Private Graph Algorithms in the Semi-Honest Model

Justin Brickell

November 24, 2004

Two Party Graph Algorithms

- Parties P_1 and P_2 own graphs G_1 and G_2
- *f* is a *two-input* graph algorithm
- Compute $f(G_1,G_2)$ without revealing "unnecessary information"

Unnecessary Information

- Intuitively, the protocol should function *as if a trusted third party computed the output*
- We use *simulation* to prove that a protocol is private

The Semi-Honest Model

- A *malicious* adversary can alter his input
- A semi-honest adversary
  - adheres to protocol
  - tries to learn extra information from the message transcript

General Secure Two Party Computation

- *Any* polynomial sized functionality can be made private (in the semi-honest model)
  - Yao's Method
- What are our goals?
  - Yao's Method is inefficient
  - Efficient, private protocols to compute particular graph functionalities
  - Take advantage of information "leaked" by the result

Two-Input Graph Algorithms?

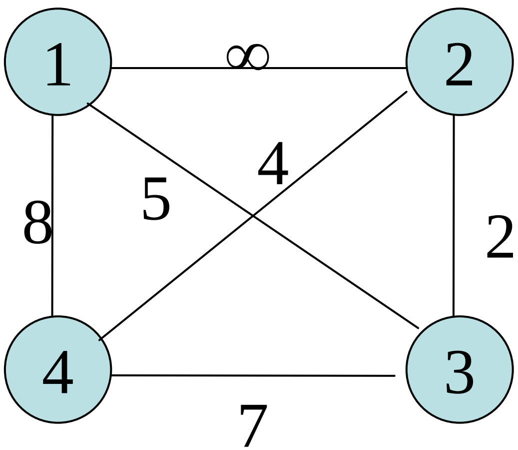
- Graph Isomorphism
- Comparison of graph statistics
  - $f(G_1) > f(G_2)$?
  - max flow,diameter,average degree
- Synthesized Graphs
  - $f(G_1 \bullet G_2)$

Graph Synthesis

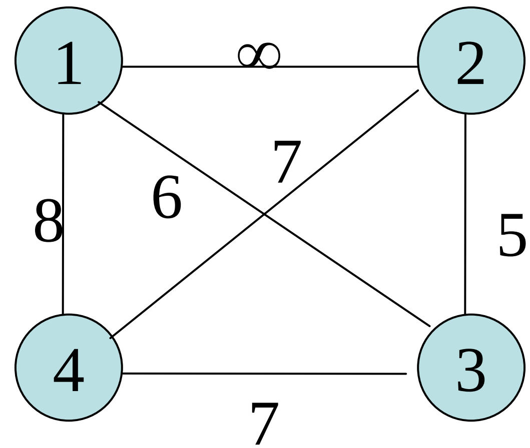
- G_1 and G_2 are weighted complete graphs on the same vertex and edge set
  - $G_1 = (V,E,w_1)$; $G_2 = (V,E,w_2)$
- $\mathrm{gmax}(G_1,G_2) = (V,E,w_{\max})$
  - $w_{\max}(e) = \max(w_1(e),w_2(e))$
- $\mathrm{gmin}(G_1,G_2) = (V,E,w_{\min})$
  - $w_{\min}(e) = \min(w_1(e),w_2(e))$

Graph Synthesis

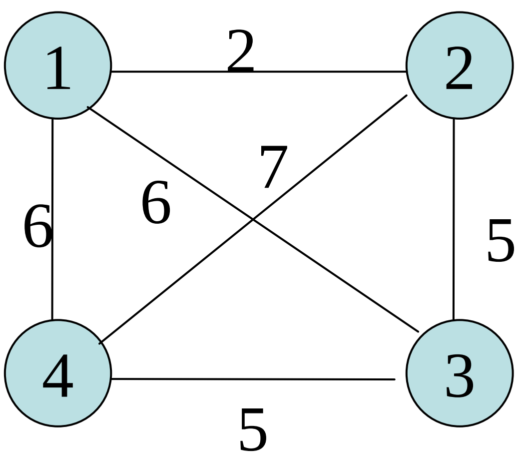
G_1



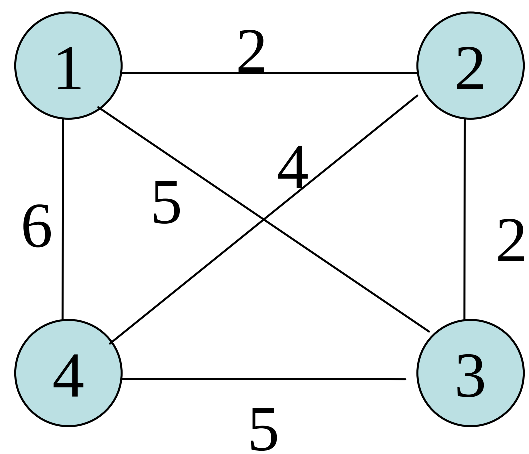
$gmax(G_1,G_2)$



G_2



$gmin(G_1,G_2)$



Graph Isomorphism

- Unlikely to find a private protocol
  - No known poly-time algorithm

Comparison of Graph Statistics

1. Compute statistic on own graph
   - Semi-honest participants can’t lie
2. Use a private comparison protocol
   - Yao’s Millionaire Protocol
   - Yao’s method (circuit protocol)

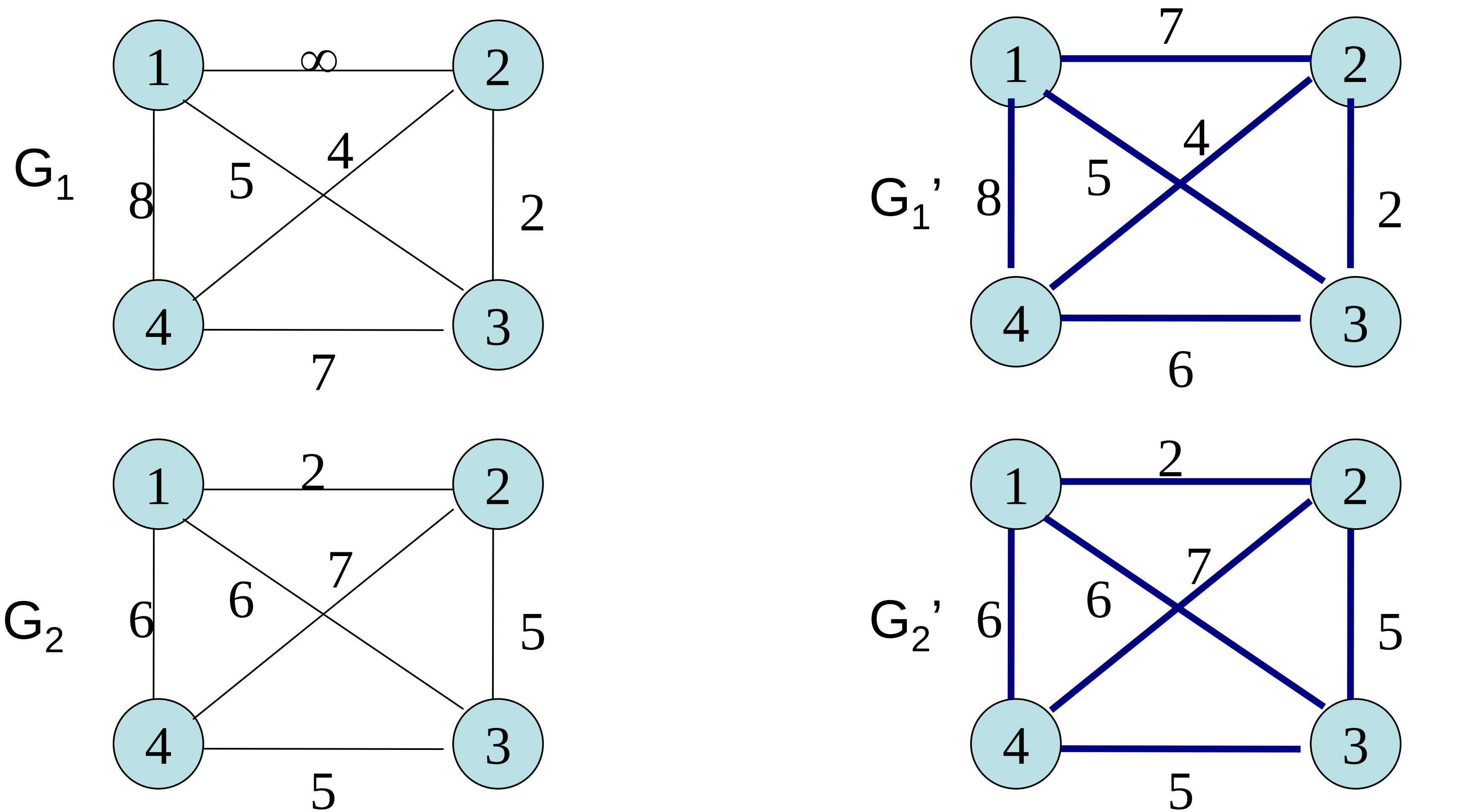
Synthesized Graphs

- This is the interesting case
- All Pairs Shortest Distance and Single Source Shortest Distance both “leak” significant useful information
  - Solved: APSD(gmin),SSSD(gmin)
  - Solved with leaks: APSD(gmax),SSSD(gmax)

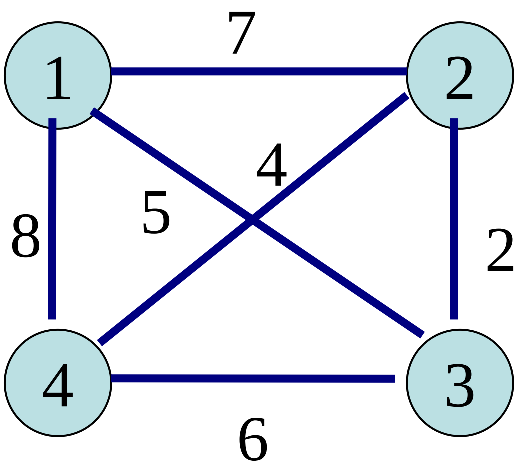
APSD(gmin(G_1,G_2))

- Basic Idea: Add edges to the solution graph in order of smallest to largest
- Private, because we can recover the order from the final solution graph

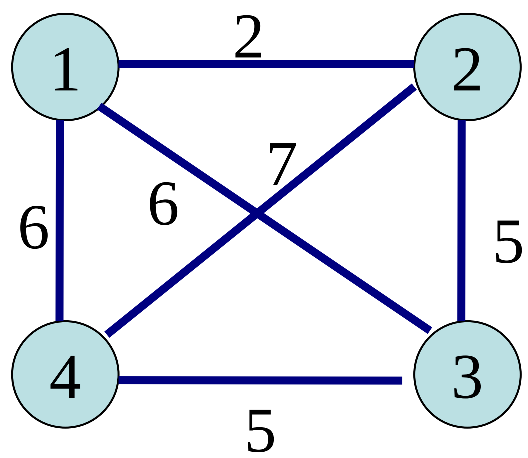
Run APSD on G_1 and G_2



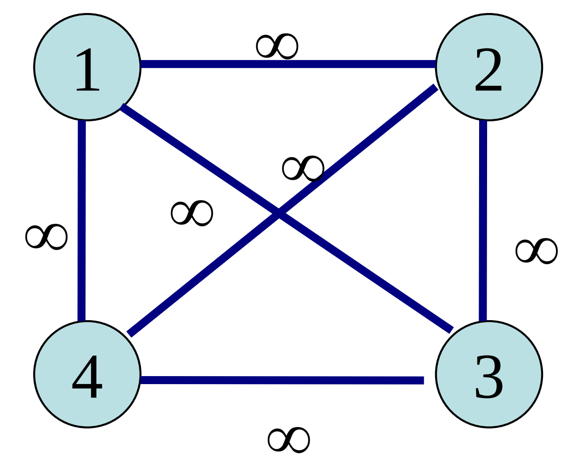
Initialize G_0'



G_1'

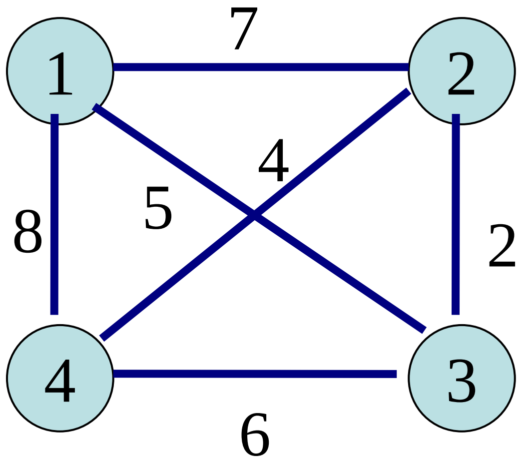


G_2'



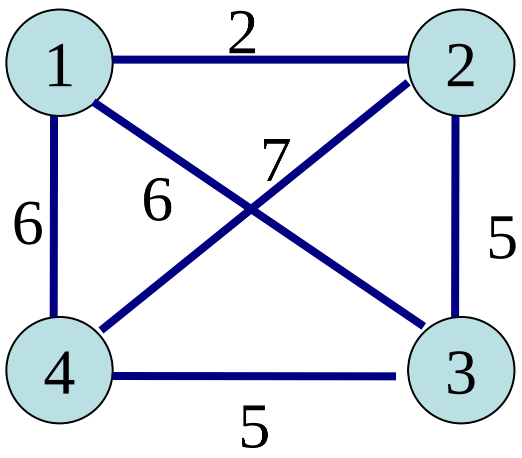
G_0'

Find shortest blue edge lengths



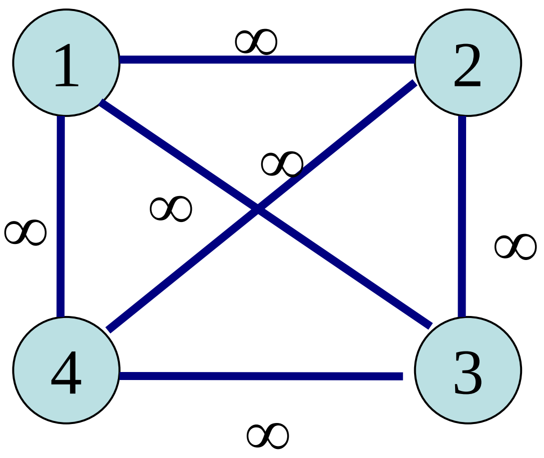
G_1'

min1 = 2



G_2'

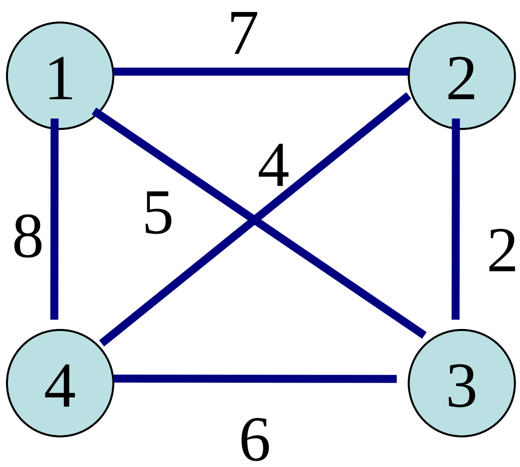
min2 = 2



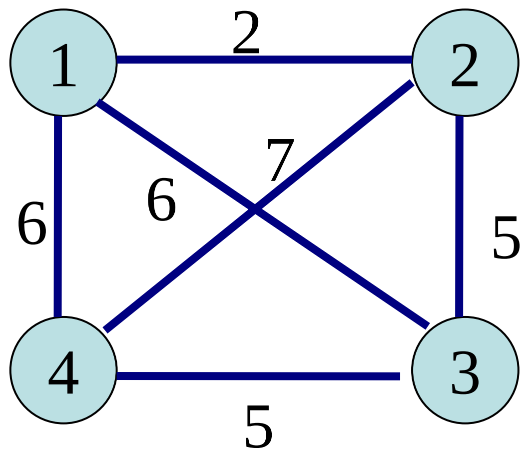
G_0'

min0 = ∞

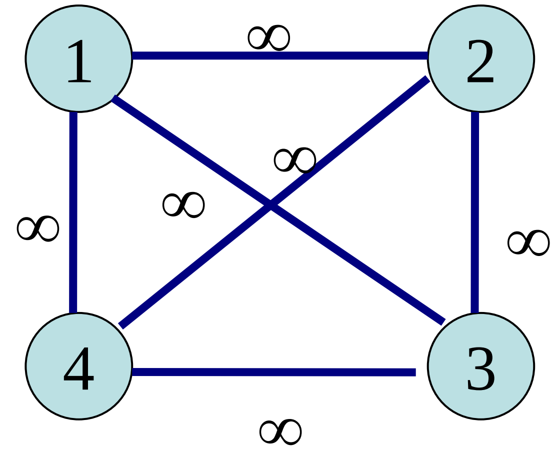
Privately find global shortest length



G$_1$'



G$_2$'



G$_0$'

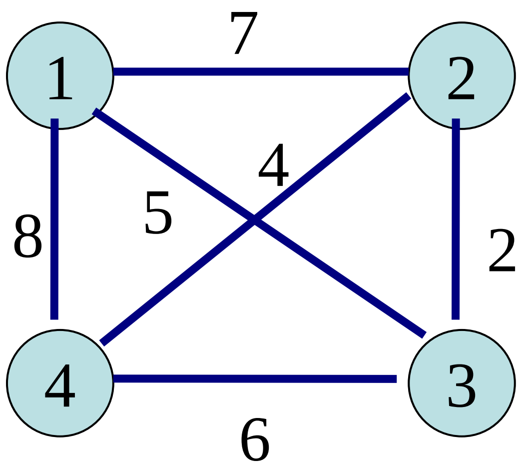
min1 = 2

min2 = 2

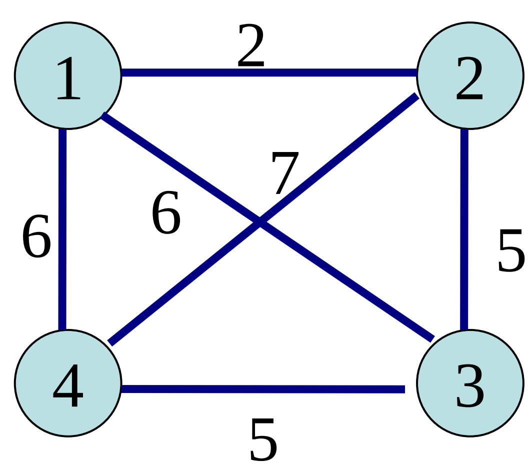
min0 = ∞

bluemin = min(min0,min1,min2) =2

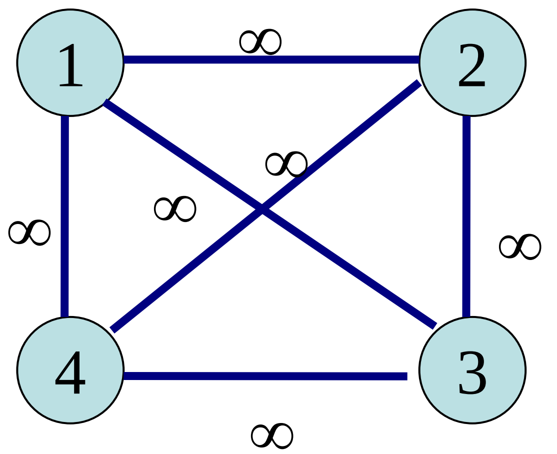
Find edges of length bluemin



G_1'



G_2'

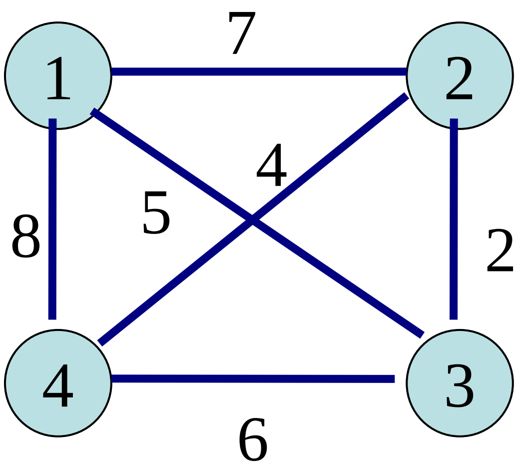


G_0'

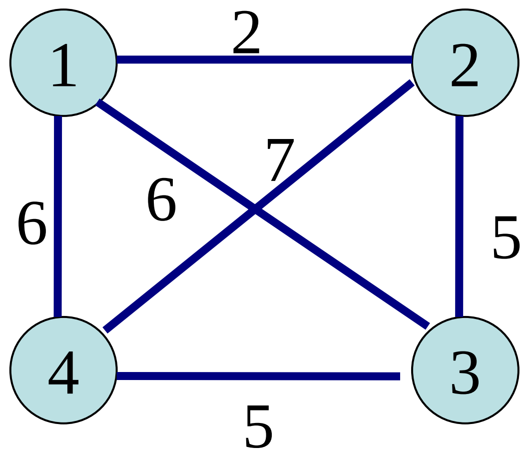
$S_1 = \{e_{23}\}$ $S_2 = \{e_{12}\}$ $S_0 = \{\}$

bluemin = min(min0,min1,min2) =2

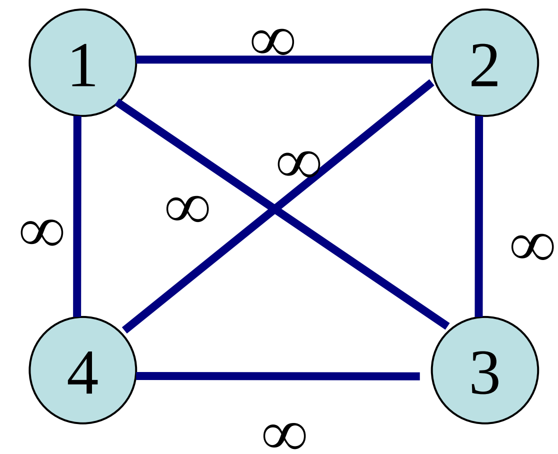
Privately find all edges of length bluemin



G$_1$'



G$_2$'



G$_0$'

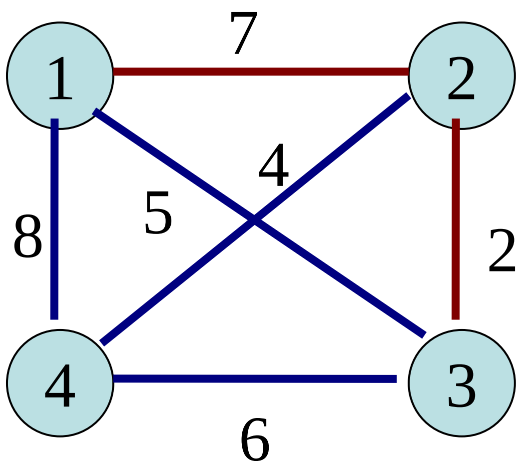
$S_1 = \{e_{23}\}$

$S_2 = \{e_{12}\}$

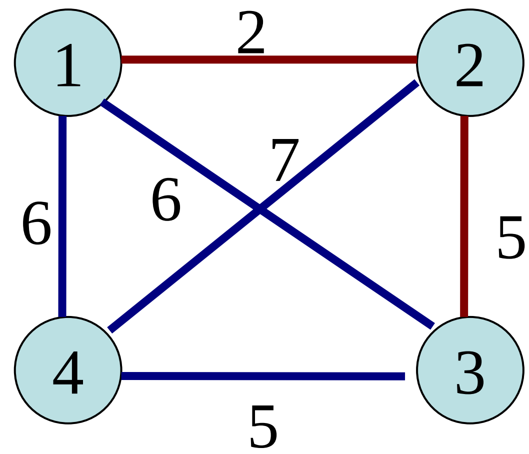
$S_0 = \{\}$

$S = S_1 \cup S_2 \cup S_3 = \{e_{12}, e_{23}\}$

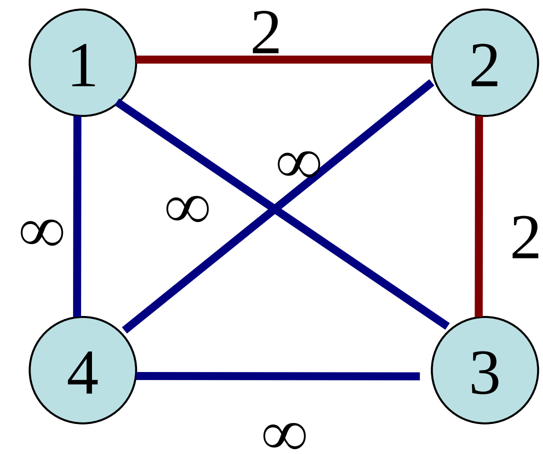
Update S edges in G_0'



G_1'



G_2'



G_0'

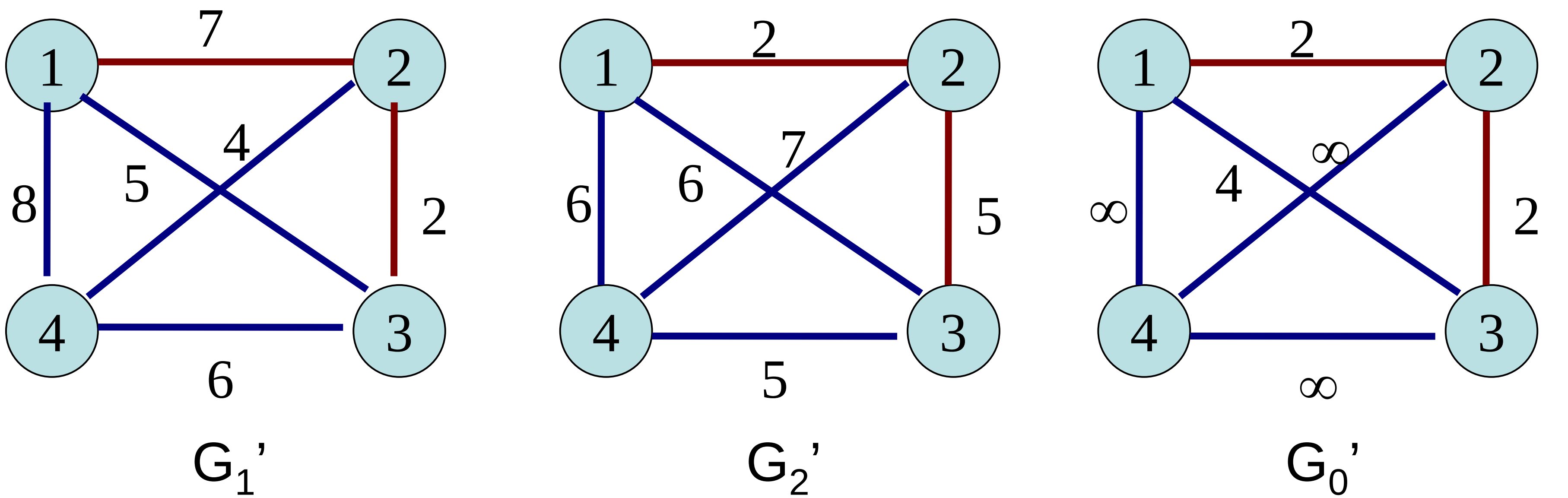
$S_1 = \{e_{23}\}$

$S_2 = \{e_{12}\}$

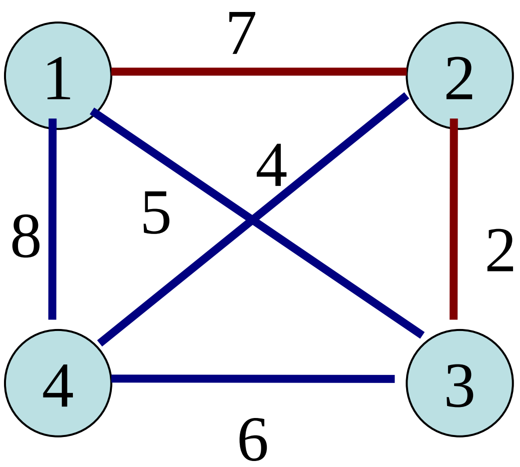
$S_0 = \{\}$

$$S = S_1 \cup S_2 \cup S_3 = \{e_{12}, e_{23}\}$$

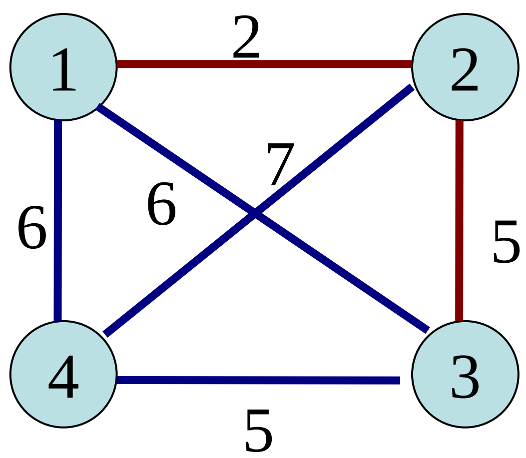
Run APSD on G_0'



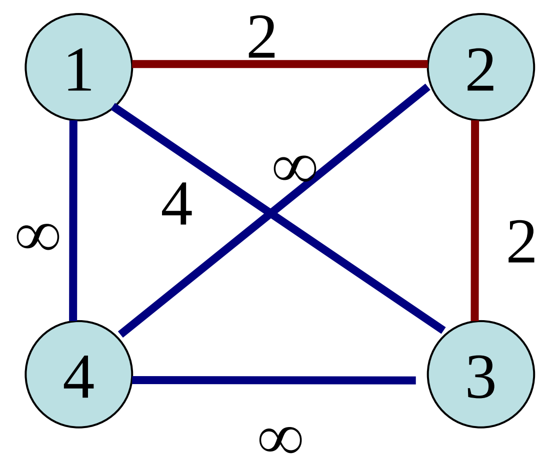
Repeat!



G_1'



G_2'

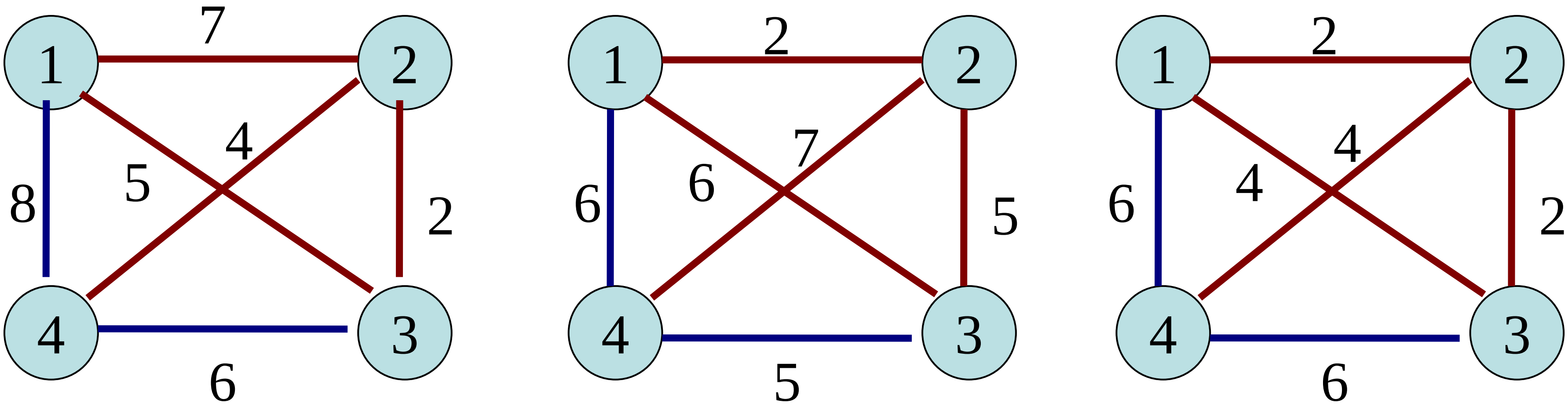


G_0'

bluemin = min(min0,min1,min2) =4

$S = S_1 \cup S_2 \cup S_3 = \{e_{13}, e_{24}\}$

…

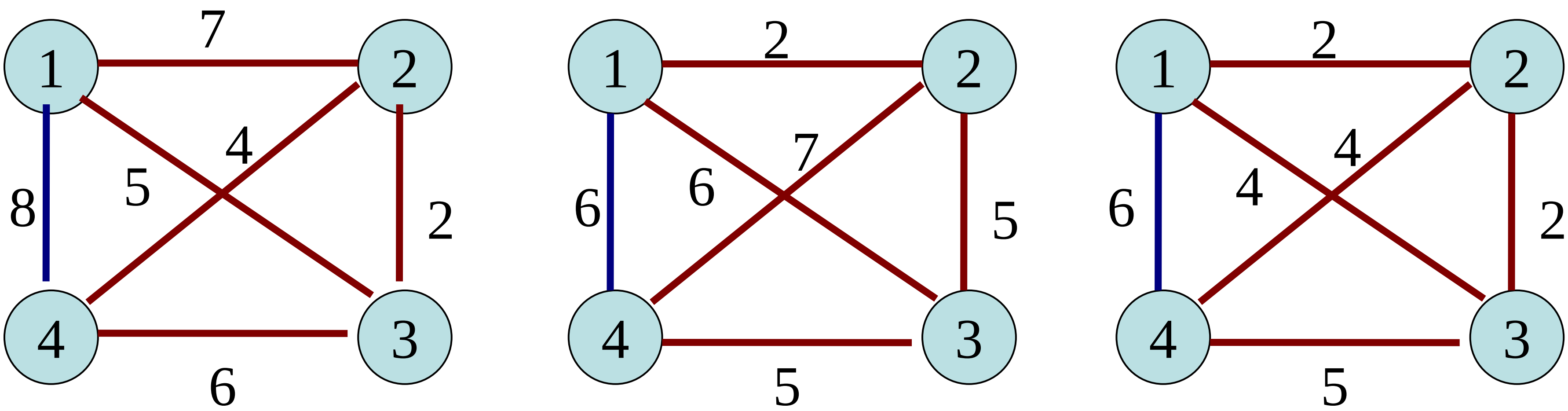


G_1' G_2' G_0'

bluemin = min(min0,min1,min2) =5

$S = S_1 \cup S_2 \cup S_3 = \{e_{34}\}$

…

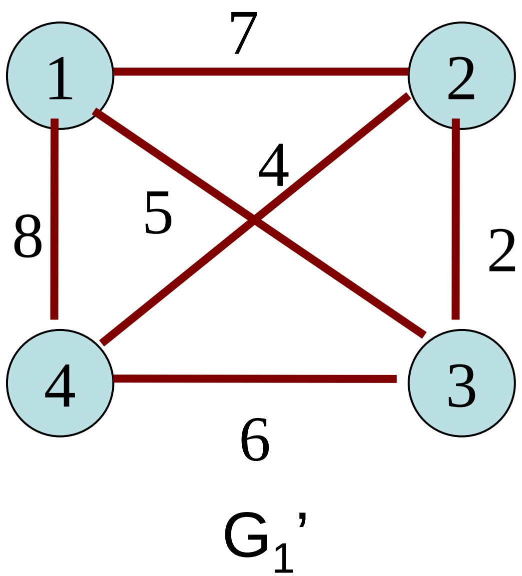


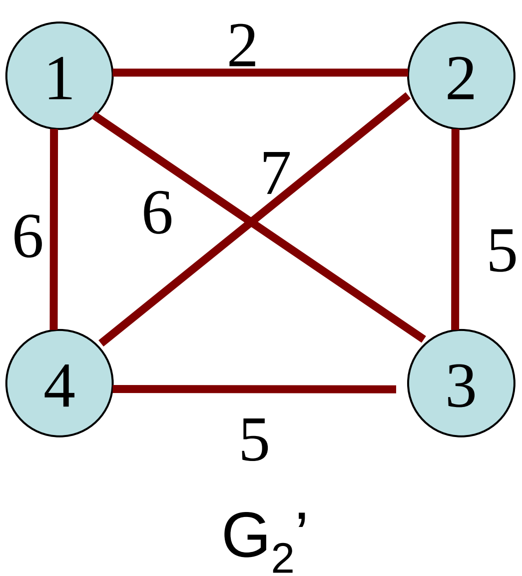
G_1' G_2' G_0'

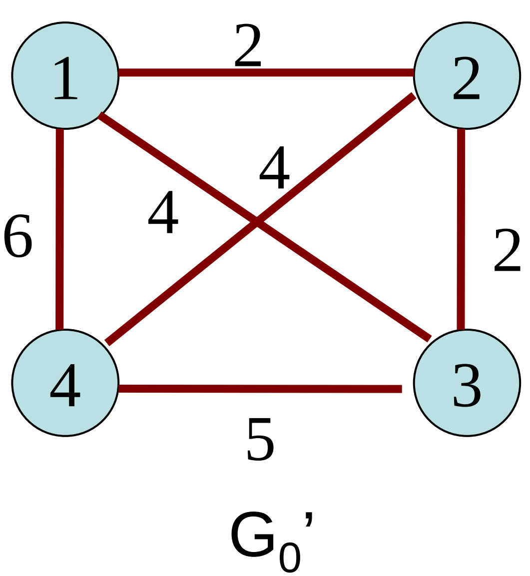
bluemin = min(min0,min1,min2) =6

$S = S_1 \cup S_2 \cup S_3 = \{e_{14}\}$

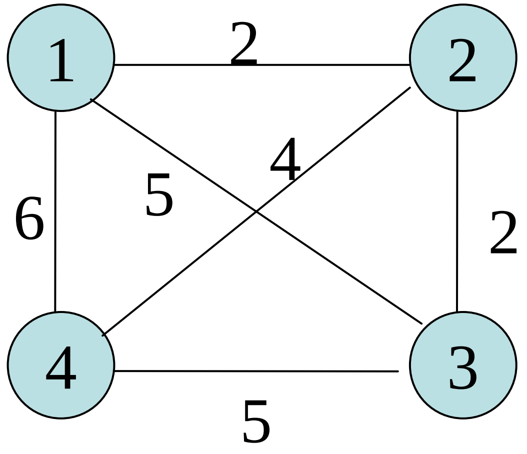
… until all edges are red



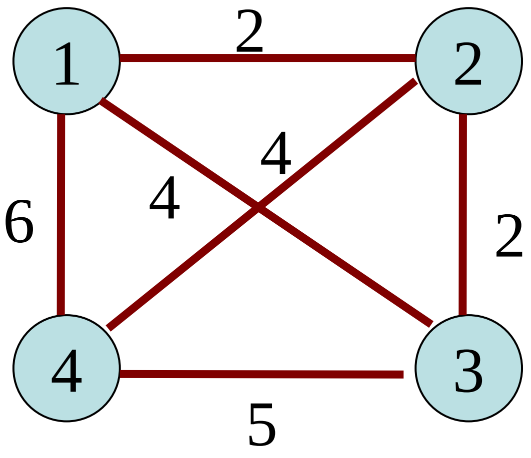




The solution is correct!



gmin(G_1,G_2)



G_0'

Other Results

- A similar protocol for SSSD(gmin)
  - This isn't free!
- Protocol for special case of APSD(gmax) and SSSD(gmax)
  - Input graphs obey triangle inequality
- "Leaky" protocol for APSD(gmax) and SSSD(gmax) in the general case

Final Thought

- Other graph algorithms don’t leak enough information
- Questions?