

# Graph Algorithms

Danupon Nanongkai

Max Planck Institute for Informatics

Saarland Informatics Campus, Saarbrücken, Germany



1. Open Problem (€100)

# Open: Cut Query for Reachability

$V = \# \text{ nodes}$ ,  $E = \# \text{ edges}$ ,  $\tilde{O}$  hides  $\text{polylog}(V)$

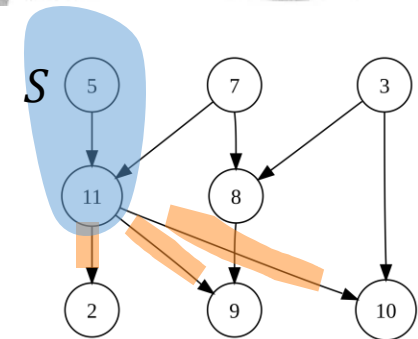
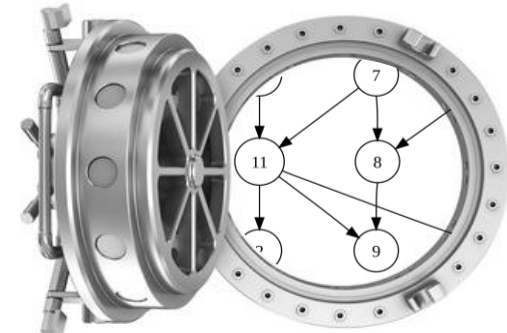
- Input: **Hidden** unweighted directed graph  $G=(V, E)$
- Query access: For  $S \subseteq V$ , **cut(S)** = # edges going out from  $S$
- Return: Exist path from nodes  $s$  to  $t$ ?

Exercise 1:  $\tilde{O}(V)$  for undirected graph

Exercise 2:  $\tilde{O}(V^2)$  even for more general problems [Cunningham'83]

Open:  $\tilde{O}(V^{1.99})$

Ultimate:  $\tilde{O}(V)$  & more



$\text{cut}(\{5, 11\}) = 3$

**Solution.** Observe: [Mehra-Mukhopadhyay'24]

1.  $\text{cut}(S) + \text{cut}(V \setminus S) = \sum_{u \in S, v \in V \setminus S} w(uv) + w(vu)$   
where  $w(uv)$  is the "weight" of edge from  $u$  to  $v$ .

2.  $\text{cut}(S) - \text{cut}(V \setminus S) = \sum_{u \in S} \text{outdegree}(u) - \text{indegree}(u)$

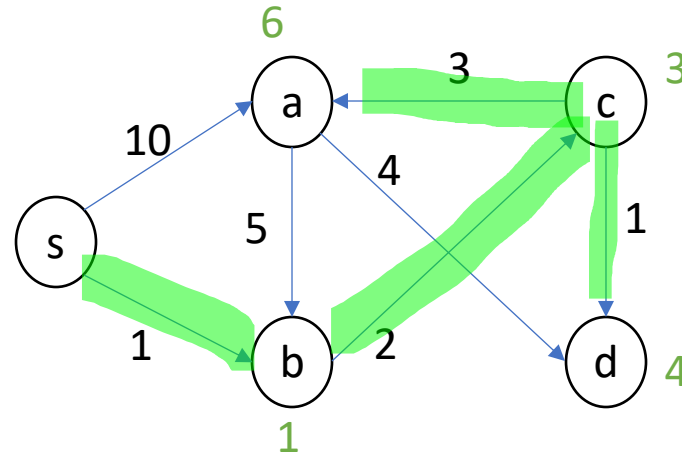
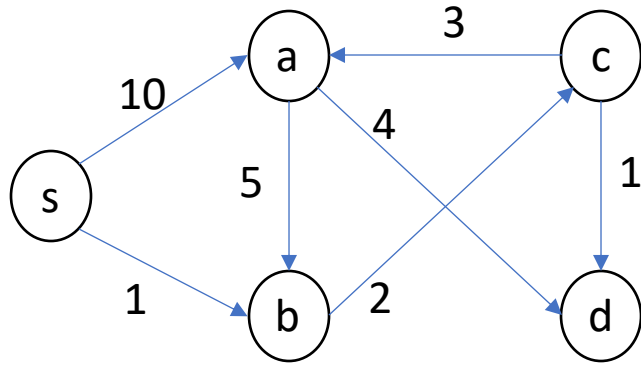
**Thus:** compute all possible  $w(uv)$ ,  $\text{outdegree}(u)$  and  $\text{indegree}(u)$

# Why? Submodular Function Minimization (SFM)

- Recall: For sets  $X \subseteq Y$  and  $x \notin Y$ ,
$$f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y)$$
- Special cases: Reachability, Min-cut/Max-flow, Matching, Matroid intersection, Disjoint trees, ...
- Known for SFM:  $\tilde{O}(n^2)$  &  $\Omega(n \log n)$  value queries.
  - Grötschel, Lovász and Schrijver 1984 & 1988; Lee, Sidford and Wong 2015; Jiang 2021; Chakrabarty, Graur, Jiang, and Sidford 2022
- Open:  $O(n^{1.99})$  and  $\Omega(n^{1.01})$ 
  - We can't even solve reachability in  $O(n^{1.99})$  queries

## 2. Shortest Paths

# Single-Source Shortest Paths



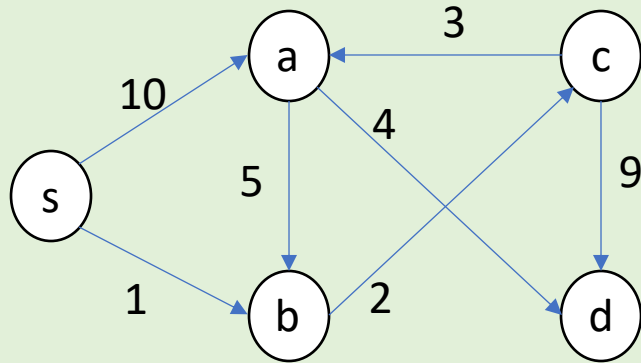
- Input: Directed weighted graph  $G = (V, E, w)$ , source  $s \in V$
- Output: Minimum-cost path from  $s$  to every  $v \in V$



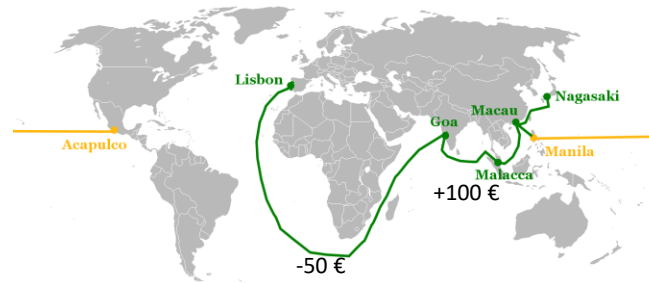
# Textbook algorithms

## Dijkstra

- + Near-optimal time  
( $O(E + V \log V)$  time)
- Only **non**negative weights.



But sometimes we need both **positive** and **negative** weights:



Ships/airplanes routing



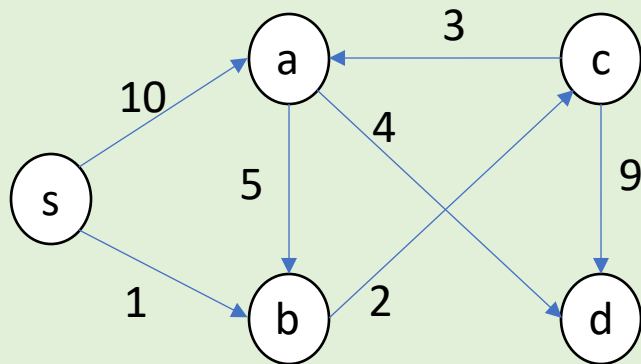
Arbitrage trading

Also job scheduling, periodic optimization, control theory, ...

# Textbook algorithms

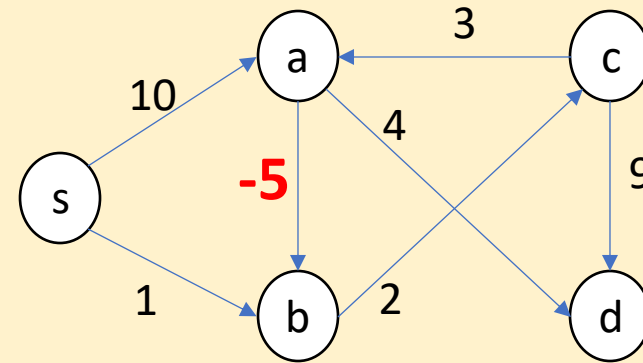
## Dijkstra

- + Near-optimal time  
( $O(E + V \log V)$  time)
- Only **non**negative weights.



## Bellman-Ford


- + Allow **negative weights**
- **Far** from near-linear time  
( $O(VE)$  time)



Near linear =  $E \log^{O(1)} E$   
e.g.  $E \log^{10} E$



# Research question: Near-linear time algorithm for **negative-weight** SSSP?



**Textbook (from 1950s):**  $O(EV)$  [Shimbel'55, Ford'56, Bellman'58, Moore'59]

Assume **integer** weights  
-W, -W+1, -W+2..., 0, 1, 2, ...



**1980s-1990s Approximation/scaling techniques:**  $O(E\sqrt{V} \log W)$

Gabow'85 ( $EV^{\frac{3}{4}} \log V$ ), Gabow-Tarjan'89 ( $EV^{\frac{1}{2}} \log(VW)$ ), Goldberg'95 ( $EV^{\frac{1}{2}} \log(W)$ )



**2000s Special graphs:**

$\tilde{O}(E)$  time for planar graphs [Fakcharoenphol-Rao'06, Mozes-Wulff-Nilsen'10]

$\tilde{O}(V^{4/3} \log W)$  time for bounded-genus & minor-free graphs [Wulff-Nilsen'11]



**2010+: Continuous Methods + Dynamic Graphs:**  $E^{\frac{4}{3}+o(1)} \log W$ ,  $\tilde{O}(E + V^{1.5} \log W)$



Cohen, Madry, Sankowski, Vladu'17 ( $(E^{10/7+o(1)} \log W)$ ), Axiotis, Madry, Vladu, 2020 ( $E^{4/3+o(1)} \log W$ ), van den Brand, Lee, N, Peng, Saranurak, Sidford, Song, Wang 2020 ( $E + V^{1.5} \log W$ )

# Our results: Near-linear time shortest path

(2022)  $O(E \log^8 V \log(W))$  time in expectation

Bernstein, N., Wulff-Nilsen

(2023)  $O(E \log^2 V \log(VW) \log \log V)$  time in expectation

Bringmann, Cassis, Fischer

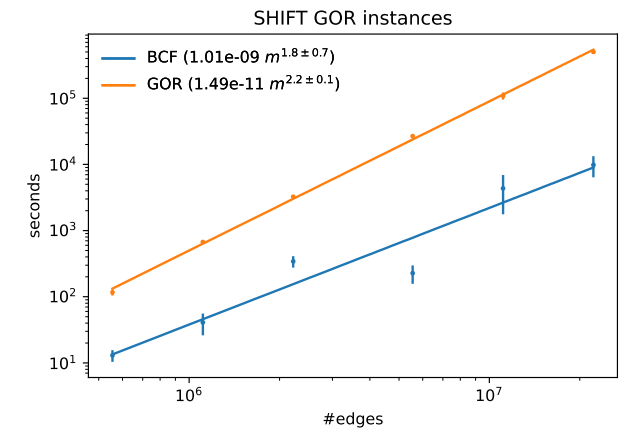
(2024) Efficient in Parallel, Distributed, and Quantum Settings

Ashvinkumar, Bernstein, Cao, Grunau, Haeupler, Jiang, N., Su

(2024) Experimental results

Bringmann, Cassis, Karrenbauer, Nusser, Rinaldi

Codable. Teachable. Efficient in many settings.



Key Techniques:  
Low-Diameter Decomposition (LDD)

# Definition of LDD( $G, D$ )

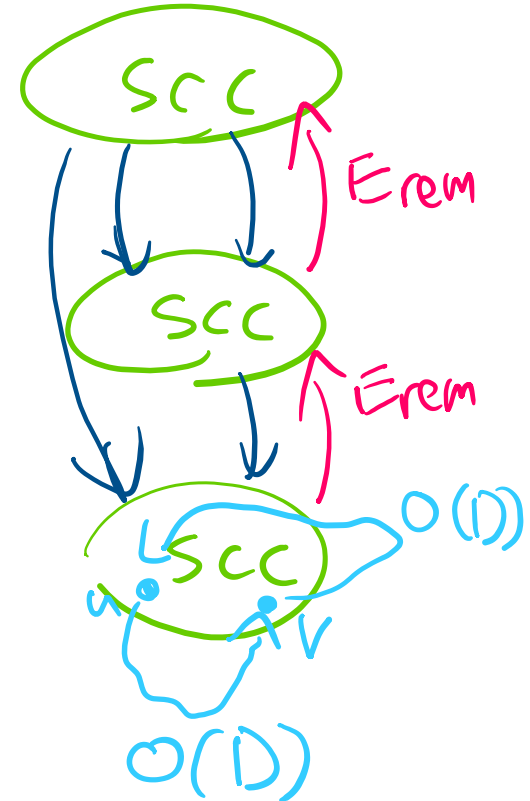
(Oversimplified)

Input: Directed graph  $G = (V, E)$  & positive integer  $D$

Output:  $E_{Rem} \subseteq E$  such that

1. each **SCCs** of  $G \setminus E_{rem}$  has diameter  $O(D)$ 
  - i.e. for  $u, v$  in the same SCC,  
 $distance(u, v) = O(D)$  &  $distance(v, u) = O(D)$
2.  $\forall e \in E, Pr[e \in E_{rem}] = O\left(\frac{1}{D}\right)$ .

Runtime:  $\tilde{O}(E)$  in expectation.

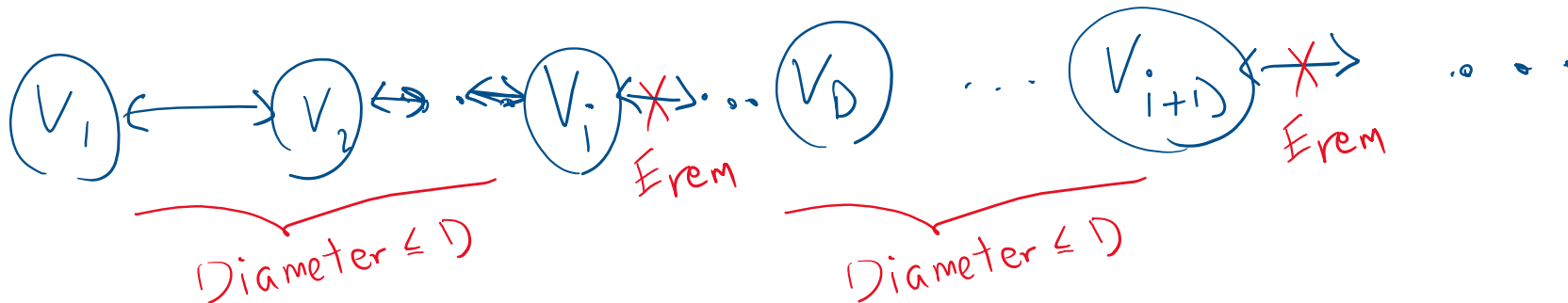


# Example (1)

$G =$  undirected path  $(v_1, v_2, \dots, v_n)$

Getting LDD( $G, D$ ):

- randomly select  $i \in [1, D]$
- add edges  $(v_i, v_{i+1}), (v_{i+D}, v_{i+D+1}), (v_{i+2D}, v_{i+2D+1}), \dots$  to  $E_{rem}$

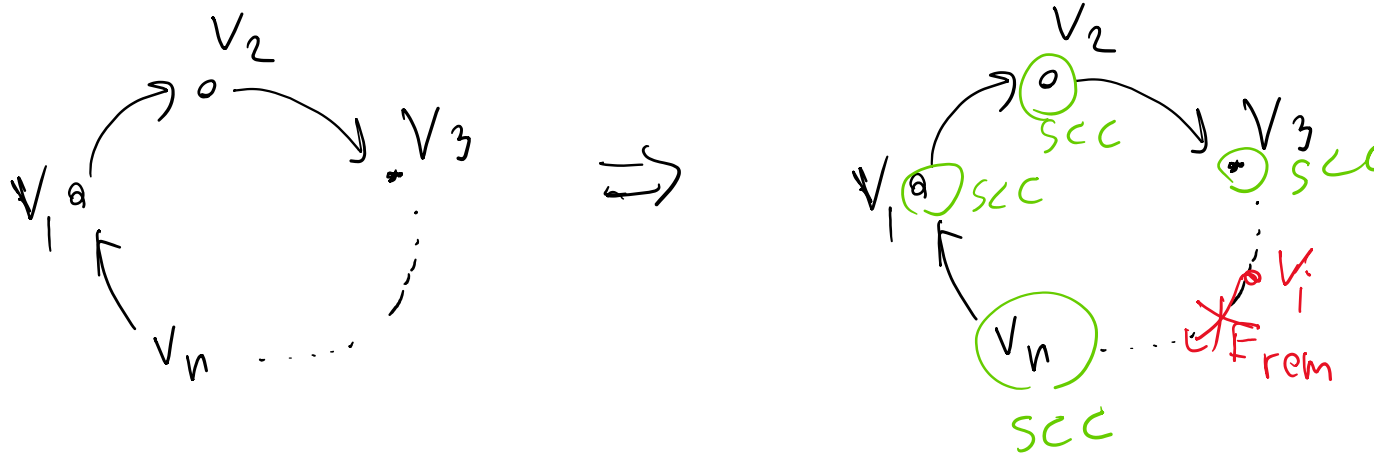


# Example (2)

$G =$  directed cycle  $(v_1, v_2, \dots, v_n)$

Getting LDD( $G, D$ ): randomly add one edge to  $E_{rem}$

→ Each node becomes an SCC



# How to use LDD? - Solve this first (€1)

Solve this problem in near-linear time:

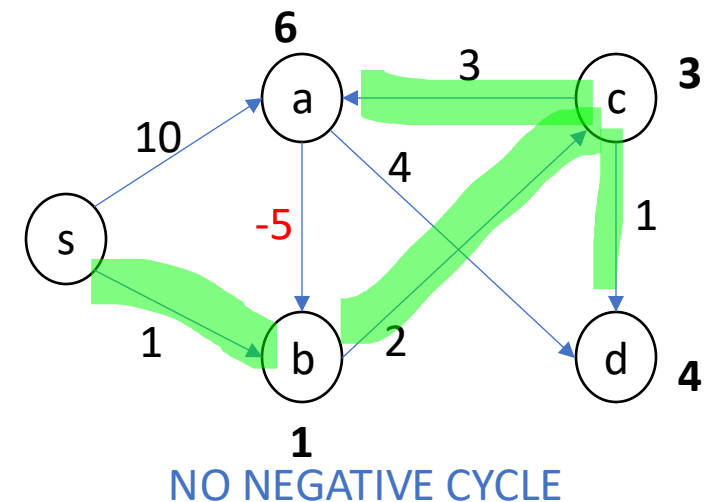
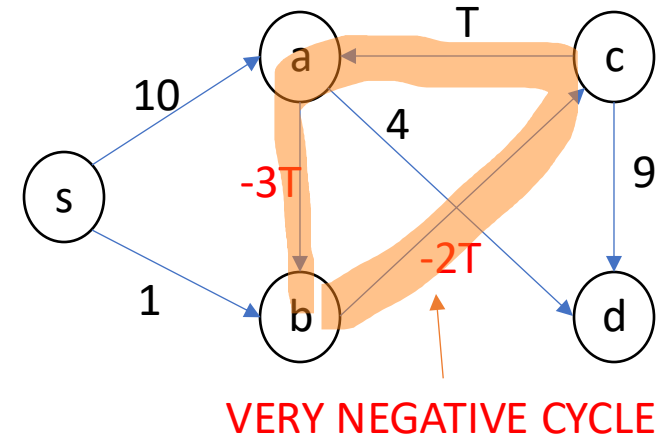
**Input:** Strongly-connected directed graph  $G = (V, E)$  with integer edge weight  $w$  for some integer  $T > 0$ .

**Promise:** The diameter of  $G$  is at most  $10T$ ; i.e. for every vertices  $u$  and  $v$ , there is a path from  $u$  to  $v$  with weight at most  $10T$ .

**Output:** Either report a cycle whose average edge weight is  $< -T$  or report that no negative cycle exists:

- Output "VERY NEGATIVE CYCLE" if there is a cycle (sequence of vertices)  $v_1, v_2, \dots, v_k = v_1$ , such that  $\sum_{i=1}^{k-1} w(v_i, v_{i+1}) < -T$ .
- Output "NO NEGATIVE CYCLE" if there is for every cycle  $v_1, v_2, \dots, v_k = v_1$ ,  $\sum_{i=1}^{k-1} w(v_i, v_{i+1}) \geq 0$ .

For other cases, you can output any of the above.



Solution to this problem and LDD are two key ideas