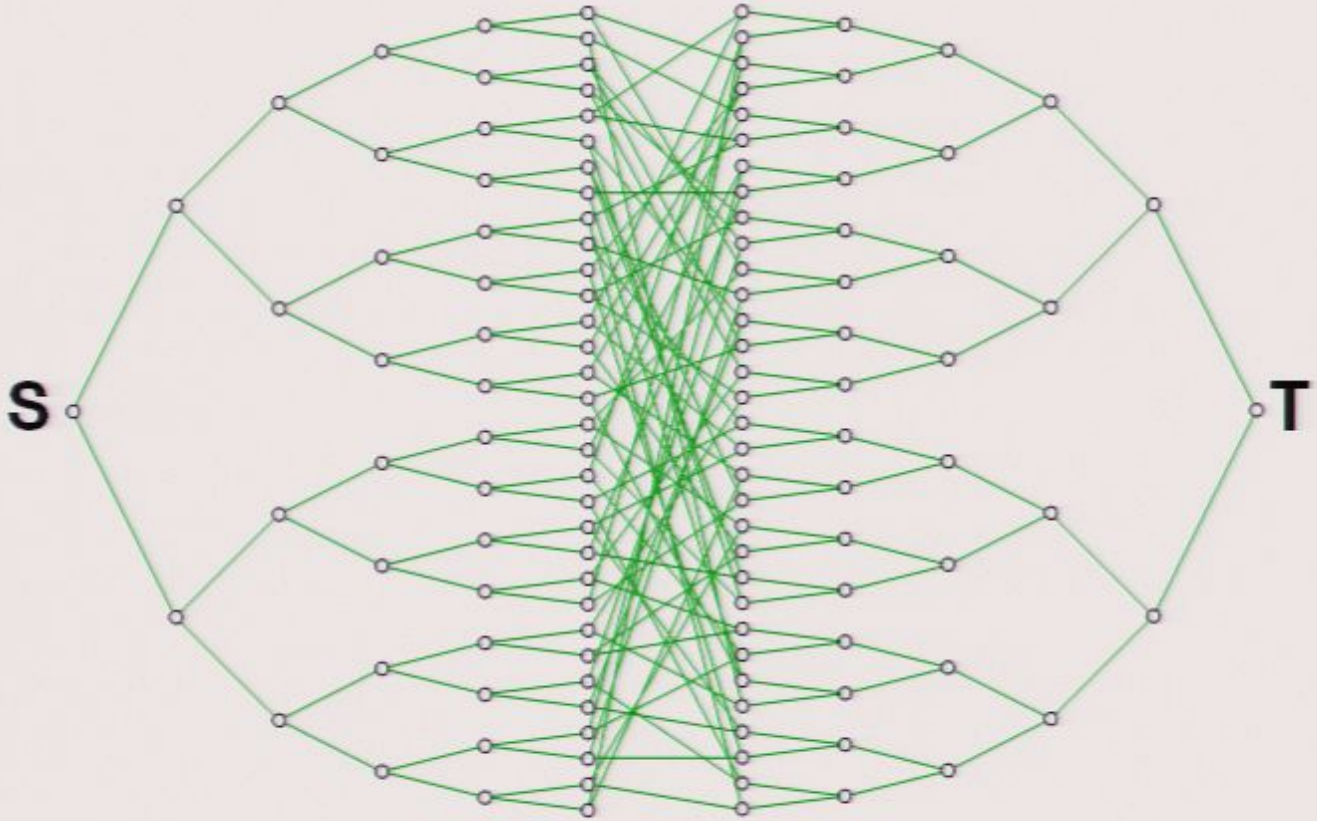


Column subspace:

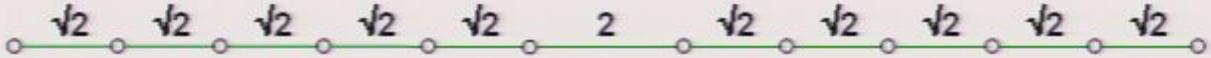


Choose $t \in [0, O(n^4)]$ uniformly, run QW for time t, and measure

Analysis of continuous-time QW



Column subspace:

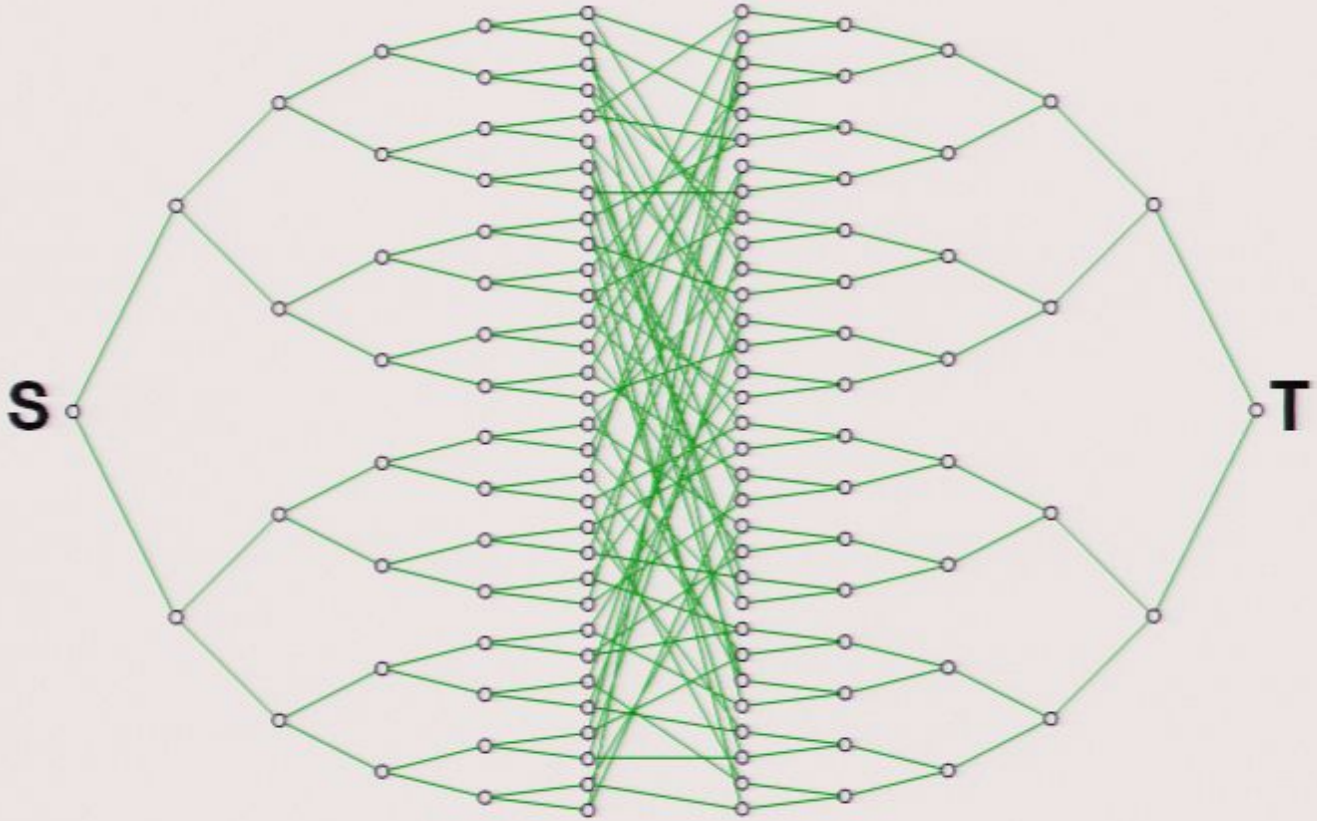


Choose $t \in [0, O(n^4)]$ uniformly, run QW for time t, and measure

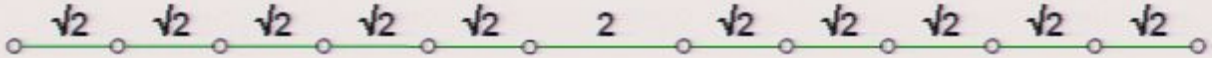
Theorem: the outcome is T with probability $\Omega(1/n)$

How to simulate the quantum walk

Analysis of continuous-time QW



Column subspace:



Choose $t \in [0, O(n^4)]$ uniformly, run QW for time t, and measure

Theorem: the outcome is T with probability $\Omega(1/n)$

How to simulate the quantum walk

Goal: simulate $e^{-iHt}|\mathbf{s}\rangle$, where $\langle x|H|y\rangle = \begin{cases} 1 & \text{if } (x,y) \text{ edge} \\ 0 & \text{otherwise} \end{cases}$

given **s** and a black box for G

How to simulate the quantum walk

Goal: simulate $e^{-iHt}|\mathbf{s}\rangle$, where $\langle x|H|y\rangle = \begin{cases} 1 & \text{if } (x,y) \text{ edge} \\ 0 & \text{otherwise} \end{cases}$

given **s** and a black box for G

Tools for simulating Hamiltonian evolution:

How to simulate the quantum walk

Goal: simulate $e^{-iHt}|\mathbf{s}\rangle$, where $\langle x|H|y\rangle = \begin{cases} 1 & \text{if } (x,y) \text{ edge} \\ 0 & \text{otherwise} \end{cases}$

given **s** and a black box for G

Tools for simulating Hamiltonian evolution:

Tensor product simulations of $H_1, H_2, \ldots, H_n$ and a method for efficiently computing product of their eigenvalues enables simulation of $H_1 \otimes H_2 \otimes \ldots \otimes H_n$

How to simulate the quantum walk

Goal: simulate $e^{-iHt}|\mathbf{s}\rangle$, where $\langle x|H|y\rangle = \begin{cases} 1 & \text{if } (x,y) \text{ edge} \\ 0 & \text{otherwise} \end{cases}$

given **s** and a black box for G

Tools for simulating Hamiltonian evolution:

Tensor product simulations of $H_1, H_2, \ldots, H_n$ and a method for efficiently computing product of their eigenvalues enables simulation of $H_1 \otimes H_2 \otimes \ldots \otimes H_n$

Unitary conjugation simulation of H and U, $U^\dagger$ efficiently computable enables simulation of $U^\dagger H U$ (that is, $e^{-iU^\dagger HUt}$)

How to simulate the quantum walk

Goal: simulate $e^{-iHt}|\mathbf{s}\rangle$, where $\langle x|H|y\rangle = \begin{cases} 1 & \text{if } (x,y) \text{ edge} \\ 0 & \text{otherwise} \end{cases}$

given **s** and a black box for G

Tools for simulating Hamiltonian evolution:

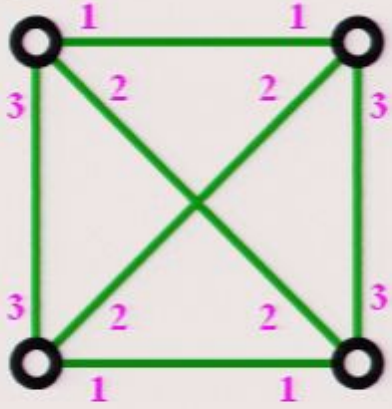
Tensor product simulations of $H_1, H_2, \ldots, H_n$ and a method for efficiently computing product of their eigenvalues enables simulation of $H_1 \otimes H_2 \otimes \ldots \otimes H_n$

Unitary conjugation simulation of H and U, $U^\dagger$ efficiently computable enables simulation of $U^\dagger H U$ (that is, $e^{-iU^\dagger HUt}$)

Linear combination simulation of $H_1, H_2, \ldots, H_n$ enables simulation of $H_1+H_2+\ldots+H_n$ (that is, $e^{-i(H_1+H_2+\ldots+H_n)t}$)

Simulation of quantum walk for *symmetric* black-boxes

Simulation of quantum walk for *symmetric* black-boxes



Symmetric: $v_k(x) = y$ iff $v_k(y) = x$

Define **SWAP** $|x\rangle|y\rangle = |y\rangle|x\rangle$

Simulate evolution by **SWAP** (tensor product)

Simulate evolution by $\boldsymbol{V}_k^{\dagger}\textbf{SWAP}\boldsymbol{V}_k$ (unitary conjugation)

$$\boldsymbol{V}_k^{\dagger}\textbf{SWAP}\boldsymbol{V}_k\,|x\rangle|0\rangle = \boldsymbol{V}_k^{\dagger}\textbf{SWAP}\,|x\rangle|v_k(x)\rangle \quad (\boldsymbol{V}_k^{\dagger} \text{ reversible black-box})$$

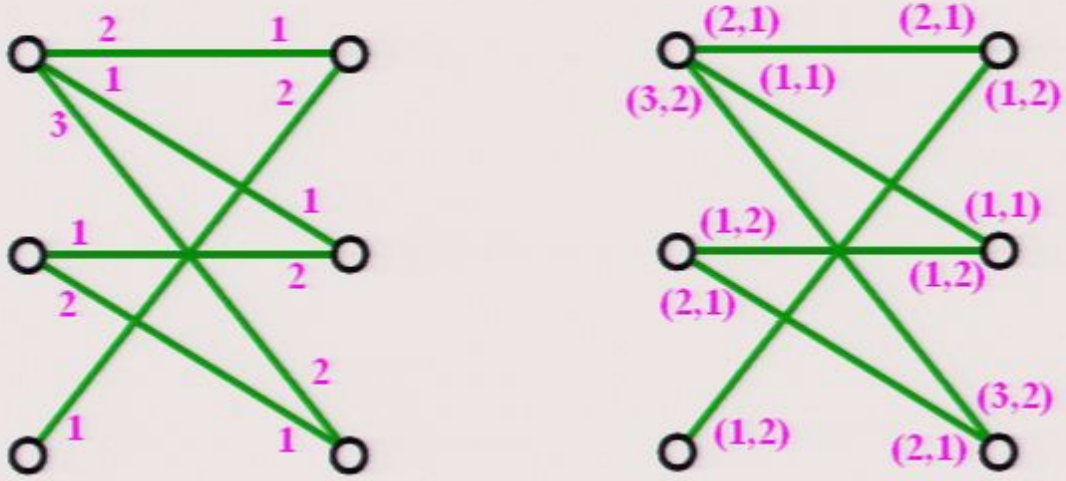
$$= \boldsymbol{V}_k^{\dagger}|v_k(x)\rangle|x\rangle$$

$$= |v_k(x)\rangle|x \oplus v_k(v_k(x))\rangle \;=\; |v_k(x)\rangle|0\rangle$$

Simulate evolution by $A \otimes I = \sum_k \boldsymbol{V}_k^{\dagger}\textbf{SWAP}\boldsymbol{V}_k$ (linear combination)

Simulation for *bipartite* graphs

Idea: reduce to the symmetric case



Conclusion: there is a quantum walk algorithm that solves **ST**-traversal with polynomially many queries and other operations

Simulation for *arbitrary* graphs

[Aharonov & Ta-Shma '03]: $O(n^9 d^3 t^{1.5}/\sqrt{\varepsilon})$ queries, where:

- $n = \log|G|$
- d is the maximum degree
- t is the time duration
- ε is the precision

Simulation for *arbitrary* graphs

[Aharonov & Ta-Shma '03]: $O(n^9 d^3 t^{1.5}/\sqrt{\varepsilon})$ queries, where:

- $n = \log|G|$
- d is the maximum degree
- t is the time duration
- ε is the precision

[Ahokas, Berry, C & Sanders '04]: $O((\log^* n)\, d^{2+2\delta}\, t^{1+\delta}/\varepsilon^{\delta})$ queries, for any constant $\delta > 0$

Simulation for *arbitrary* graphs

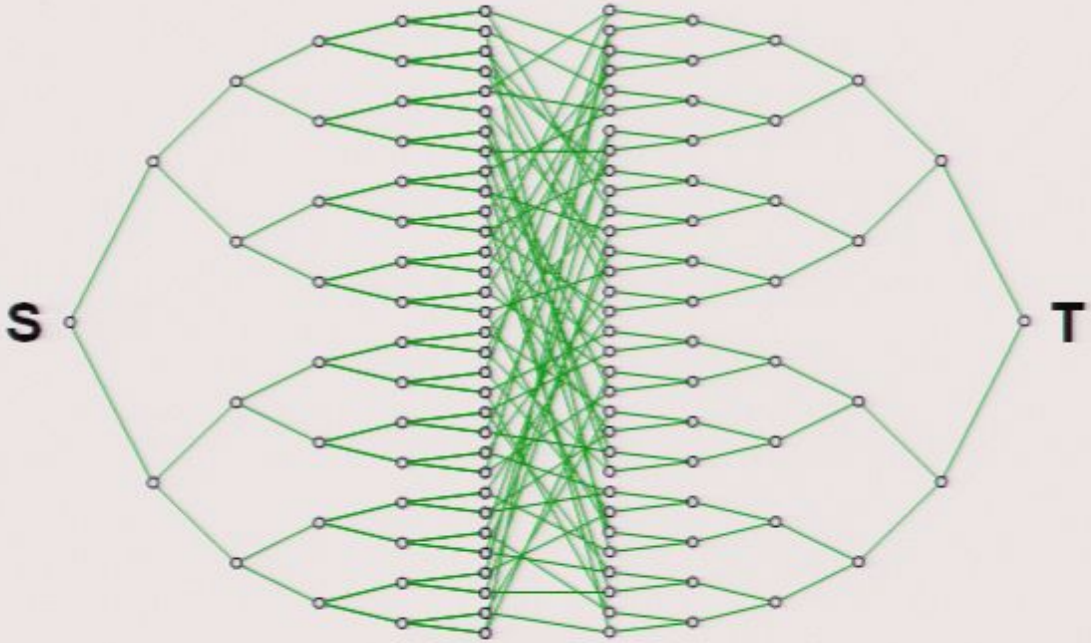
[Aharonov & Ta-Shma '03]: $O(n^9 d^3 t^{1.5}/\sqrt{\varepsilon})$ queries, where:

- $n = \log|G|$
- d is the maximum degree
- t is the time duration
- ε is the precision

[Ahokas, Berry, C & Sanders '04]: $O((\log^* n)\, d^{2+2\delta}\, t^{1+\delta}/\varepsilon^{\delta})$ queries, for any constant $\delta > 0$

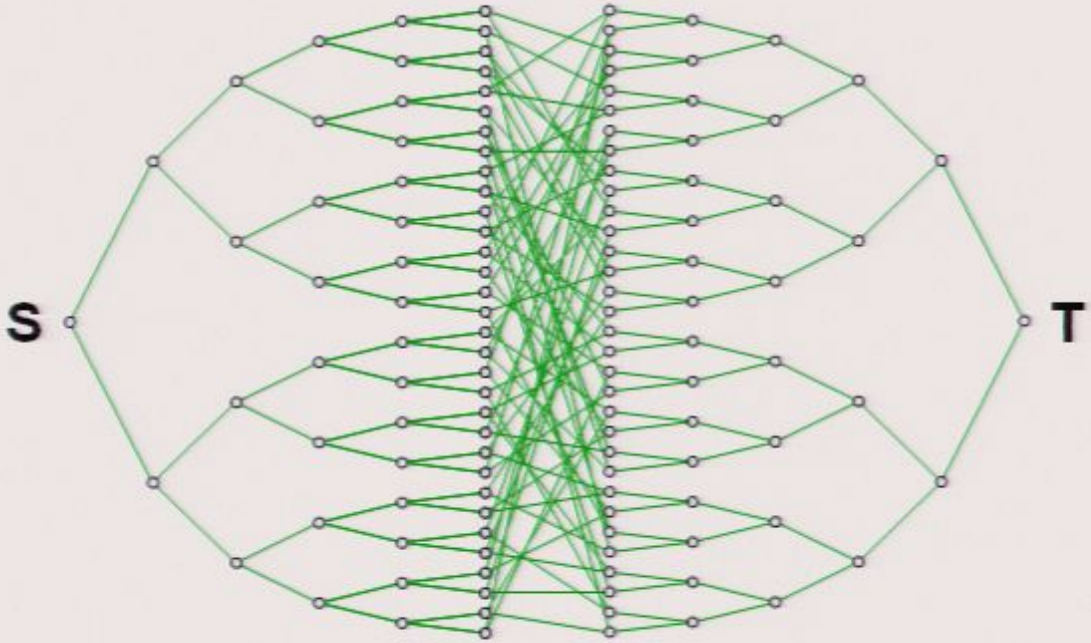
Consequence: for algorithms that employ continuous-time quantum walks, the overhead for the QW simulation can be made negligible in many cases

Classical hardness of ST-traversal

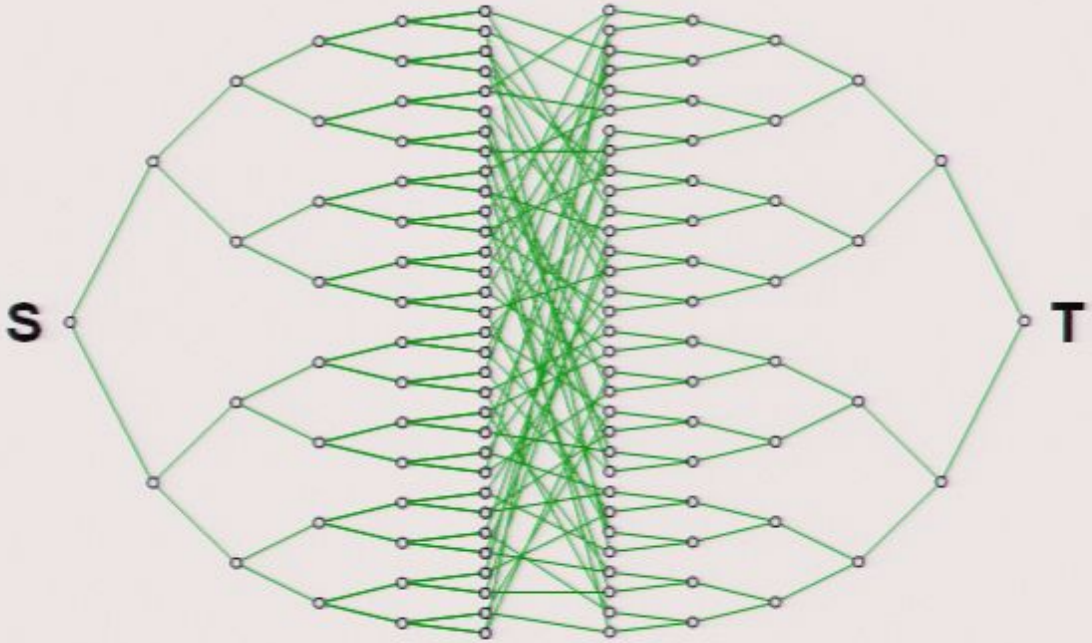


Classical hardness of ST-traversal

Define ***success*** as either finding T or finding a cycle



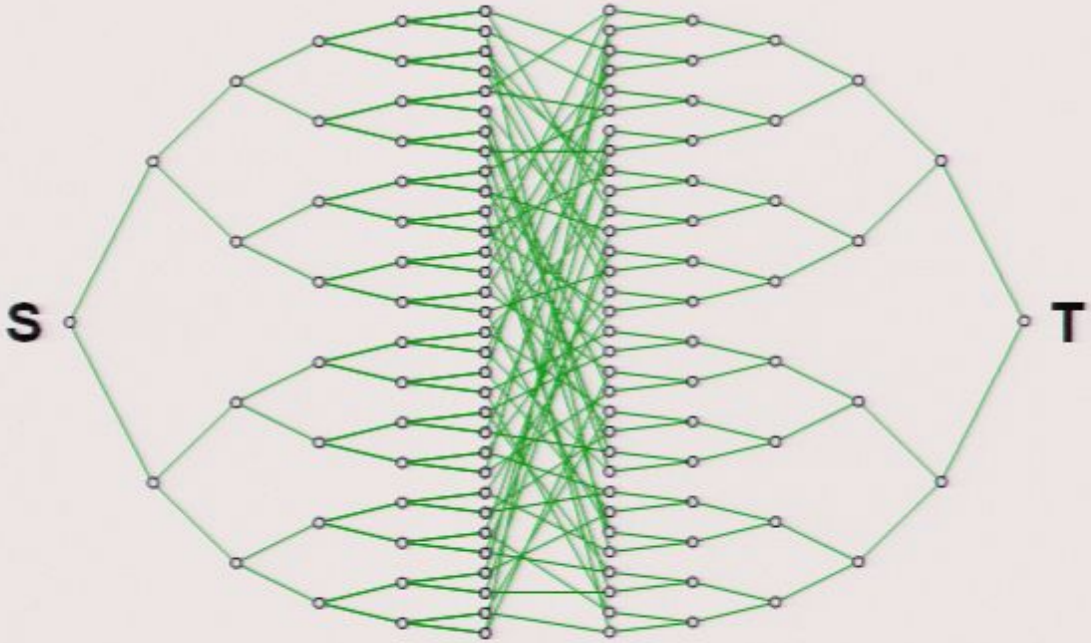
Classical hardness of ST-traversal



Classical hardness of ST-traversal

Define ***success*** as either finding T or finding a cycle

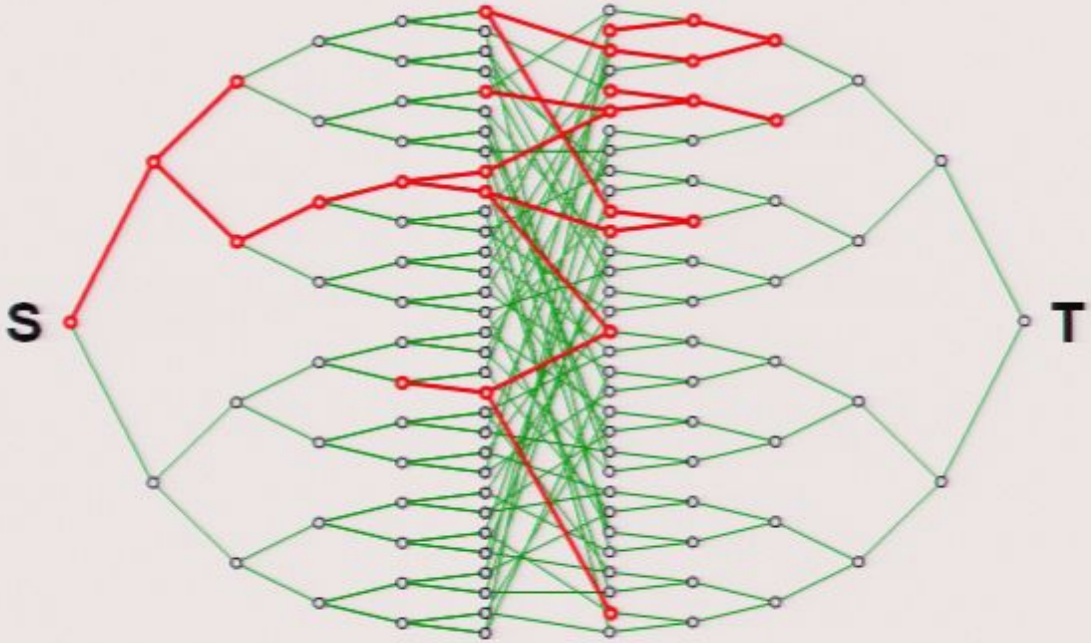
Then, prior to success, algorithm is growing a tree on G



Classical hardness of ST-traversal

Define ***success*** as either finding T or finding a cycle

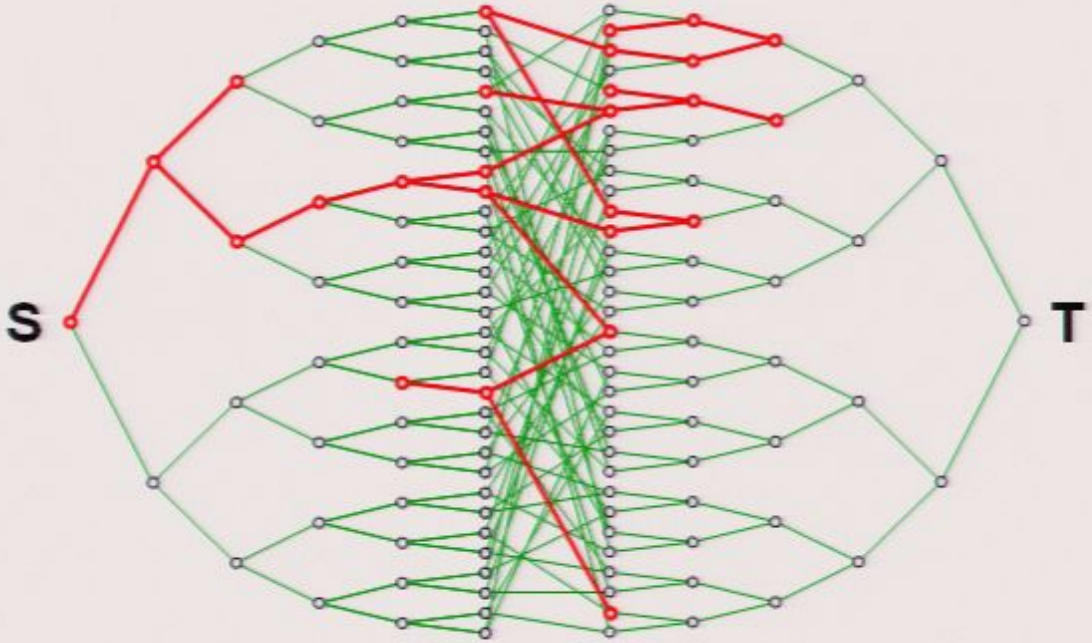
Then, prior to success, algorithm is growing a tree on G

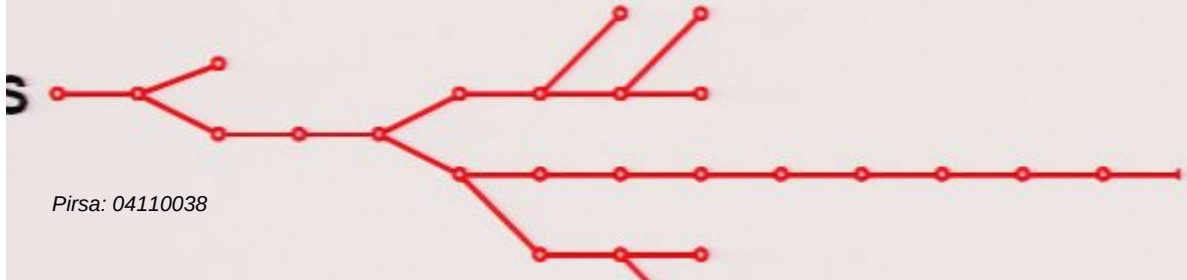


Classical hardness of ST-traversal

Define ***success*** as either finding T or finding a cycle

Then, prior to success, algorithm is growing a tree on G

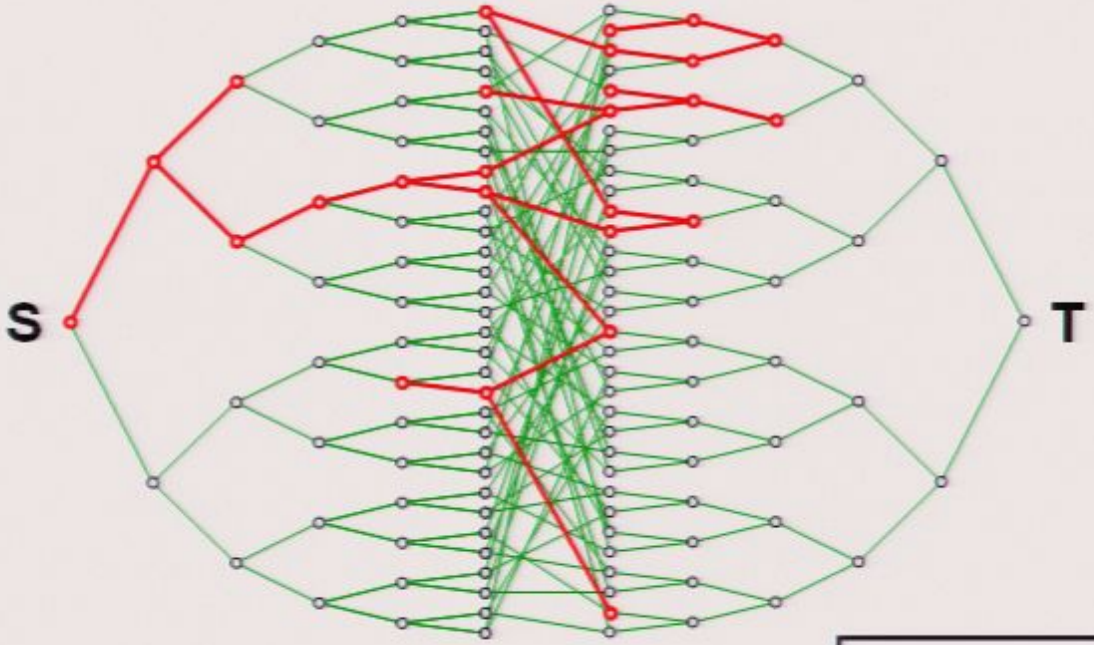


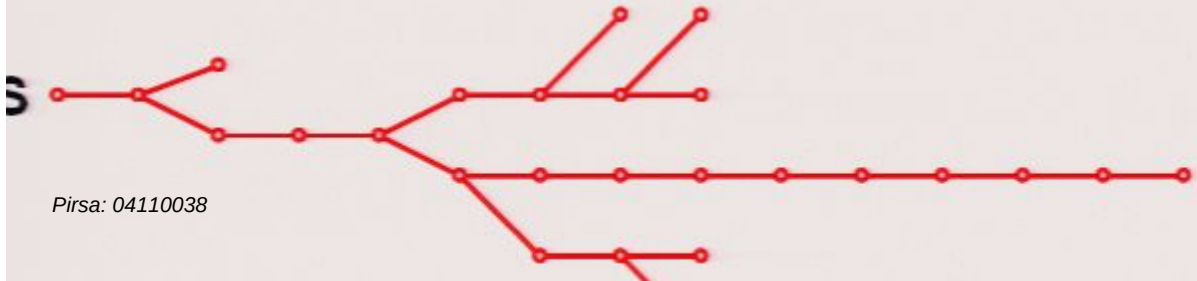


Classical hardness of ST-traversal

Define ***success*** as either finding T or finding a cycle

Then, prior to success, algorithm is growing a tree on G

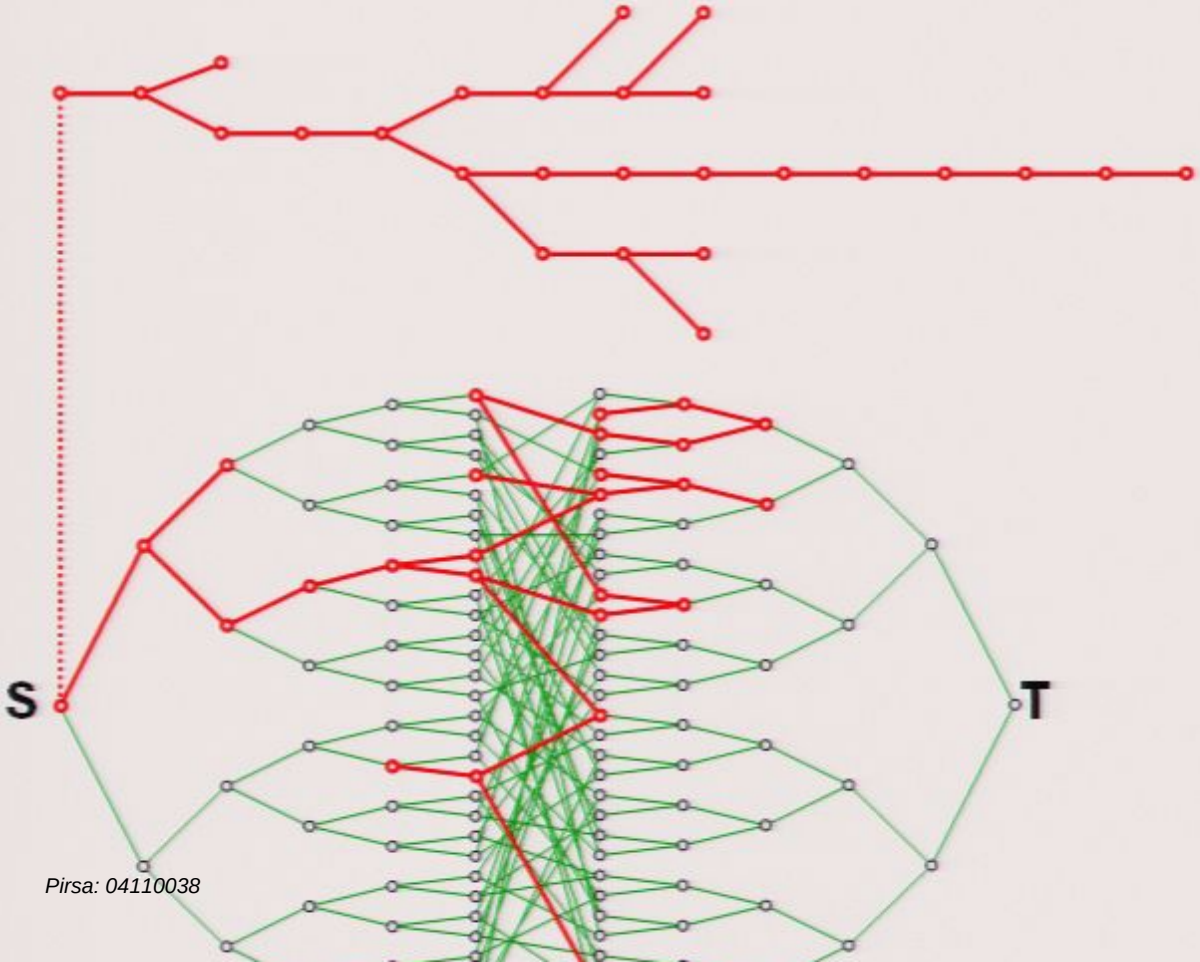




Can assume tree is pre-computed, since black box is random

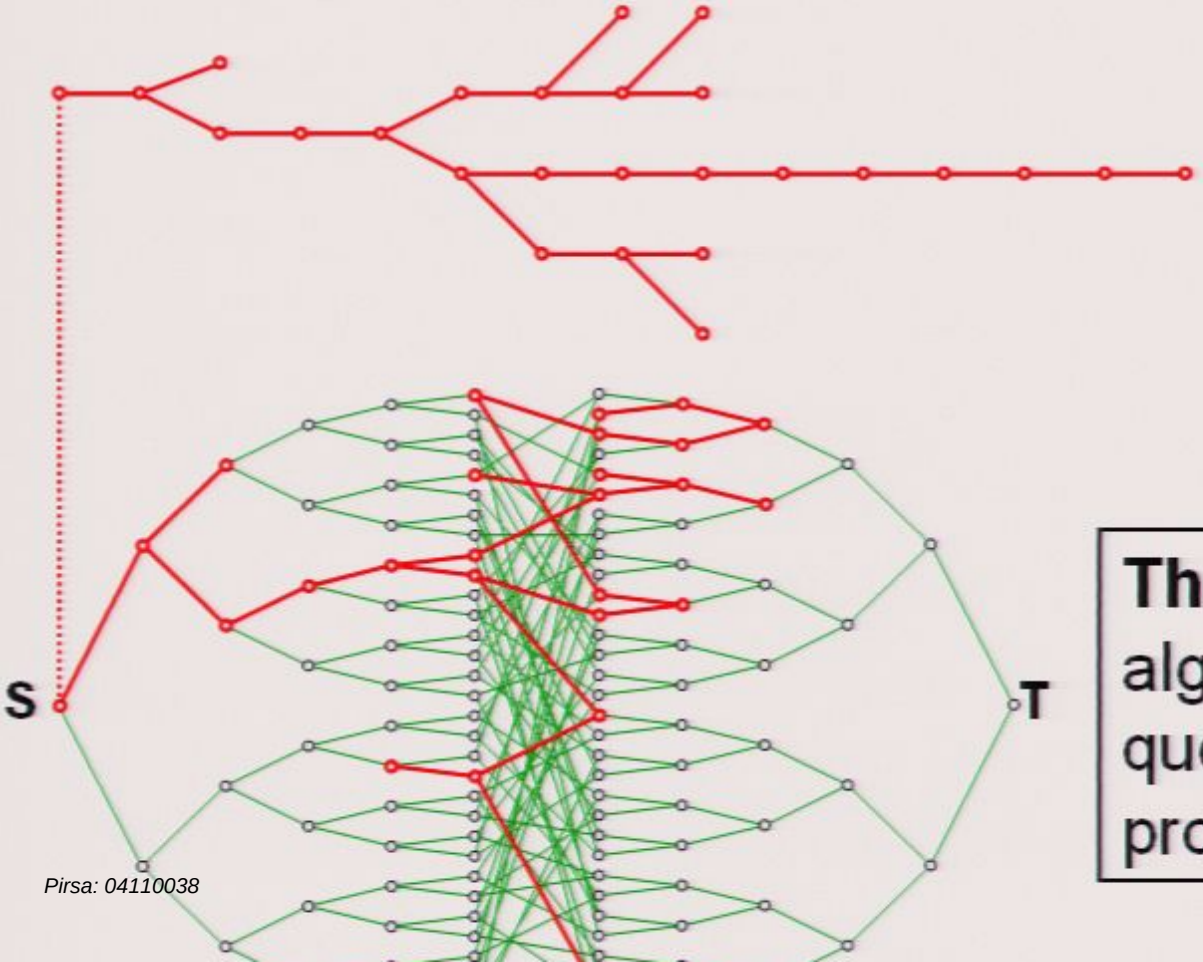
Classical hardness of ST-traversal

Lemma: randomly embedding ***any*** tree of size m into G "succeeds" with probability at most $m^2/2^{n/2}$



Classical hardness of ST-traversal

Lemma: randomly embedding ***any*** tree of size m into G "succeeds" with probability at most $m^2/2^{n/2}$



Theorem: *any* classical algorithm making $\leq 2^{n/6}$ queries succeeds with probability $\leq 4 \cdot 2^{-n/6}$

Concluding remarks

- Exponential quantum vs. classical speedup using a quantum walk algorithm
- Results easily adaptable to ***ST-connectivity*** problem
- Quantum algorithm solves **ST**-traversal without finding a path from **S** to **T**!

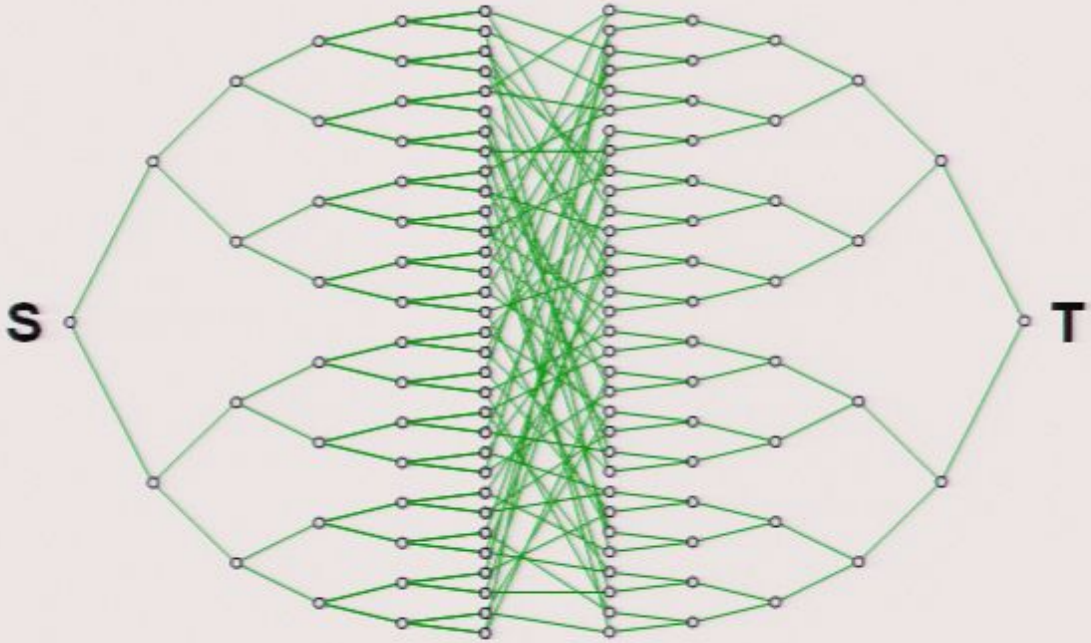
Open questions:

- Are there ***useful*** problems that can solved efficiently by quantum walks? Progress by [Ambainis '04]
- Is there a graph where **ST**-connectivity can be ***decided*** efficiently, but an **ST**-path cannot be found efficiently?

Classical hardness of ST-traversal

Define ***success*** as either finding T or finding a cycle

Then, prior to success, algorithm is growing a tree on G



Classical hardness of ST-traversal