Path Detection: A Quantum Algorithmic 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

Primitives!

- Quantum algorithmic primitives
  1. Widely applicable
  2. Can be used in a black box manner (with easily analyzable behavior)

Primitives!

- Quantum algorithmic primitives
  1. Widely applicable
  2. Can be used in a black box manner (with easily analyzable behavior)

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

Primitives!

- Quantum algorithmic primitives
  1. Widely applicable
  2. Can be used in a black box manner (with easily analyzable behavior)

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

Good primitive: ***st*-connectivity**

Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
    1. Widely applicable
    2. Easy to analyze

Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
    1. Widely applicable
    2. Easy to analyze

Applications:

- Read-once Boolean formulas (query optimal) [J**K**]
- Total connectivity (query optimal) [JJ**K**P]
- Cycle detection (query optimal) [D**K**W]
- Even length cycle detection [D**K**W]
- Bipartiteness (query optimal) [D**K**W]
- Directed st-connectivity (query optimal) (Beigi et al '19)
- Directed smallest cycle (query optimal) (Beigi et al '19)

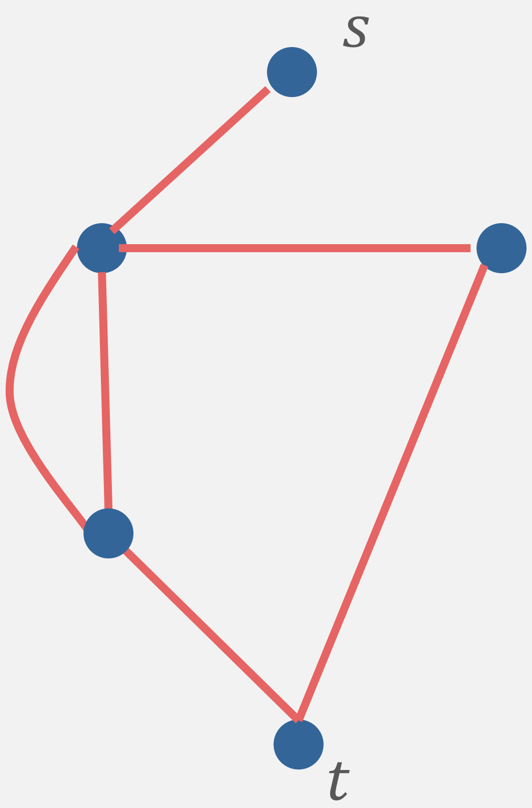
Outline:

Applications:

- Topological sort (Beigi et al '19)
- Connected components (Beigi et al '19)
- Strongly connected components (Beigi et al '19)
- k-cycle at vertex v (Beigi et al '19)
- st-connectivity (Reichardt, Belovs '12)

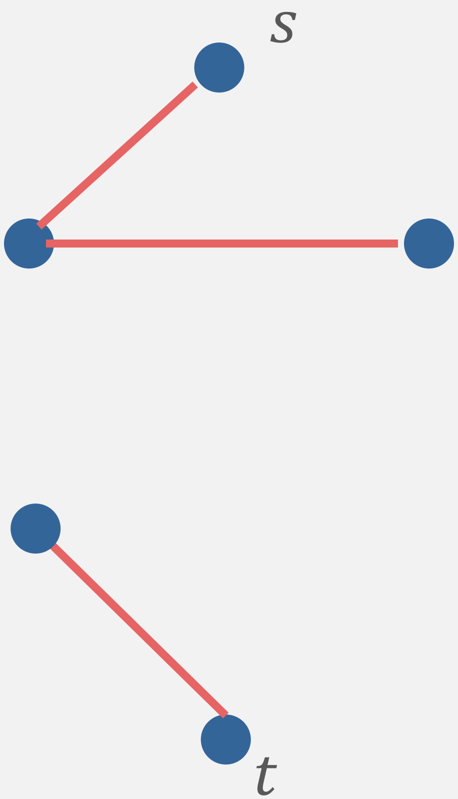
st-connectivity

$\boldsymbol{st - connectivity}$:
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

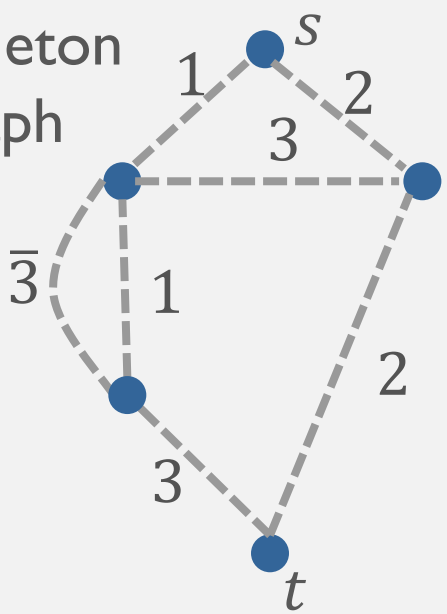
Input to Algorithm

Bit String:

x_1	x_2	…	x_n
1	0		1

Input to Algorithm

Skeleton Graph

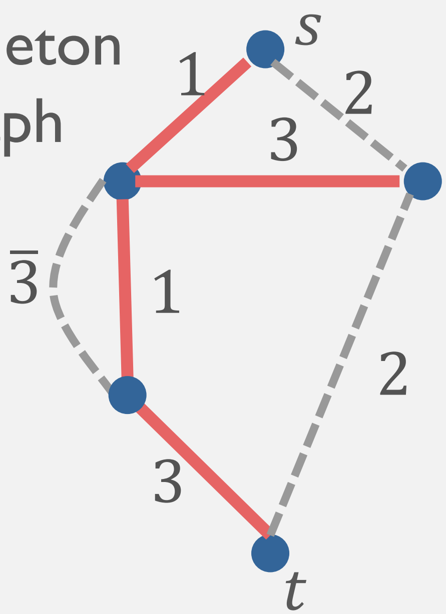


Bit String:

x_1	x_2	x_3

Input to Algorithm

Skeleton Graph

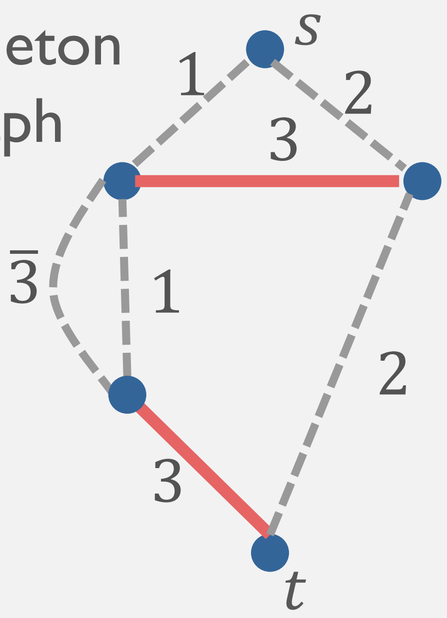


Bit String:

x_1	x_2	x_3
1	0	1

Input to Algorithm

Skeleton Graph

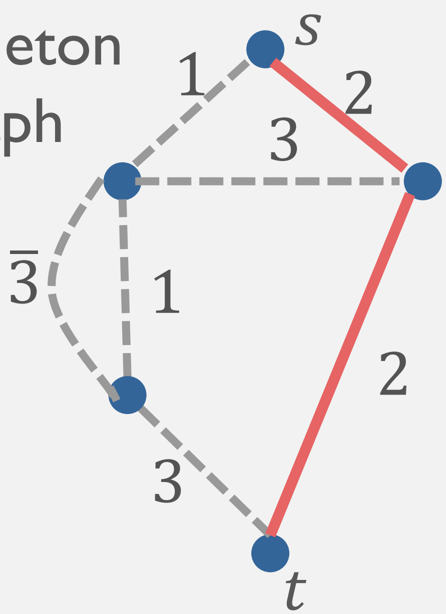


Bit String:

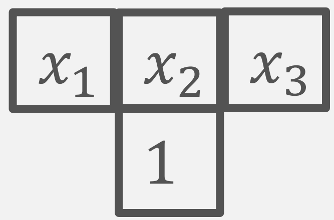
x_1	x_2	x_3
0	0	1

Input to Algorithm

Skeleton Graph



Bit String:

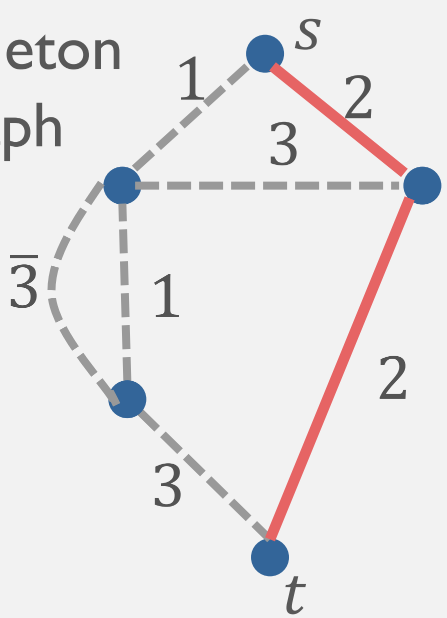


- Bit string initially hidden, can query value of string at each bit.

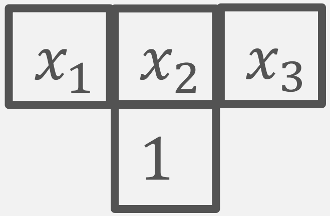
Input to Algorithm

$$O_x|i\rangle|b\rangle = |i\rangle|b + x_i\rangle$$

Skeleton Graph



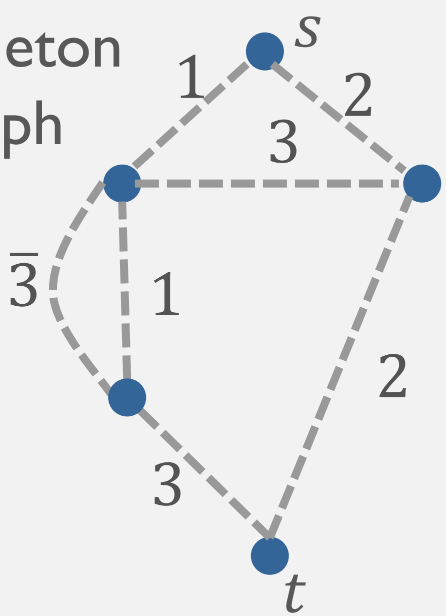
Bit String:

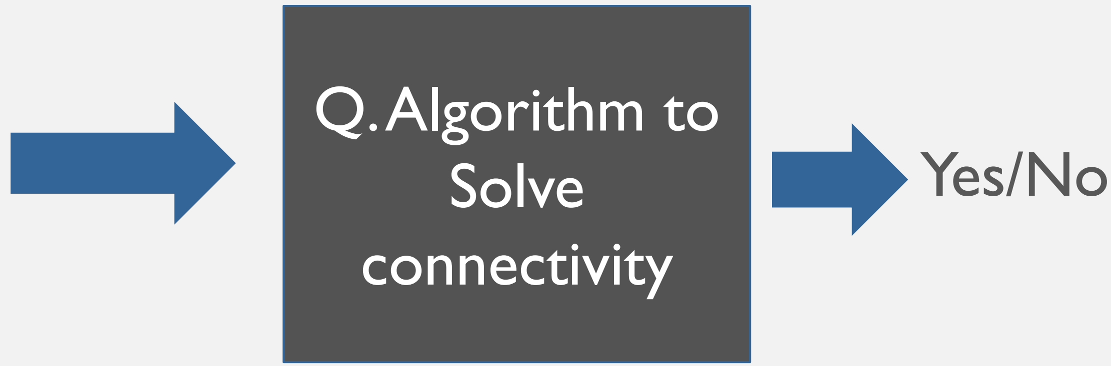


- Bit string initially hidden, can query value of string at each bit.

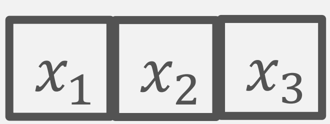
Algorithm

Skeleton Graph





O_x for Bit String:

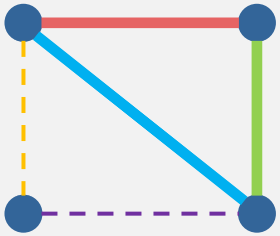


Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
    1. Widely applicable
    2. Easy to analyze

Cycle Detection

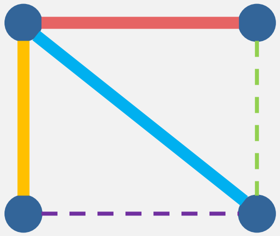
Is there a cycle?



Yes

Cycle Detection

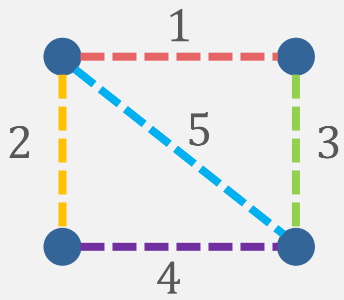
Is there a cycle?



No

Cycle Detection

Is there a cycle?



Input to Cycle Detection:

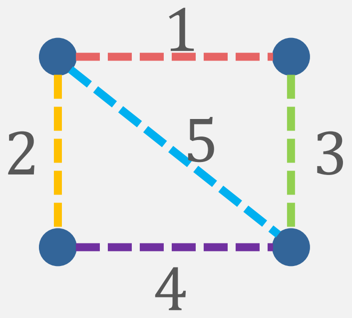
- Skeleton graph
- Hidden bit string

$$x_1 x_2 x_3 x_4 x_5$$

$x_i = 1 \leftrightarrow$ edge i is present

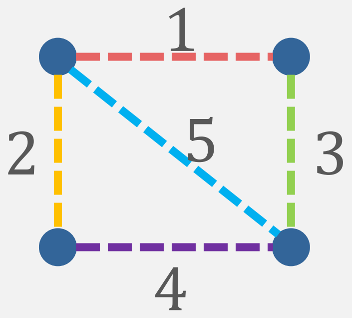
Cycle Detection

Is there a cycle through edge 1?



Cycle Detection

Is there a cycle through edge 1?

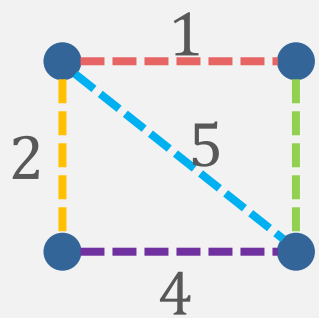


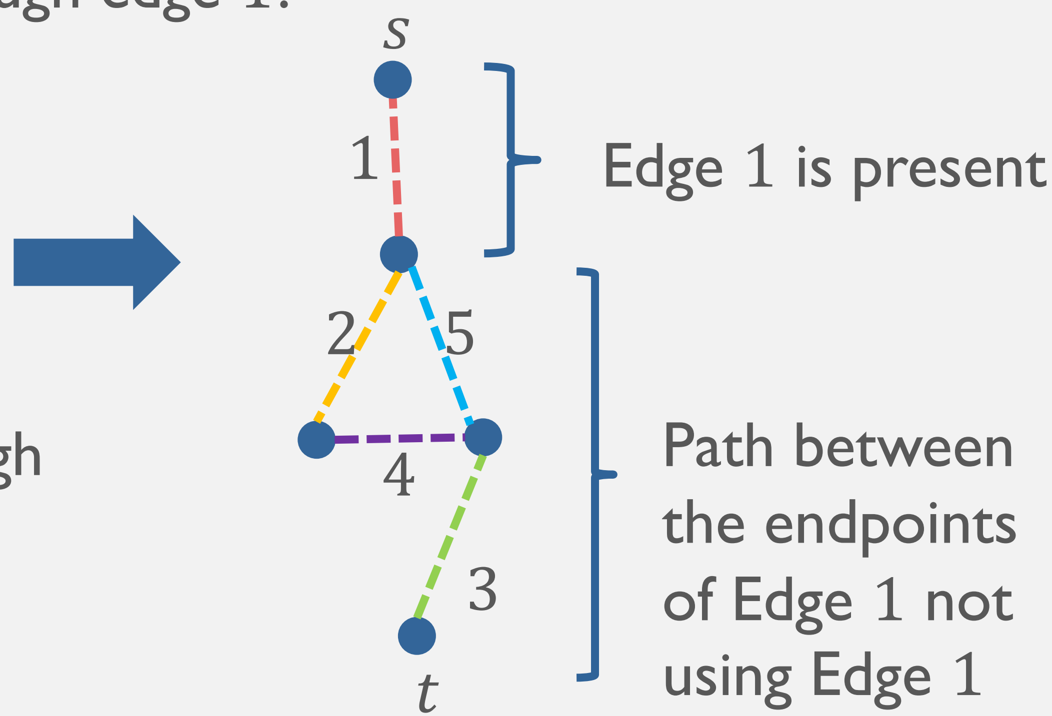
There is a cycle through Edge 1 iff

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

Cycle Detection

Is there a cycle through edge 1?



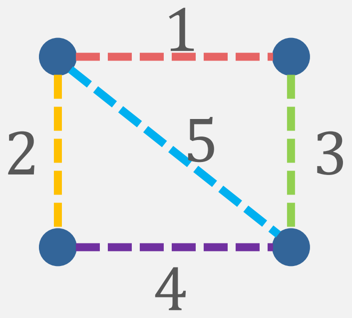


There is a cycle through Edge 1 iff

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

Cycle Detection

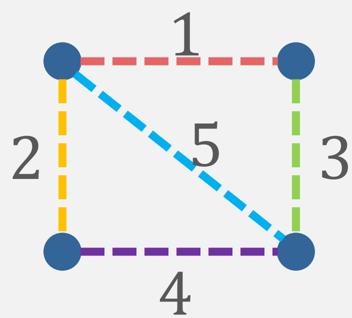
Is there a cycle?

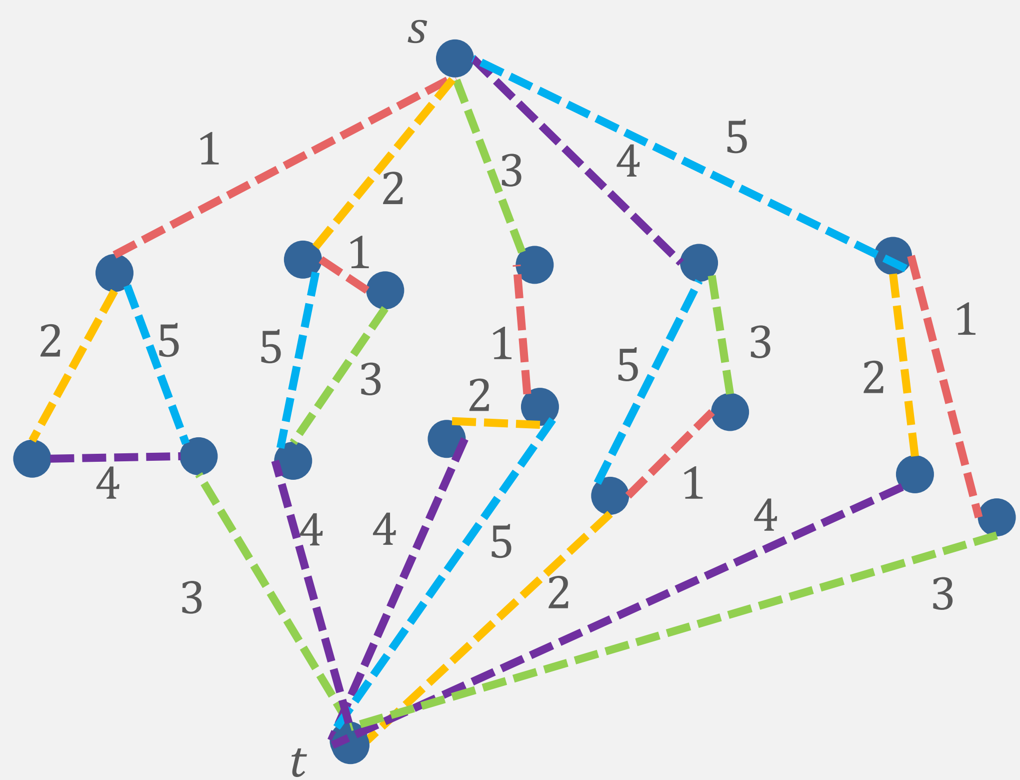


There is a cycle if there is a cycle through some edge

Cycle Detection

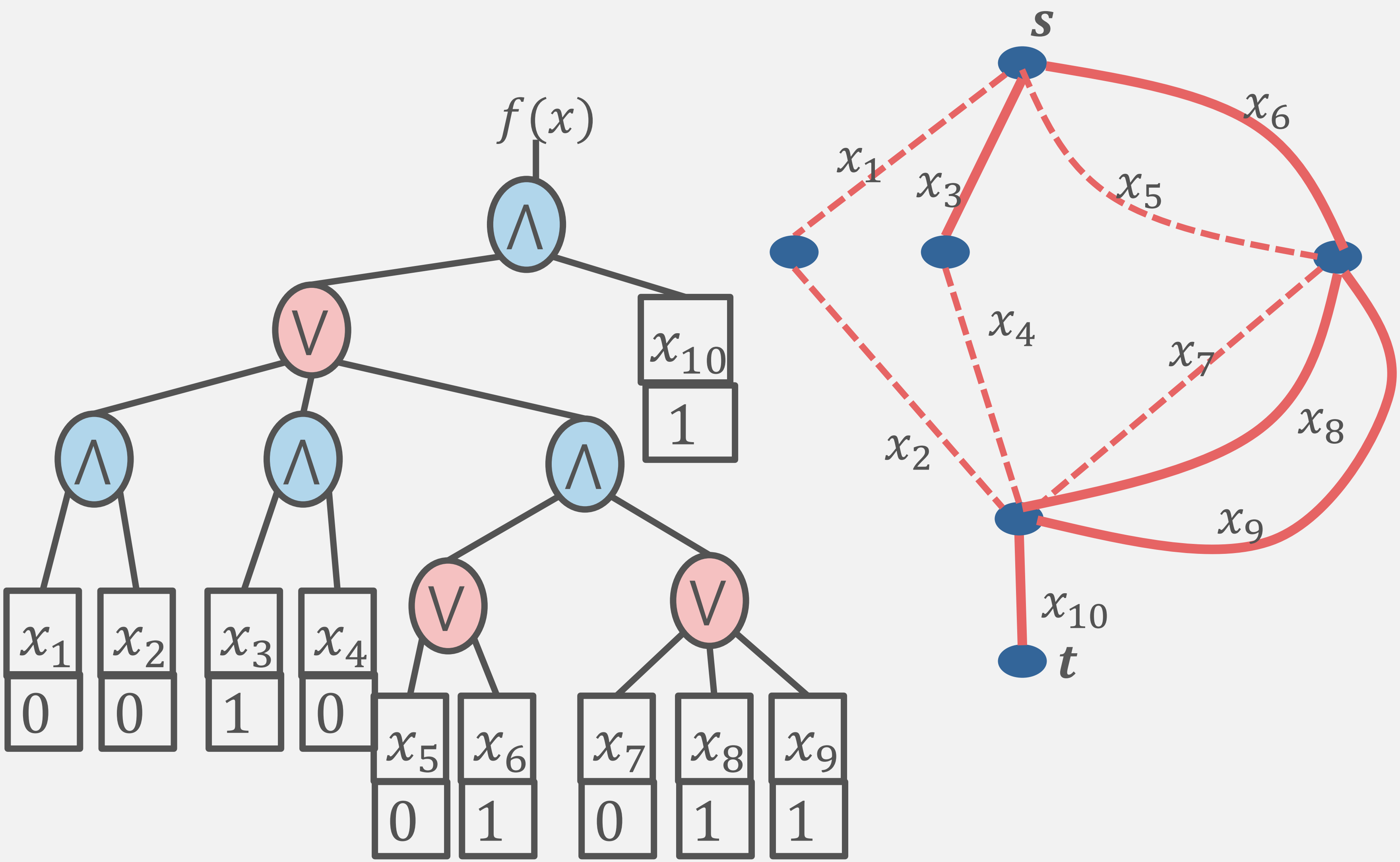
Is there a cycle?





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

Boolean Formulas



Outline:

A. Introduction to st-connectivity
B. st-connectivity makes a good algorithmic primitive
    1. Widely applicable
    2. Easy to analyze

Algorithm Analysis:

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

Algorithm Analysis:

Query Complexity:

- Bit string initially unknown
- Minimum # of oracle uses to determine w.h.p. on worst input

Algorithm Analysis:

Query Complexity:

- Bit string initially unknown
- Minimum # of oracle uses to determine w.h.p. on worst input

$\approx$

Time Complexity

Algorithm Analysis:

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 Analysis:

Query Complexity:

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

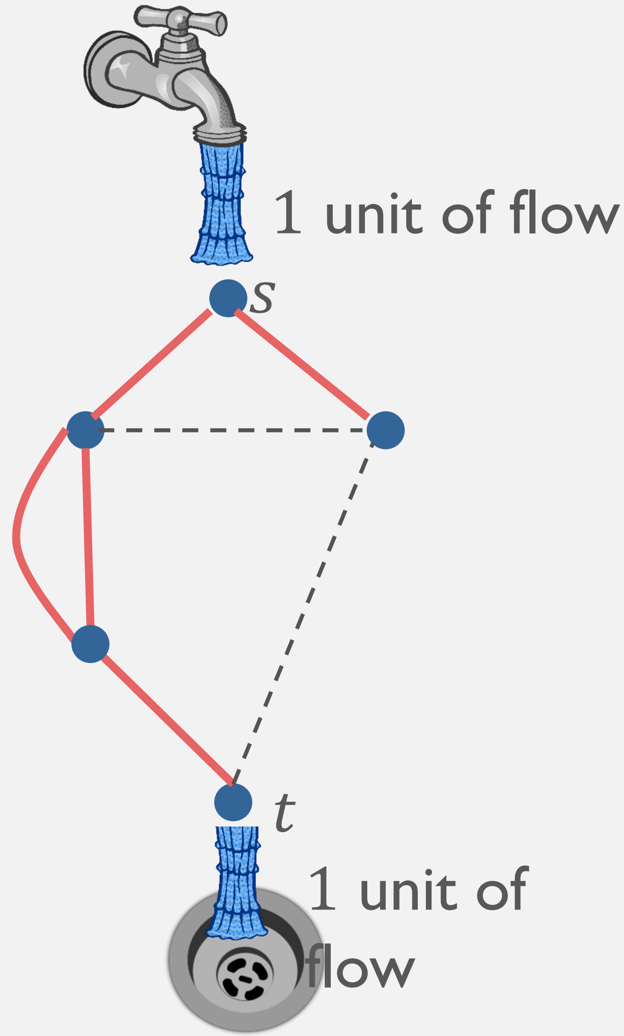
Effective resistance

[Belovs, Reichardt, '12]

Effective capacitance

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

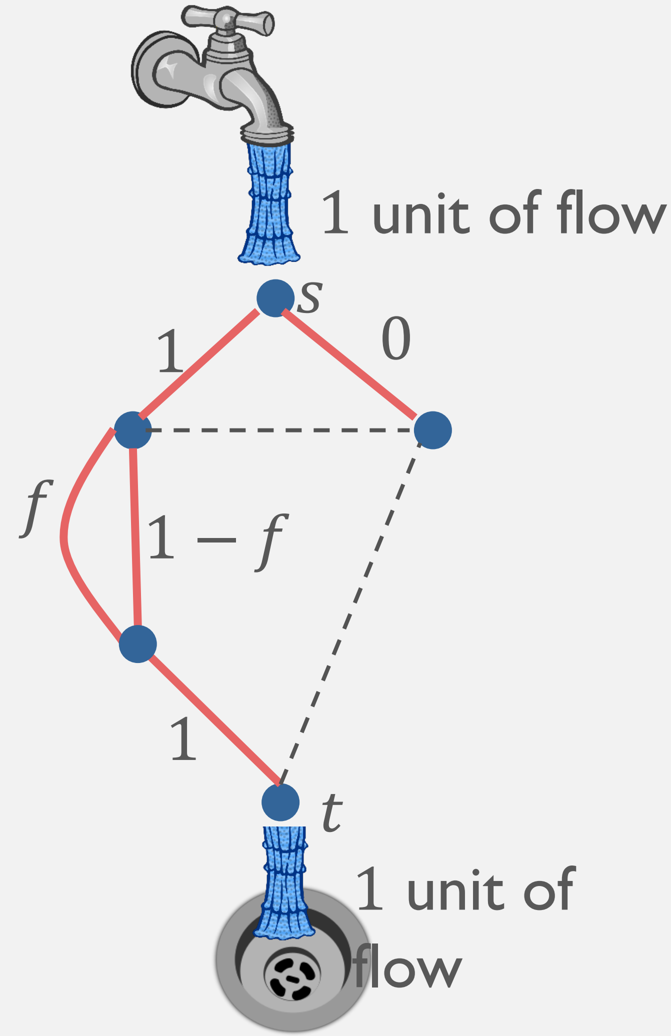
Effective Resistance



Effective Resistance

Valid flow:

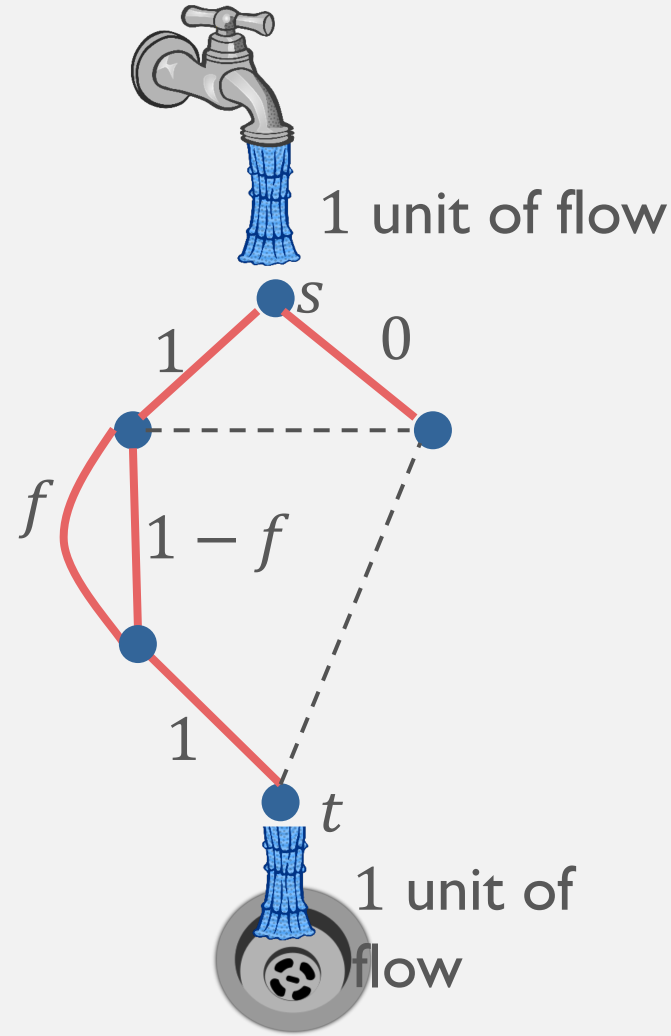
- 1 unit in at s
- 1 unit out at t
- At all other nodes, zero net flow



Effective Resistance

Flow energy:

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



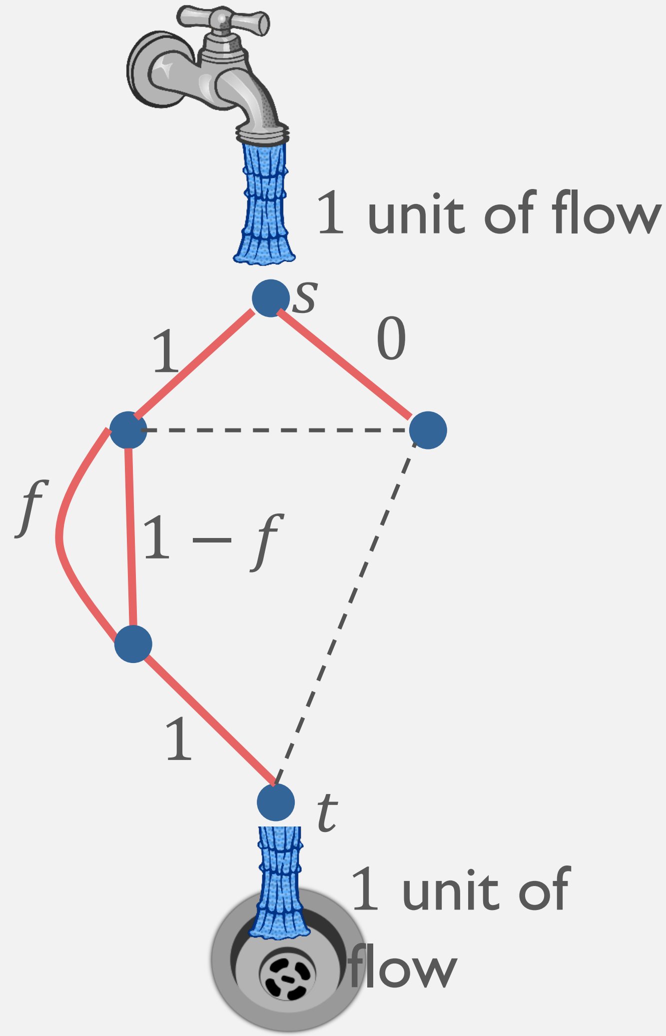
Effective Resistance

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.



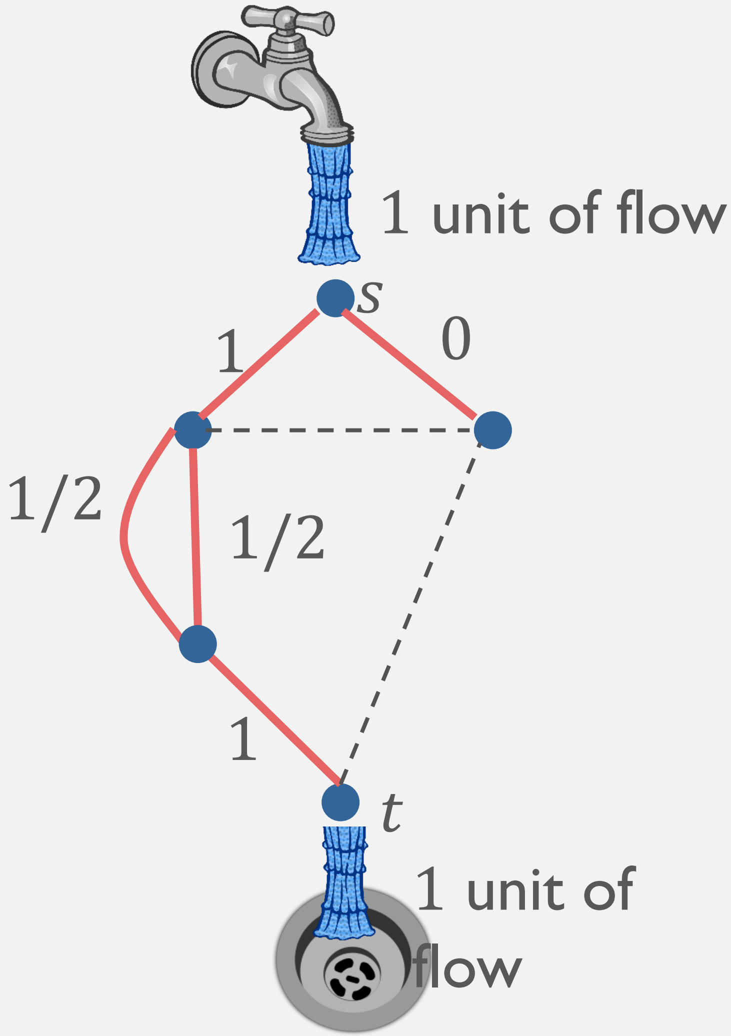
Effective Resistance

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 Analysis:

Query Complexity:

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

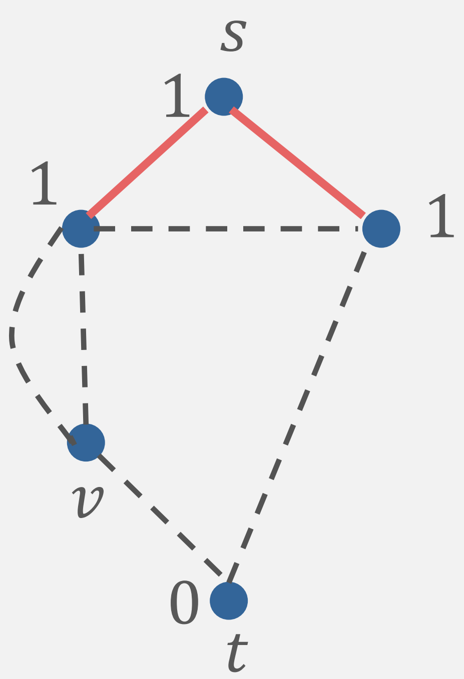
Effective resistance

Effective capacitance

Effective Capacitance

Generalized cut:

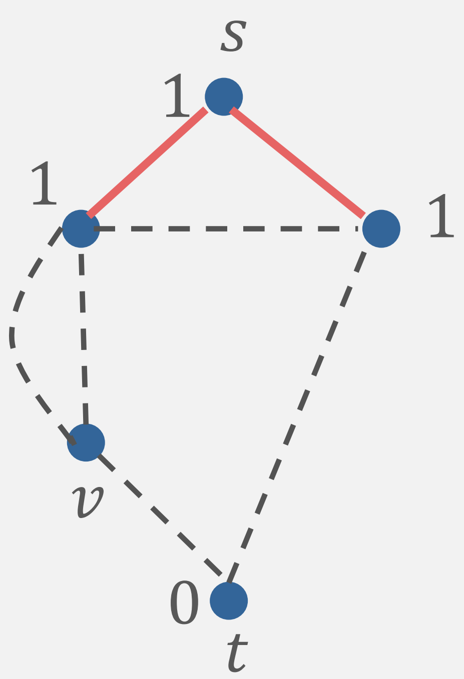
- 1 at s
- 0 at t
- Difference is 0 across edge



Effective Capacitance

Potential energy:

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



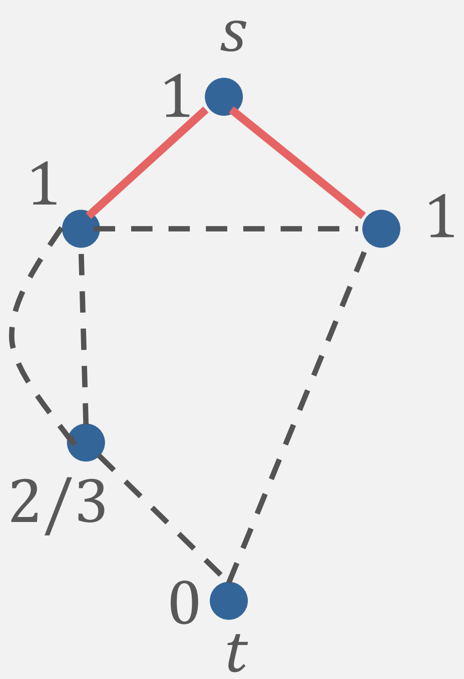
Effective Capacitance

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 Analysis:

Query Complexity:

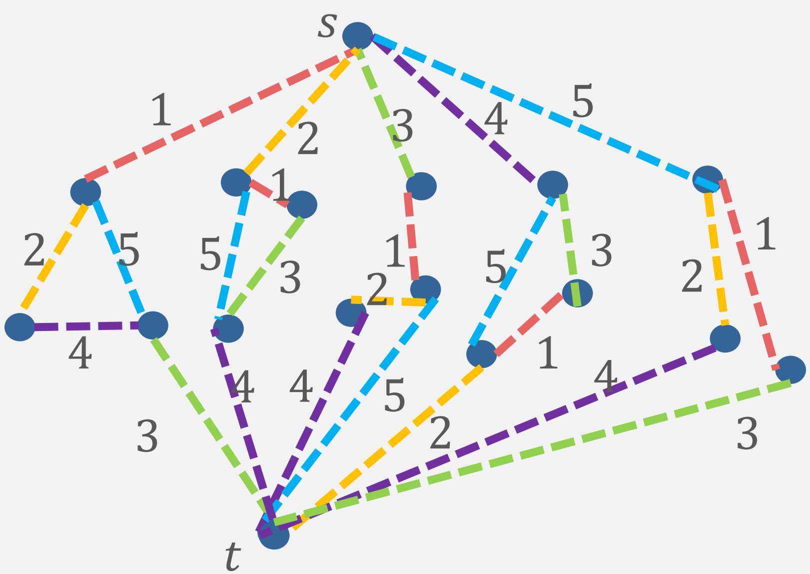
$$O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$$

Effective resistance

Effective capacitance

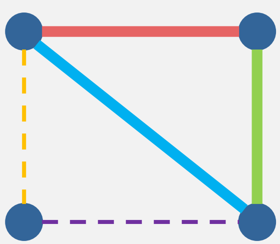
Example

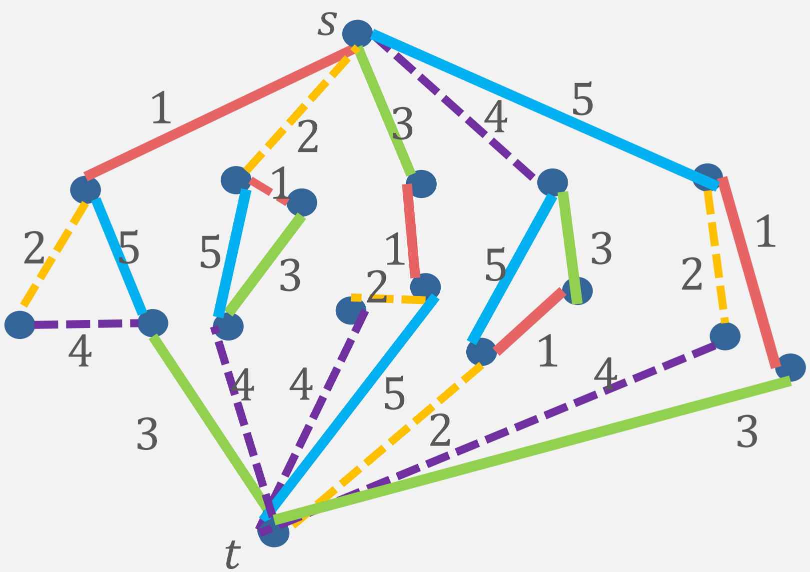
Cycle Detection $$O\left(\sqrt{\max_{\text{connected } G} R_{s,t}(G)}\ \sqrt{\max_{\text{not connected } G} C_{s,t}(G)}\right)$$



Example

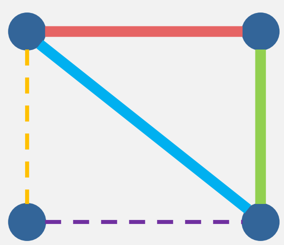
Cycle Detection $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\ \sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$

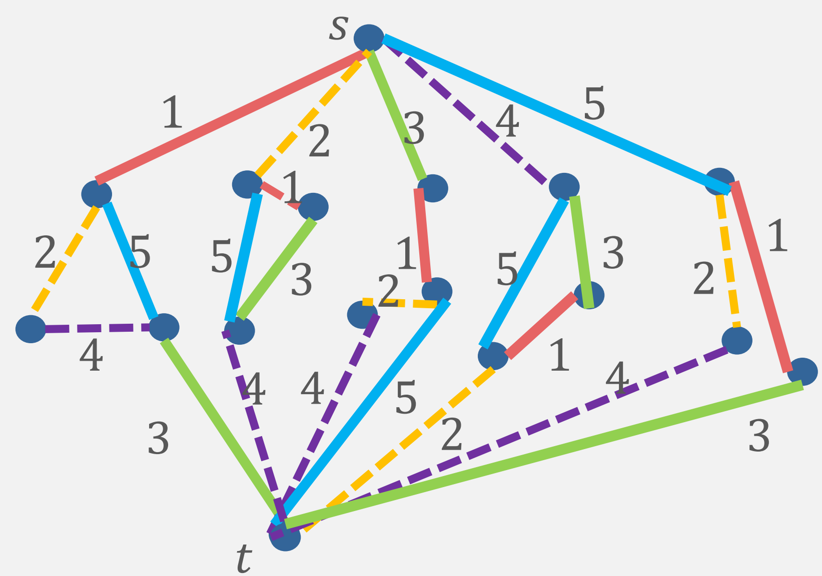




Example

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



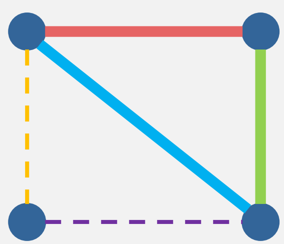


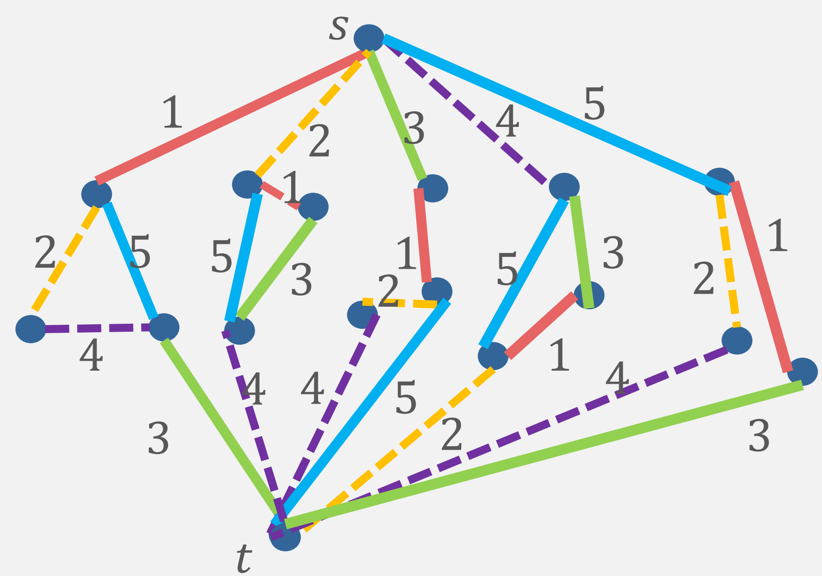
Length k cycle

- k parallel paths of length k
- $1/k$ flow on each path
- Effective resistance is 1

Example

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

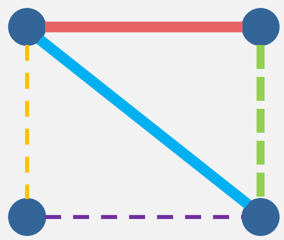


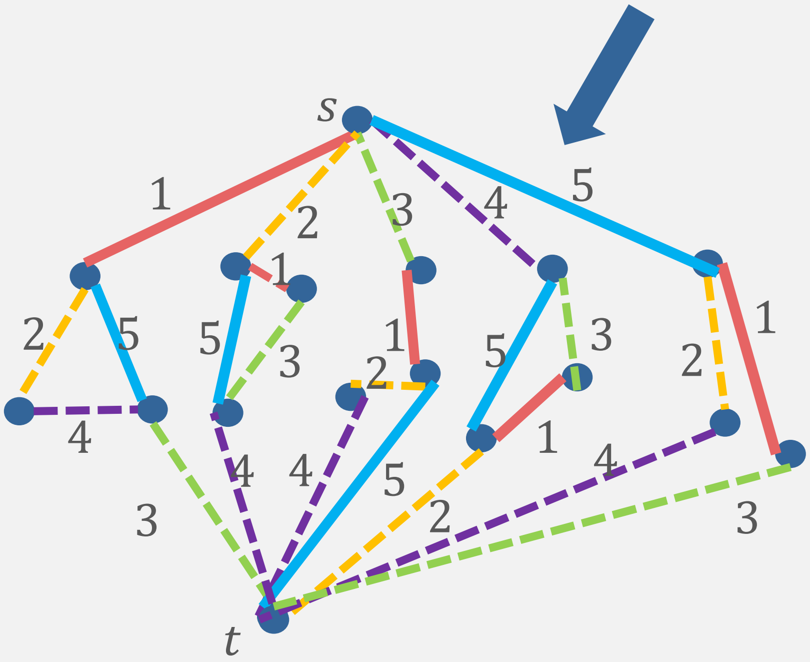


Multiple cycles → Effective resistance is < 1

Example

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

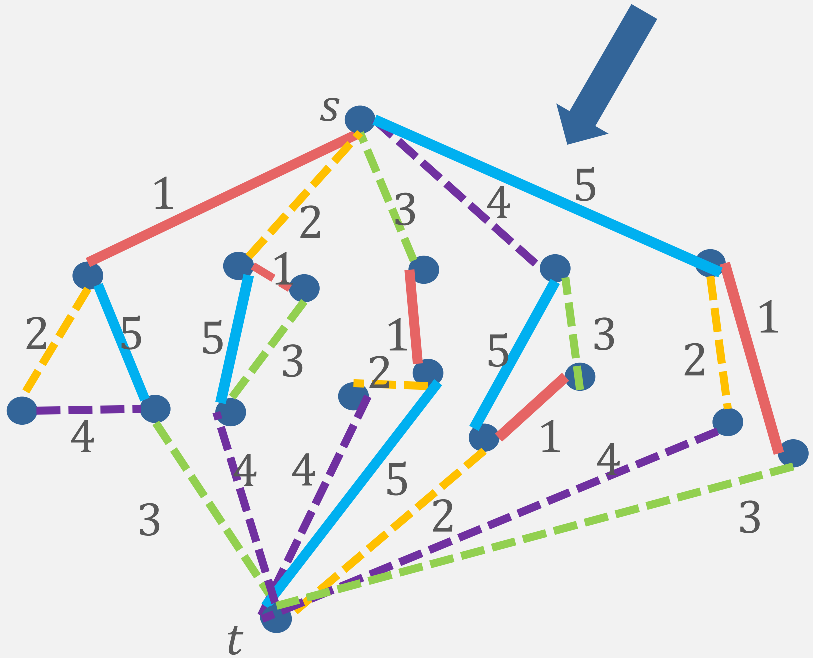




Example

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

- $O(n^2)$ subgraphs corresponding to non-present edges: cut at top
- $O(n)$ subgraphs corresponding to present edges, cut could have $O(n^2)$ edges in cut



Example

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

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

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

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

(optimal – logarithmic improvement over previous algorithm)

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 to cut to create a cycle free graph

- Quantum algorithm picks out critical topological parameter
- If promised either large circuit rank or no cycle, then cycle detection algorithm runs faster
- Proved by 2nd year undergrads

Estimation Algorithm:

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

Estimation 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.

Umm…Algorithm?

Umm…Algorithm?

Span Program

$$\begin{aligned}\forall i \in [N], b \in \{0,1\} :\ H_{i,b} &= \text{span}\{|e\rangle : e \in \overrightarrow{E}_{i,b}\} \\ U &= \text{span}\{|v\rangle : v \in V(G)\} \\ \tau &= |s\rangle - |t\rangle\end{aligned}$$

$$\forall e = (u,v,\ell) \in \overrightarrow{E}(G) :\ A|u,v,\ell\rangle = \sqrt{c(u,v,\ell)}(|u\rangle - |v\rangle)$$

Span Program->Unitary U = (reflection about space that depends on skeleton graph)(reflection about a space that depends on input)

Do phase estimation on U to precision $O\left(\sqrt{\max_{connected\ G} R_{s,t}(G)}\sqrt{\max_{not\ connected\ G} C_{s,t}(G)}\right)$

Open Questions and Current Directions

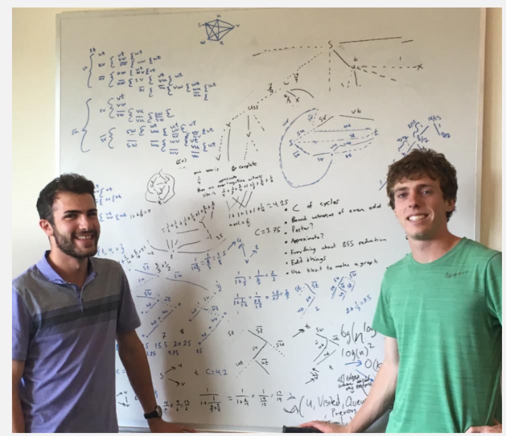
- How to choose edge weights? (Beigi et al '19)
- Conditions when st-connectivity reduction optimal?
- What is the classical time/query complexity of st-connectivity in the black box model? Under the promise of small capacitance/resistance?
- Better estimation algorithm for st-connectivity effective resistance/capacitance
- Primitives/Pedagogical Problems?

Thank you!

Stacey Jeffery

Michael Jarret

Alvaro Piedrafita



Teal Witter

Kai De Lorenzo