Path Detection: A Quantum Computing Primitive

Shelby Kimmel

Middlebury College

Based on work with
Stacey Jeffery: arXiv: 1704.00765 (Quantum vol 1 p 26)
Michael Jarret, Stacey Jeffery, Alvaro Piedrafita, arXiv:1804.10591 (ESA 2018)
Kai DeLorenzo, Teal Witter, arXiv:1904.05995 (TQC 2019)

Middlebury

How to make quantum algorithms accessible?

How to make quantum algorithms accessible?

- Need quantum algorithmic primitives

How to make quantum algorithms accessible?

- Need quantum algorithmic primitives
  1. Widely applicable
  2. Easy to understand and analyze (without knowing quantum mechanics)

How to make quantum algorithms accessible?

- Need quantum algorithmic primitives
  1. Widely applicable
  2. Easy to understand and analyze (without knowing quantum mechanics)

  – Ex: Searching unordered list of n items
    – Classically, takes $\Omega(n)$ time
    – Quantumly, takes $O(\sqrt{n})$ time

How to make quantum algorithms accessible?

- Need quantum algorithmic primitives
  1. Widely applicable
  2. Easy to understand and analyze (without knowing quantum mechanics)

  – Ex: Searching unordered list of n items
    – Classically, takes $\Omega(n)$ time
    – Quantumly, takes $O(\sqrt{n})$ time

- New primitive: ***st*-connectivity**

Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Widely applicable
   2. Easy to analyze (without knowing quantum mechanics)
C. Examples

Outline:

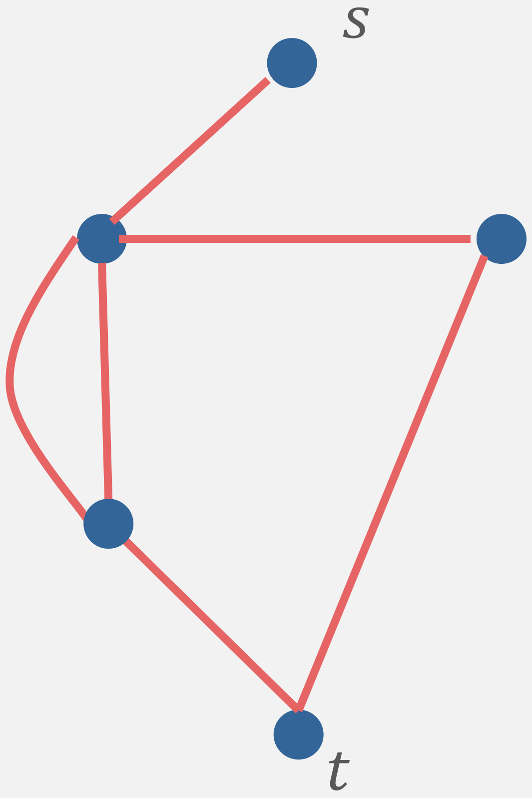
A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Widely applicable
   2. Easy to analyze (without knowing quantum mechanics)
C. Examples

Applications:

- Read-once Boolean formulas (query optimal)
- Total connectivity (query optimal)
- Cycle detection (query optimal)
- Bipartiteness (query optimal)

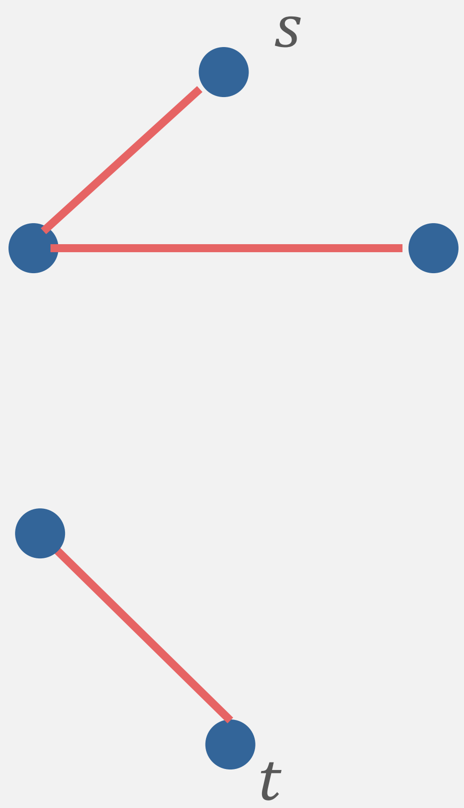
st-connectivity

$\boldsymbol{st - connectivity}$:
is there a path from s to t?



st-connectivity

$\boldsymbol{st - connectivity}$:
is there a path from s to t?



Input to Algorithm

Bit String:

x_1 x_2 … x_n

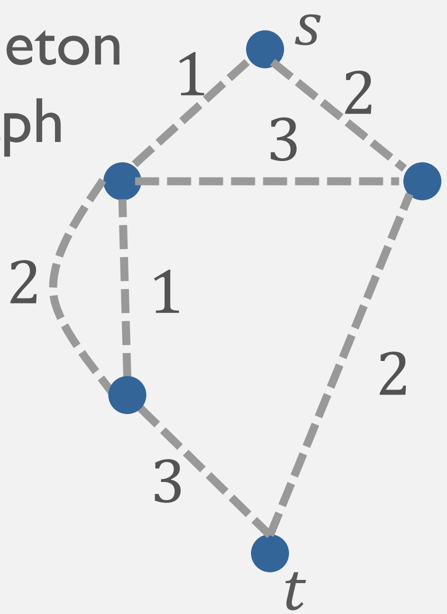
Input to Algorithm

Bit String:

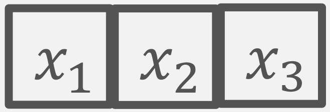
x_1	x_2	…	x_n
1	0		1

Input to Algorithm

Skeleton Graph

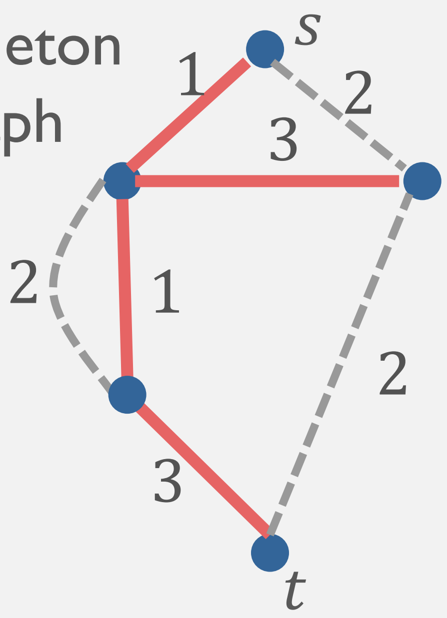


Bit String:



Input to Algorithm

Skeleton Graph

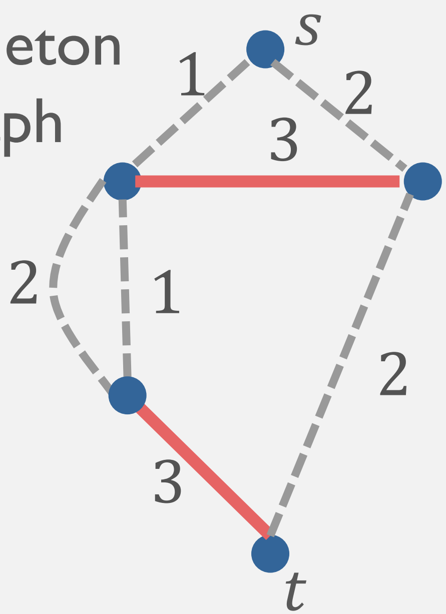


Bit String:

x_1	x_2	x_3
1	0	1

Input to Algorithm

Skeleton Graph

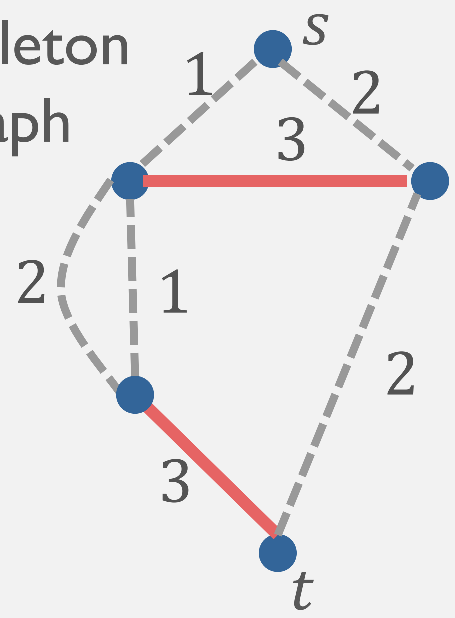


Bit String:

x_1	x_2	x_3
0	0	1

Input to Algorithm

Skeleton Graph



Bit String:

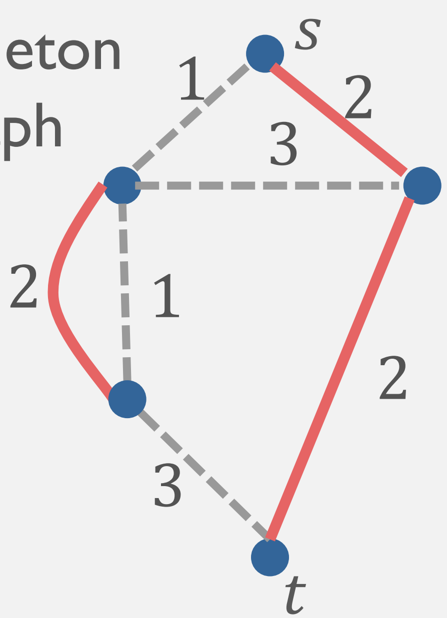
x_1	x_2	x_3
0	0	1

Catch:

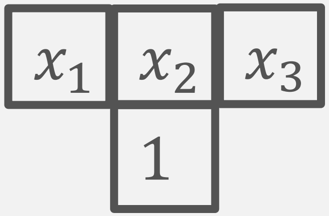
- Bit string initially hidden
- Goal: solve while revealing as few bits as possible → minimize **Query Complexity**

Input to Algorithm

Skeleton Graph



Bit String:

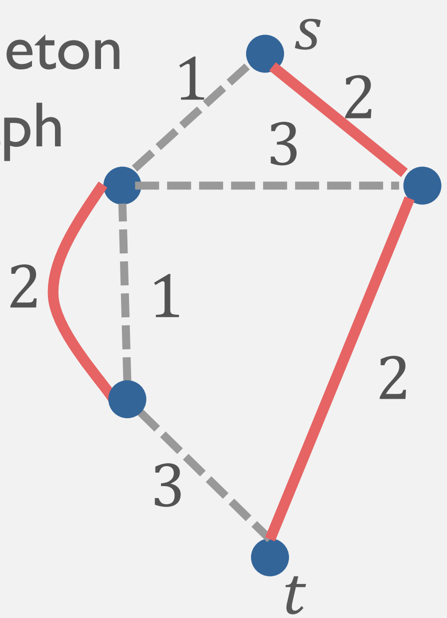


Catch:

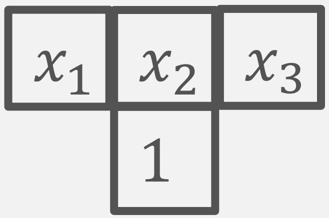
- Bit string initially hidden
- Goal: solve while revealing as few bits as possible → minimize **Query Complexity**

Input to Algorithm

Skeleton Graph



Bit String:



Catch:

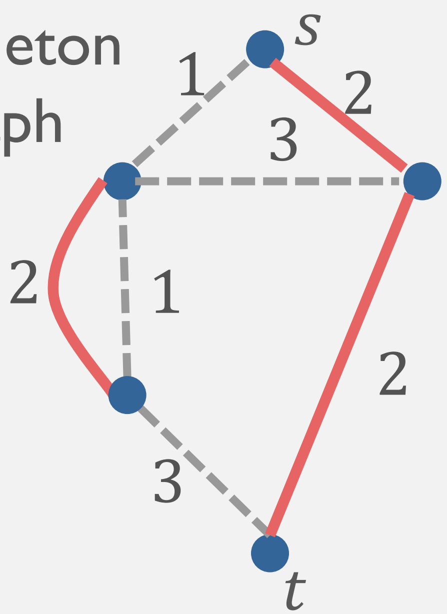
- Bit string initially hidden
- Goal: solve while revealing as few bits as possible → minimize **Query Complexity**

$\stackrel{?}{\approx}$

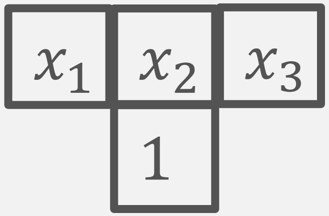
Time Complexity

Input to Algorithm

Skeleton Graph



Bit String:



Catch:

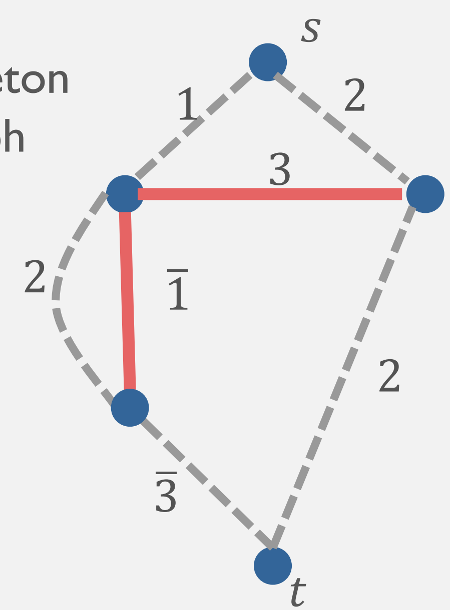
- Bit string initially hidden
- Goal: solve while revealing as few bits as possible → minimize **Query Complexity**

$\approx$?

Time Complexity

Input to Algorithm

Skeleton Graph



Bit String:

x_1	x_2	x_3
0	0	1

Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
   1. Widely applicable
      - Boolean Formulas
      - Cycle Detection
   2. Easy to analyze (without knowing quantum mechanics)
C. Example

Boolean Formulas

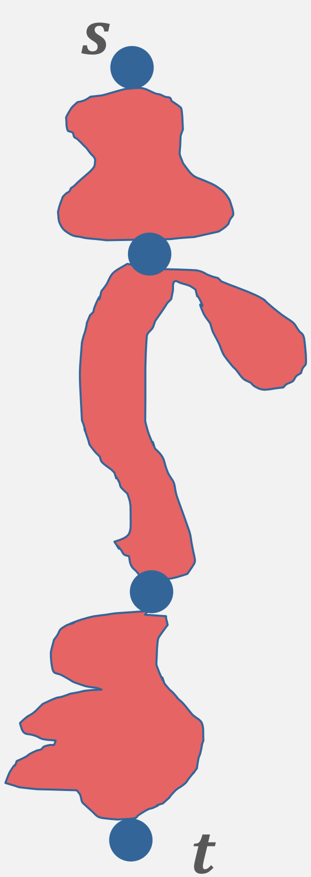
$\wedge$ AND: outputs 1 if all input subformulas have value 1

$\vee$ OR: outputs 1 if any input subformulas have value 1

Boolean Formulas

∧ AND: outputs 1 if all input subformulas have value 1

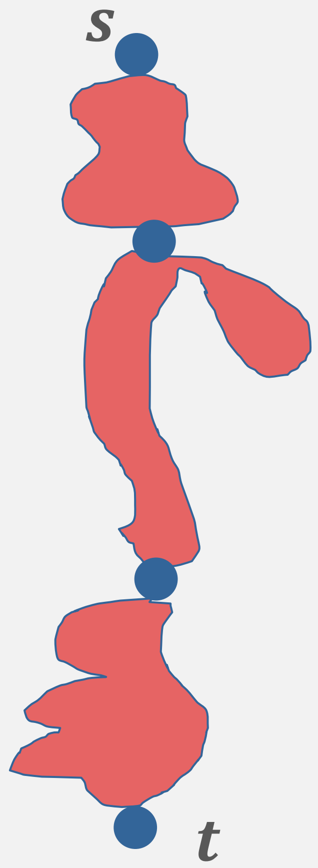
∨ OR: outputs 1 if any input subformulas have value 1



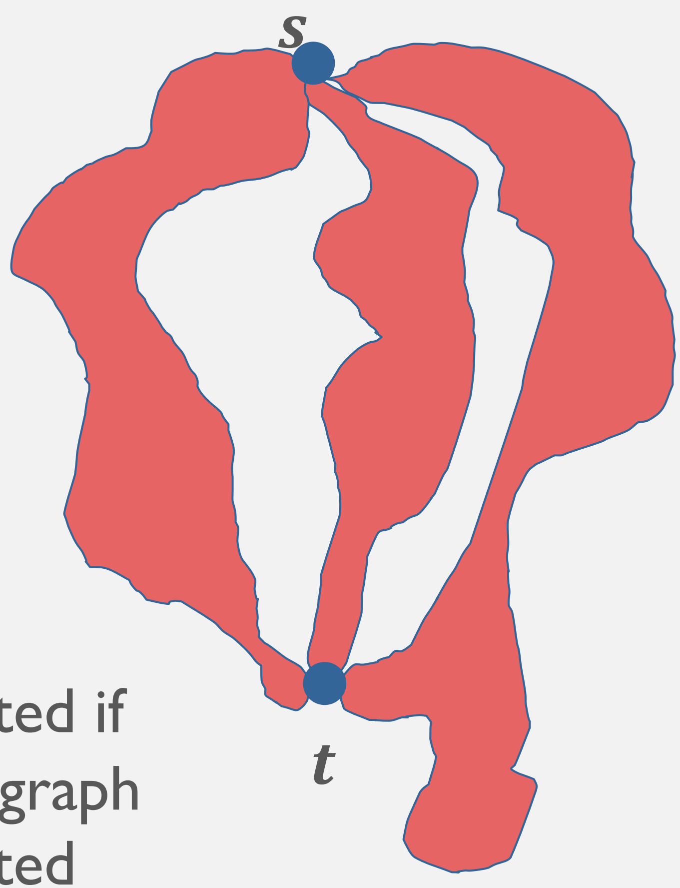
s and t connected if all subgraphs connected

Boolean Formulas

$\wedge$ AND: outputs 1 if all input subformulas have value 1

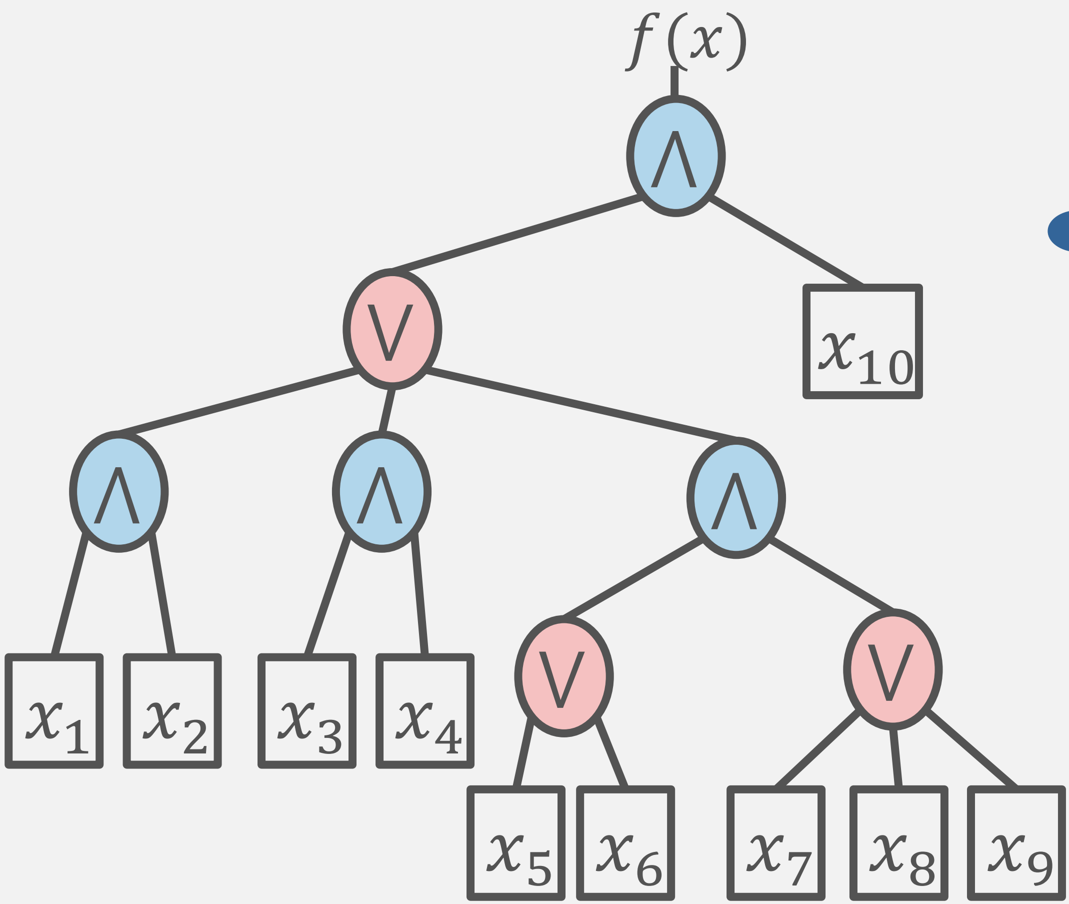


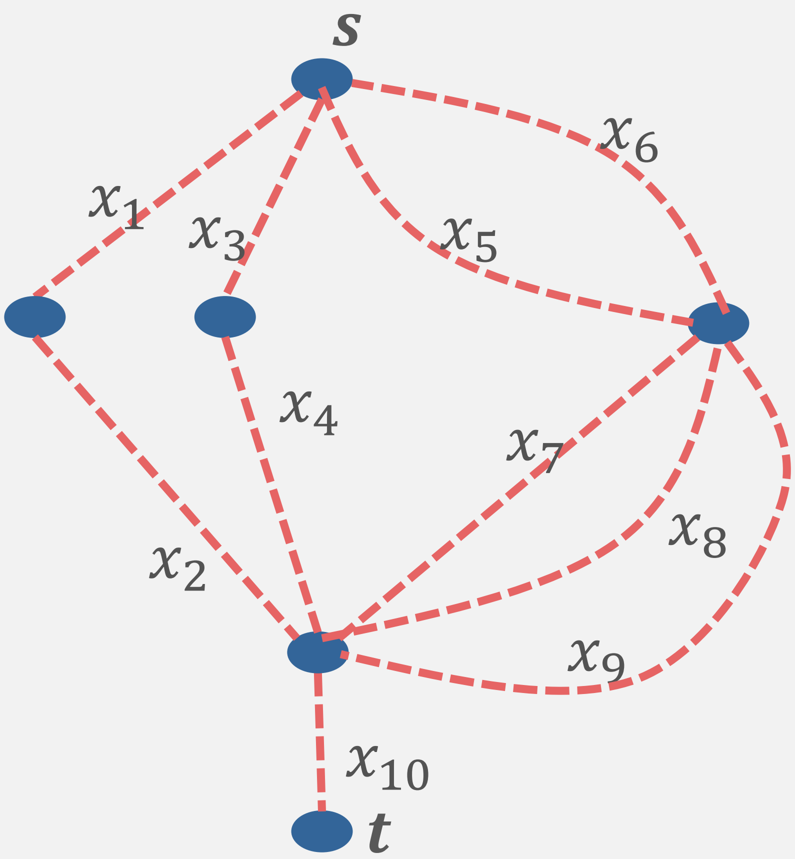
$\vee$ OR: outputs 1 if any input subformulas have value 1



s and t connected if any subgraph connected

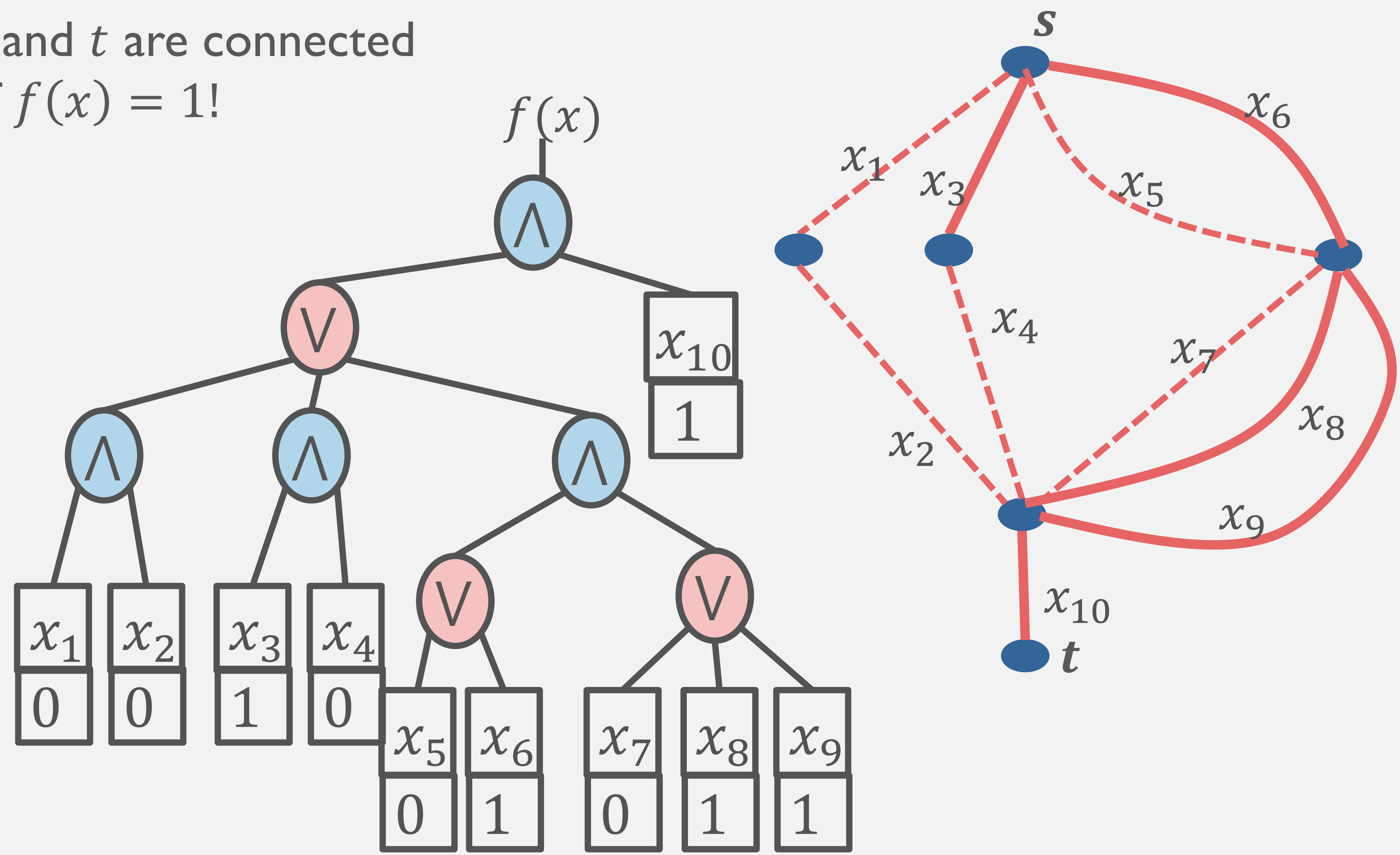
Boolean Formulas



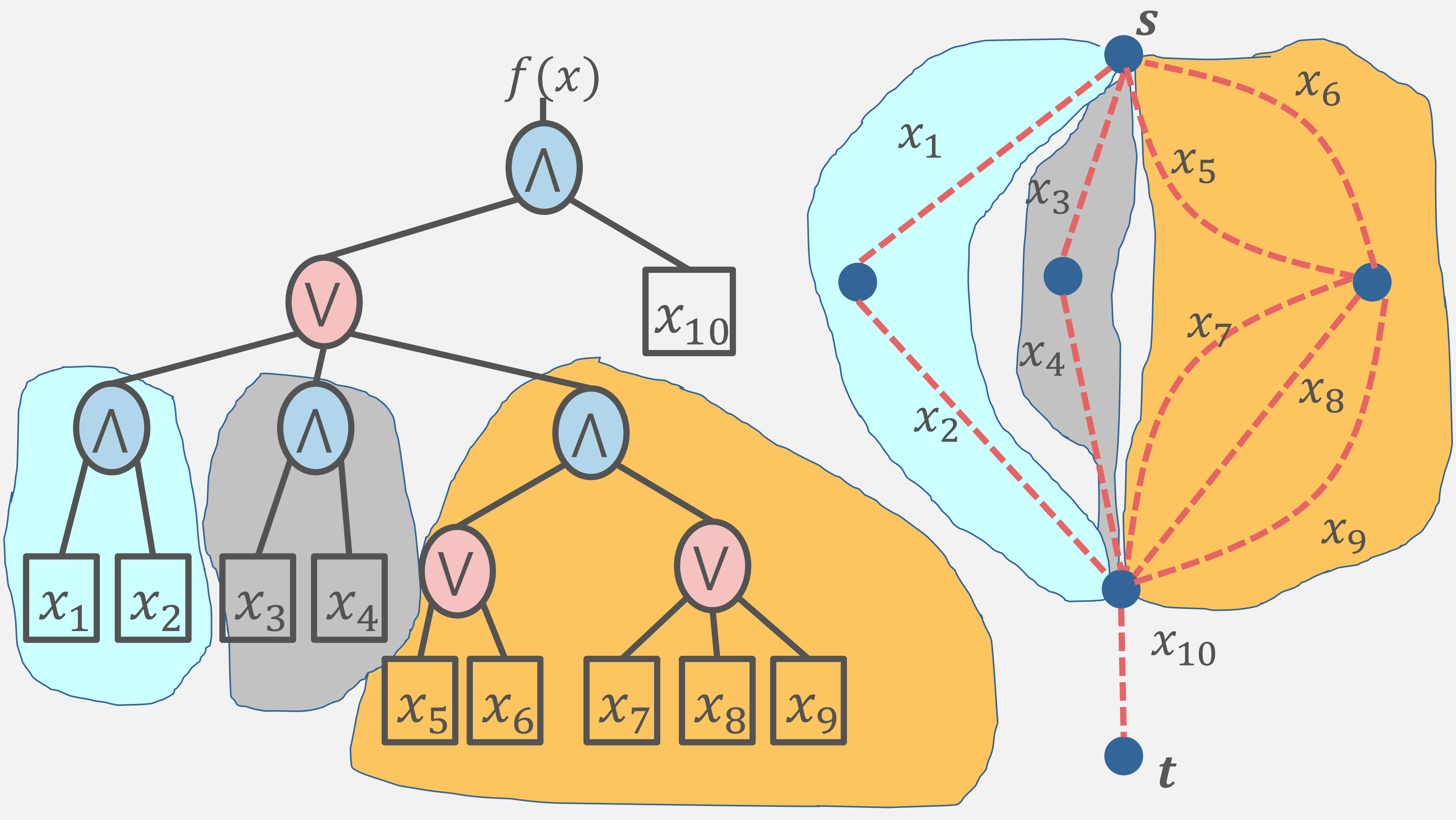


Boolean Formulas

s and t are connected
iff $f(x) = 1$!



Boolean Formulas

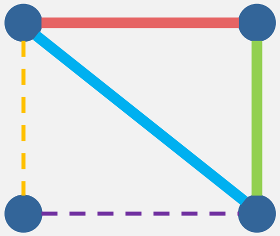


Boolean Formula Applications

- Logic
- Designing electrical circuits
- Game theory (deciding who will win a game)
- Combinatorics and graph problems
- Linear programming
- Testing potential solution to an NP-complete problem

Cycle Detection

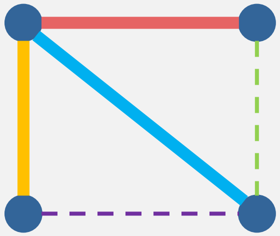
Is there a cycle?



Yes

Cycle Detection

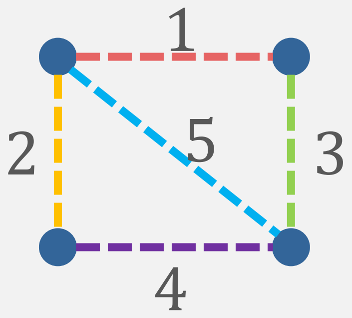
Is there a cycle?



No

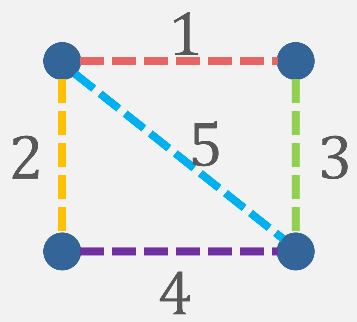
Cycle Detection

Is there a cycle through edge 1?



Cycle Detection

Is there a cycle through edge 1?

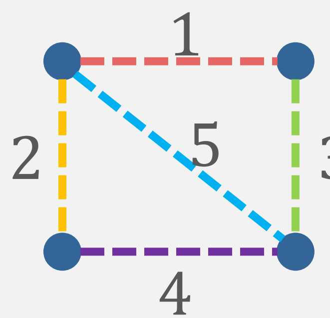


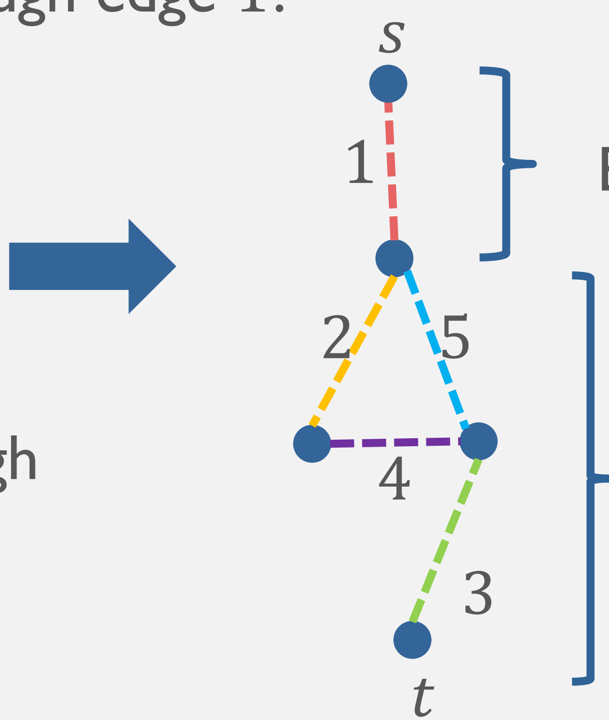
There is a cycle through Edge 1 iff

- Edge 1 is present
- Path between the endpoints of Edge 1 not using Edge 1

Cycle Detection

Is there a cycle through edge 1?





Edge 1 is present

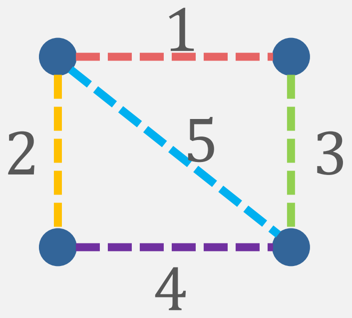
Path between the endpoints of Edge 1 not using Edge 1

There is a cycle through Edge 1 iff

- Edge 1 is present
- Path between the endpoints of Edge 1 not using Edge 1

Cycle Detection

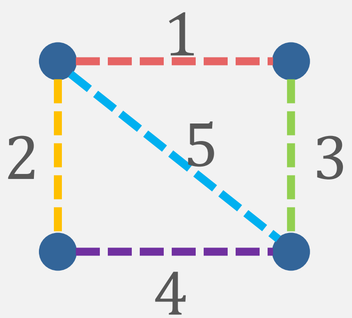
Is there a cycle?

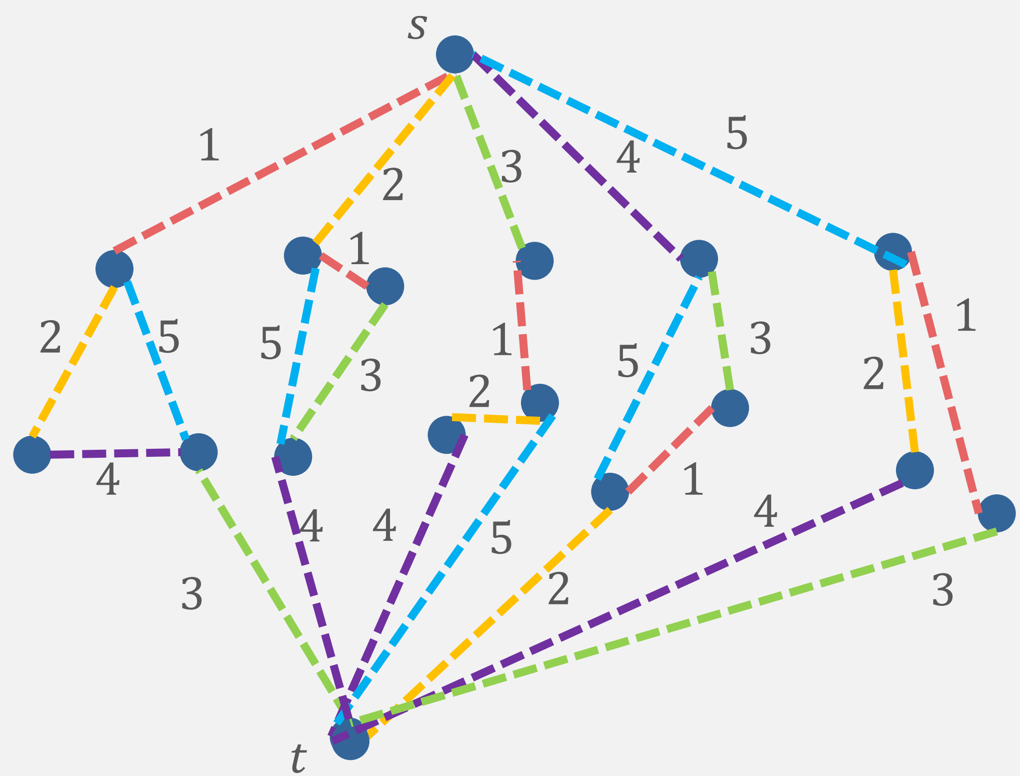


There is a cycle if there is a cycle through some edge

Cycle Detection

Is there a cycle?





There is a cycle if there is a cycle through some edge

Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
    1. Widely applicable
    2. Easy to analyze (without knowing quantum mechanics)
C. Extra example

Algorithm Complexity:

Space Complexity: $O(\log(\#\ edges\ in\ skeleton\ graph))$

Algorithm Complexity:

Query Complexity:

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

Effective resistance

Effective capacitance

Algorithm Complexity:

Query Complexity:

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

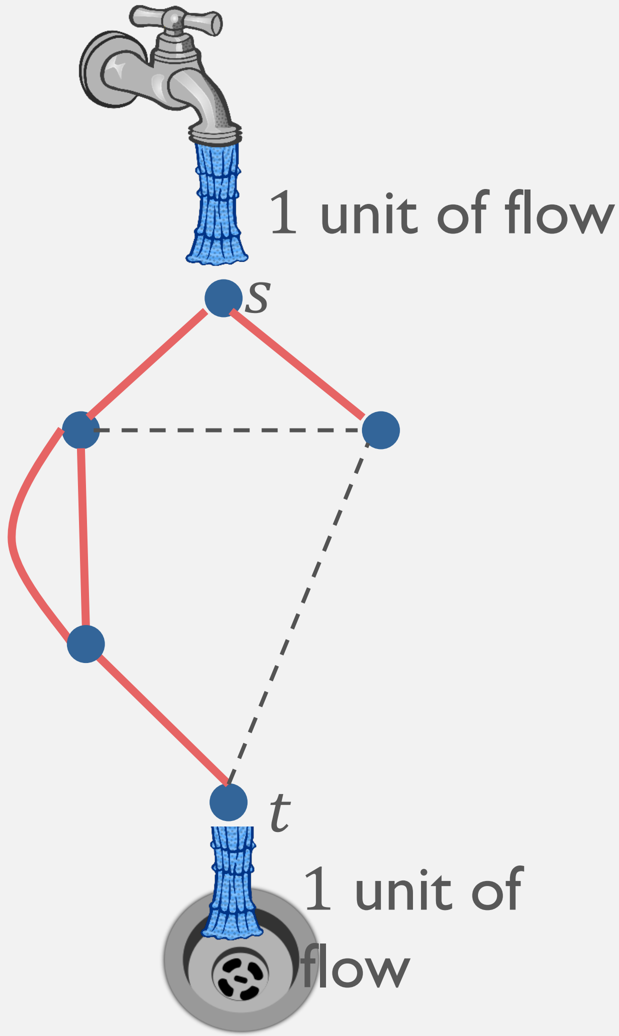
Effective resistance

[Belovs, Reichardt, '12]

Effective capacitance

[Jarret, Jeffery, Kimmel, Piedrafita, '18]

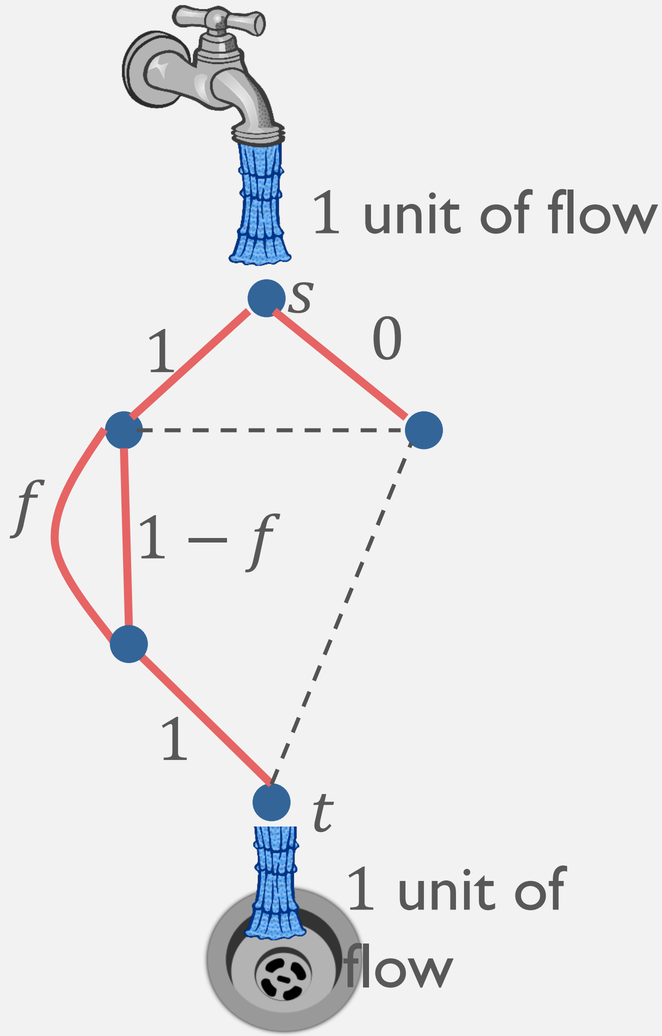
Effective Resistance



Effective Resistance

Valid flow:

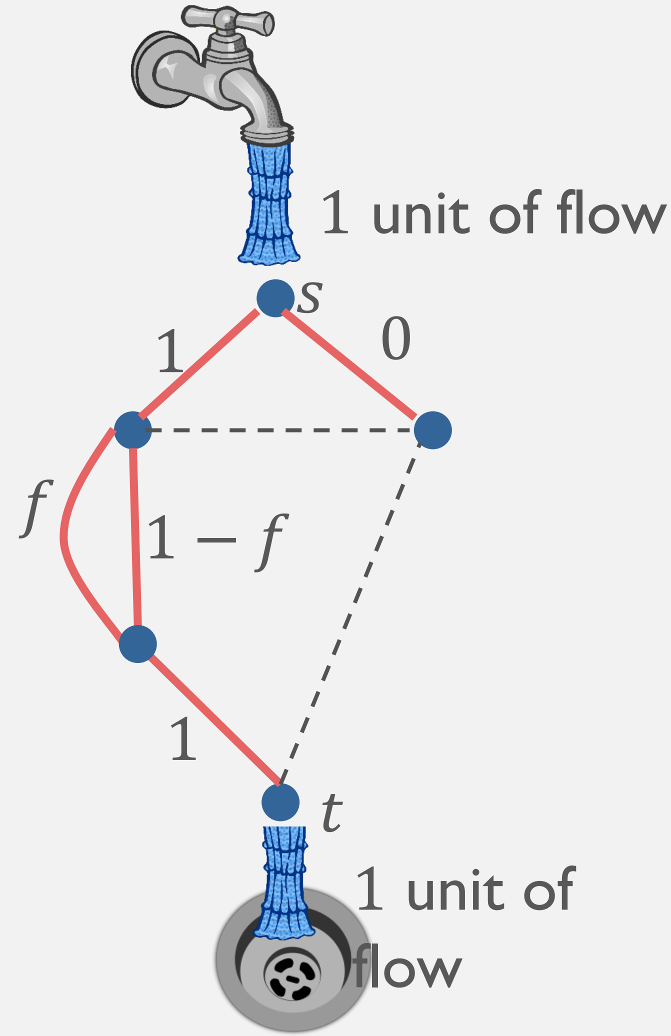
- 1 unit in at s
- 1 unit out at t
- At all other nodes, zero net flow



Effective Resistance

Flow energy:

$$\sum_{edges} (flow\ on\ edge)^2$$



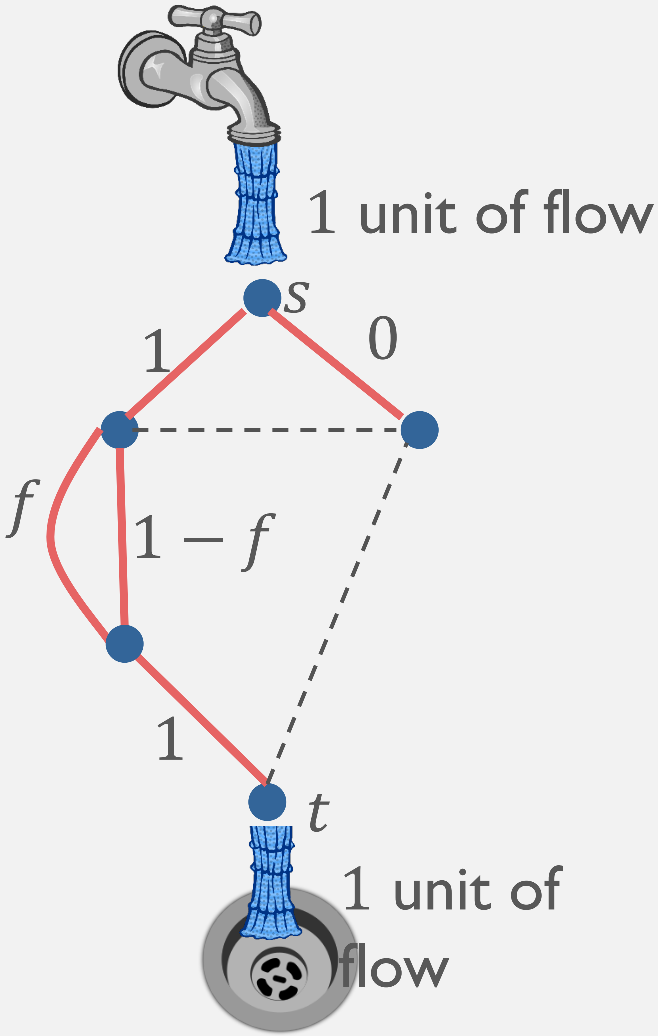
Effective Resistance

Flow energy:

$$\sum_{edges} (flow\ on\ edge)^2$$

Effective Resistance: $R_{s,t}(G)$

- Smallest energy of any valid flow from s to t on G.



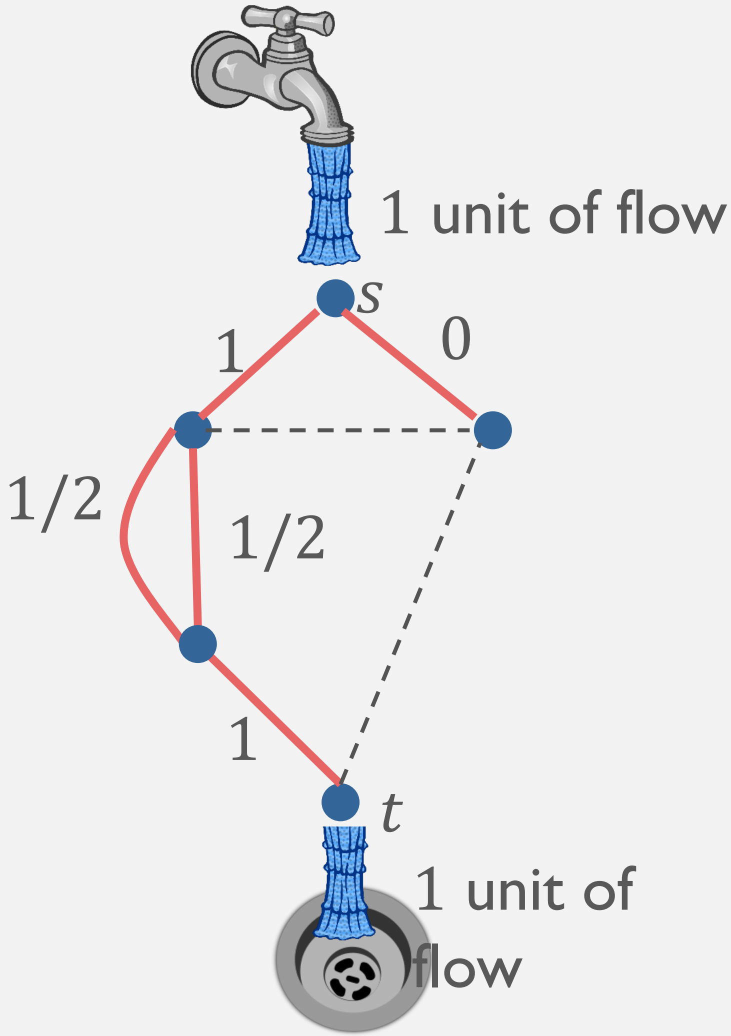
Effective Resistance

Flow energy:

$$\sum_{edges} (flow\ on\ edge)^2$$

Effective Resistance: $R_{s,t}(G)$

- Smallest energy of any valid flow from s to t on G.



Algorithm Complexity:

Query Complexity:

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

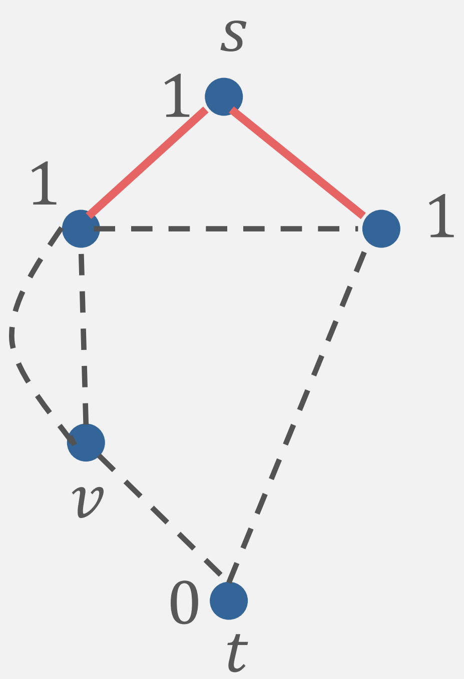
Effective resistance

Effective capacitance

Effective Capacitance

Generalized cut:

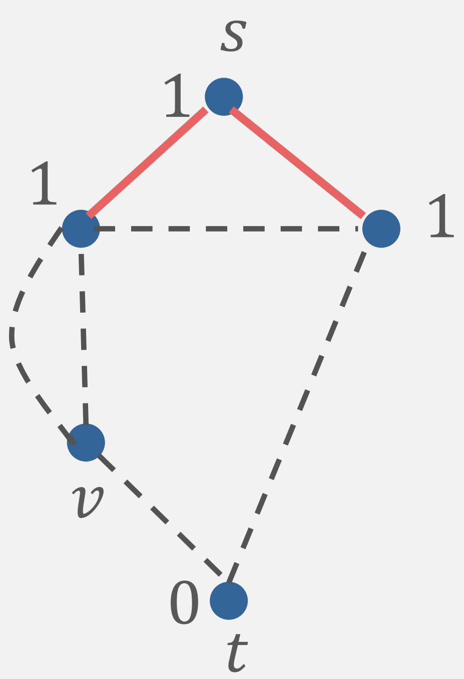
- 1 at s
- 0 at t
- Difference is 0 across edge



Effective Capacitance

Potential energy:

$$\sum_{\substack{edges\ in \\ skeleton\ graph}} (cut\ difference)^2$$



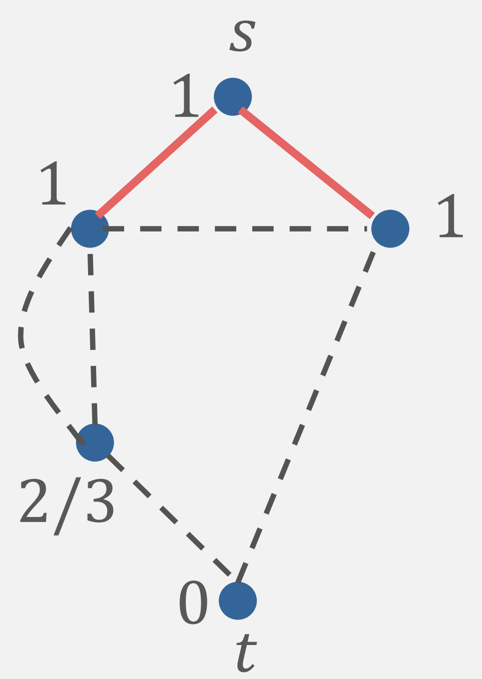
Effective Capacitance

Potential energy:

$$\sum_{\substack{edges\ in \\ skeleton\ graph}} (cut\ difference)^2$$

Effective Capacitance: $C_{s,t}(G)$

- Smallest potential energy of any valid generalized cut between s and t on G.



Algorithm Complexity:

Query Complexity:

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

Effective resistance

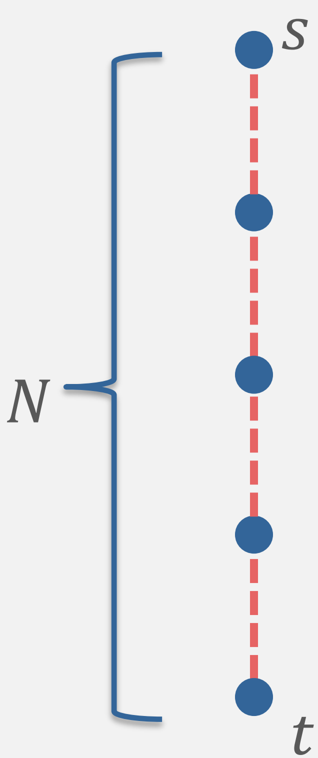
Effective capacitance

Example

Decide $AND(x_1, x_2, \ldots, x_N)$, if

- All $x_i = 1$, or
- At least k input bits are 0.

Example



Decide $AND(x_1, x_2, \dots, x_N)$, if

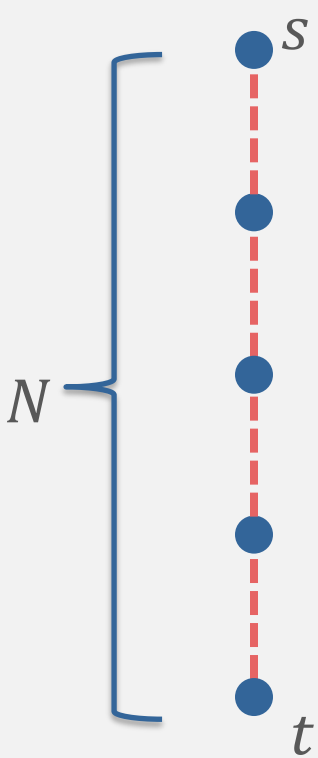
- All $x_i = 1$, or
- At least k input bits are 0.

⇕

Decide if

- s and t are connected, or
- At least k edges are missing

Example

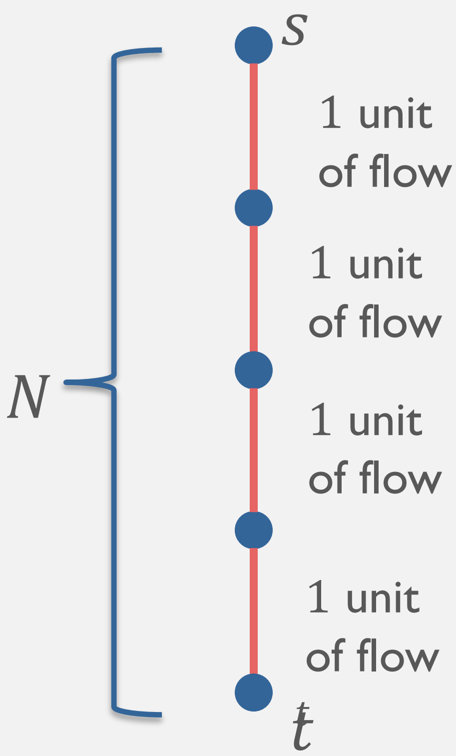


Decide if

- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

Example

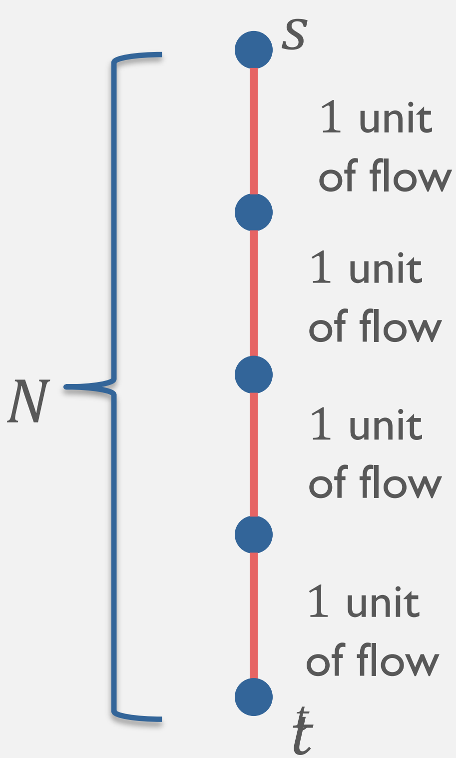


Decide if

- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

Example



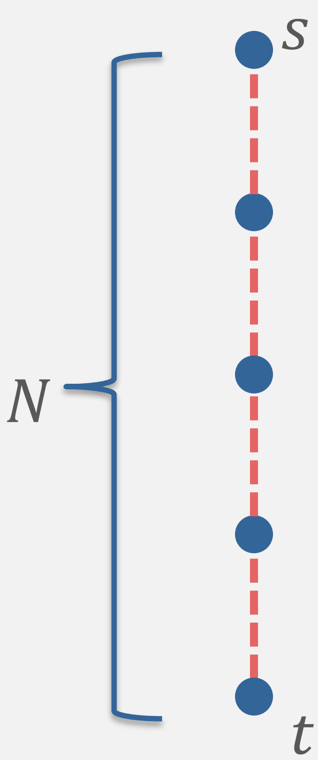
Decide if

- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

$$\max_{connected\ G} R_{s,t}(G) = N$$

Example

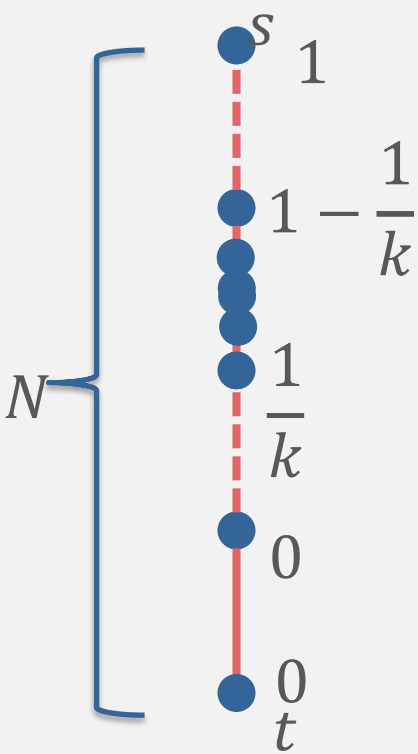


Decide if

- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

Example



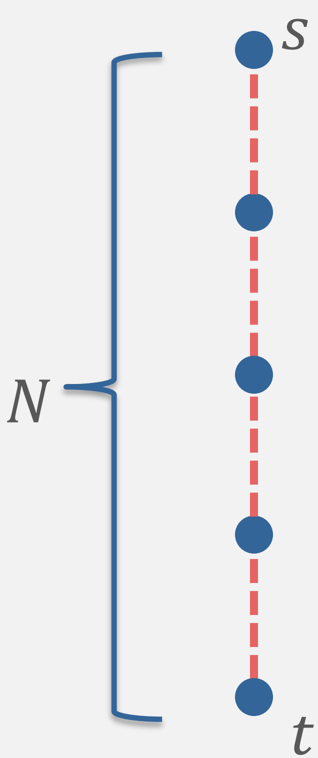
Decide if

- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{\text{connected } G} R_{s,t}(G)}\sqrt{\max_{\text{not connected } G} C_{s,t}(G)}\right)$$

$$\max_{\text{not connected } G} C_{s,t}(G) = k \times \left(\frac{1}{k}\right)^2 = \frac{1}{k}$$

Example



Decide if

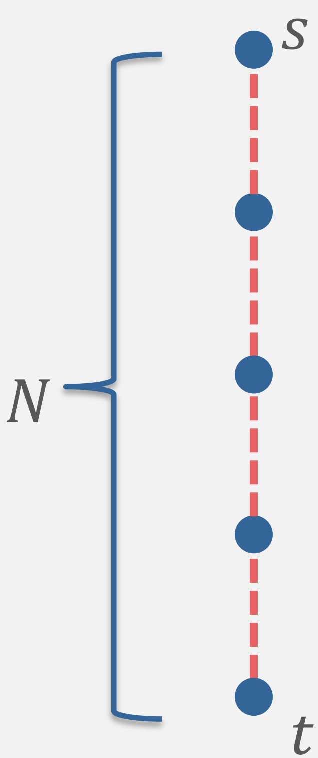
- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

$\max_{connected\ G} R_{s,t}(G) \rightarrow N$

$\max_{not\ connected\ G} C_{s,t}(G) \rightarrow 1/k$

Example



Decide if

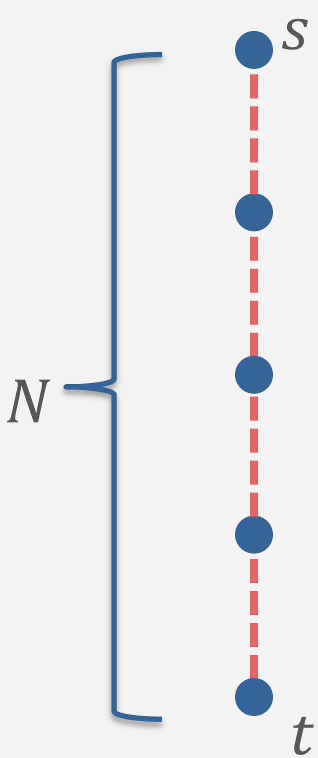
- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

N $1/k$

Quantum complexity is $O\left(\sqrt{N/k}\right)$ (optimal)

Example



Decide if

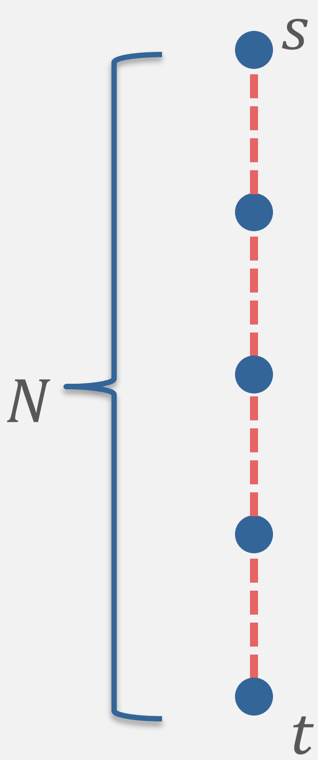
- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

$\downarrow$ N $\qquad$ $\downarrow$ $1/k$

Quantum complexity is $O\left(\sqrt{N/k}\right)$ (optimal)

Example



Decide if

- s and t are connected, or
- At least k edges are missing

$$O\left(\sqrt{\max_{\text{connected } G} R_{s,t}(G)}\sqrt{\max_{\text{not connected } G} C_{s,t}(G)}\right)$$

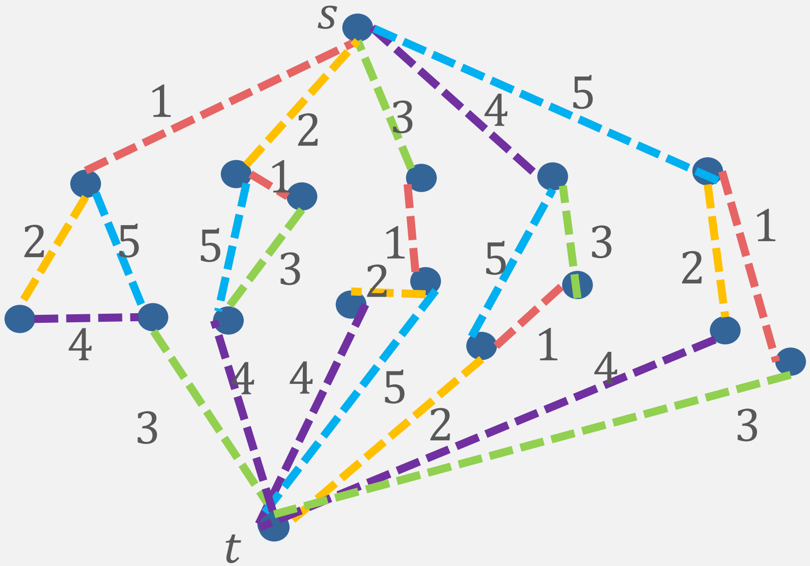
$\downarrow$ N $\downarrow$ $1/k$

Quantum complexity is $O\left(\sqrt{N/k}\right)$ (optimal)

Randomized classical complexity is $\Omega(N/k)$

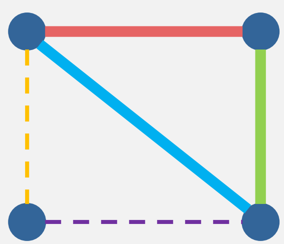
Example

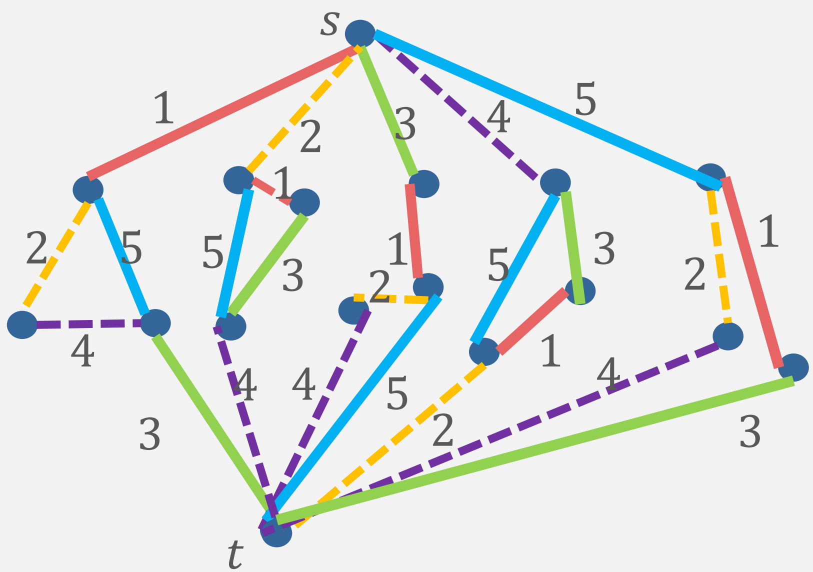
Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$



Example

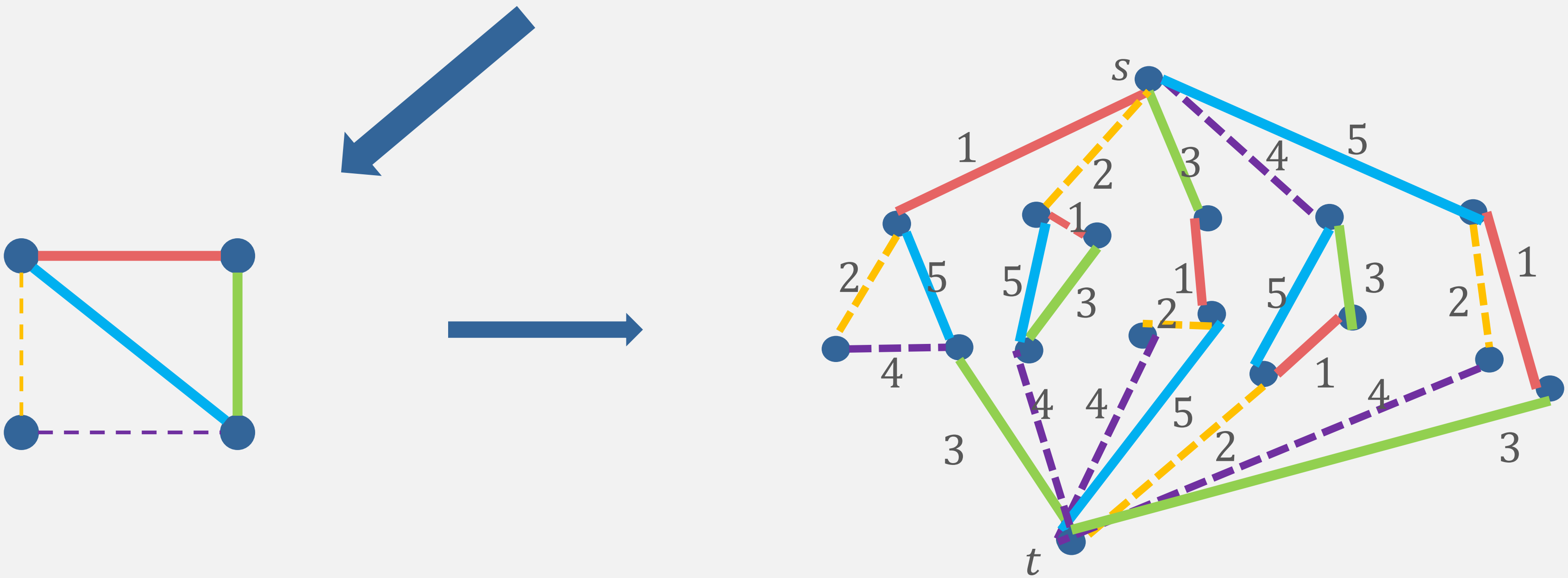
Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$





Example

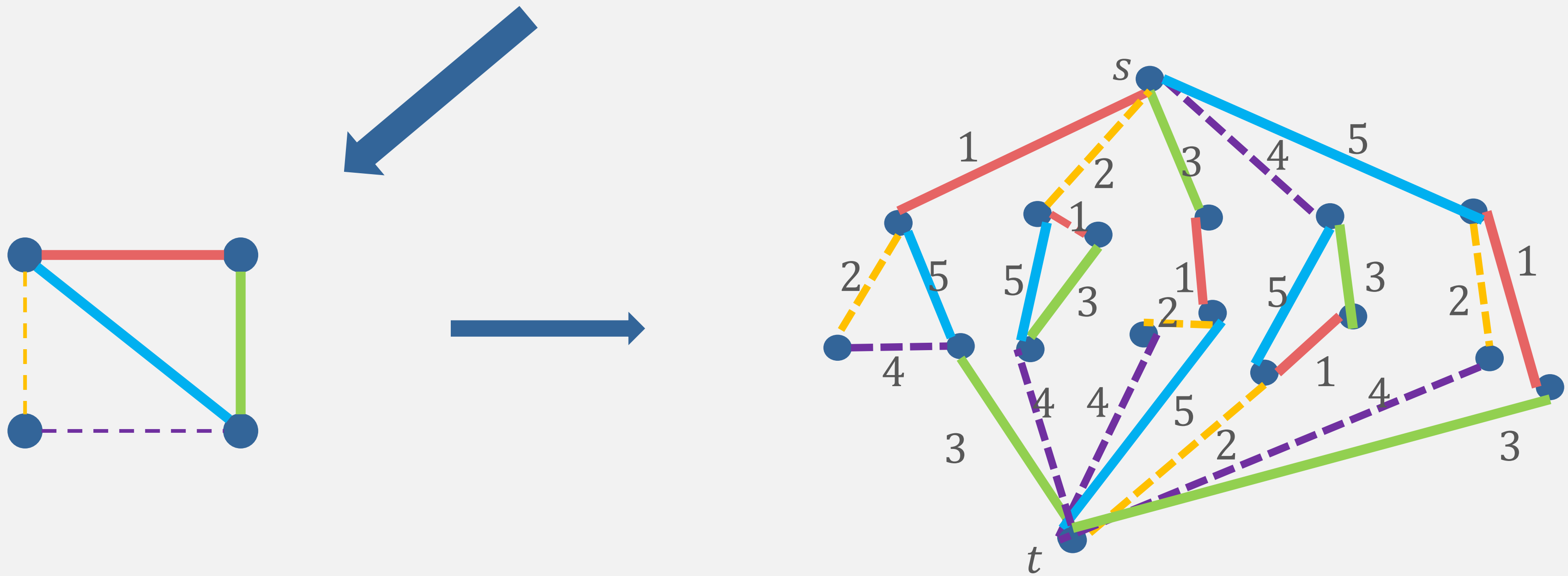
Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$



$R_{s,t}(G) = 1$

Example

Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$



More generally: $R_{s,t}(G) = (circuit\ rank)^{-1} \leq 1$

Circuit rank = min # of edges that must be cut to create a cycle free graph

Example

Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$

$$R_{s,t}(G) = (circuit\ rank)^{-1}$$

Circuit rank = min # of edges that must be cut to create a cycle free graph

- Quantum algorithm picks out critical topological parameter
- If promised large circuit rank (if cycle exists), then cycle detection algorithm runs faster
- Proved by 2nd year university students

Example

Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$

$R_{s,t}(G) = (circuit\ rank)^{-1}$

$C_{s,t}(G) = O(n^3)$

Query complexity: $O\left(n^{3/2}\right)$

(optimal – logarithmic improvement over previous algorithm)

Bonus Algorithm:

Quantum query algorithm to estimate effective resistance or effective capacitance of G. (Jeffery, Ito '15)

Bonus Algorithm:

Quantum query algorithm to estimate effective resistance or effective capacitance of G. (Jeffery, Ito '15)

Because effective resistance depends directly on circuit rank, we now have a quantum algorithm to estimate circuit rank.

Recap

st-connectivity makes a good algorithmic primitive

1. Widely applicable
2. Easy to analyze (without knowing quantum mechanics)

Open Questions and Current Directions

- Time complexity (current research at QuSoft)
- How to choose edge weights?
- When is st-connectivity reduction optimal?
- What is the classical time/query complexity of st-connectivity in the black box model? Under the promise of small capacitance/resistance?

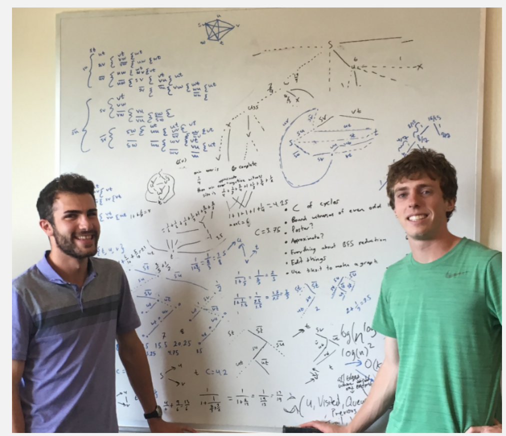
Thank you!

Stacey Jeffery

Michael Jarret

Lizeth Lucero

Alvaro Piedrafita



Teal Witter

Kai De Lorenzo