Algorithm Design and Analysis

David N. JANSEN, Bohua ZHAN
名 姓

算法设计与分析

詹博华，杨大卫

This week's content

这周的内容

- Today Wednesday:
  - Chapter 26: Maximum Flow （only 26.1–26.3)
  - Exercises
- Tomorrow Thursday:
  - Exercise solutions
  - Chapter 34: NP-completeness (B. Zhan)

- 今天周三：
  - 第26章：最大流 (仅26.1–26.3）
  - 练习
- 明天周四：
  - 练习题解答
  - 第34章：NP完全性 （詹老师）

Algorithm Design and Analysis

Maximum Flow

David N. JANSEN
名 姓

算法设计与分析

最大流

杨大卫

Ch. 26

26章

Weighted Graphs 权重的图

Shortest Path 最短路径

Minimum Spanning Tree 最小生成树

weight = length or cost of edge
权重 = 边的长度或者代价

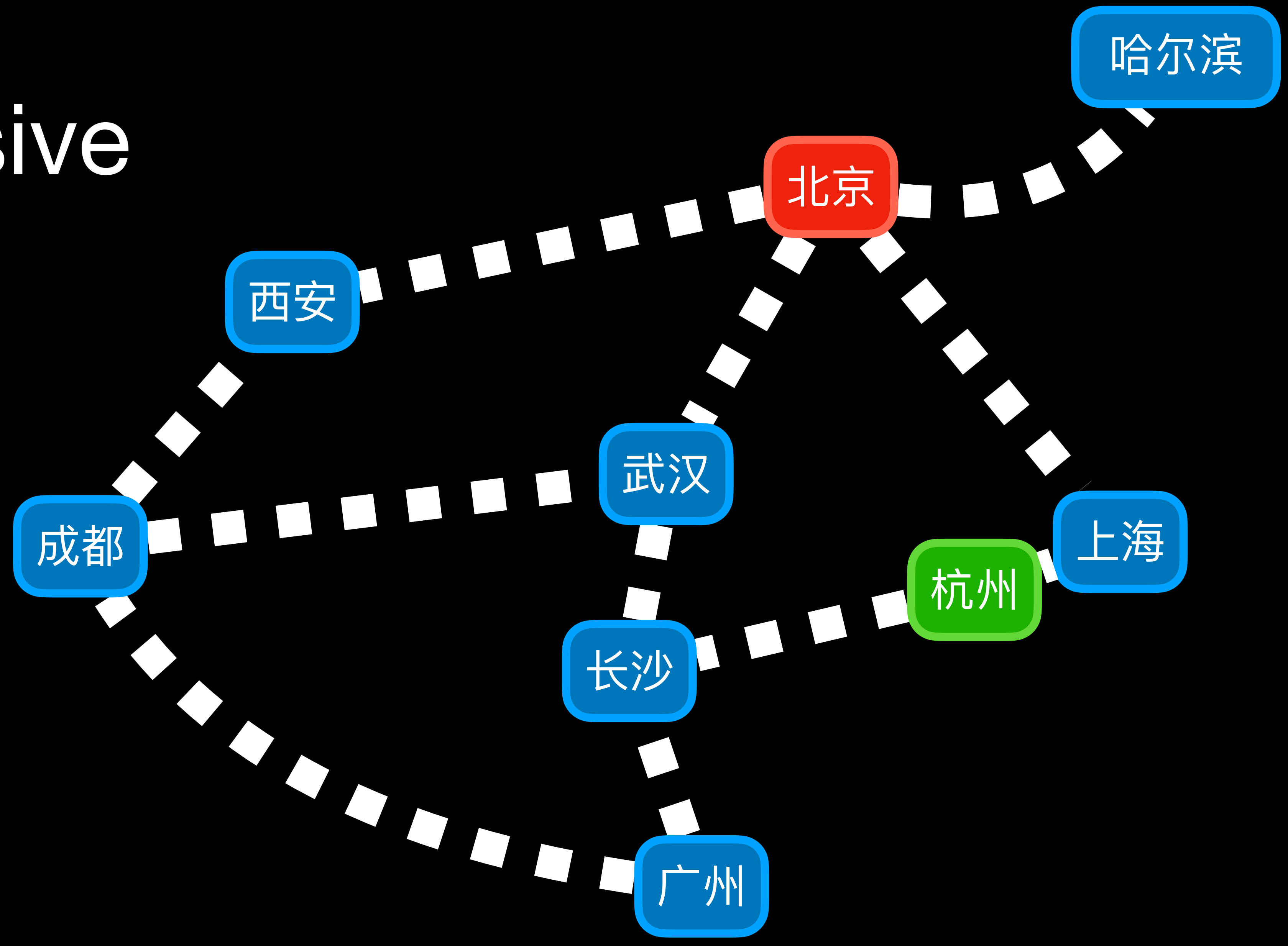
How far / how expensive is the trip from Beijing to Hangzhou?

What is the cheapest connected network?

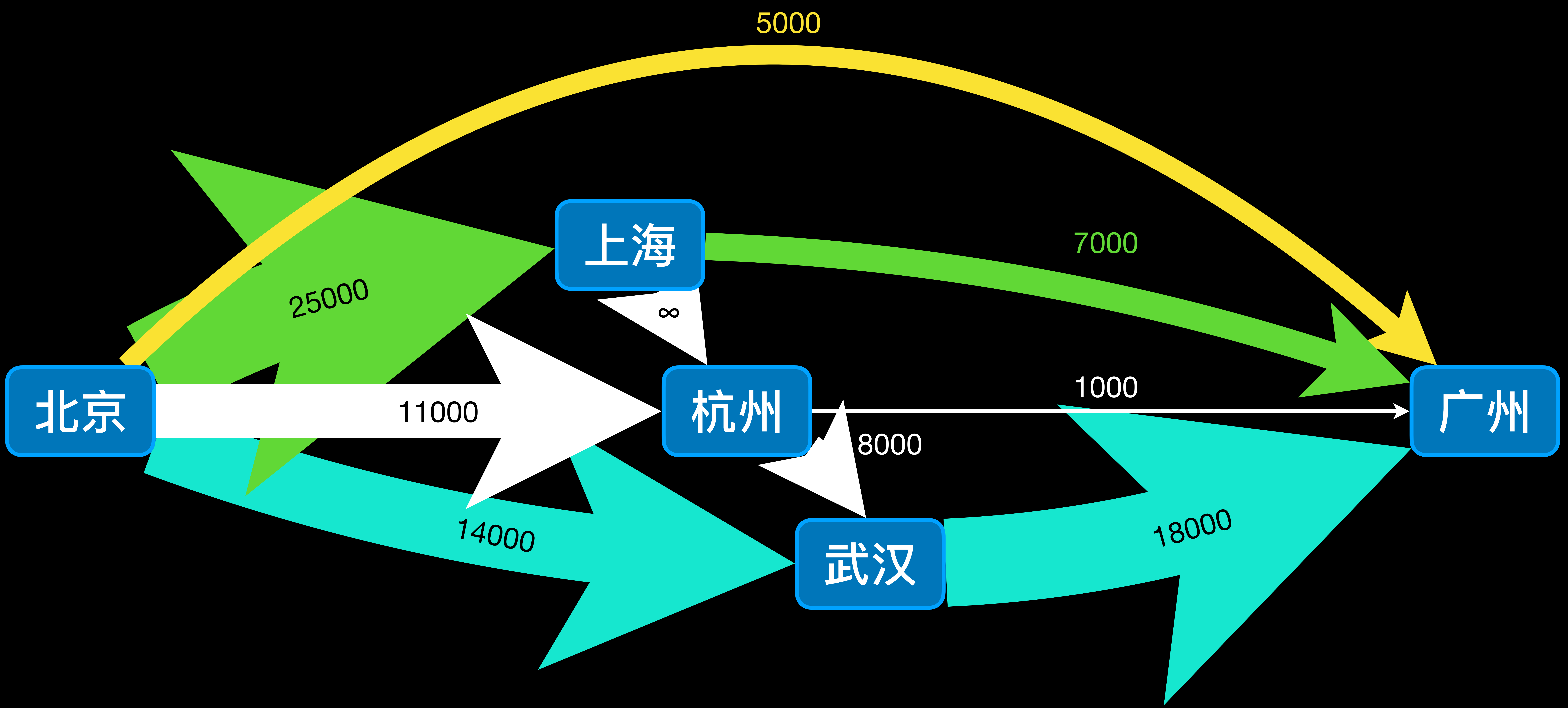
Maximum Flow 最大流

weight = capacity or width of edge
权重 = 边的容量或者宽度

How many people can travel each day from Beijing to Guangzhou?



How many people can travel?



Flow Network

流网络

- weighted graph $G = (V,E)$:
  every edge has a capacity, c: $E \to \mathbb{R}^{>0} \cup \{\infty\}$.
  (If there is no edge (u,v), then $c(u,v) = 0$.)

- source s and sink $t \in V$.

- simplifying technical assumptions:
  - no edge in both directions
    $(u,v) \in E \implies (v,u) \notin E$
    (= no antiparallel edges)
  - no self-loops $(u,u) \notin E$
  - every vertex v is on a path from source to sink $s \rightsquigarrow v \rightsquigarrow t$

- 权重的图 $G = (V,E)$：
  每条边有个容量值, c: $E \to \mathbb{R}^{>0} \cup \{\infty\}$.
  （如果没有变 (u,v)，定义 $c(u,v) = 0$。）

- 源点 s 和汇点 $t \in V$。

- 简化技术假设：
  - 没有两方向的边 $(u,v) \in E \implies (v,u) \notin E$
    （= 没有反平行边）
  - 没有自环 $(u,u) \notin E$
  - 所有的结点 v 有从源点到汇点的路径
    $s \rightsquigarrow v \rightsquigarrow t$

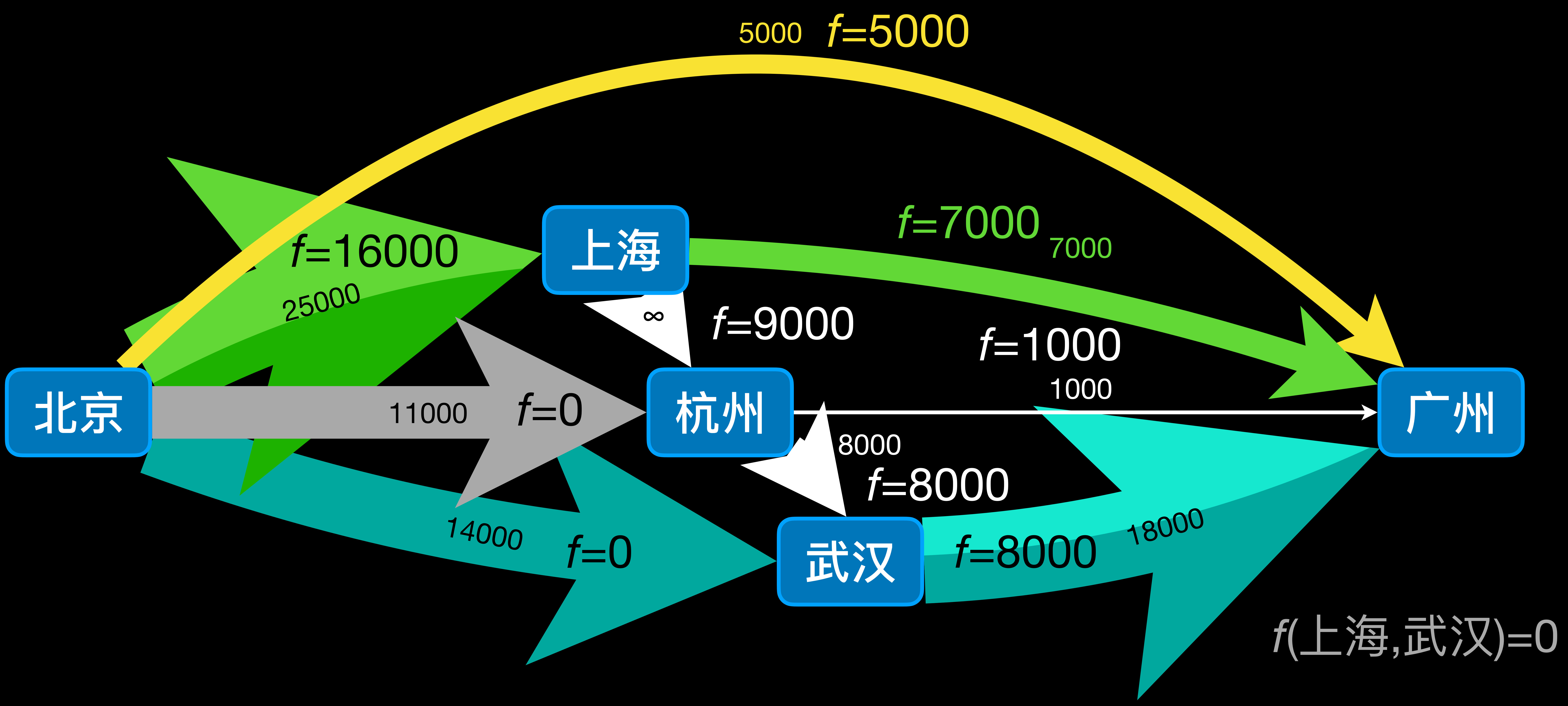
(Network) Flow

（网络的）流

- A flow $f: V \times V \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ is a function that indicates how much material can flow through the network.
- **Capacity constraint:** for all $u,v \in V$, we have $0 \leq f(u,v) \leq c(u,v)$.
- **Flow conservation:** For all $u \in V \setminus \{s,t\}$, we have $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$.
- Value of the flow: $|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$.
- Q: What is the maximum possible value of $|f|$?

- 流 $f: V \times V \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ 是一个表示多少物质可以流过网络的函数。
- **容量限制：**对于所有 $u,v \in V$，我们有 $0 \leq f(u,v) \leq c(u,v)$。
- **流量守恒：**对于所有 $u \in V \setminus \{s,t\}$，我们有 $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$。
- 流的值： $|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$。
- 问： $|f|$ 的最大可能值是多少?

Flow Example



Example: Simulation Relation 类似关系

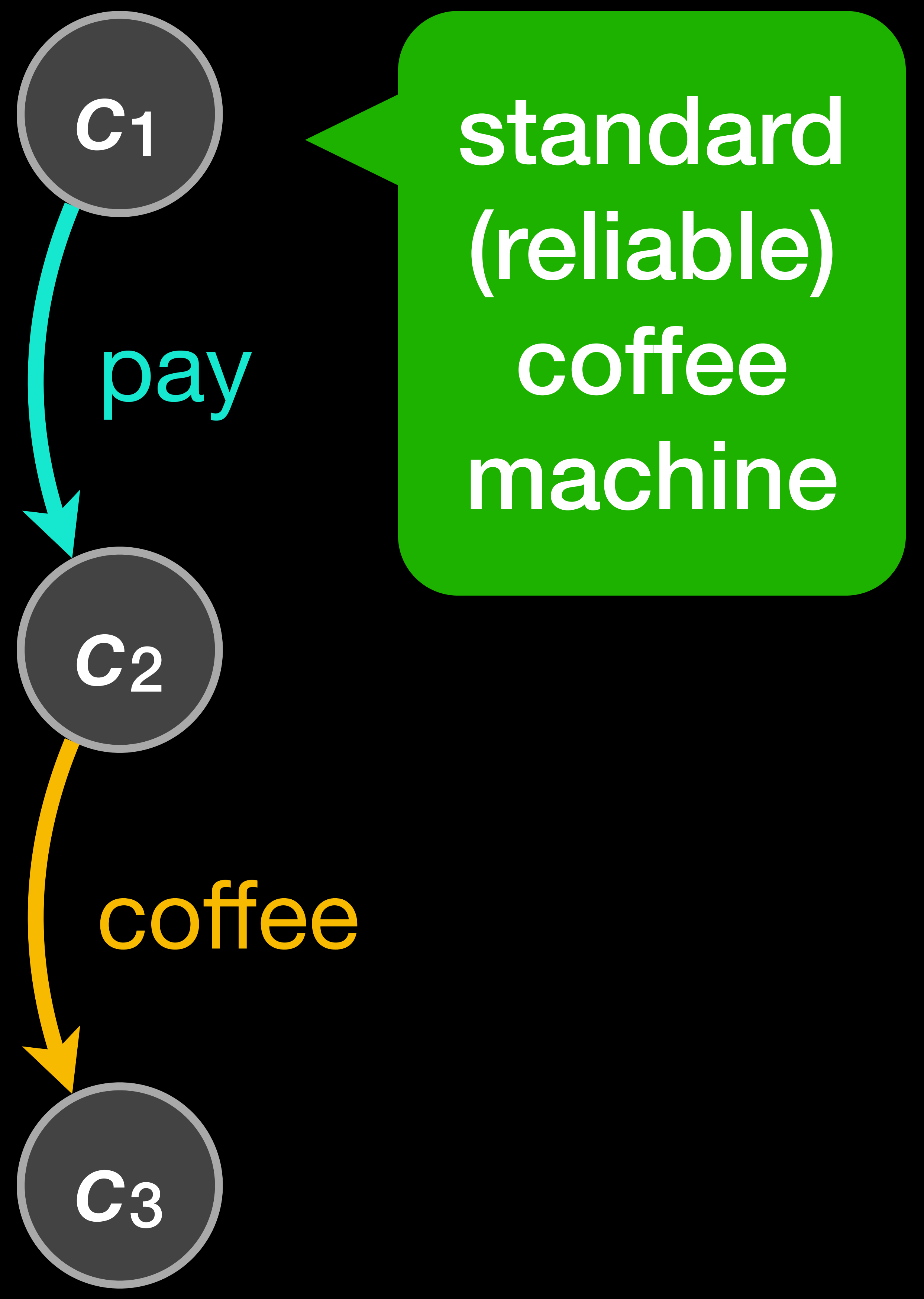
- Simulation relations compare behaviour models.
  They answer the question “Can *Impl* do everything that *Spec* can do?”

- They are a way to describe whether an implementation satisfies the specification:
  If the implementation can do everything required by the specification, the system is correct.

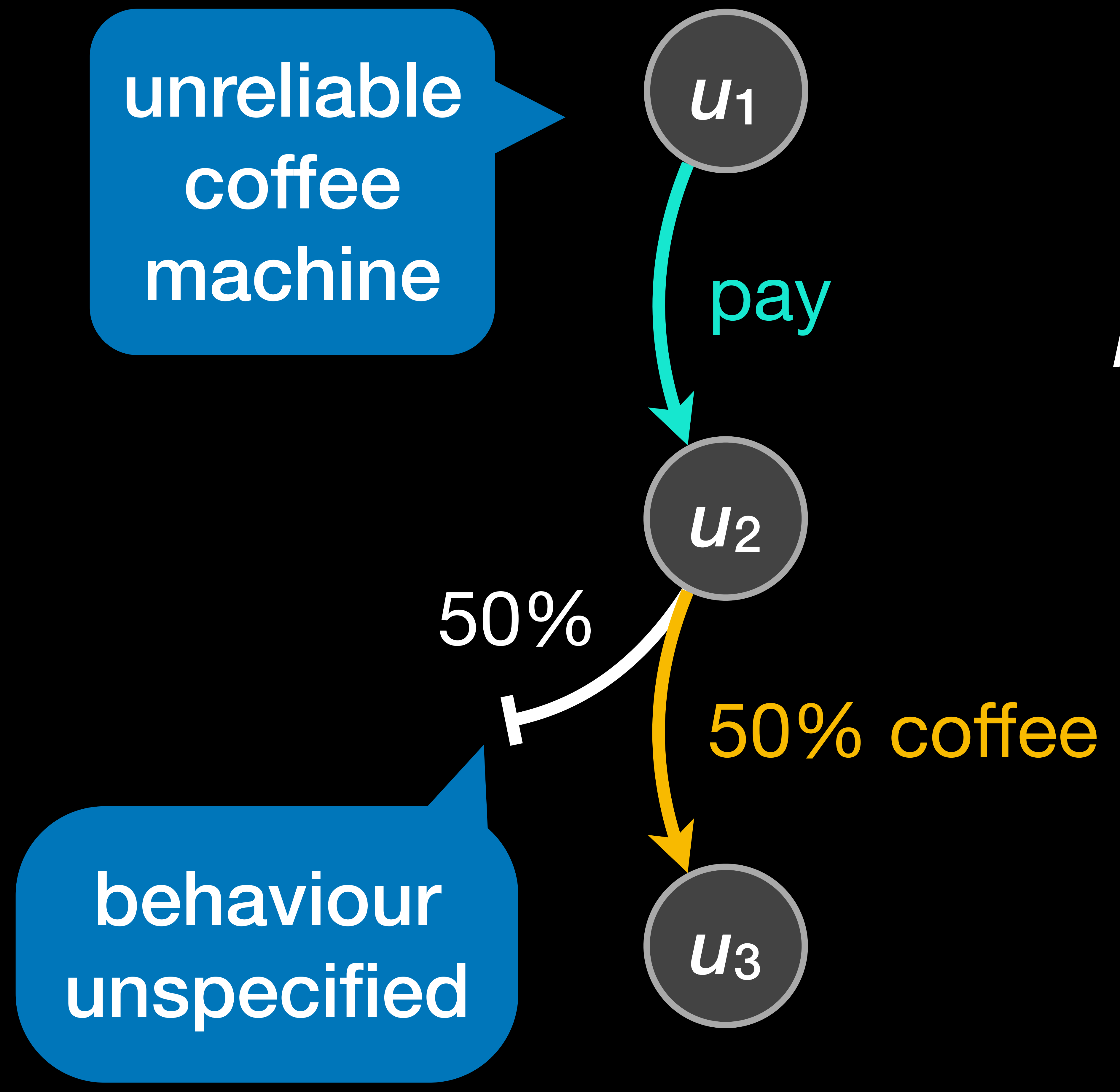
- Probabilistic simulation relation

Zhang, Lijun; Jansen, David N.: **A space-efficient simulation algorithm on probabilistic automata.**
Information and computation 249, 2016. pp. 138–159. http://dx.doi.org/10.1016/j.ic.2016.04.002

Reliable Coffee Machine

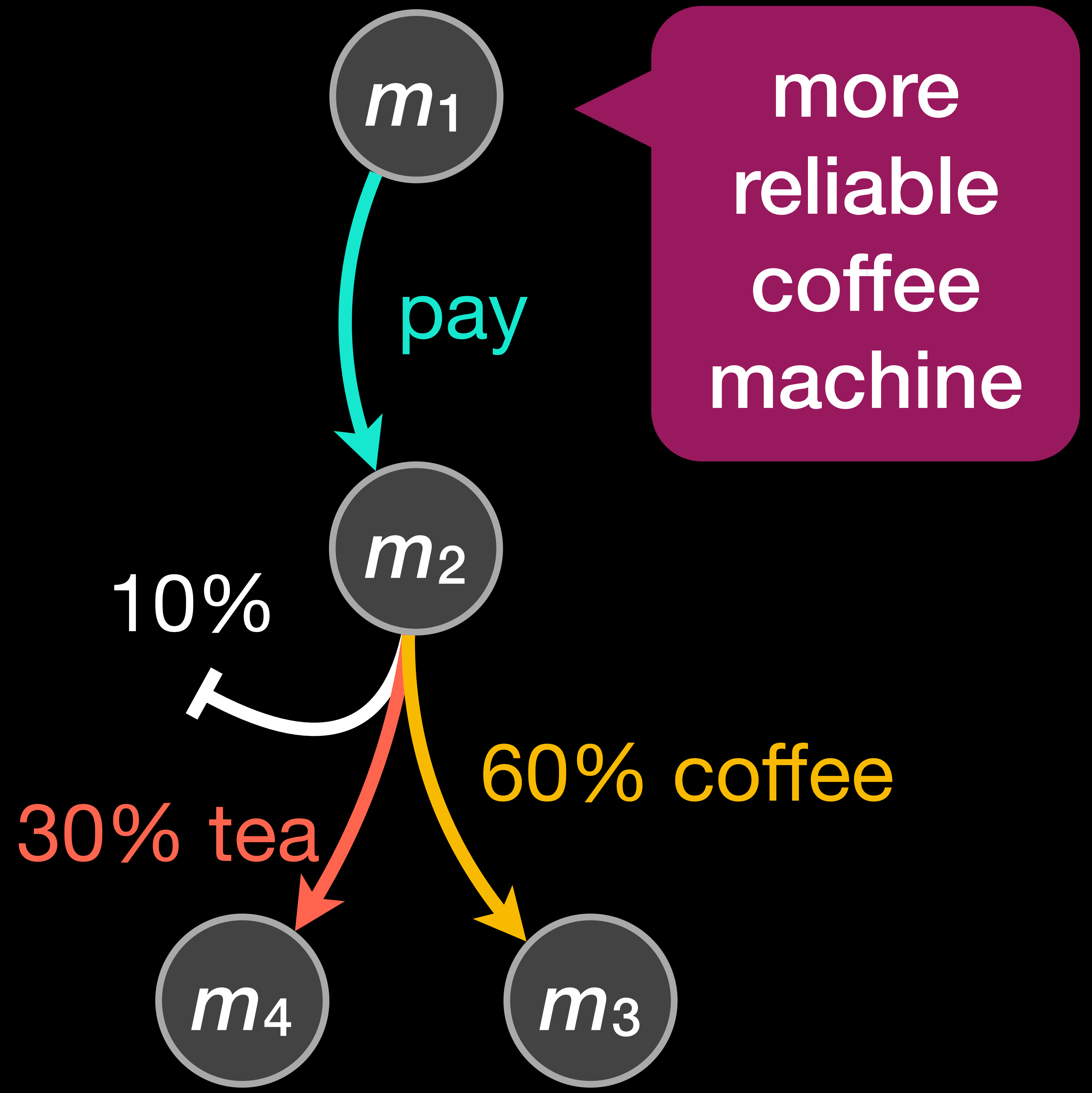


Unreliable Coffee Machines 摩卡墨彩

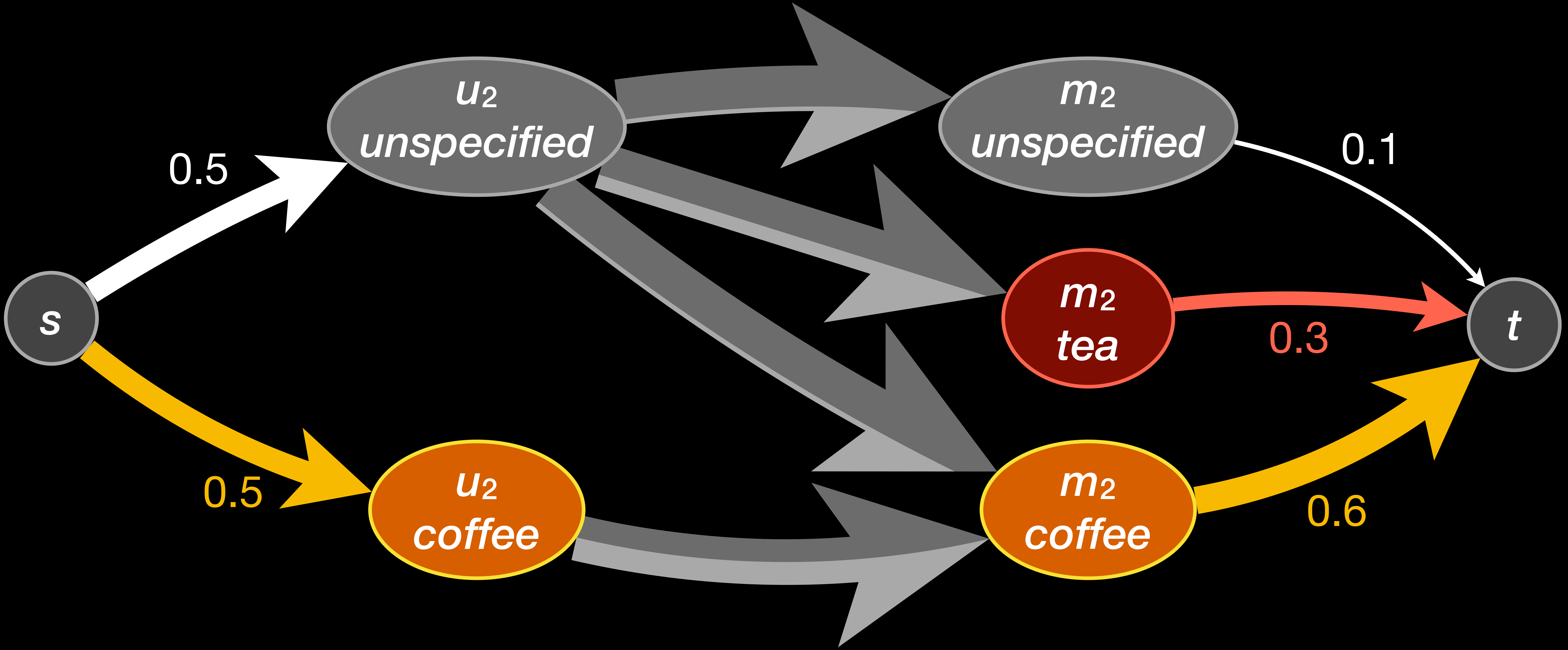


$\precsim$

m_1 simulates u_1.
m_1 is a better coffee machine than u_1.



Coffee Machine Comparison



This network has a flow with value 1, so we have $u_2 \precsim m_2$.

Overview

- **Ford–Fulkerson**
  Idea: The method keeps a legal flow f
  and tries to improve it.

  Augmenting path =
  path from s to t that can carry
  more flow than currently in f.

- **Edmonds–Karp**
  Idea: Choose shortest augmenting
  paths to improve running time.

概述

- **Ford–Fulkerson**
  理念：方法保持法律流 f
  并试改进它。

  增广路径 = 从 s 到 t 的路径，
  可以承载比 f 中当前
  更多的流量。

- **Edmonds–Karp**
  理念：选择最短的增加路径
  改进运行时间。

Ford–Fulkerson

- Idea: Assume given always a legal flow *f*. This flow may not yet exhaust the capacity of the network.
  If possible, improve it.

- Try to find an augmenting path, i.e. a path from *s* to *t* that can carry more flow than currently in *f*.

- To find an augmenting path, use the residual network. This network indicates how *f* can be changed.

- 想法：假设总是有一个法律流 *f*。
  这种流量可能还没有耗尽网络的容量。

  如果可能的话，改进它。

- 尝试找到一条增广路径，
  即从 *s* 到 *t* 的路径，该路径可以承载比当前f中更多的流量。

- 要找到增广路径，使用残存网络。
  此网络指示如何更改 *f*。

Ford–Fulkerson

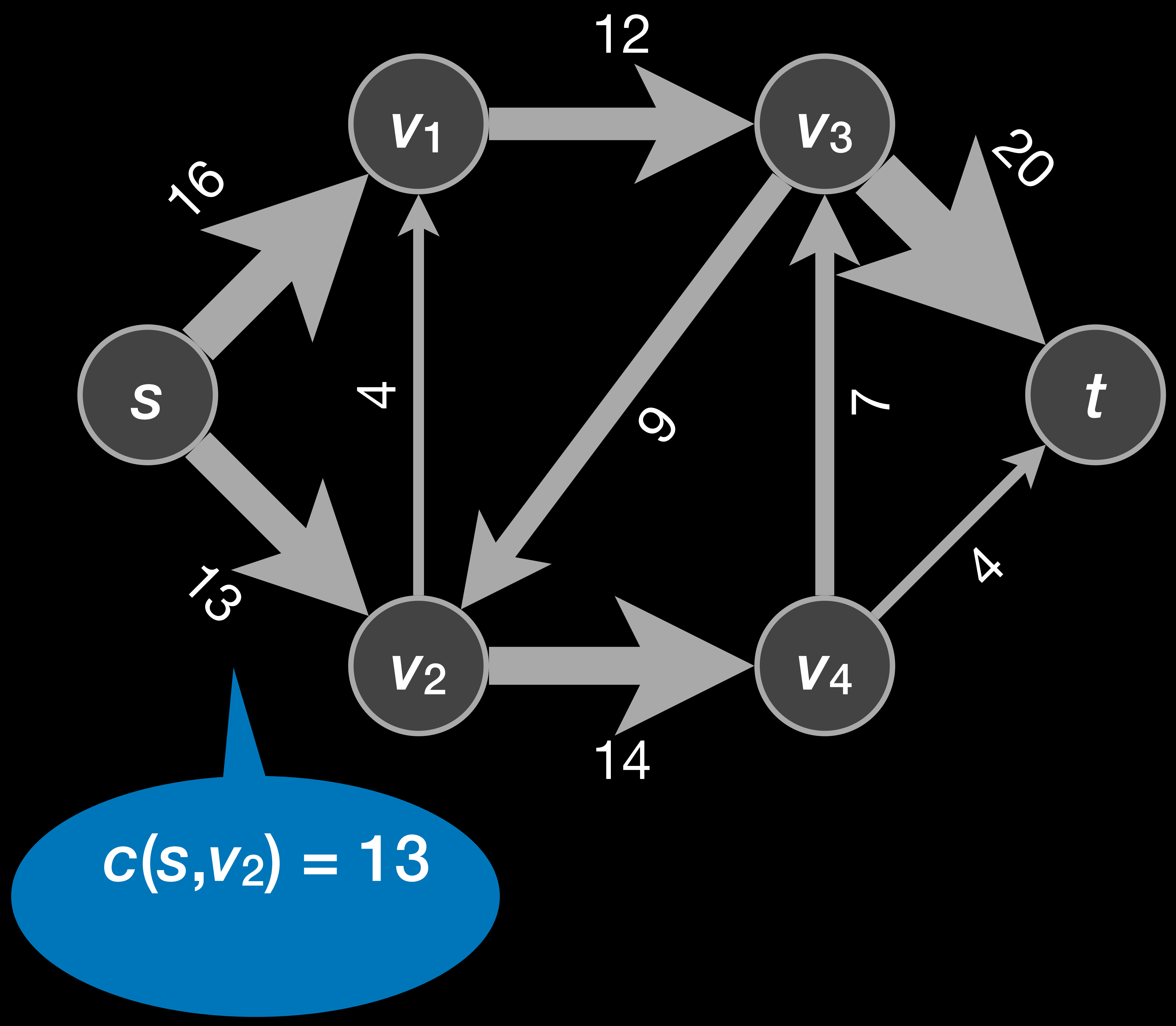
- Idea: Assume given always a legal flow *f*.
  This flow may not yet exhaust the capacity of the network.
  If possible, improve it.

- 想法：假设总是有一个法律流 *f*。
  这种流量可能还没有耗尽网络的容量。

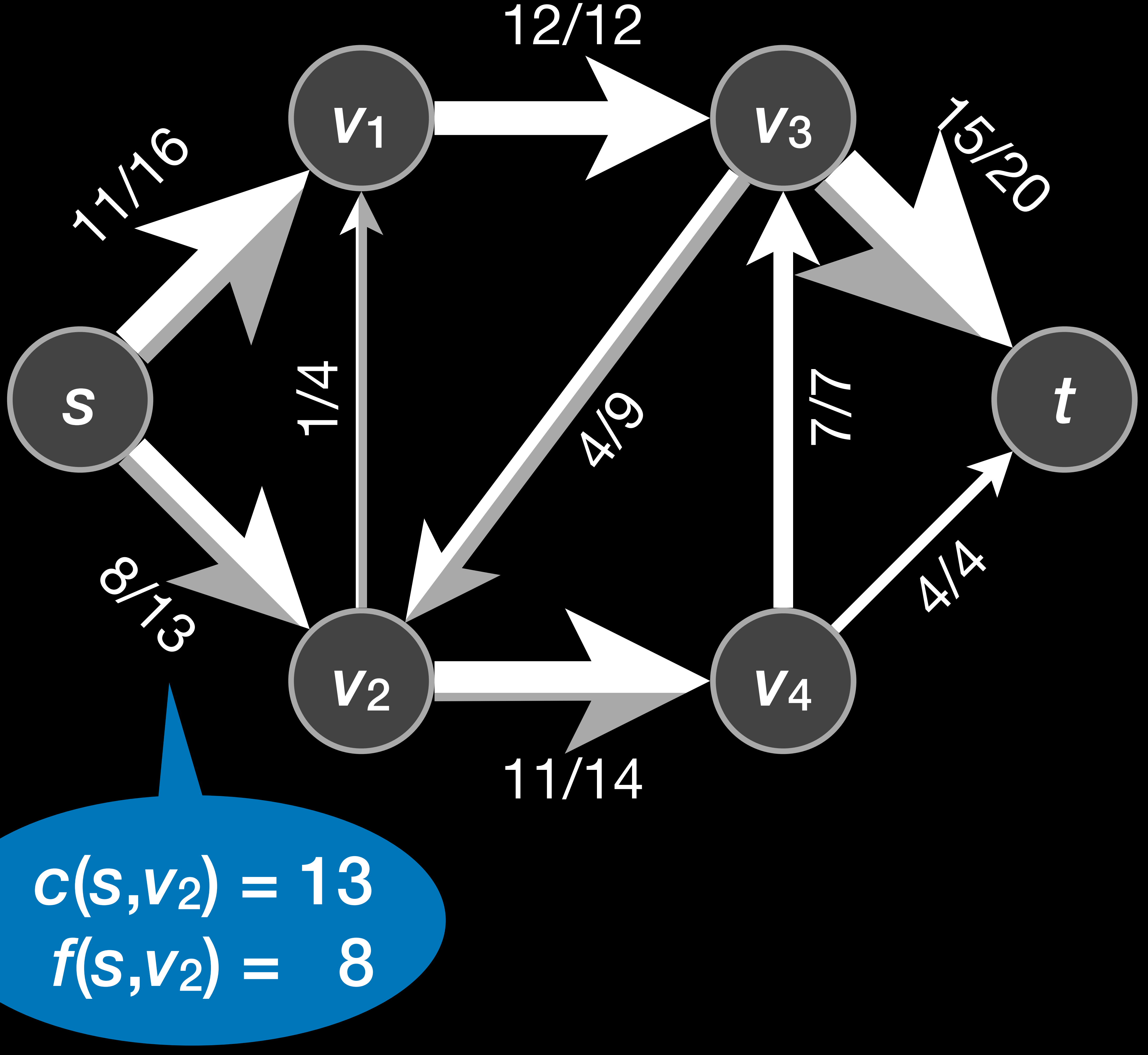
  如果可能的话，改进它。

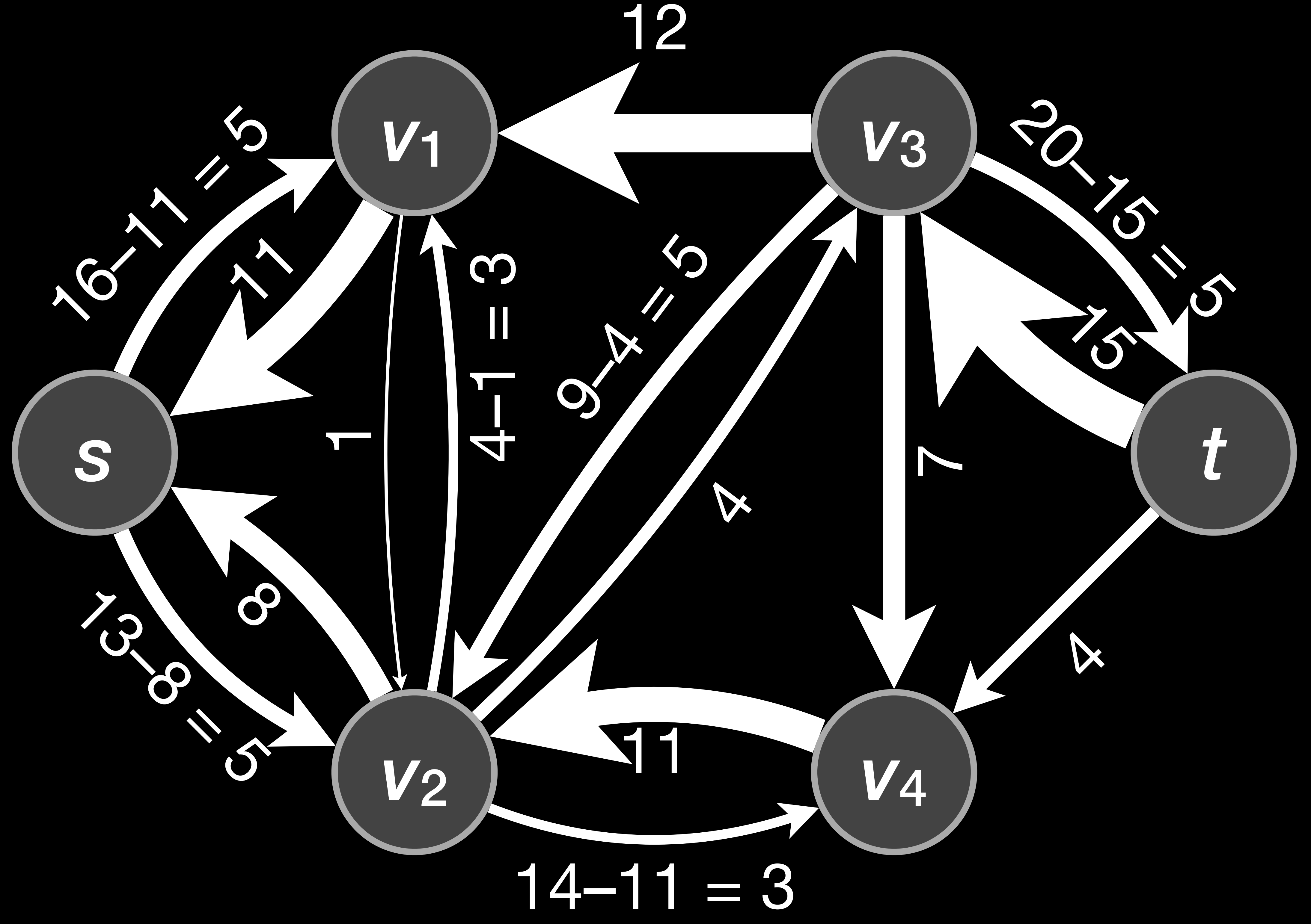
FORD-FULKERSON(G,c,s,t)
initialize flow f to 0
while there exists an augmenting path p in the residual network G_f
    augment f by p
return f

Residual Network 残存网络

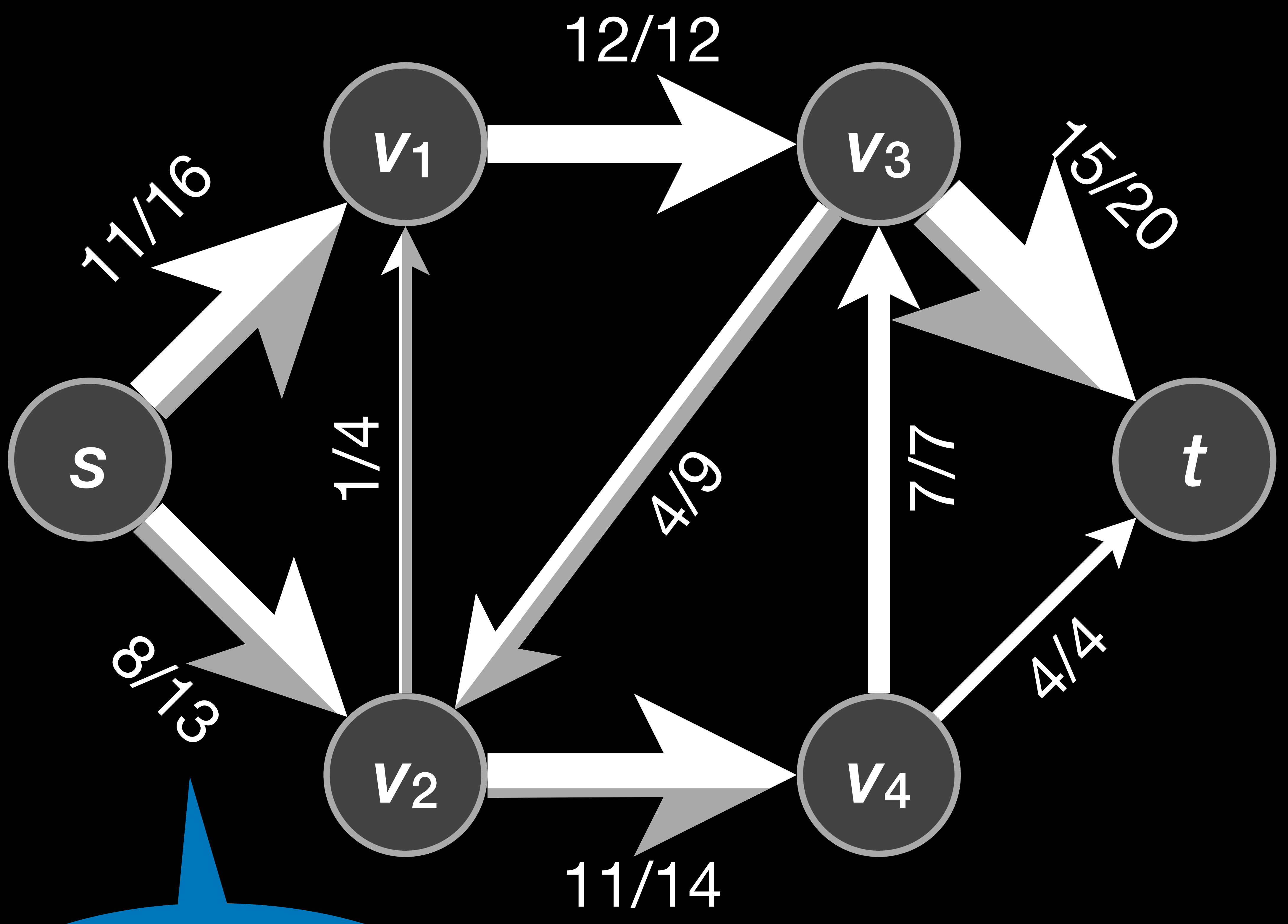


Residual Network 残存网络

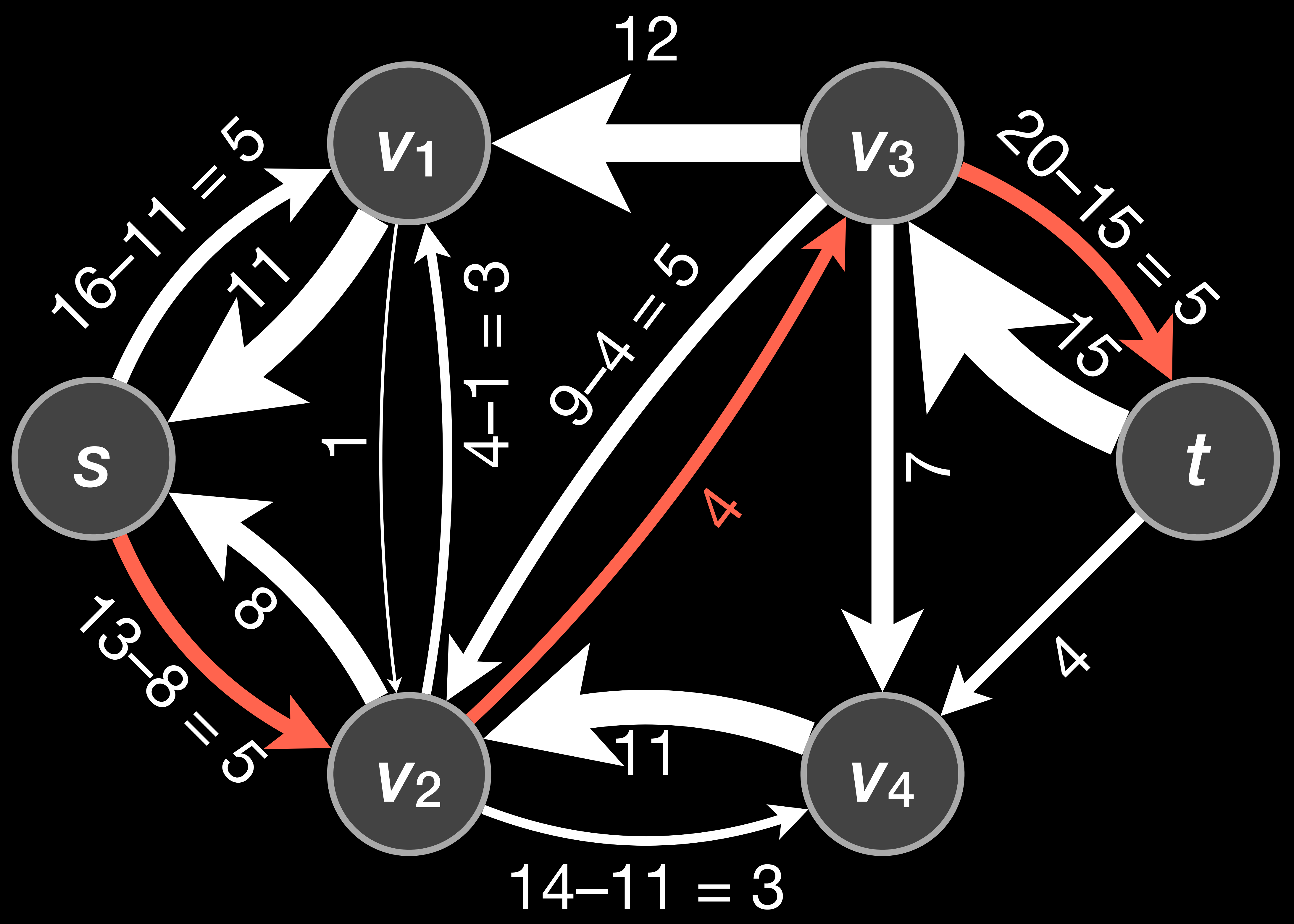




Residual Network 残存网络

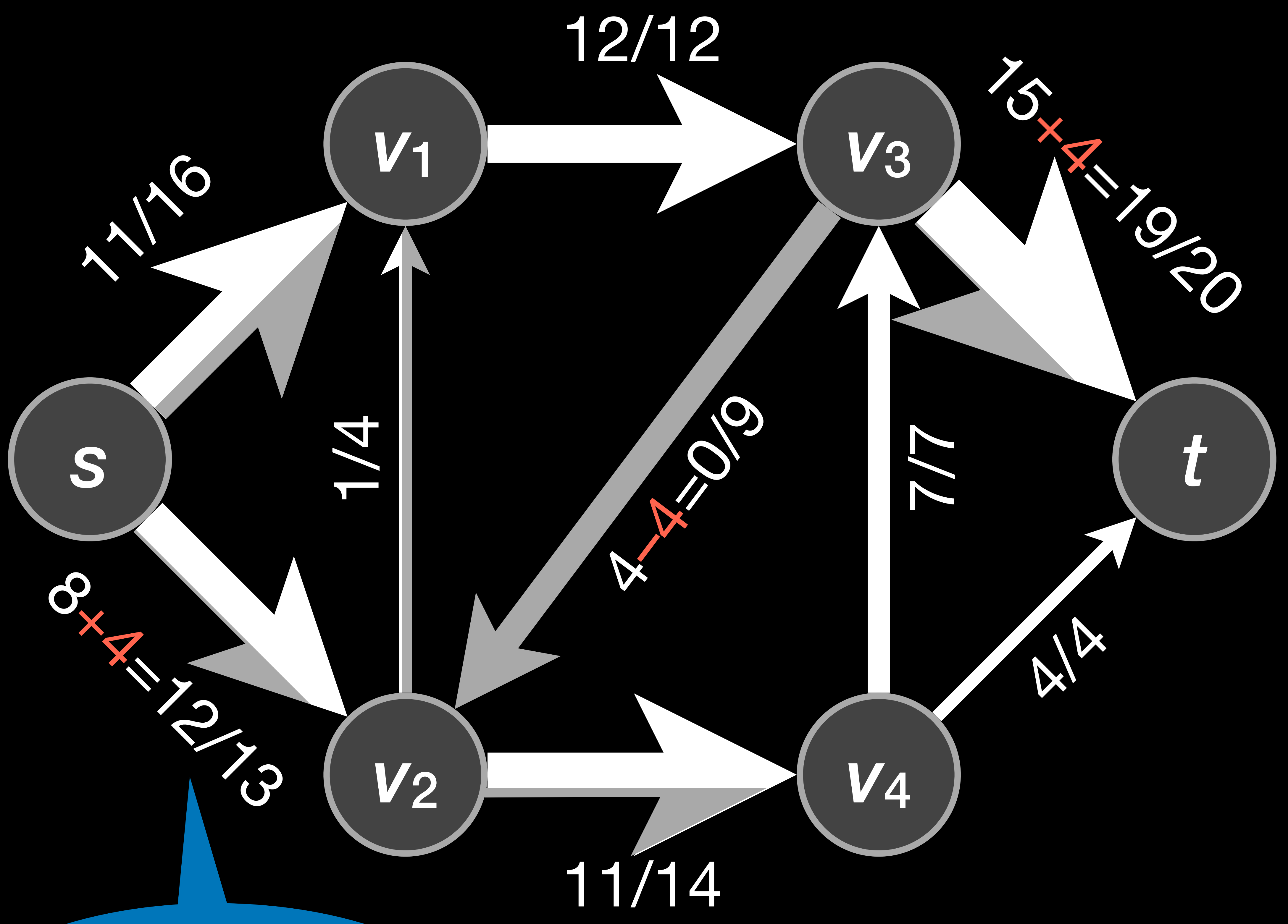


$c(s, v_2) = 13$
$f(s, v_2) = \ \ 8$

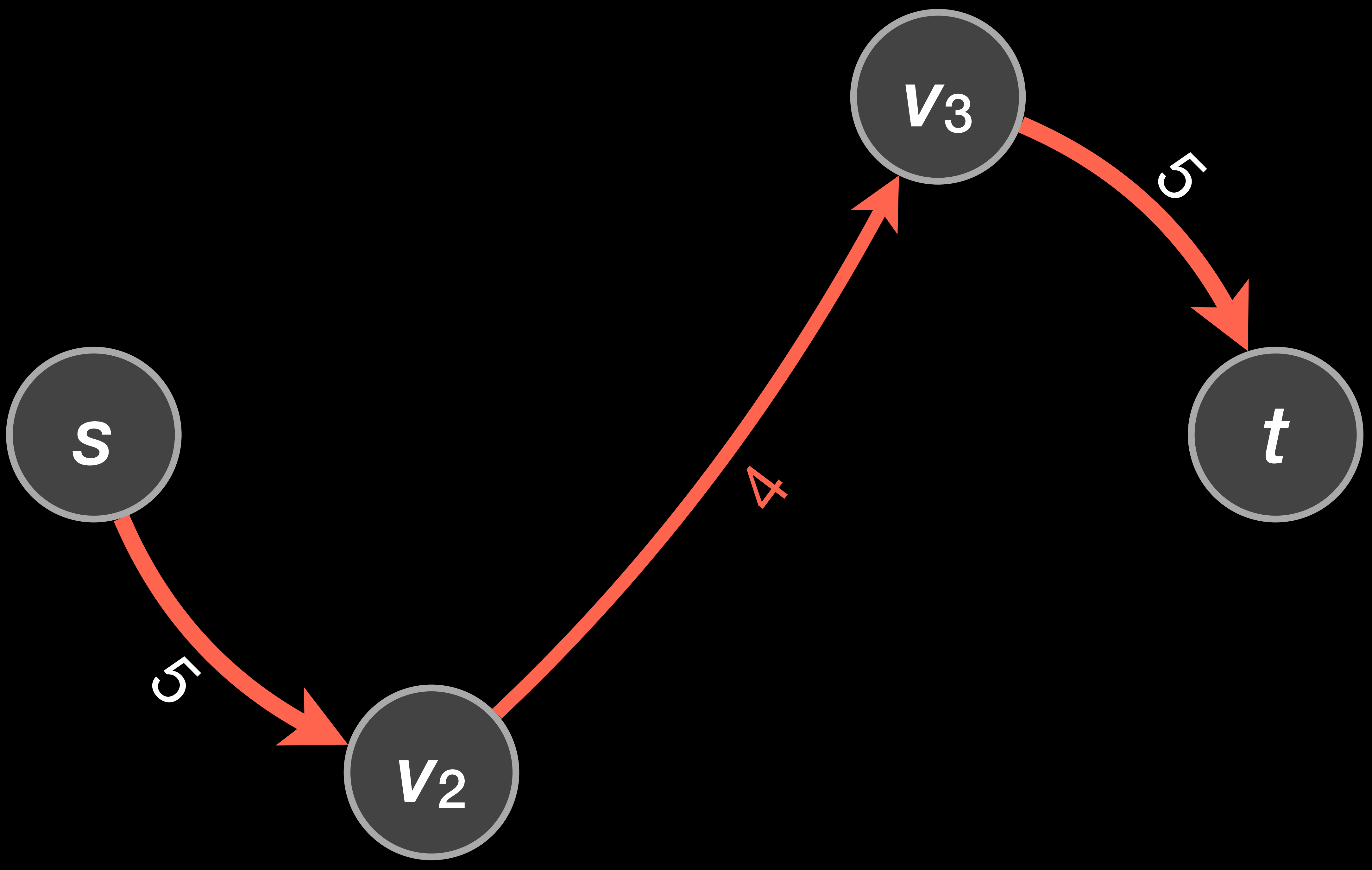


The red path is an augmenting path with capacity 4.
红色的路径是容量4的增广路径。

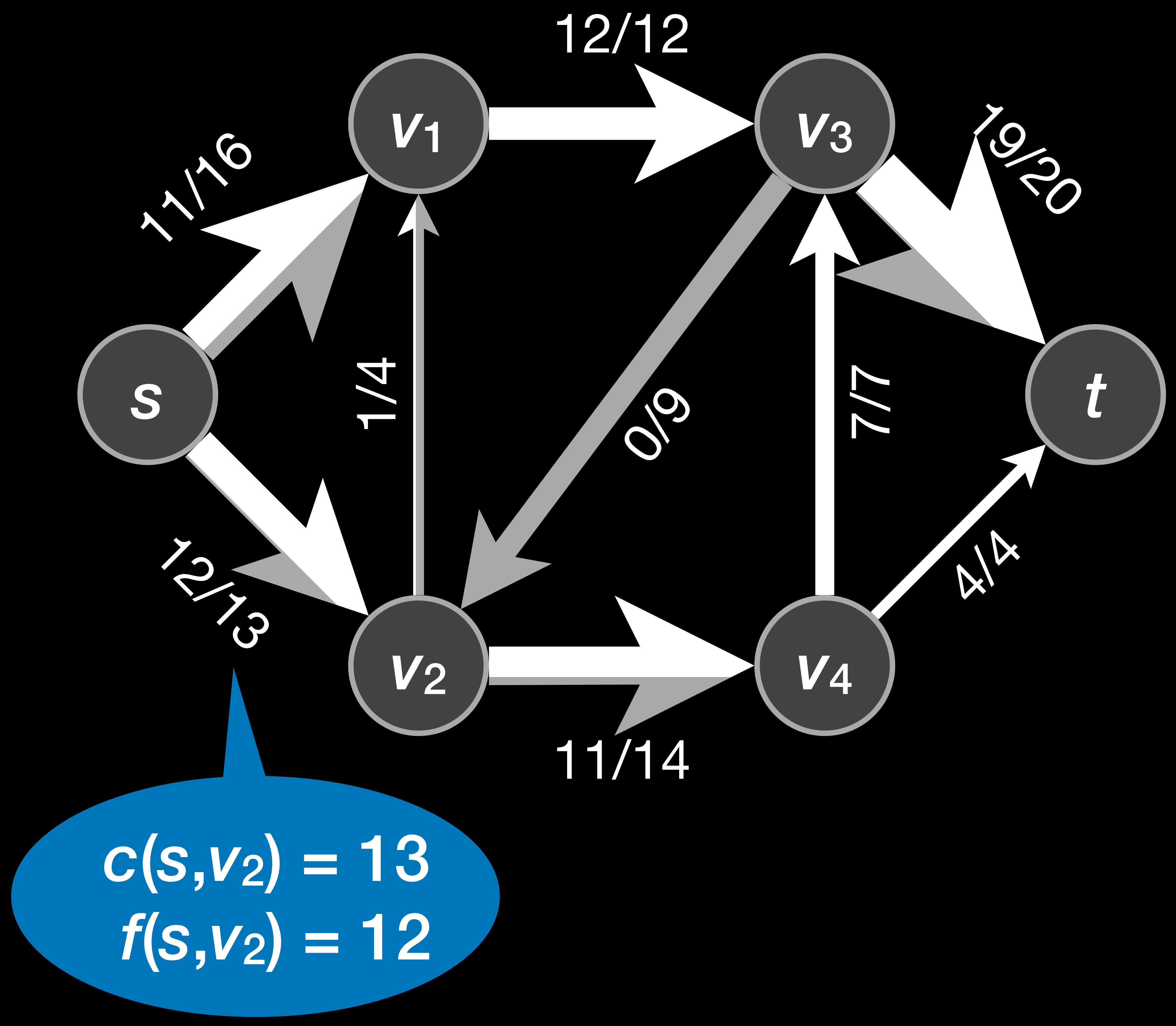
Augmentation 递增

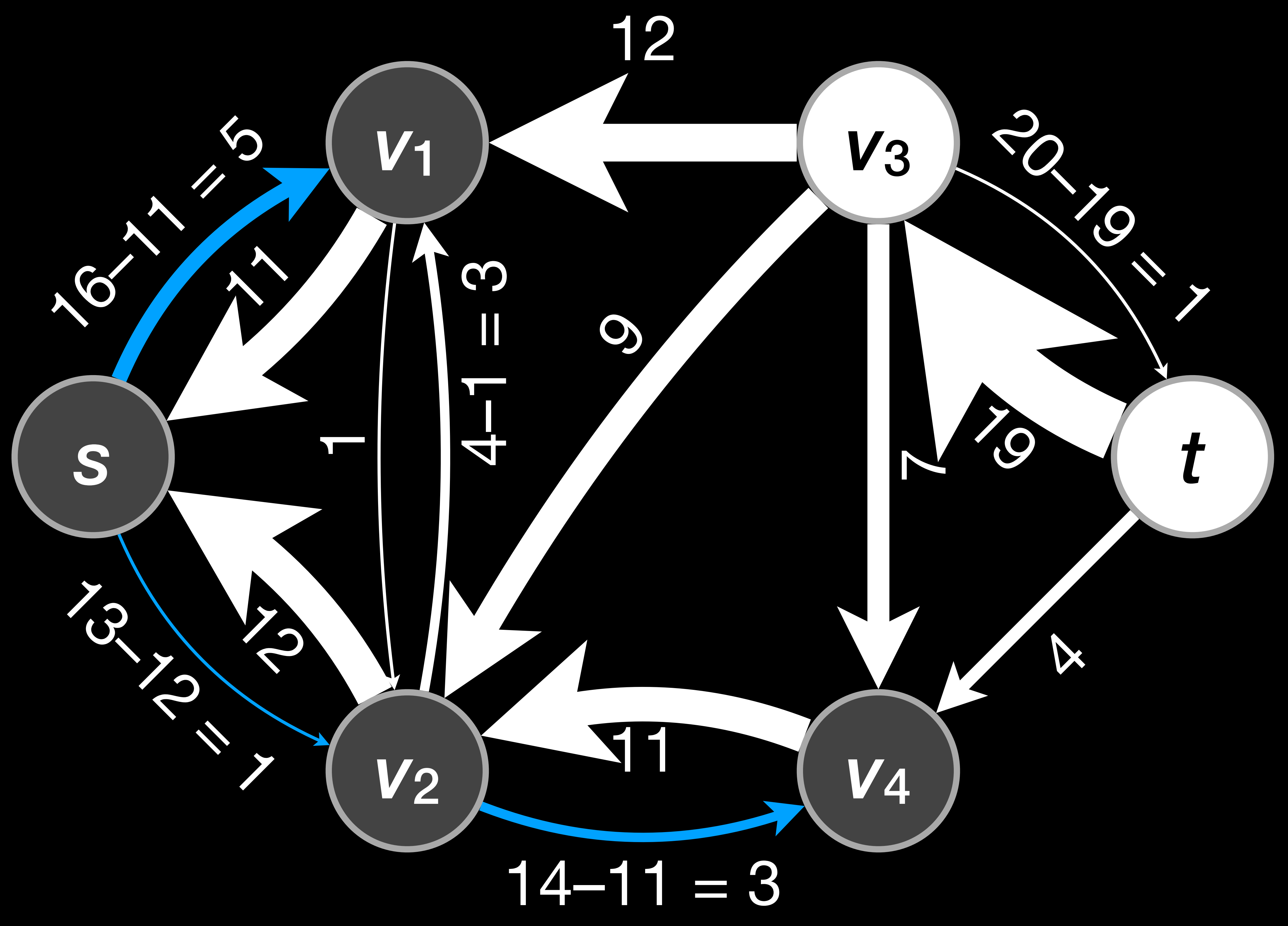


$c(s,v_2) = 13$
$f(s,v_2) = 12$



New Residual Network 残存网络





No more augmenting paths
增广路径没有了

Residual Network 残存网络

- For flow network $G = ((V,E),c)$ and network flow f, the residual network $G_f = ((V,E_f),c_f)$ contains the same vertices V as the flow network and all edges and antiparallel edges $E_f = E \cup \{(v,u) \mid (u,v) \in E\}$.
- The edges have capacity:
  $c_f(u,v) = c(u,v) - f(u,v)$ if $(u,v) \in E$;
  $c_f(u,v) = f(v,u)$ if $(v,u) \in E$.
- A residual network is almost a flow network—only there can be antiparallel edges.

- 对于流网络 $G = ((V,E),c)$ 和网络流 f，残存网络 $G_f = ((V,E_f),c_f)$ 包含与流网络相同的顶点 V，以及所有边和反平行边 $E_f = E \cup \{(v,u) \mid (u,v) \in E\}$。
- 边的容量定义：
  $c_f(u,v) = c(u,v) - f(u,v)$ 如果 $(u,v) \in E$；
  $c_f(u,v) = f(v,u)$ 如果 $(v,u) \in E$。
- 残存网络几乎就是流网络—只有反平行边能存在。

Augmenting Path 增广路径

- A flow f' in the residual network G_f can be used to augment flow f.
  $(f \uparrow f')(u,v) = f(u,v) + f'(u,v) - f'(v,u)$
- An augmenting path is a simple flow in G_f—all the flow is on one path from s to t.
- Why augmenting paths?
  Easy to find with breadth-first search or depth-first search.

- 残存网络 G_f 中的流 f' 可以用于递增流 f。
  $(f \uparrow f')(u,v) = f(u,v) + f'(u,v) - f'(v,u)$
- 增广路径是 G_f 中的一个简单流—完全的流动都在独一条从 s 到 t 的路径上。
- 为什么使用增广路径?
  容易找到用于广度优先搜索或者深度优先搜索。

Questions

问一问

- Is every augmentation $f \uparrow f'$ a correct flow?
  i.e., does it satisfy the capacity constraint and flow conservation?

- How can we find augmenting paths?

- Will the method terminate?
  Will the final result be a correct maximum flow?

- How efficient is the method?

- 所有的递增 $f \uparrow f'$ 都是否正确的流?

  即，是否满足容量限制和流量守恒?

- 怎么样找到增广路径?

- 这种方法终止吗?
  最终结果是正确的最大流吗?

- 这种方法的效率有多高?

Is augmentation $f \uparrow f'$ correct?

Lemma 26.1: Given a flow network $G = ((V,E),c)$ and a flow f in G; let f' be a flow in the residual network G_f. Then $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

- Proof of capacity constraints:

  Assume that (u,v) is in E.

  Then $f(u,v) = c_f(v,u) \geq f'(v,u)$, and so

$$\begin{aligned}(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\ &\geq \cancel{f(u,v)} + f'(u,v) - \cancel{f(u,v)} \\ &\geq 0.\end{aligned}$$

递增 $f \uparrow f'$ 是否正确的?

引理 26.1：设 $G = ((V,E),c)$ 为一个流网络，设 f 为 G 中的一个流；设 f' 为残存网络 G_f 中一个流。那么 $f \uparrow f'$ 是 G 的一个流，其值为 $|f \uparrow f'| = |f| + |f'|$。

- 证明容量限制：

  假设 $(u,v) \in E$。

$$\begin{aligned}(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\ &\leq f(u,v) + f'(u,v) \\ &\leq f(u,v) + c_f(u,v) \\ &= \cancel{f(u,v)} + c(u,v) - \cancel{f(u,v)}.\end{aligned}$$

Is augmentation $f \uparrow f'$ correct?

递增 $f \uparrow f'$ 是否正确的?

Lemma 26.1: Given a flow network $G = ((V,E),c)$ and a flow f in G; let f' be a flow in the residual network G_f. Then $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

引理 26.1：设 $G = ((V,E),c)$ 为一个流网络，设 f 为 G 中的一个流；设 f' 为残存网络 G_f 中一个流。那么 $f \uparrow f'$ 是 G 的一个流，其值为 $|f \uparrow f'| = |f| + |f'|$。

- Proof of flow conservation:
  Assume given any vertex $u \in V$.

- 证明流量守恒：
  设给结点 $u \in V$。

$$\sum_{v \in V} (f \uparrow f')(u,v) - \sum_{v \in V} (f \uparrow f')(v,u) = \sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) + \sum_{v \in V} f'(u,v) - \sum_{v \in V} f'(v,u)$$

because of

因为

$$\sum_{v \in V_{out}(u)} (f \uparrow f')(u,v) - \sum_{v \in V_{in}(u)} (f \uparrow f')(v,u) = \sum_{v \in V_{out}(u)} f(u,v) - \sum_{v \in V_{in}(u)} f(v,u) + \sum_{v \in V_{out}(u)} f'(u,v) - \sum_{v \in V_{in}(u)} f'(v,u)$$

Is augmentation $f \uparrow f'$ correct?

递增 $f \uparrow f'$ 是否正确的?

Lemma 26.1: Given a flow network $G = ((V,E),c)$ and a flow f in G; let f' be a flow in the residual network G_f. Then $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

引理 26.1：设 $G = ((V,E),c)$ 为一个流网络，设 f 为 G 中的一个流；设 f' 为残存网络 G_f 中一个流。那么 $f \uparrow f'$ 是 G 的一个流，其值为 $|f \uparrow f'| = |f| + |f'|$。

- Proof of flow conservation:
  Assume given any vertex $u \in V \setminus \{s,t\}$.

- 证明流量守恒：
  设给结点 $u \in V \setminus \{s,t\}$。

$$\underbrace{\sum_{v\in V} (f \uparrow f')(u,v) - \sum_{v\in V} (f \uparrow f')(v,u)}_{\text{flow conservation of } f \uparrow f'} = \underbrace{\sum_{v\in V} f(u,v) - \sum_{v\in V} f(v,u)}_{= 0 \text{ (if } u \neq s,t) \text{ because of flow conservation of } f} + \underbrace{\sum_{v\in V} f'(u,v) - \sum_{v\in V} f'(v,u)}_{= 0 \text{ (if } u \neq s,t) \text{ because of flow conservation of } f'}$$

Is augmentation $f \uparrow f'$ correct?

递增 $f \uparrow f'$ 是否正确的?

Lemma 26.1: Given a flow network $G = ((V,E),c)$ and a flow f in G; let f' be a flow in the residual network G_f. Then $f \uparrow f'$ is a flow in G with value $\boxed{|f \uparrow f'| = |f| + |f'|}$.

引理 26.1：设 $G = ((V,E),c)$ 为一个流网络，设 f 为 G 中的一个流；设 f' 为残存网络 G_f 中一个流。那么 $f \uparrow f'$ 是 G 的一个流，其值为 $\boxed{|f \uparrow f'| = |f| + |f'|}$。

- Proof of the flow value equation:
  Specialize this equation with $u = s$.

- 证明流值等价：
  使用 $u = s$ 专门化这个等价。

$$\underbrace{\sum_{v \in V} (f \uparrow f')(s,v) - \sum_{v \in V} (f \uparrow f')(v,s)}_{= |f \uparrow f'|} = \underbrace{\sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)}_{= |f|} + \underbrace{\sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s)}_{= |f'|}$$

How to find augmenting paths

怎么样找到增广路径

- Ford and Fulkerson (1962) did not prescribe a specific method. One can use depth-first search or breadth-first search, or any other suitable method to find a path.
- Augmenting paths – instead of general augmenting flows – are used because they are easier to find.
- Many practical implementations use breadth-first search anyway. Edmonds and Karp (1972) proved that this is efficient.

- Ford 和 Fulkerson（1962）没有规定具体的方法。可以使用深度优先搜索或广度优先搜索，或任何其他合适的方法来找到路径。
- 使用增广路径，而不是一般的增广流，因为它们更容易找到。
- 无论如何，许多实际实现使用广度优先搜索。Edmonds 和 Karp（1972）证明了这是有效的。

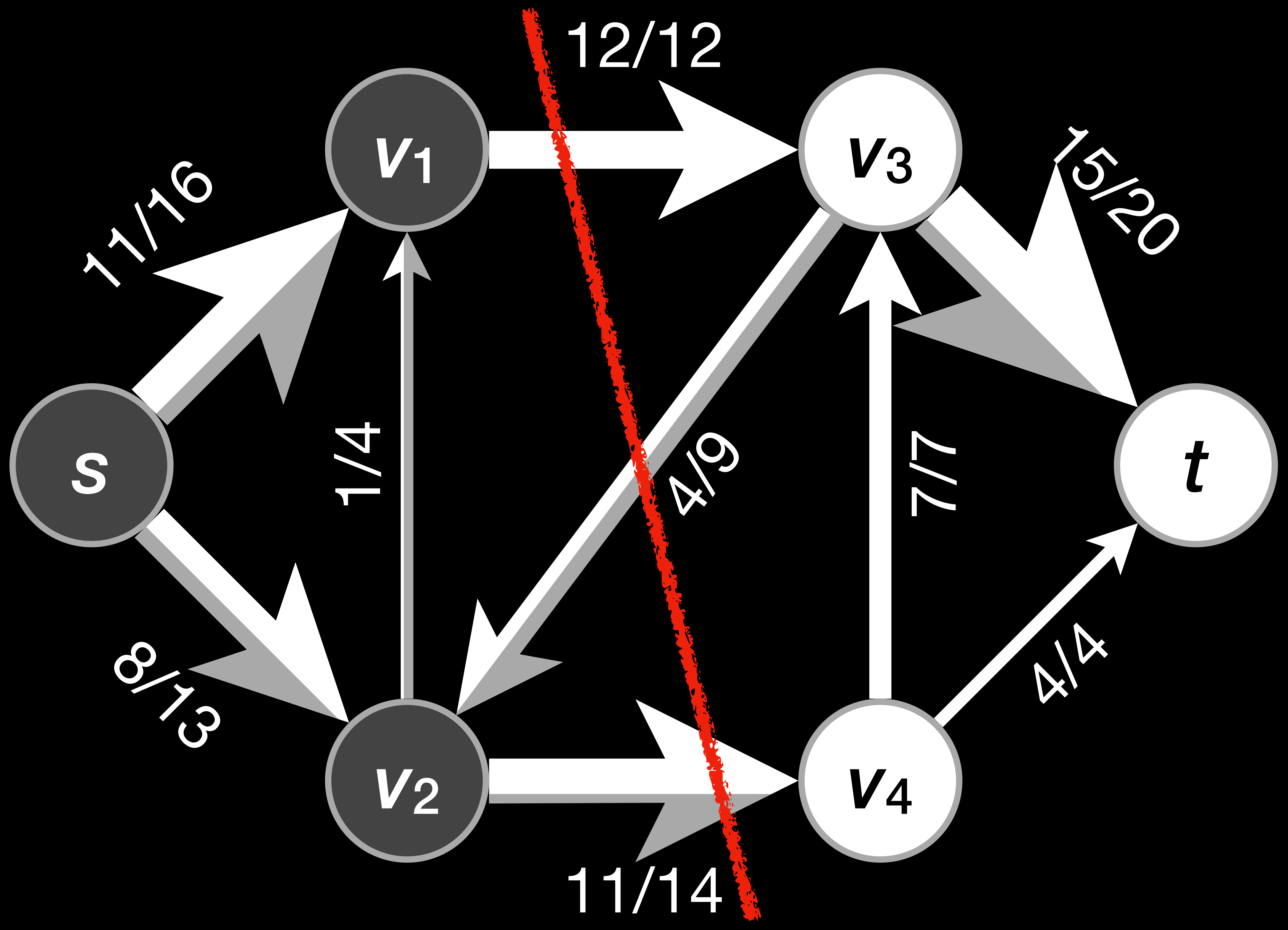
Partial Correctness 部分正确性

- When the algorithm terminates,
  is the returned flow a maximum flow?
- Proof idea: When there is no augmenting path in G_f, then $|f|$ is the capacity of a minimum cut, and then f is a maximum flow.
- uses the max-flow min-cut theorem:
  The value of a maximum flow is equal to the capacity of a minimum cut.

- 当算法终止时,
  返回的流是最大流吗?
- 证明思想：当G_f中没有增广路径时,
  $|f|$ 是最小切割的容量，f 是最大流。
- 使用最大流最小割定理：
  最大流的值等于最小切割的容量。

Cut 切割 of a Flow Network

- A cut separates the vertices into S ($\ni s$) and $T = V \setminus S$ ($\ni t$).
- Net flow across the cut:
  $f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - f(v,u)$
- Capacity of the cut:
  $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$
- Difference is intended: capacity = possible maximum net flow (if we neglect other restrictions)



$f(S,T) = 12 - 4 + 11 = 19$
$c(S,T) = 12 \quad\quad + 14 = 26$

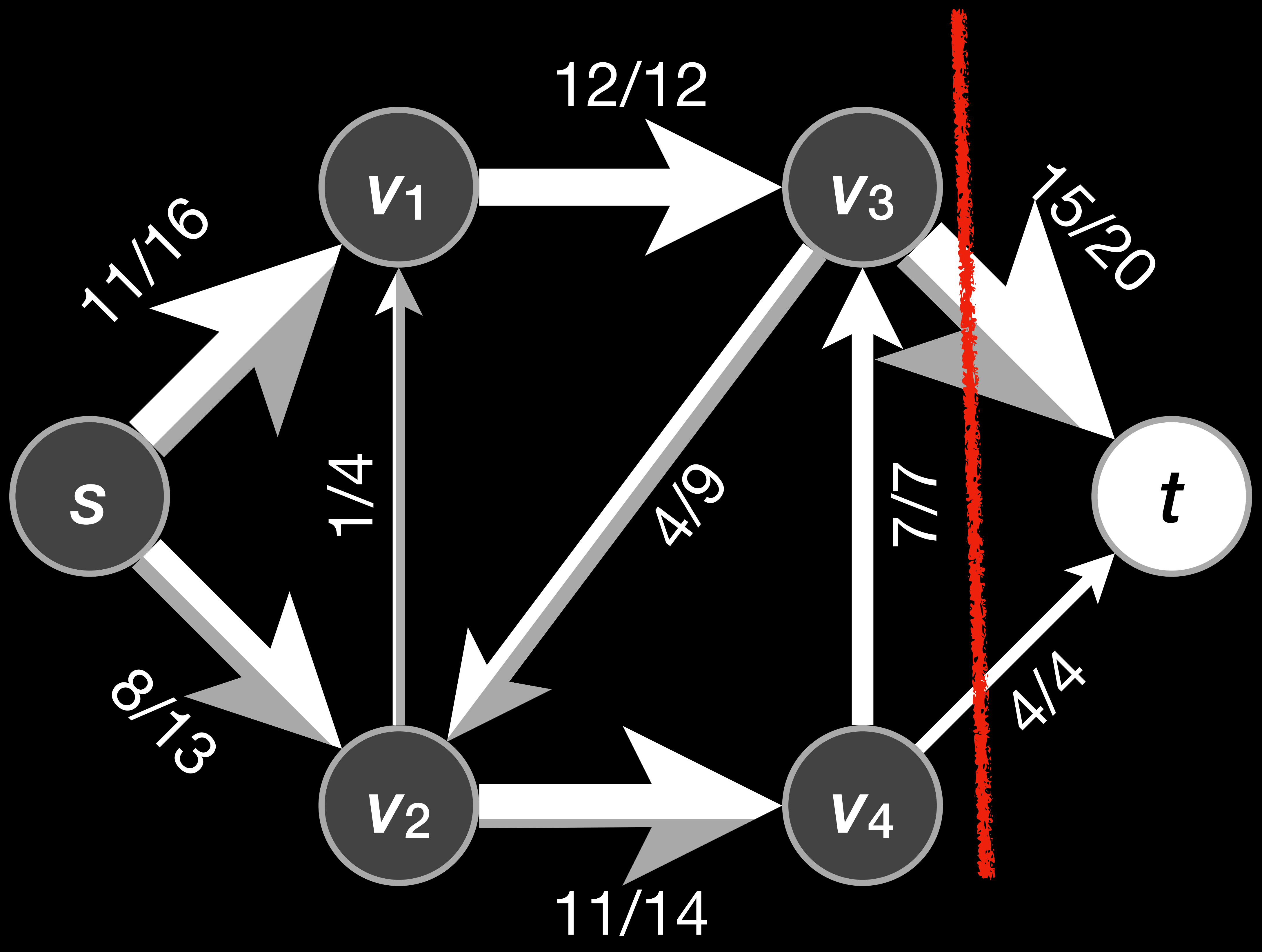
Cut 切割 of a Flow Network

- A cut separates the vertices into S ($\ni s$) and $T = V \setminus S$ ($\ni t$).

- Net flow across the cut:
  $f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - f(v,u)$

- Capacity of the cut:
  $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$

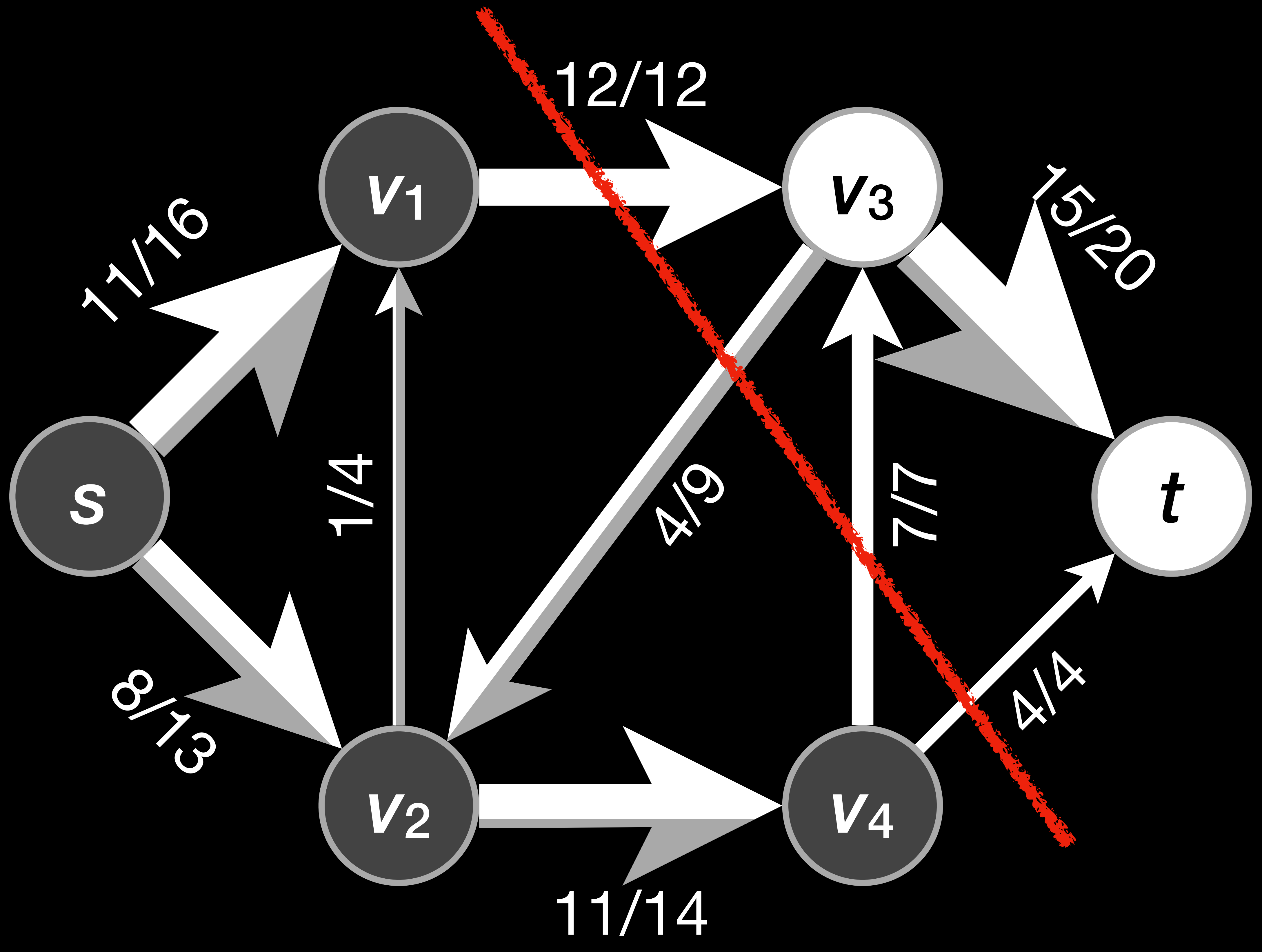
- Difference is intended: capacity = possible maximum net flow (if we neglect other restrictions)



$f(S,T) = 15 + 4 = 19$
$c(S,T) = 20 + 4 = 24$

Cut 切割 of a Flow Network

- A cut separates the vertices into S ($\ni s$) and $T = V \setminus S$ ($\ni t$).
- Net flow across the cut:
  $f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - f(v,u)$
- Capacity of the cut:
  $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$
- Difference is intended: capacity = possible maximum net flow (if we neglect other restrictions)



$f(S,T) = 12 - 4 + 7 + 4 = 19$

$c(S,T) = 12 \quad + 7 + 4 = 23$

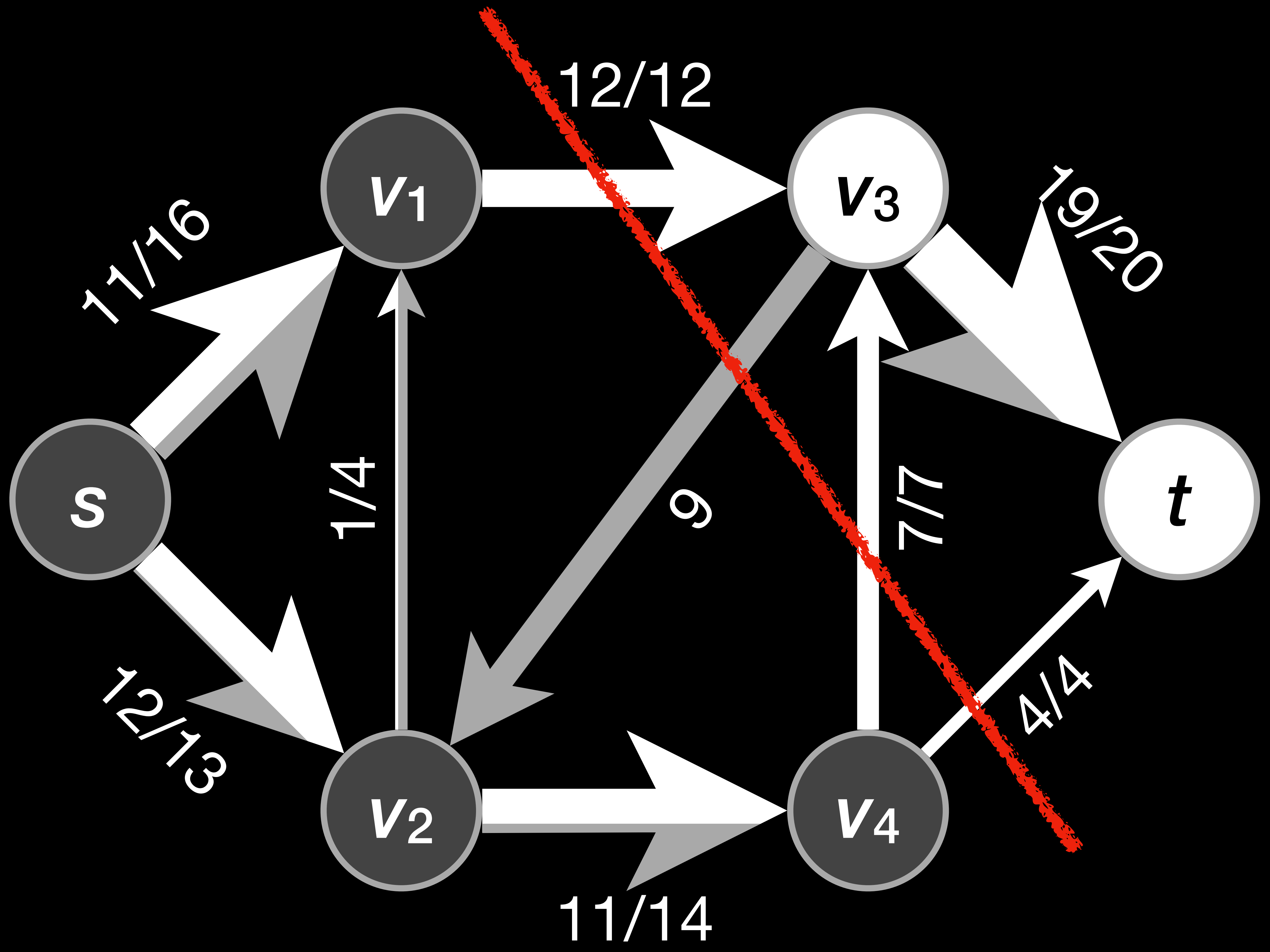
Cut 切割 of a Flow Network

- A cut separates the vertices into S ($\ni s$) and $T = V \setminus S$ ($\ni t$).

- Net flow across the cut:
  $f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - f(v,u)$

- Capacity of the cut:
  $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$

- Difference is intended: capacity = possible maximum net flow (if we neglect other restrictions)



$f(S,T) = 12 + 7 + 4 = 23$
$c(S,T) = 12 + 7 + 4 = 23$

All Net Flows Are Equal

Lemma 26.4: Let (G = (V,E),c) be a flow network and f a flow in it.
Let (S, $V \setminus S$) be a cut of the flow network.
Then $f(S,V \setminus S) = |f|$.

Proof: By induction on the number of vertices in S.

- Base case: $S = \{s\}$. By the definition of $|f|$, we have $|f| = f(\{s\},V \setminus \{s\})$.
- Induction step: Let $S' = S \cup \{x\}$. We already know that $|f| = f(S,V \setminus S)$.

$$f(S',V \setminus S') = \sum_{u \in S'} \sum_{v \in V \setminus S'} f(u,v) - f(v,u)$$

$$= \sum_{u \in S} \sum_{v \in V \setminus S} f(u,v) - f(v,u) + \cancel{\sum_{v \in V \setminus S'} f(x,v) - f(v,x) - \sum_{u \in S} f(u,x) - f(x,u)}$$

$$= f(S,V \setminus S) = |f|.$$

Max-Flow Min-Cut Theorem 最大流最小切割定理

- Based on the Lemma, we can conclude
  that the maximum flow must be ≤ the minimum cut.

- Theorem 26.6: Assume given a flow network ($G = (V,E),c$) and a flow f in it.
  Then, the following are equivalent:

  (1) f is a maximum flow in G.

  (2) The residual network G_f contains no augmenting paths.

  (3) There exists a cut (S,T) in G such that $|f| = c(S,T)$.

- So, by this theorem, we have that the maximum flow = the minimum cut.

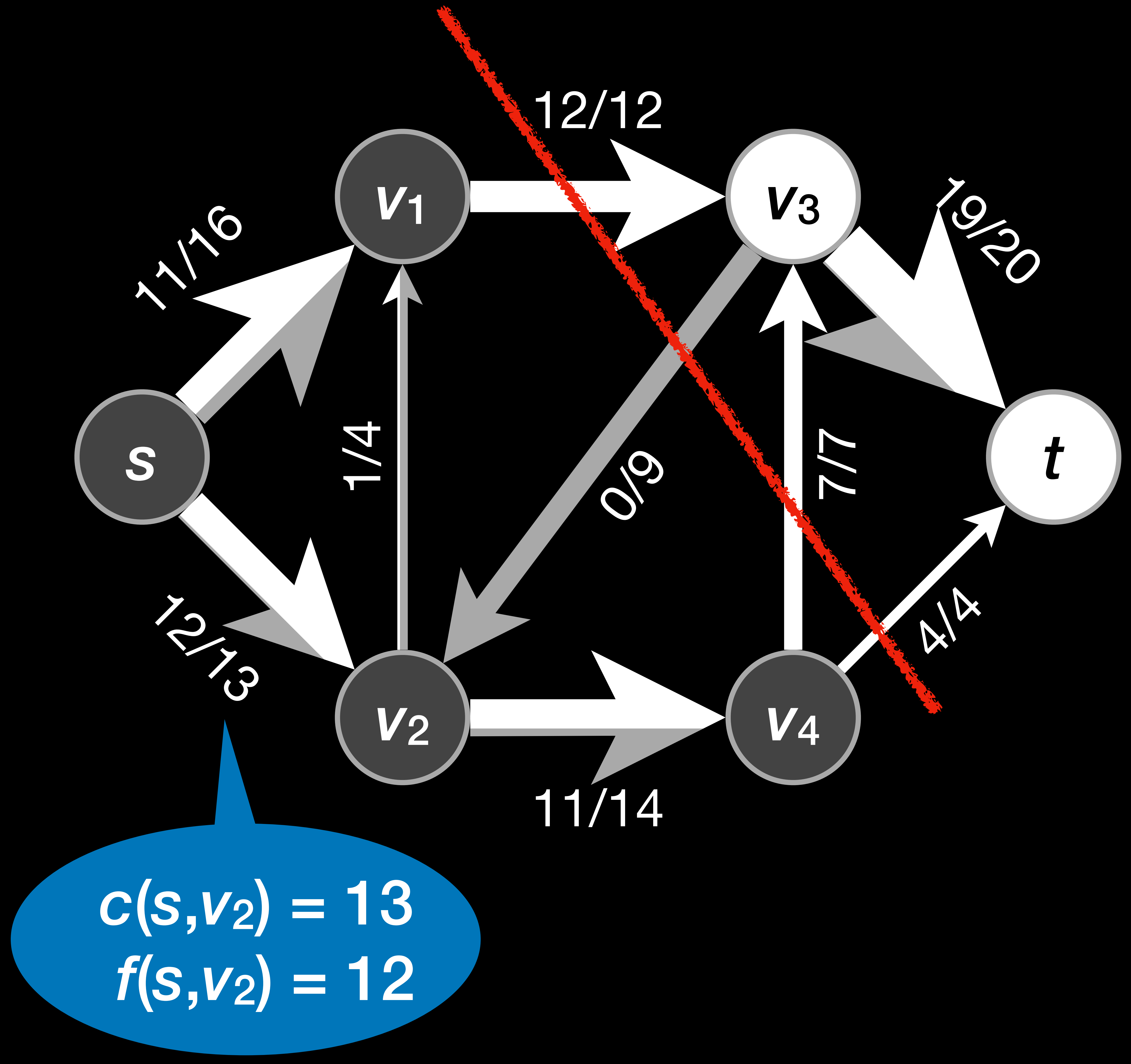
Proof of Max-Flow Min-Cut

(1) $\implies$ (2). If f is a maximum flow but G_f contains an augmenting path p, then $f \uparrow p$ would be an even larger flow. Contradiction!

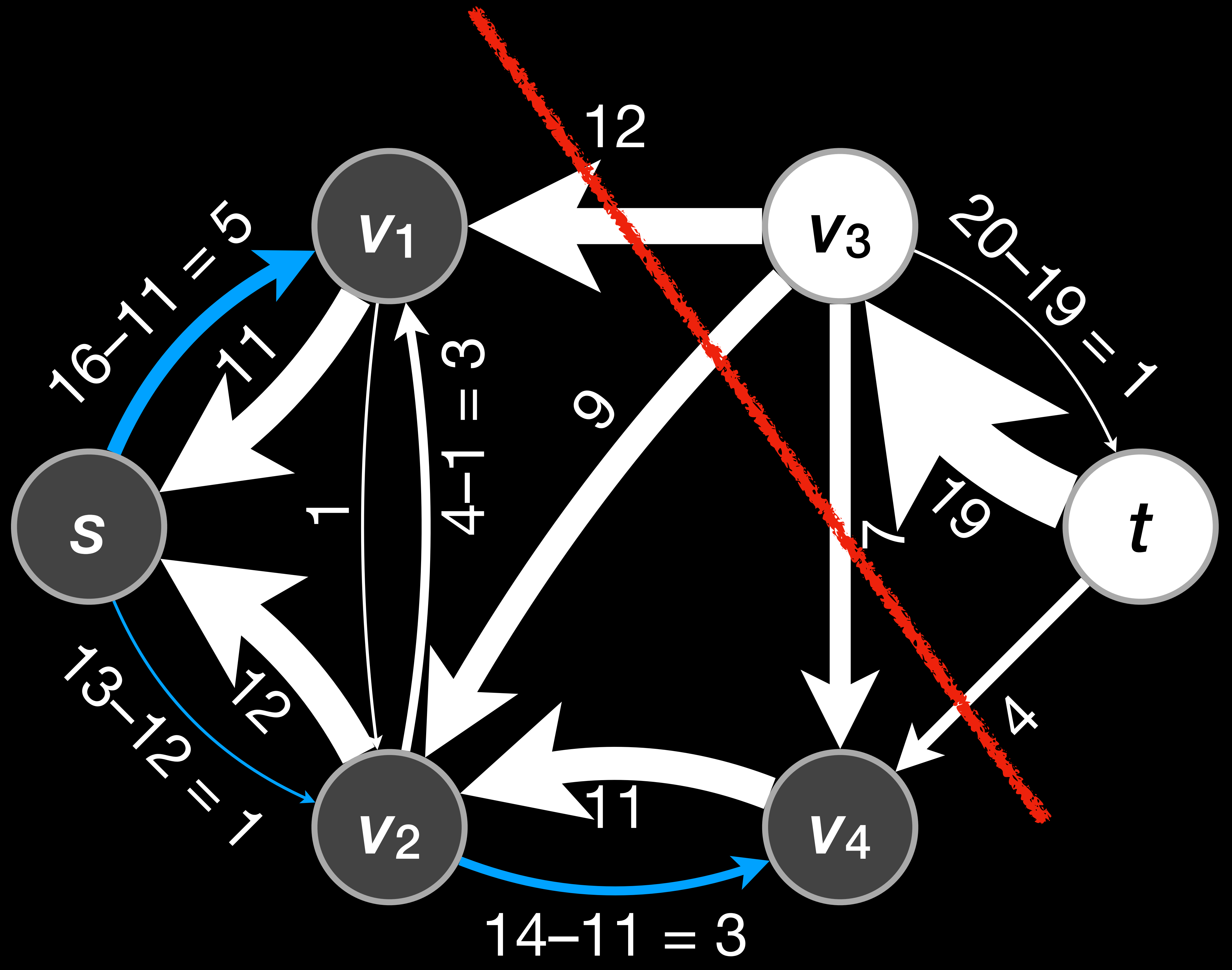
(2) $\implies$ (3). Let $S = \{v \in V \mid$ there exists a path $s \rightsquigarrow v$ in $G_f\}$.
This cut satisfies the conditions of (3) (see next slide).

(3) $\implies$ (1). As the value of flow f cannot be larger than any cut,
being equal is the highest value that can be achieved, so f is maximal.

Example (2) ⟹ (3)



s
v1
v2
v3
v4
t
11/16
12/13
1/4
12/12
0/9
7/7
19/20
11/14
4/4
c(s,v2) = 13
f(s,v2) = 12



s
v1
v2
v3
v4
t
16–11 = 5
11
1
4–1 = 3
12
9
20–19 = 1
19
7
12
13–12 = 1
11
4
14–11 = 3

Termination and Running Time

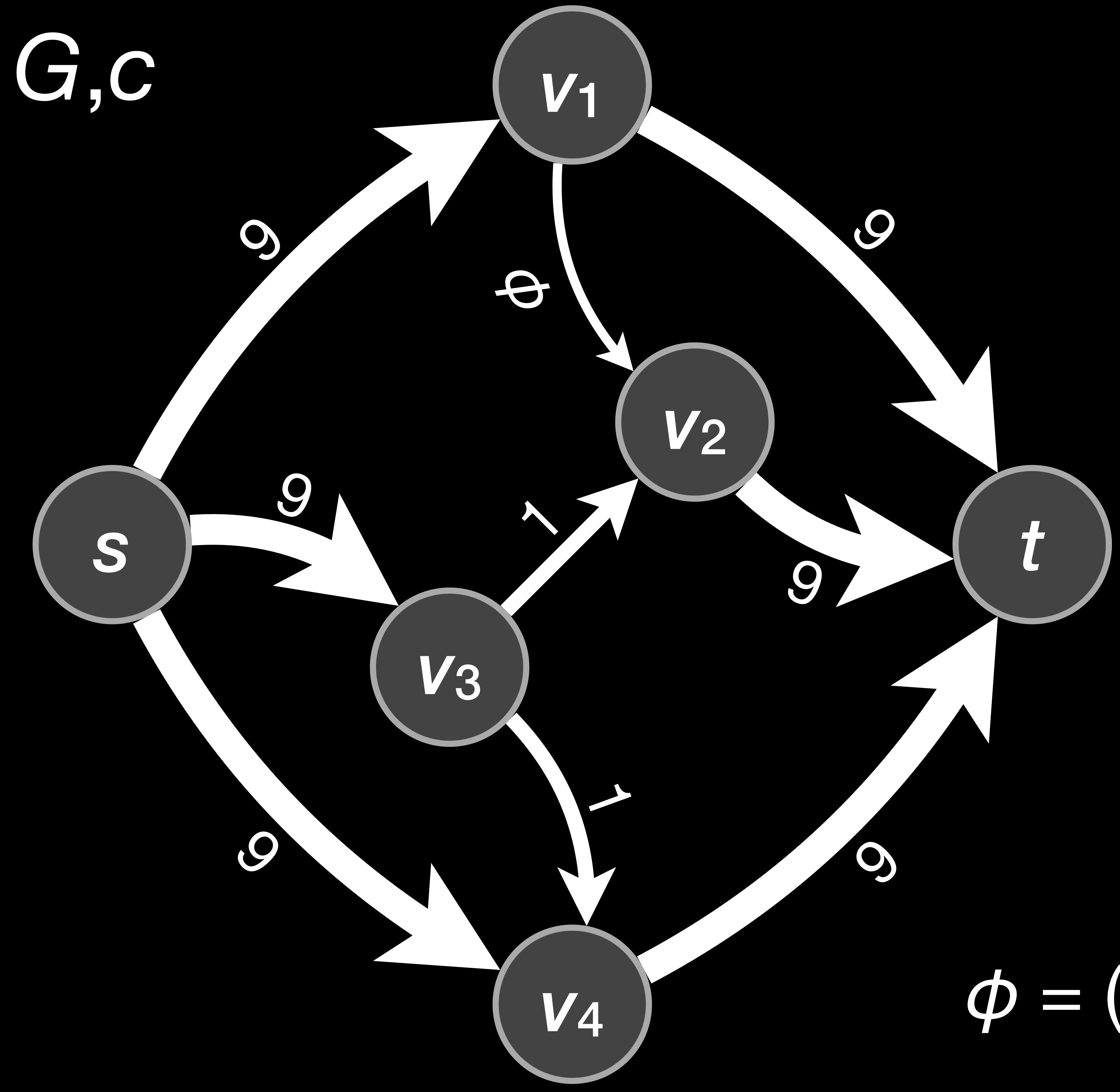
- If the edge capacities are incommensurable (irrational), Ford–Fulkerson may not terminate.
  Zwick, Uri: **The smallest networks on which the Ford–Fukerson maximum flow procedure may fail to terminate.**
  Theoretical Computer Science 148(1995)165–170.
  (constructs a network where augmenting paths have value 1, ϕ, ϕ^2, ϕ^3, ..., for $\phi = (\sqrt{5}-1)/2 \approx 0.618$.)

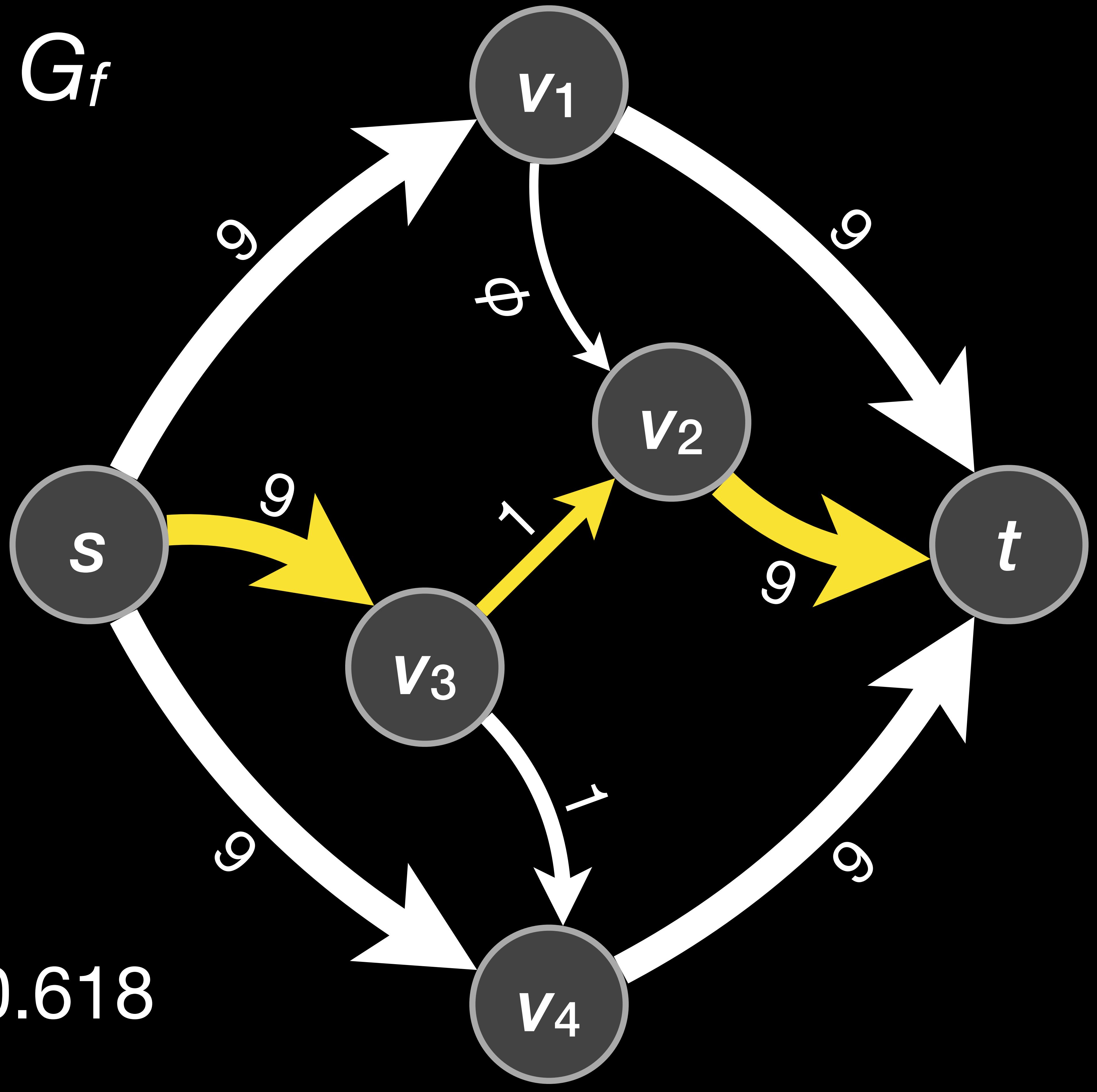
- If the edge capacities are integers, Ford–Fulkerson will terminate with an integer maximum flow f after at most $|f|$ iterations.

Integer Capacities

- Assume that all capacities in the flow network are integers $c\colon E \to \mathbb{N} \cup \{\infty\}$.
- Then, every intermediary flow and augmenting path will have integer value. The value of every augmenting path is at least 1.
- Therefore, if f^* is a maximum flow, it is constructed using at most $|f^*|$ augmenting paths.

Non-terminating Example

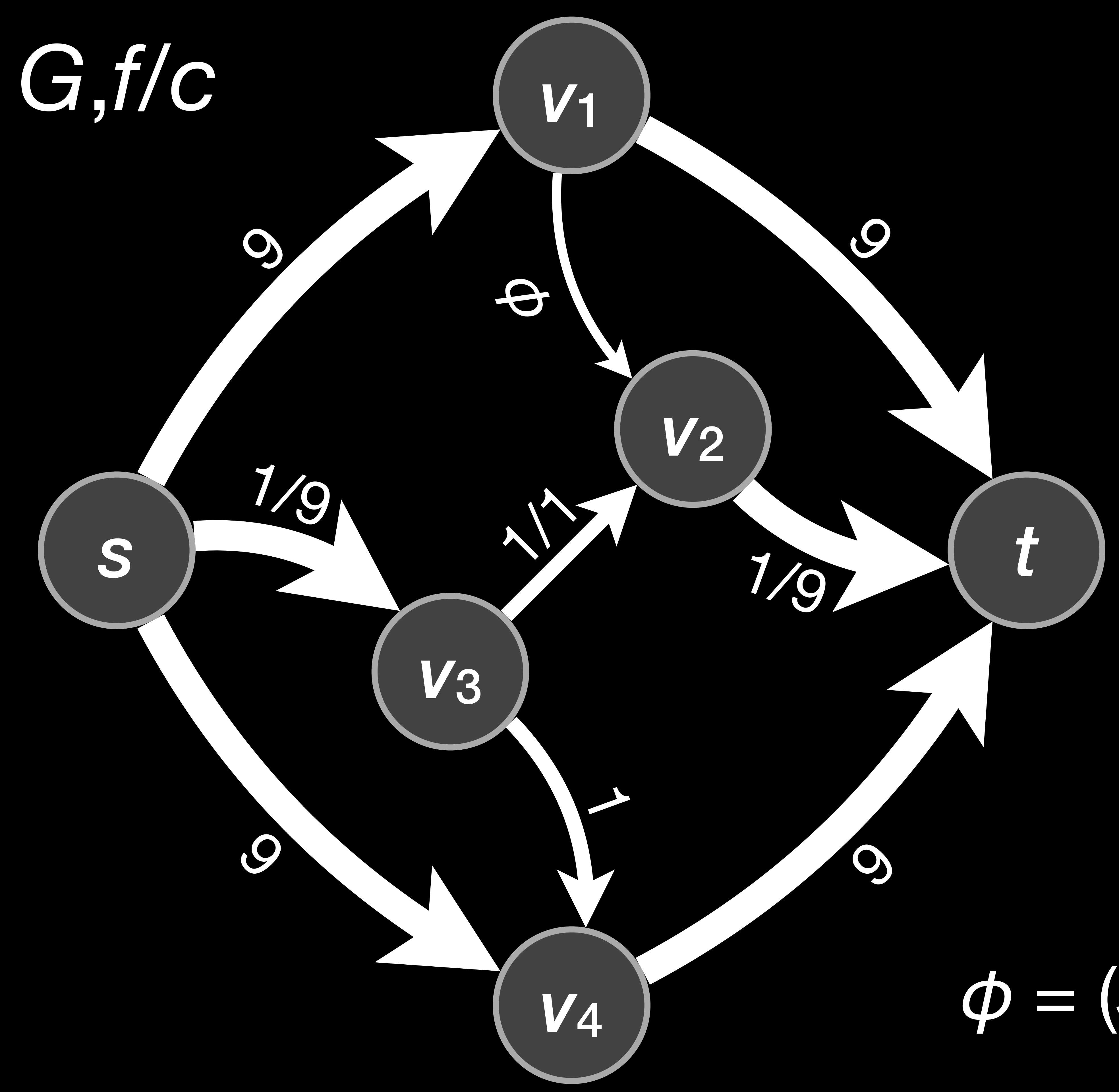


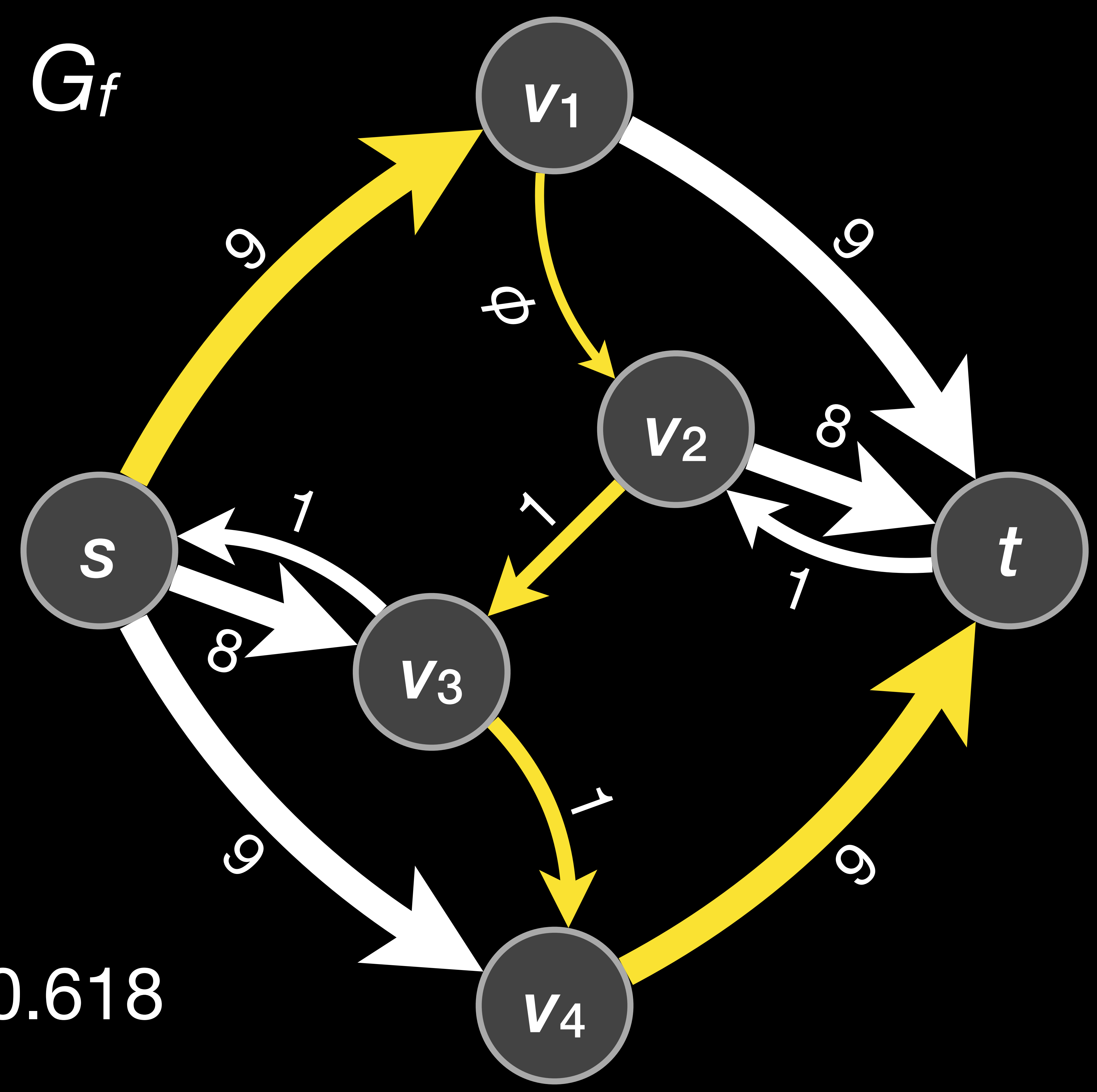


$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Non-terminating Example

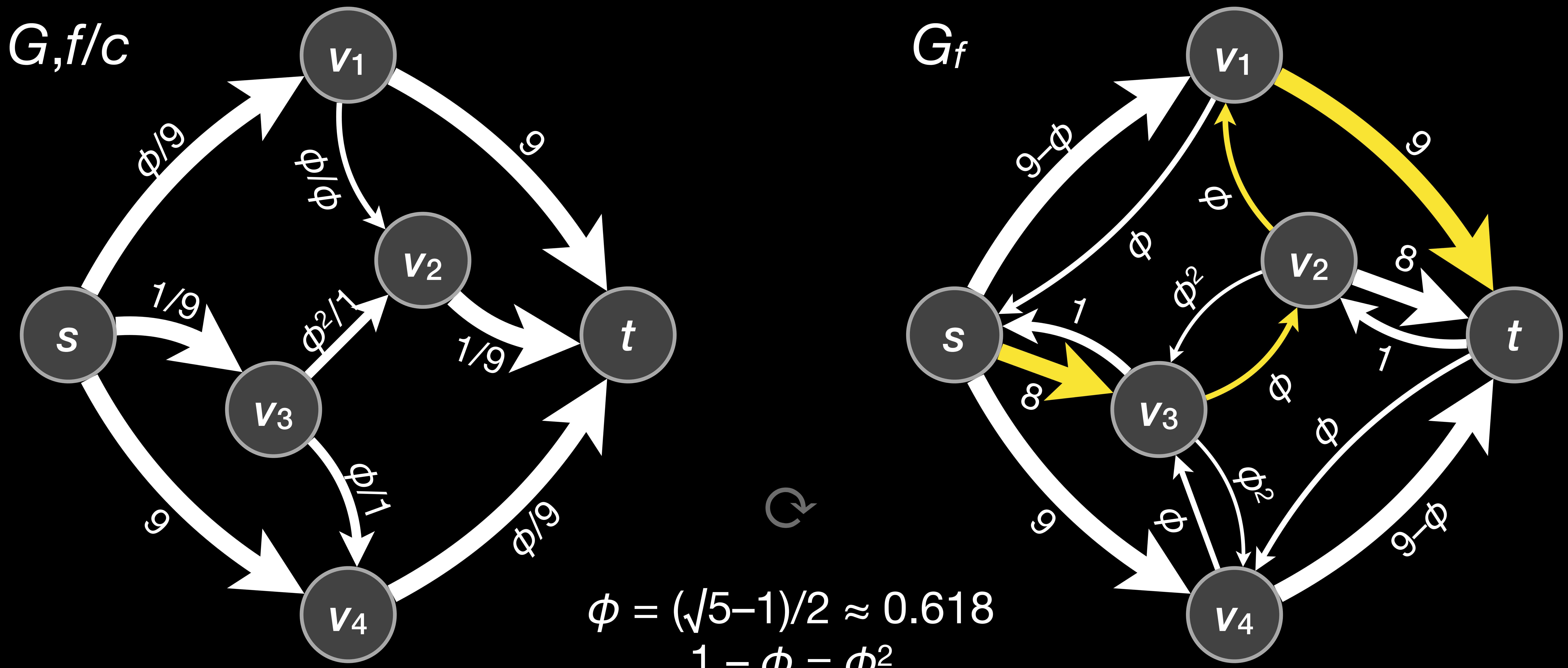




$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

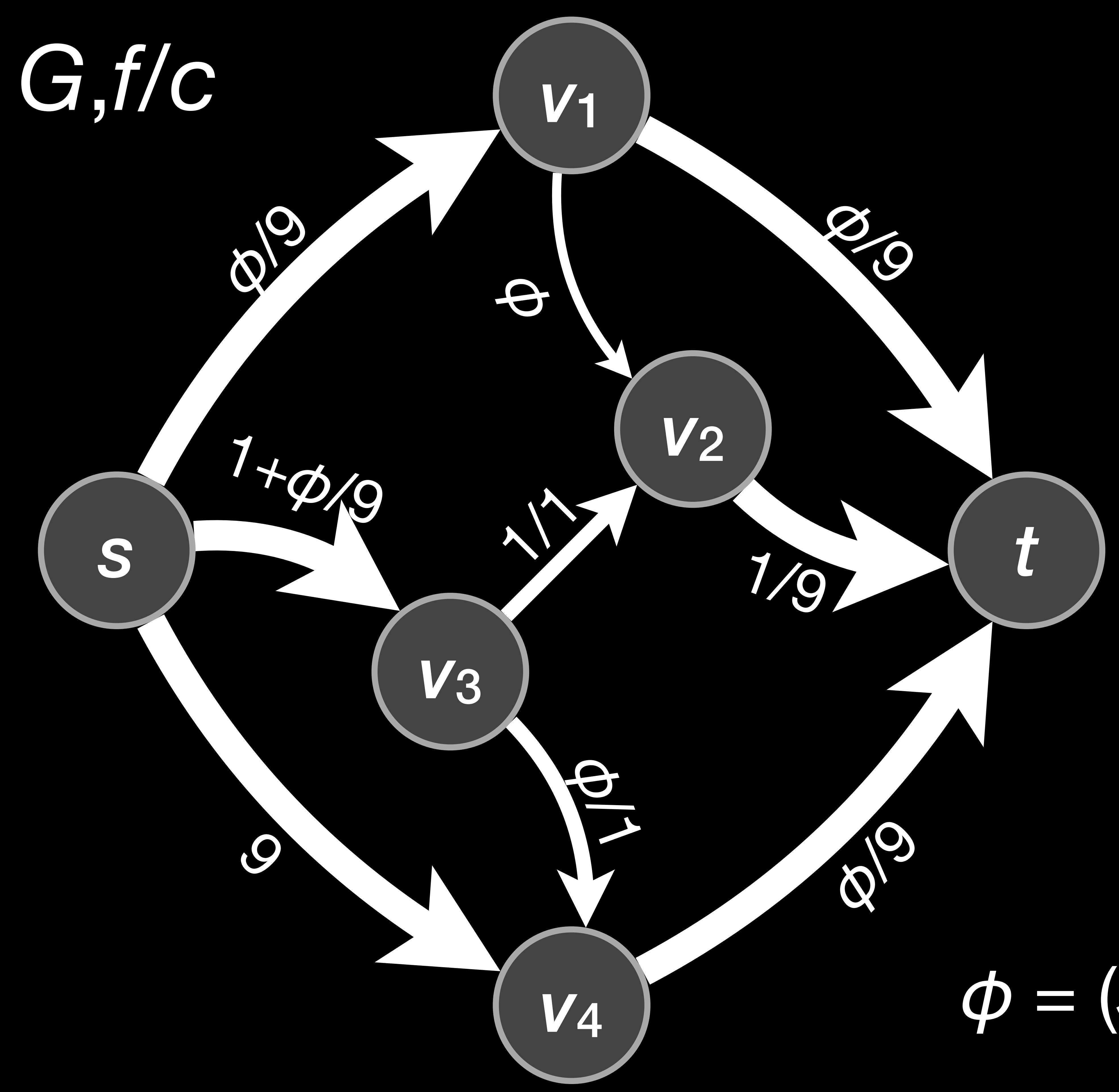
Non-terminating Example

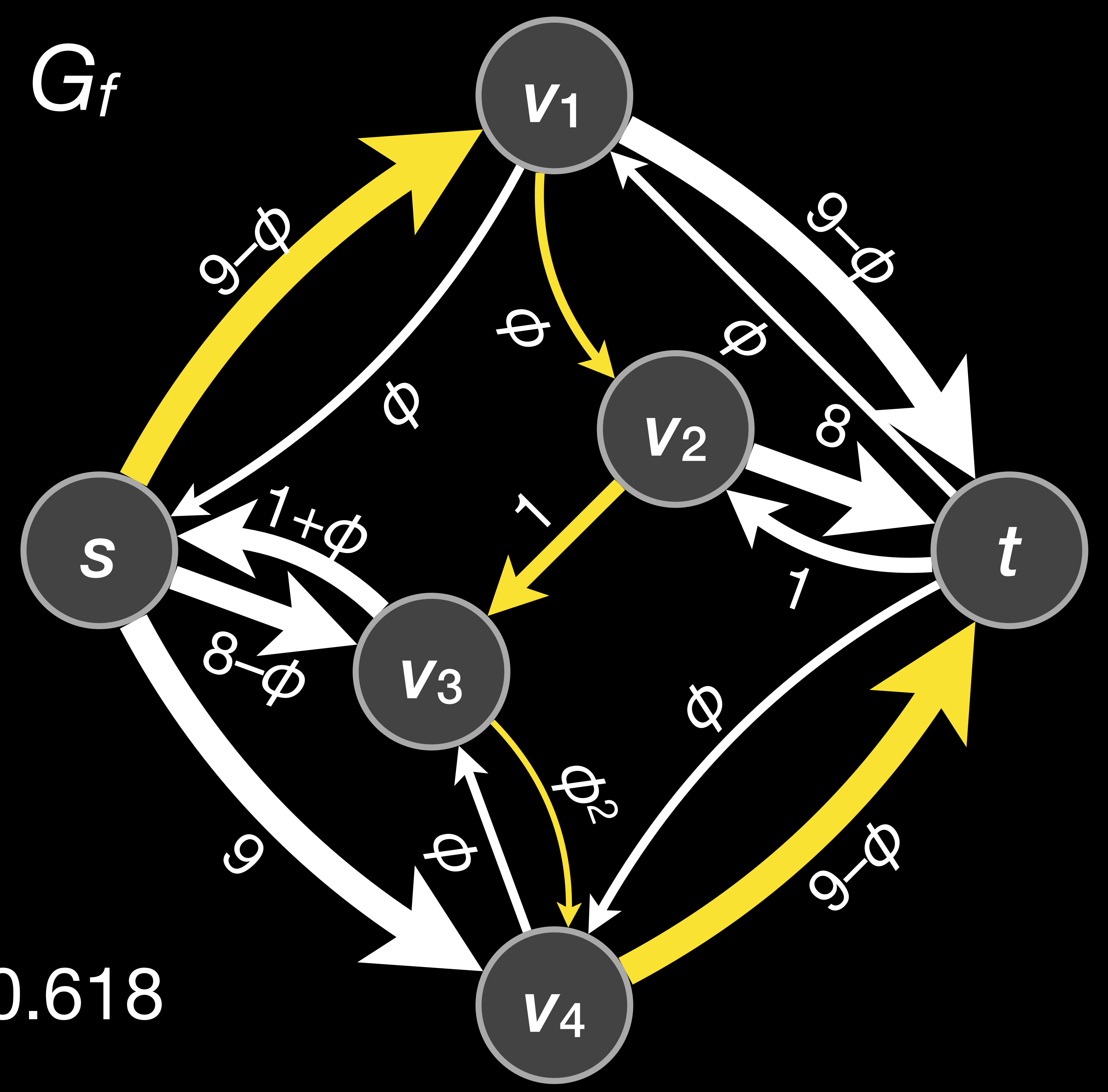


$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Non-terminating Example

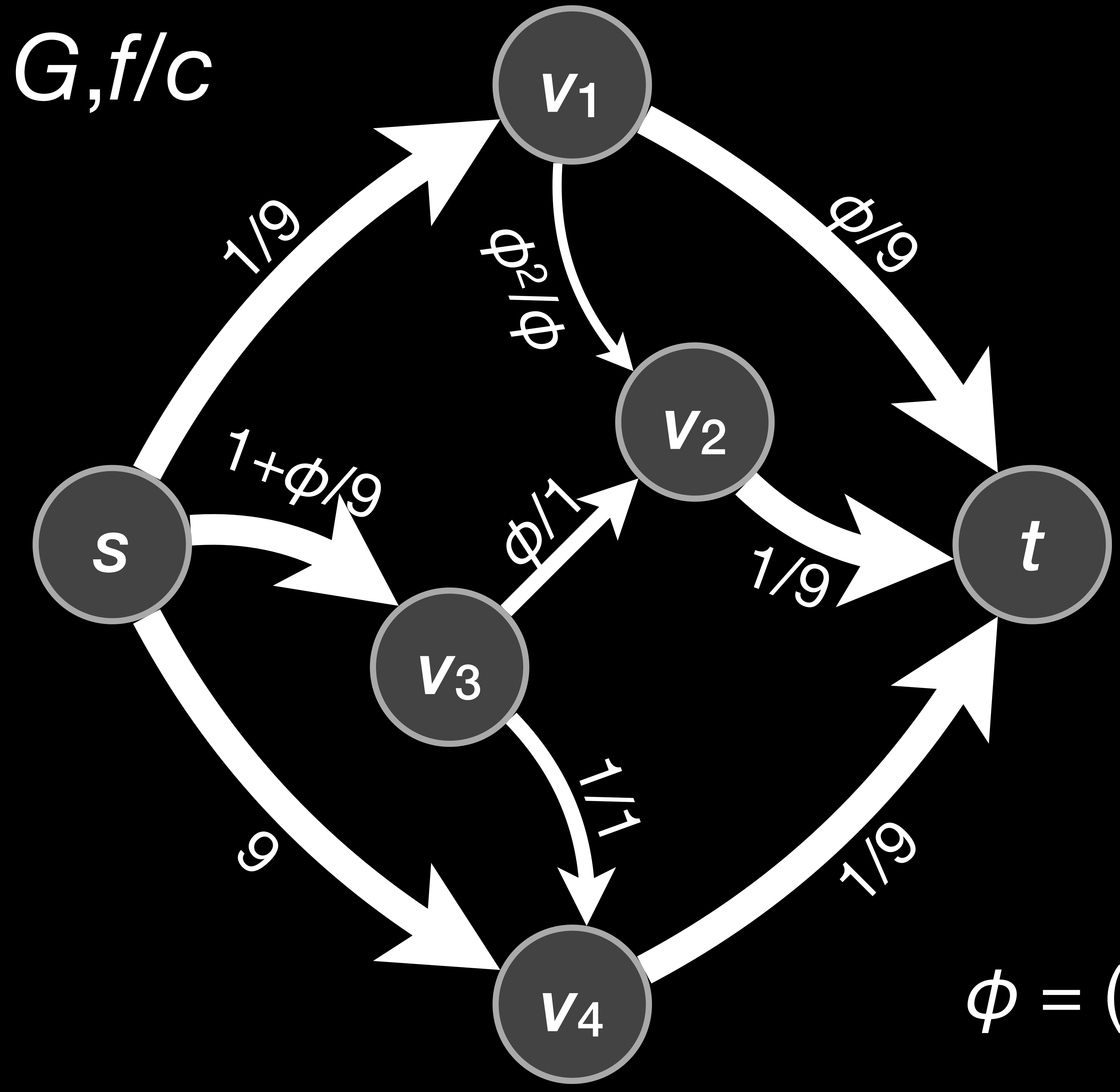


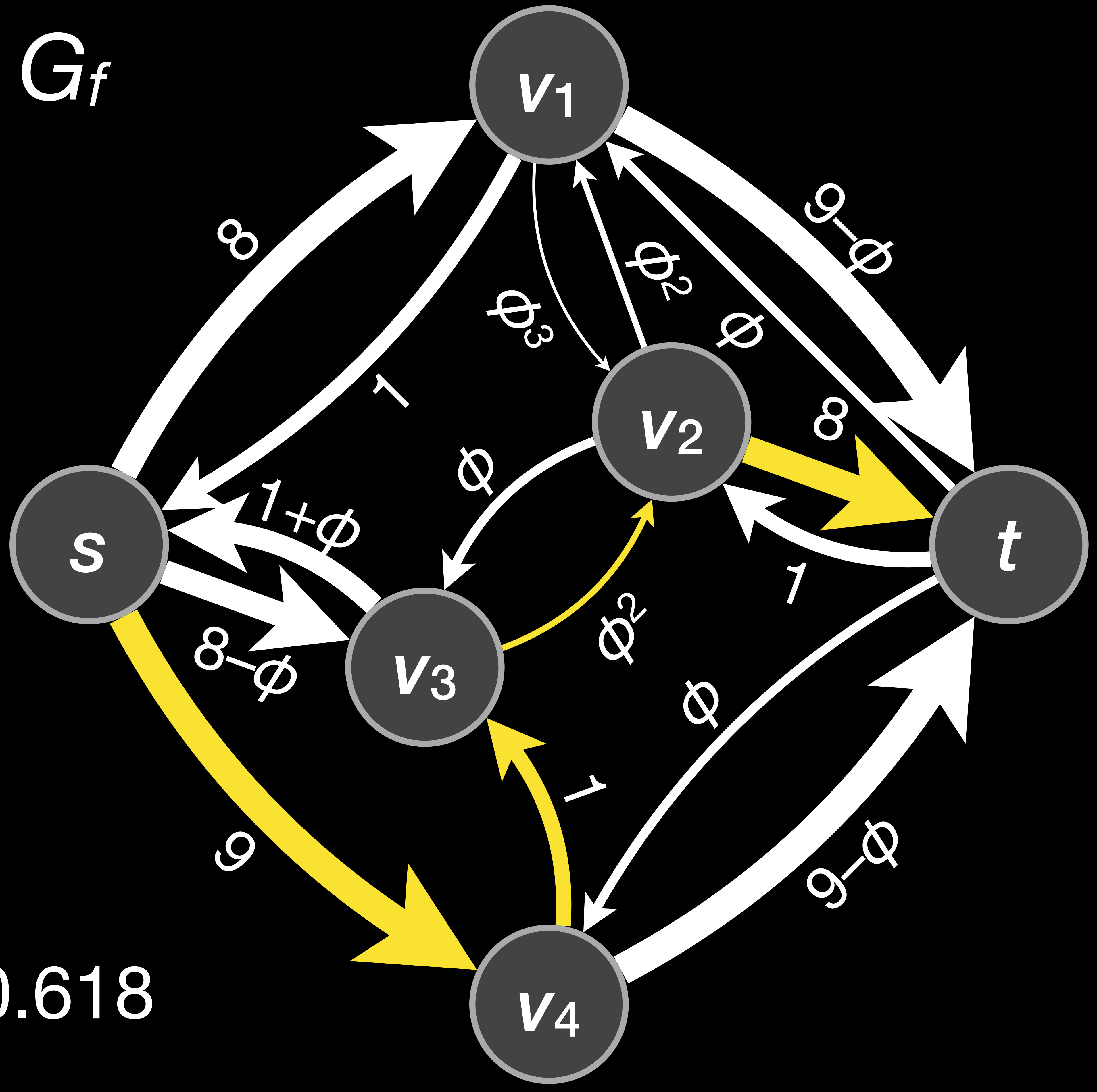


$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Non-terminating Example

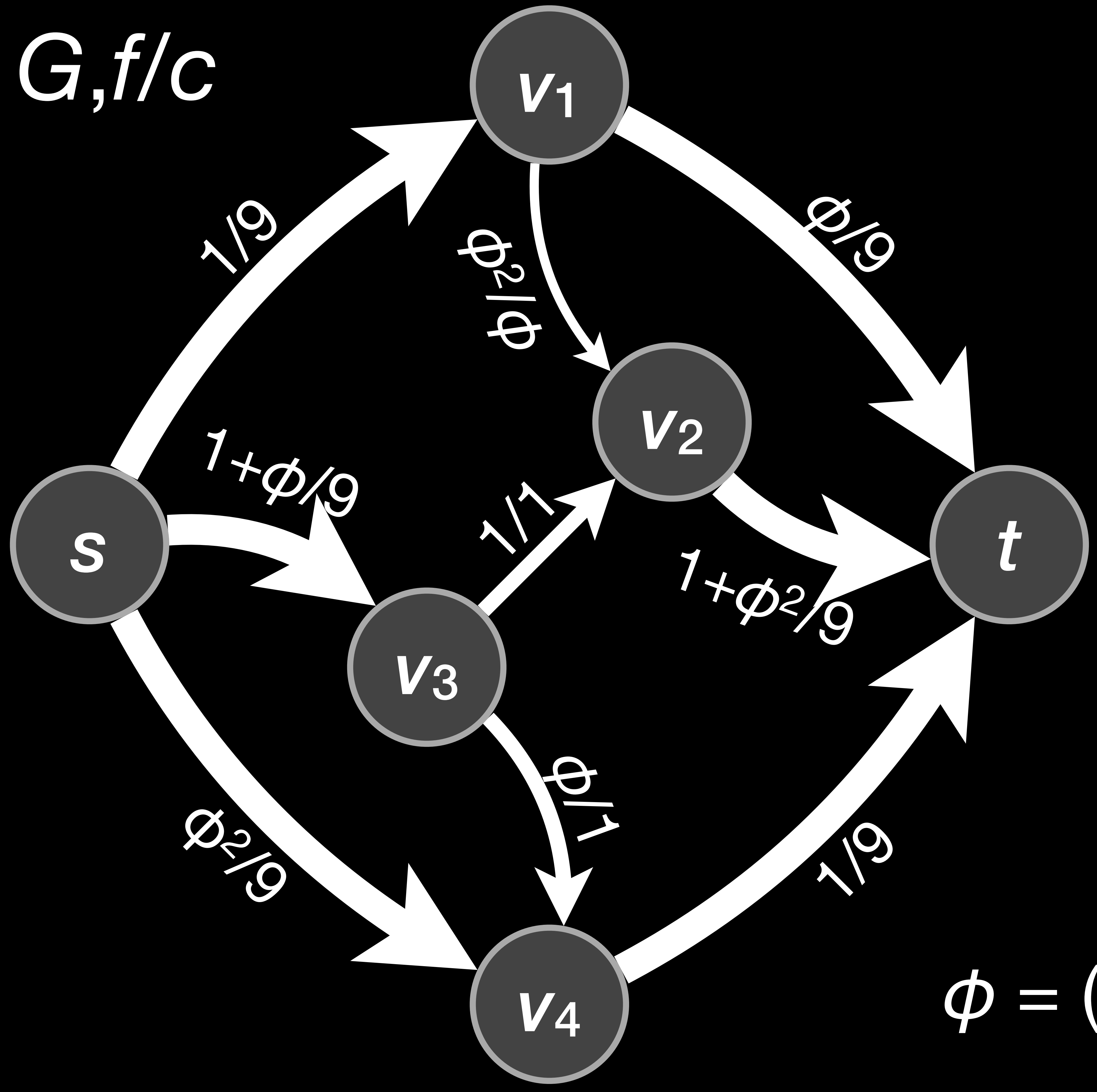


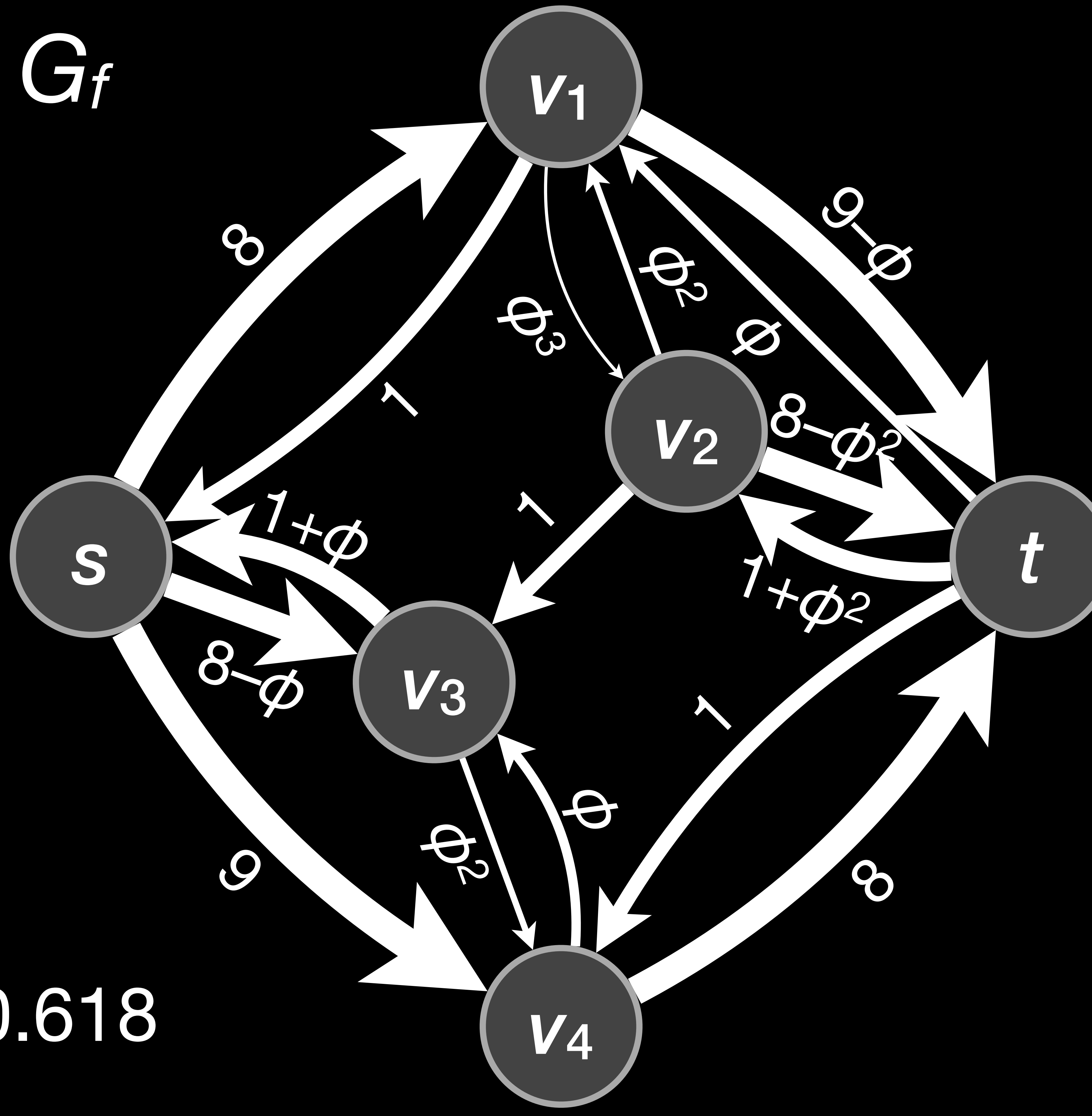


$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Non-terminating Example

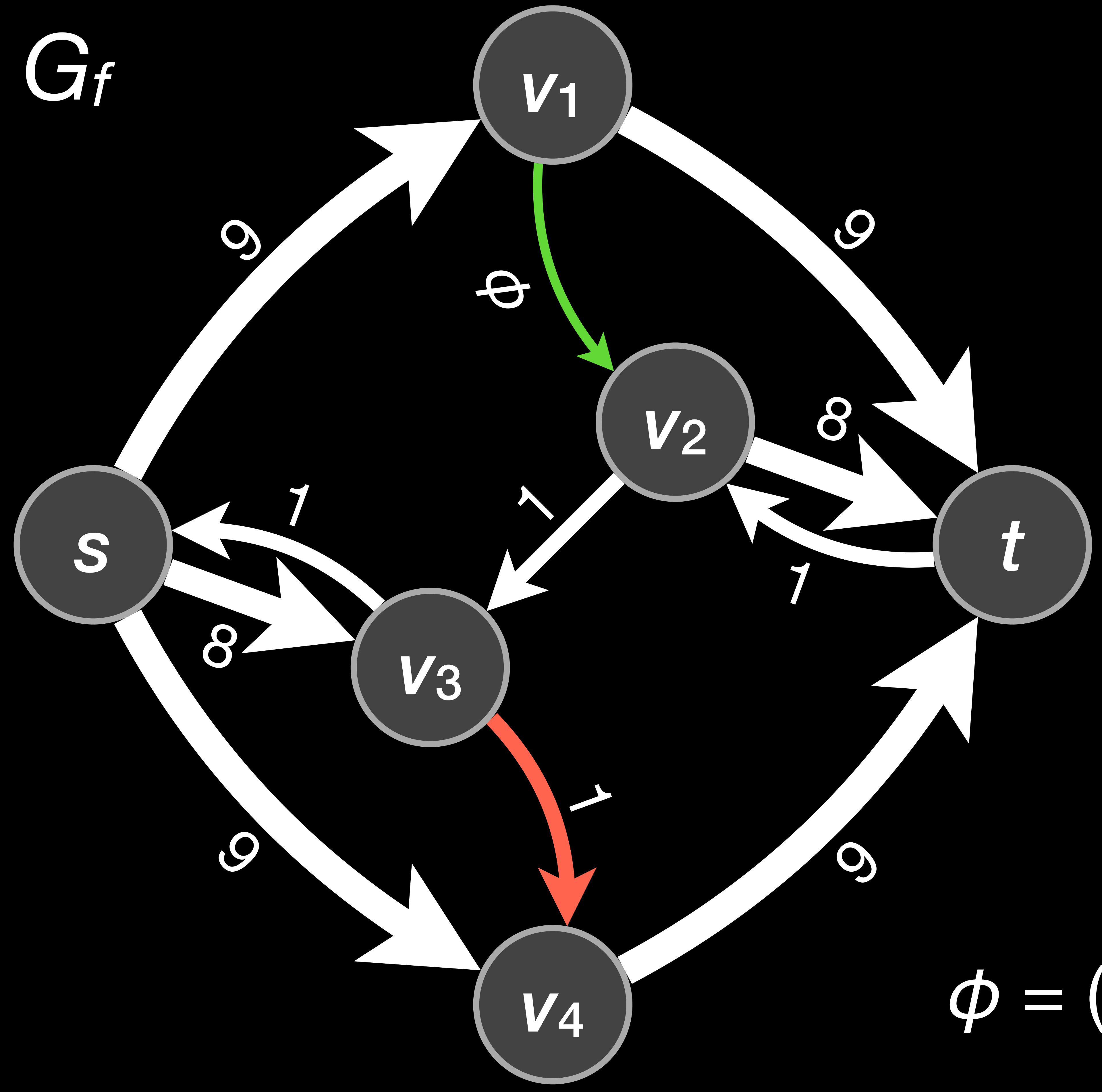


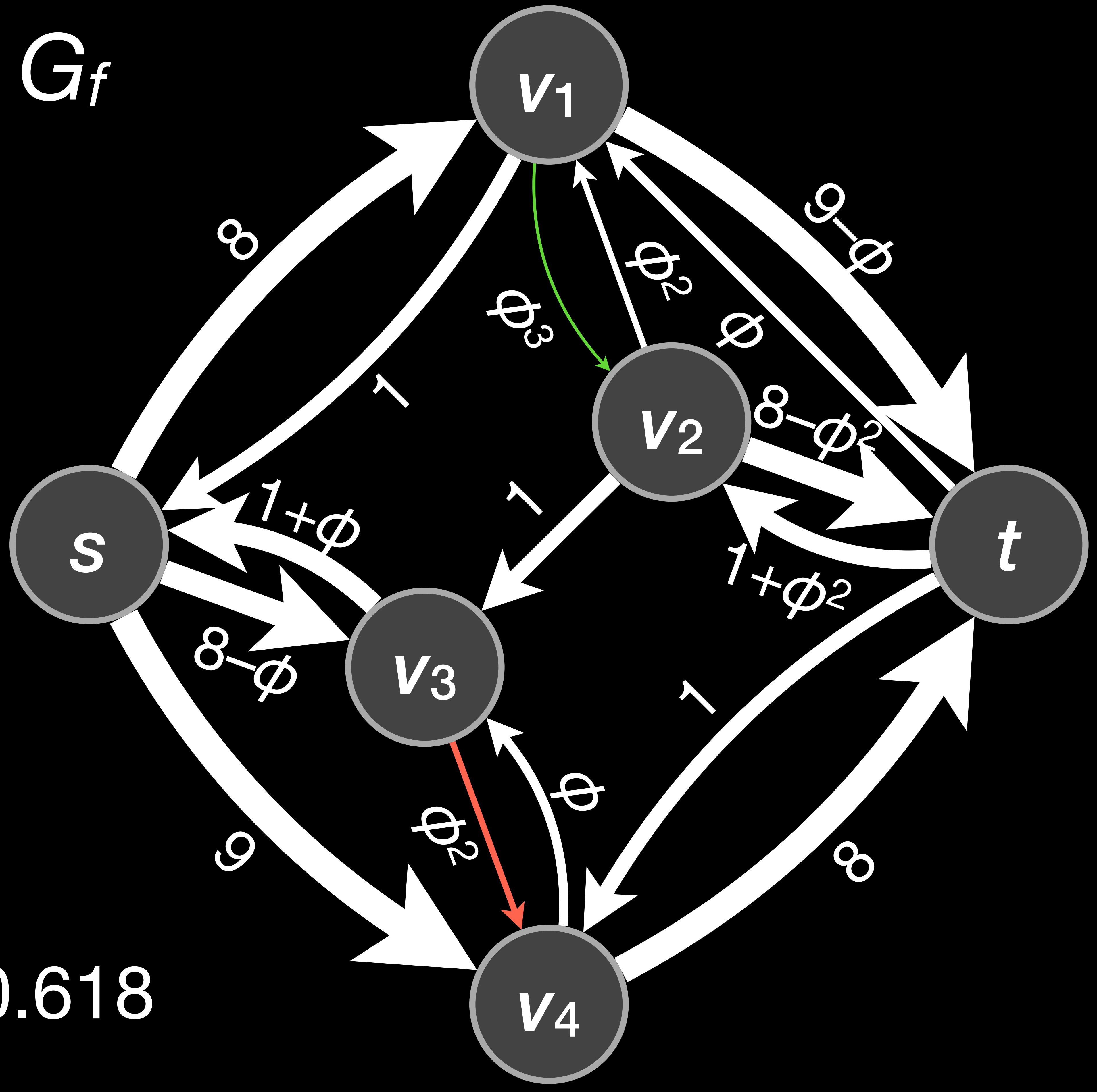


$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Non-terminating Example

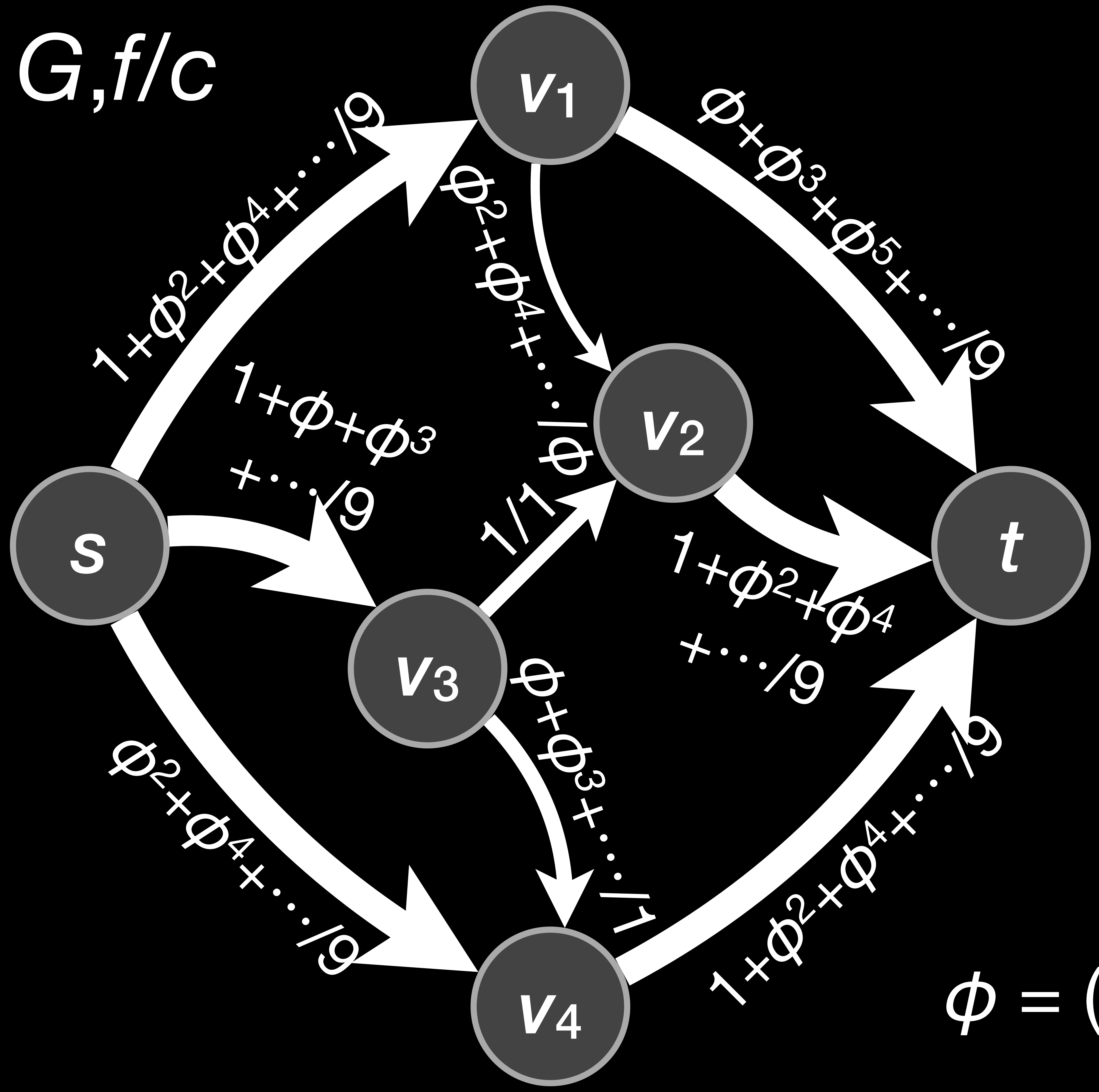




$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Non-terminating Example



- The same sequence of augmenting paths can be applied again, but every augment will be by factor ϕ^2 smaller.
- In the limit, the left flow is reached.
- Value $= 3 + 2\phi^2 + 2\phi^4 + \cdots$
  $= 3 + 2\phi \approx 4.236$

$\phi = (\sqrt{5}-1)/2 \approx 0.618$

$1 - \phi = \phi^2$

Running Time (integer capacities)

运行时间（整数的容量）

- Assume that all capacities are integers. Then, we already know there will be ≤ $|f^*|$ augmenting paths, i.e. ≤ $|f^*|$ iterations.
- How long does an iteration take?
  - Finding a path is in $O(|E|)$ (also finds the capacity of the path)
  - Adapting the capacities of the residual network is in $O(|E|)$
- So, overall the running time is in $O(|E|\cdot|f^*|)$.

- 假设所有的容量是整数。
已经知道最多有 $|f^*|$ 增广路径，
算法最多进行 $|f^*|$ 迭代。
- 一个迭代需要多少时间?
  - 找到一条路径为 $O(|E|)$（包括计算路径的容量）
  - 改变残存网络的容量在 $O(|E|)$ 中
- 总的来说，运行时间是 $O(|E|\cdot|f^*|)$。

Edmonds–Karp

- Edmonds and Karp proposed to use only breadth-first search for finding augmenting paths. They proved a better time bound.
- Proof idea: Given any vertex v, the distance $\delta(s,v)$ in G_f never decreases. If edge (u,v) is critical two times, then $\delta(s,v)$ increases by at least 2.

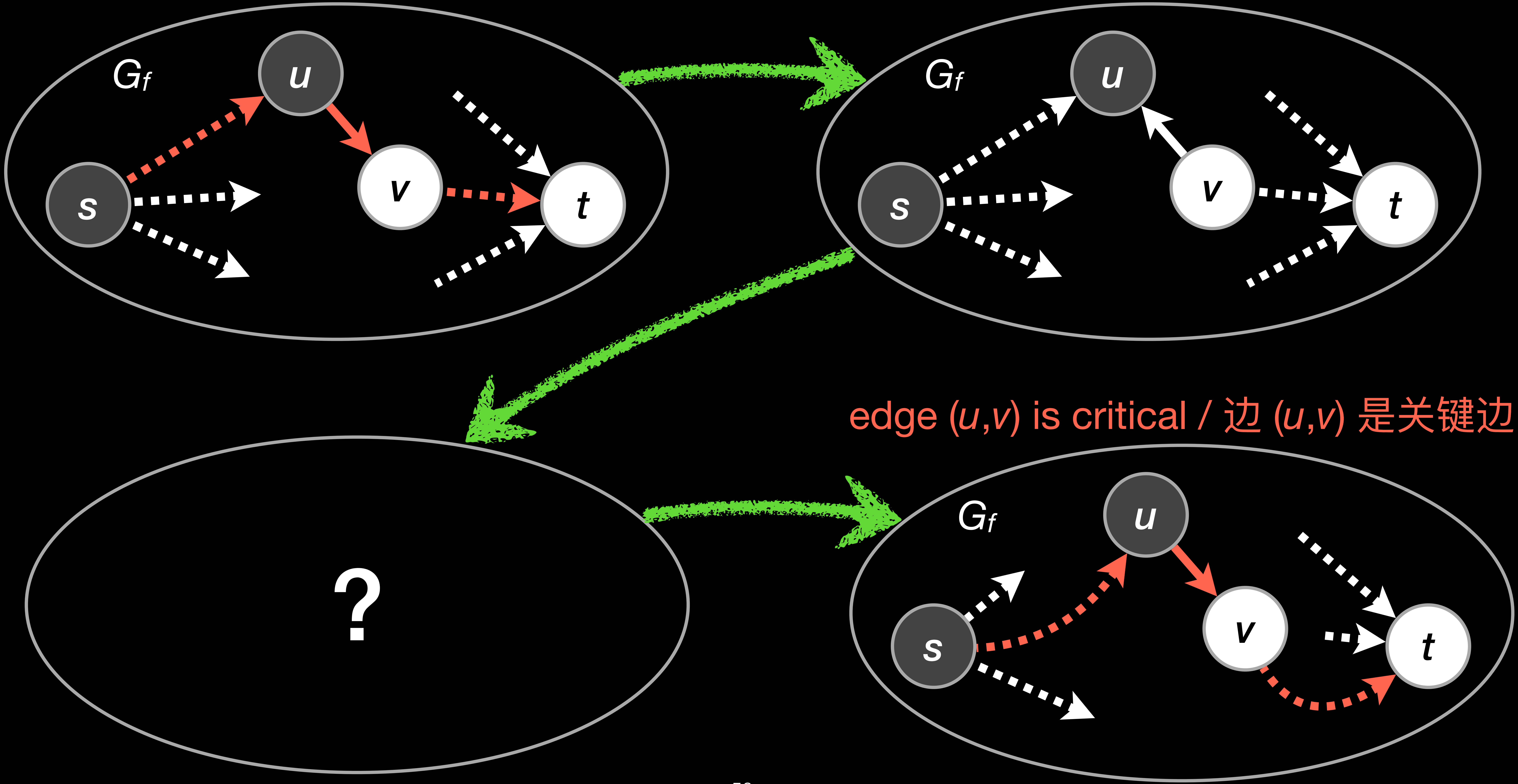
critical: augmenting path value = $c_f(u,v)$

Edmonds–Karp

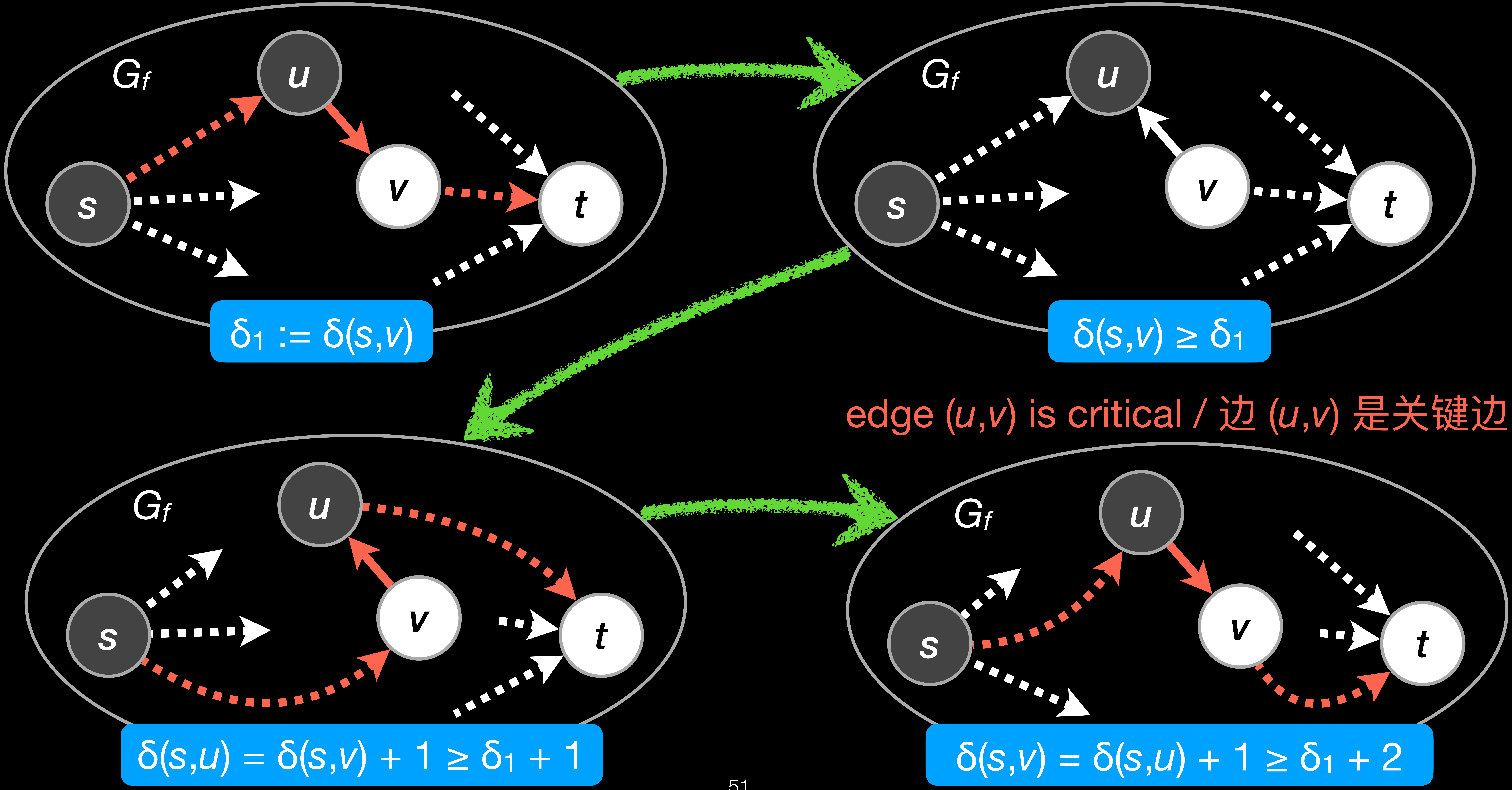
- Edmonds 和 Karp 建议总是使用广度优先搜索再找到增广路径。
这样可以证明更好的运行时间的上界。
- 证明思想：给定任何结点 v，
G_f 中的距离 $\delta(s,v)$ 都不会减小。
如果边 (u,v) 是关键边两次，
则 $\delta(s,v)$ 至少增加 2。

关键边：增广路径的值 = $c_f(u,v)$

edge (u,v) is critical / 边 (u,v) 是关键边



edge (u,v) is critical / 边 (u,v) 是关键边



Edmonds–Karp

- Lemma 26.7: For all vertices $v \in V \setminus \{s,t\}$, the distance $\delta_f(s,v)$ in the residual network G_f increases monotonically over time.

- Theorem 26.8: The total number of flow augmentations performed is in $O(|V| \cdot |E|)$.

- Corollary: The total running time of Edmonds–Karp is in $O(|V| \cdot |E|^2)$.

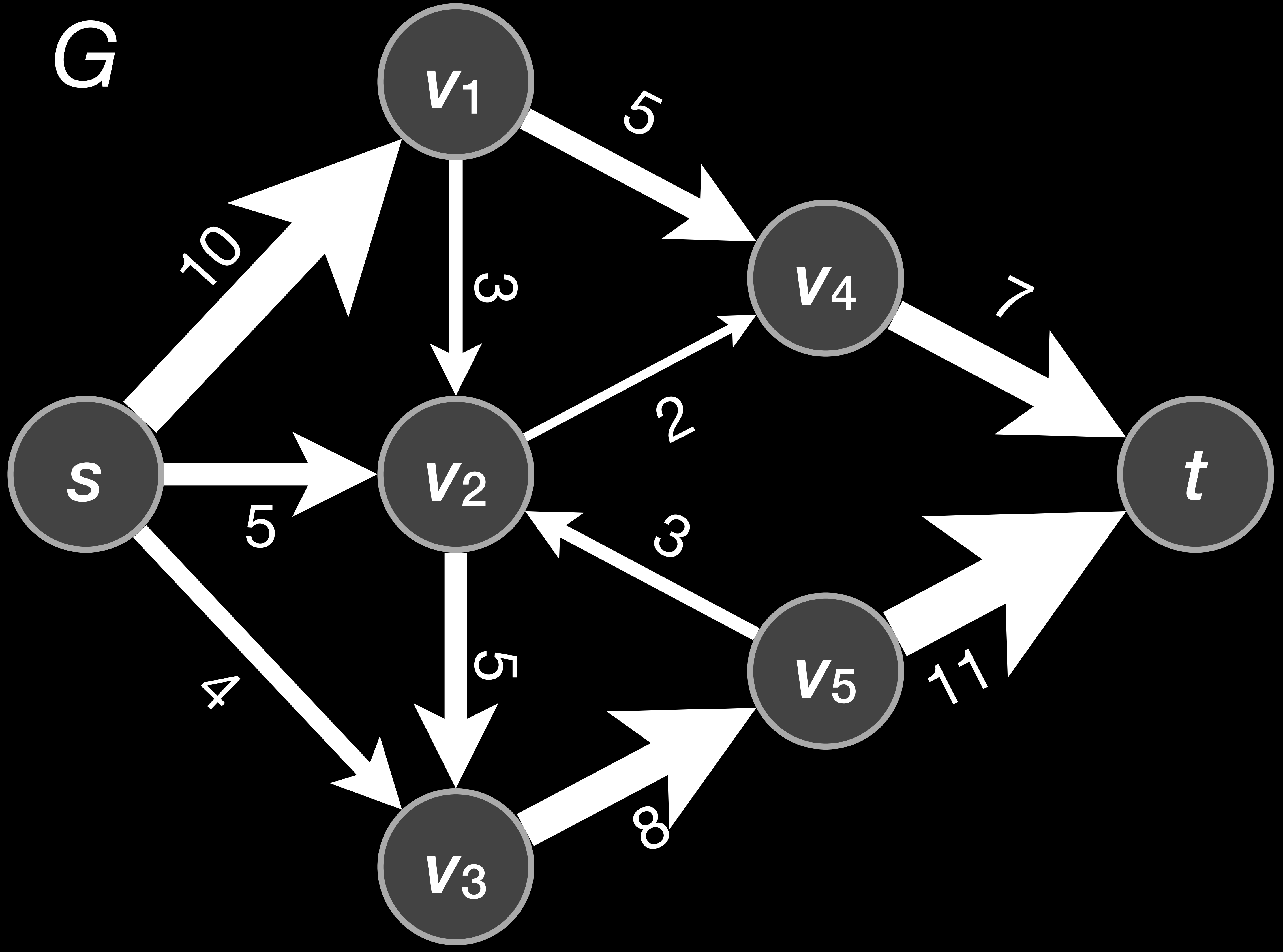
Edmonds–Karp

- 引理26.7：对于所有的结点 $v \in V \setminus \{s,t\}$，残存网络 G_f 中的做短路径距离 $\delta_f(s,v)$ 随着单调递增。

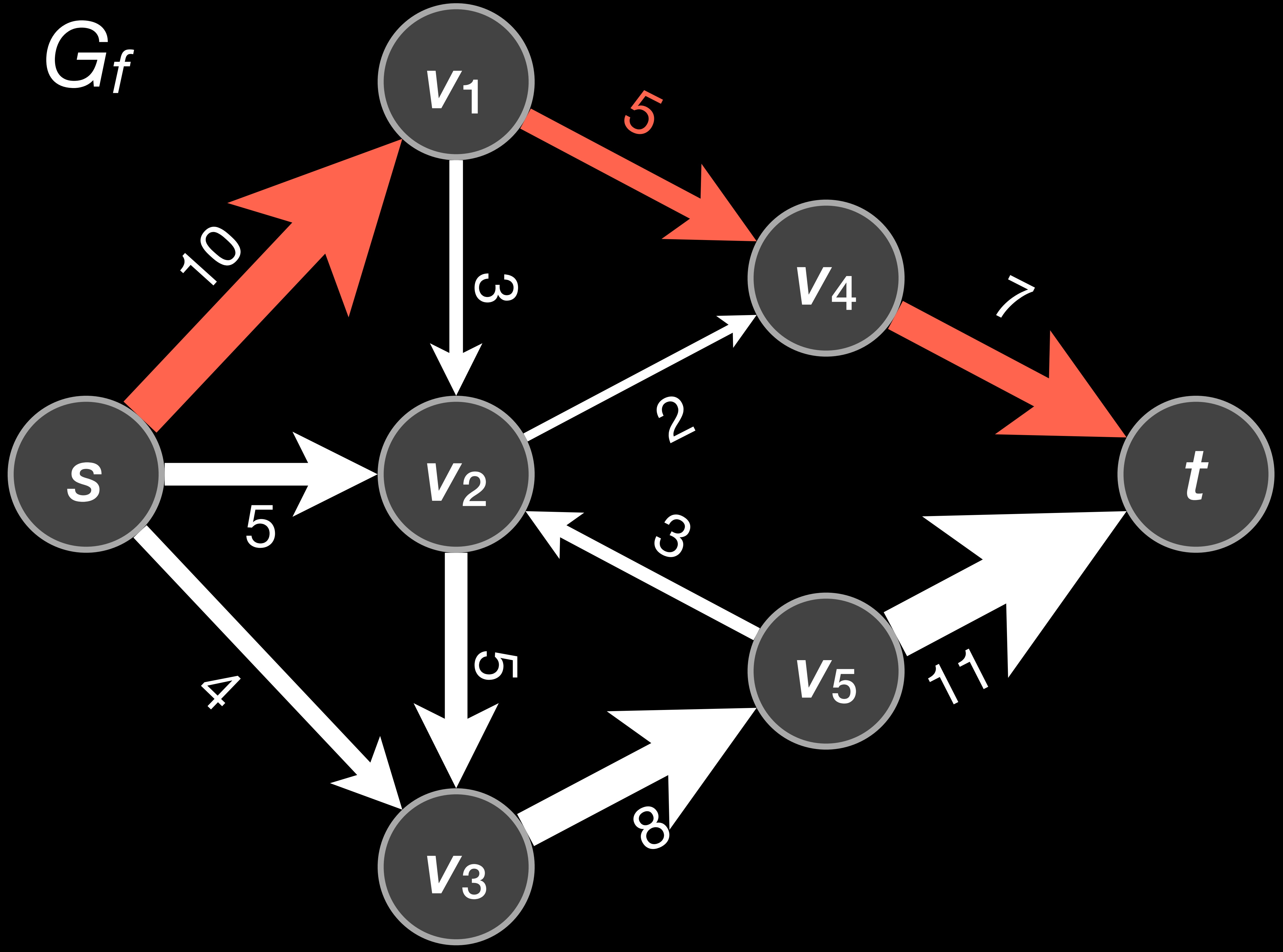
- 定理26.8：所执行的流量递增操作的总次数为 $O(|V| \cdot |E|)$。

- 推论：Edmonds–Karp的总运行时间为$O(|V| \cdot |E|^2)$。

Example

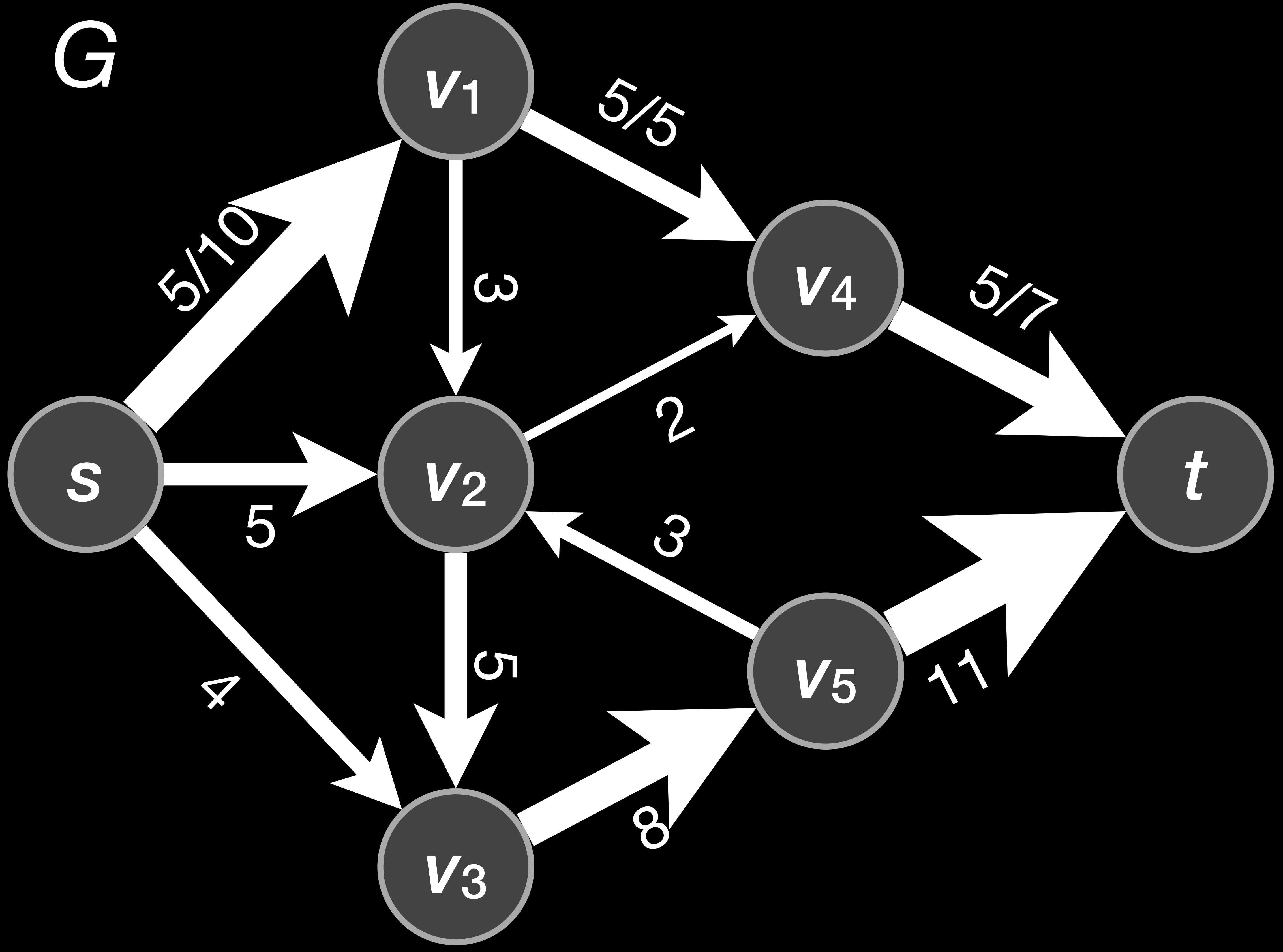


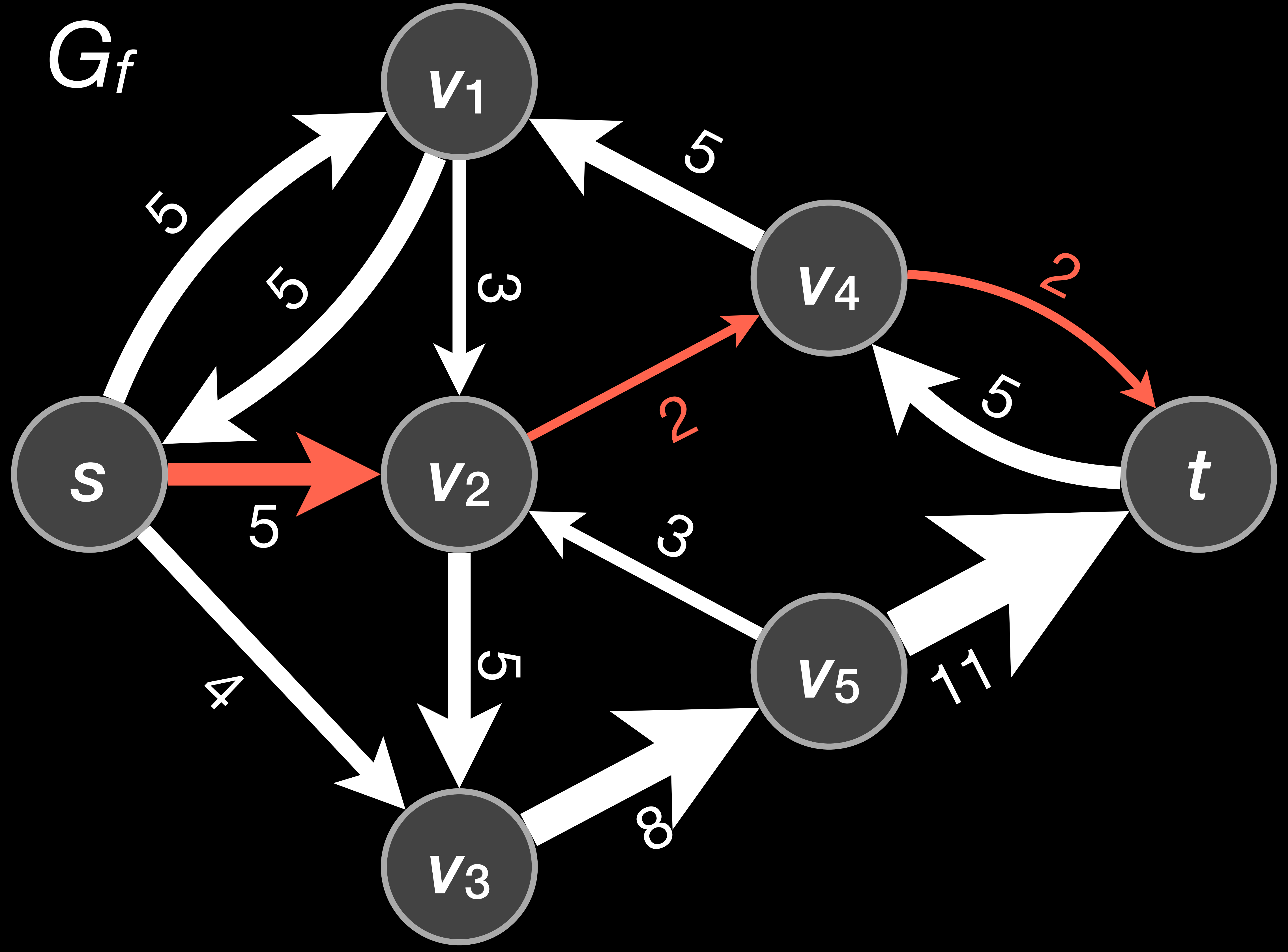
G
v1
v2
v3
v4
v5
s
t
10
5
4
3
5
2
3
8
7
11
5



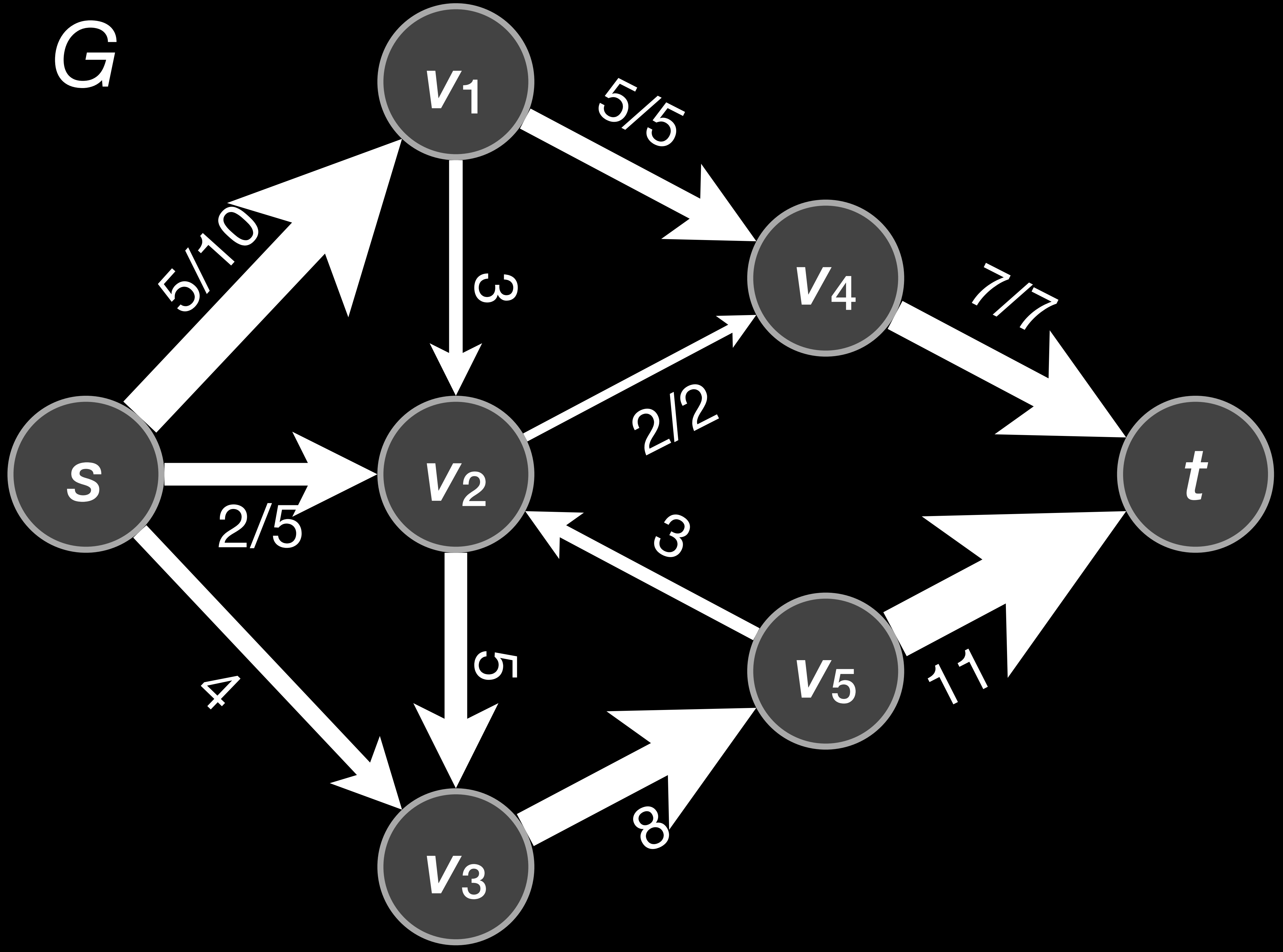
Gf
v1
v2
v3
v4
v5
s
t
10
5
4
3
5
2
3
8
7
11
5

Example

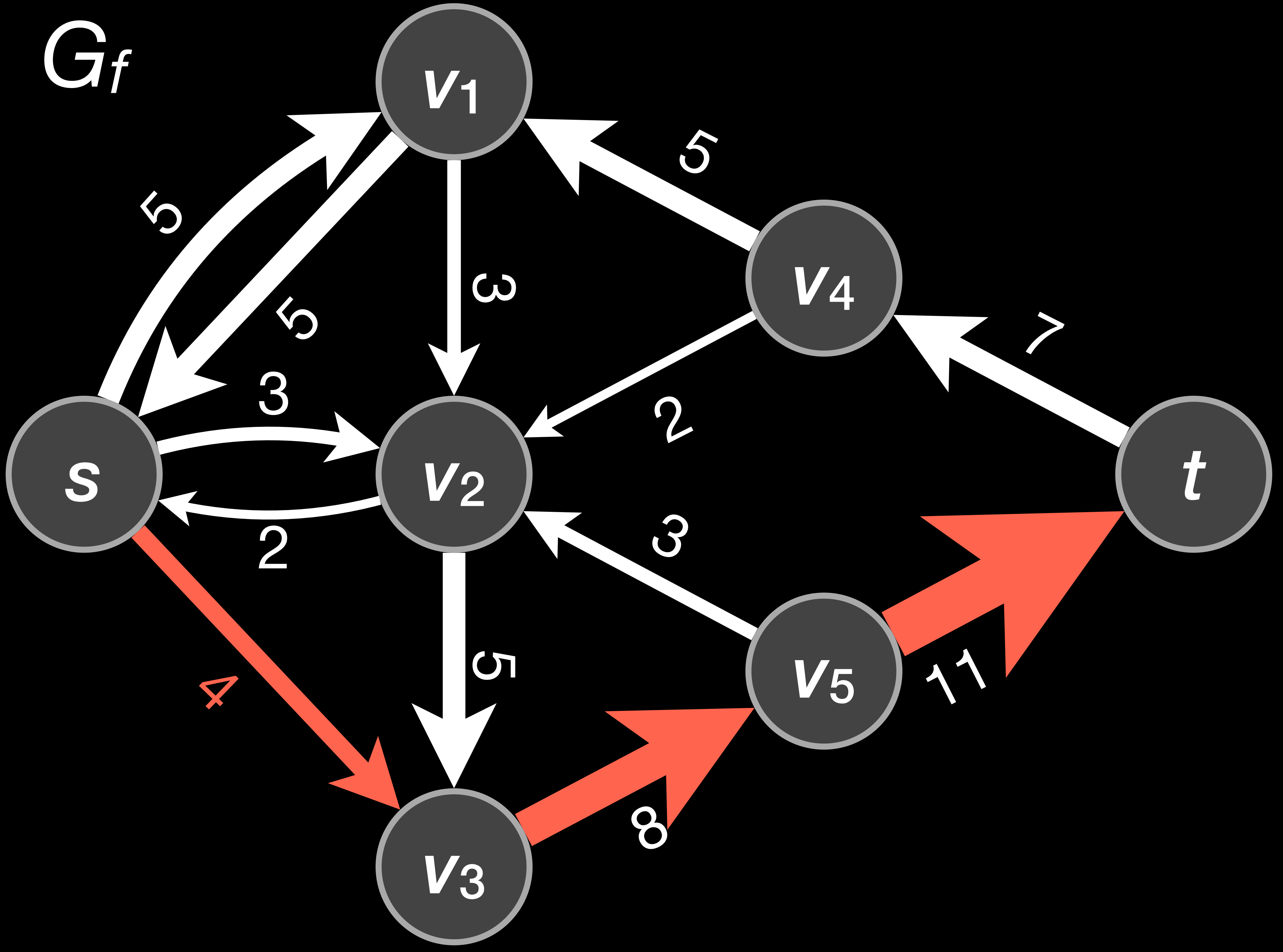

G
v_1
v_2
v_3
v_4
v_5
s
t
5/10
5/5
3
5/7
2
5
3
5
4
11
8


G_f
v_1
v_2
v_3
v_4
v_5
s
t
5
5
5
3
2
2
5
5
3
5
4
11
8

Example

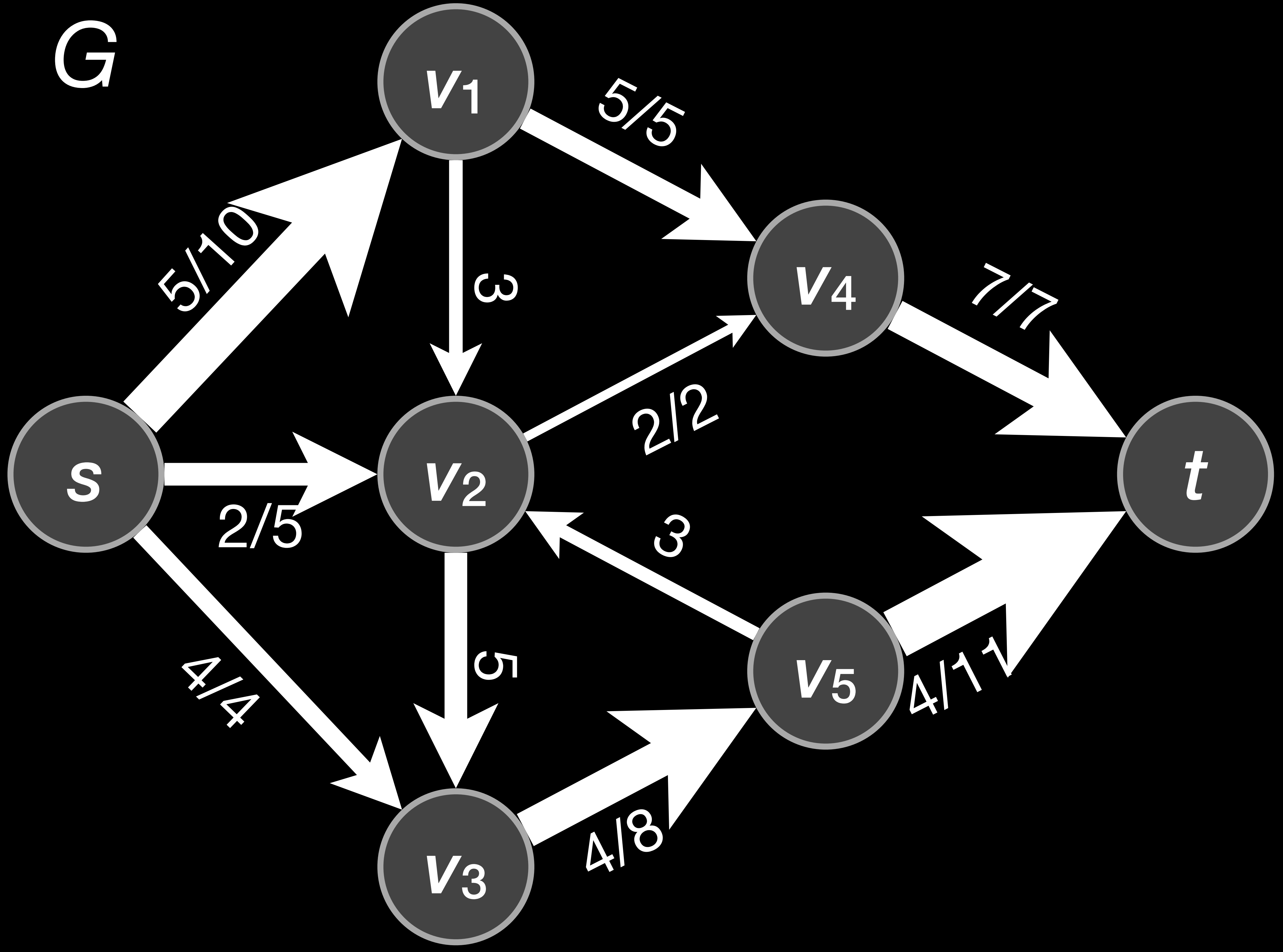


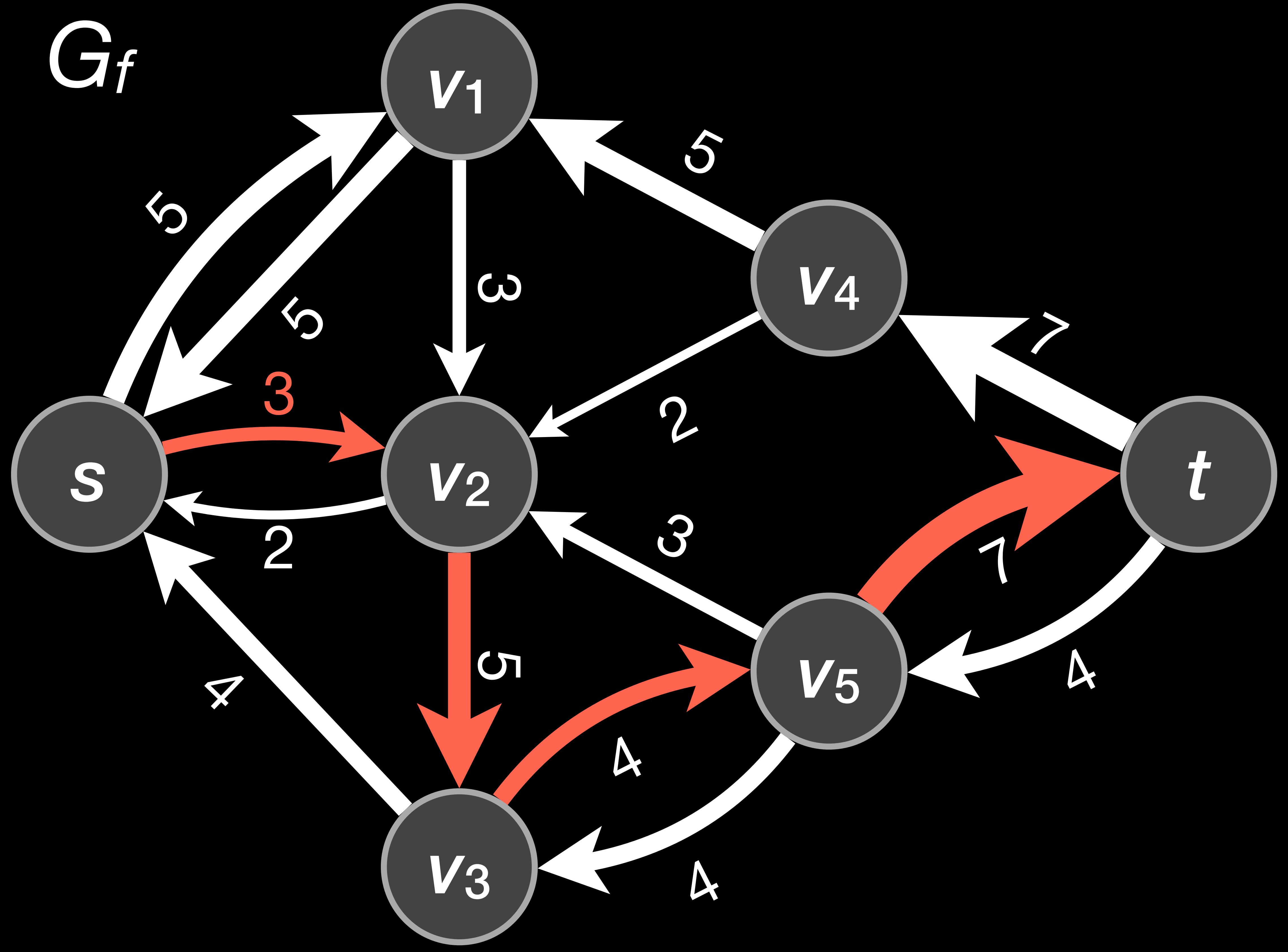
G
s
v1
v2
v3
v4
v5
t
5/10
2/5
4
3
5/5
2/2
5
3
8
7/7
11



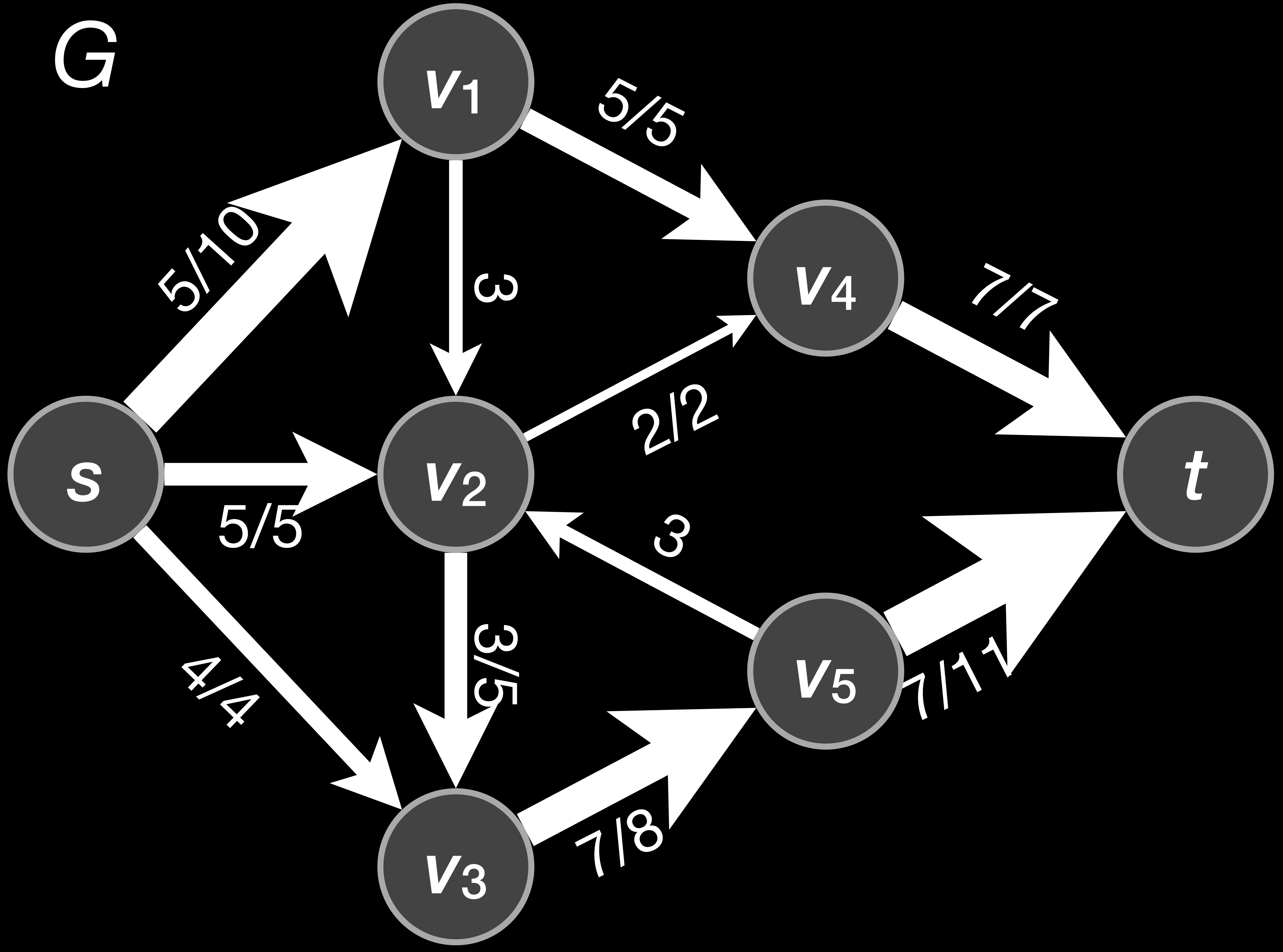
Gf
s
v1
v2
v3
v4
v5
t
5
5
3
2
4
3
5
5
2
3
8
7
11

Example

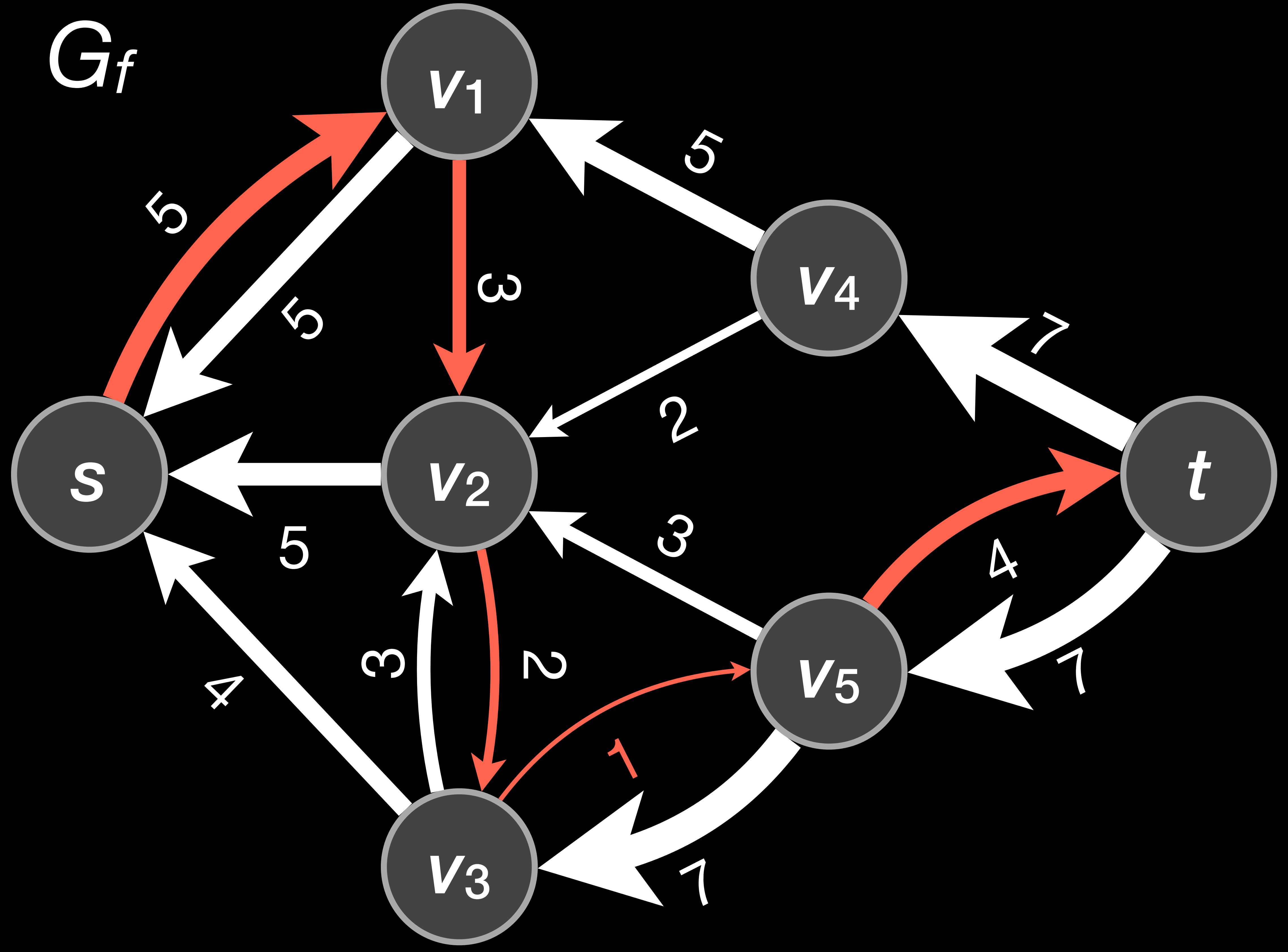

G
v_1
v_2
v_3
v_4
v_5
s
t
5/10
5/5
3
2/5
2/2
7/7
3
4/4
5
4/8
4/11


G_f
v_1
v_2
v_3
v_4
v_5
s
t
5
5
3
5
7
3
2
2
2
3
4
5
7
4
4
4

Example

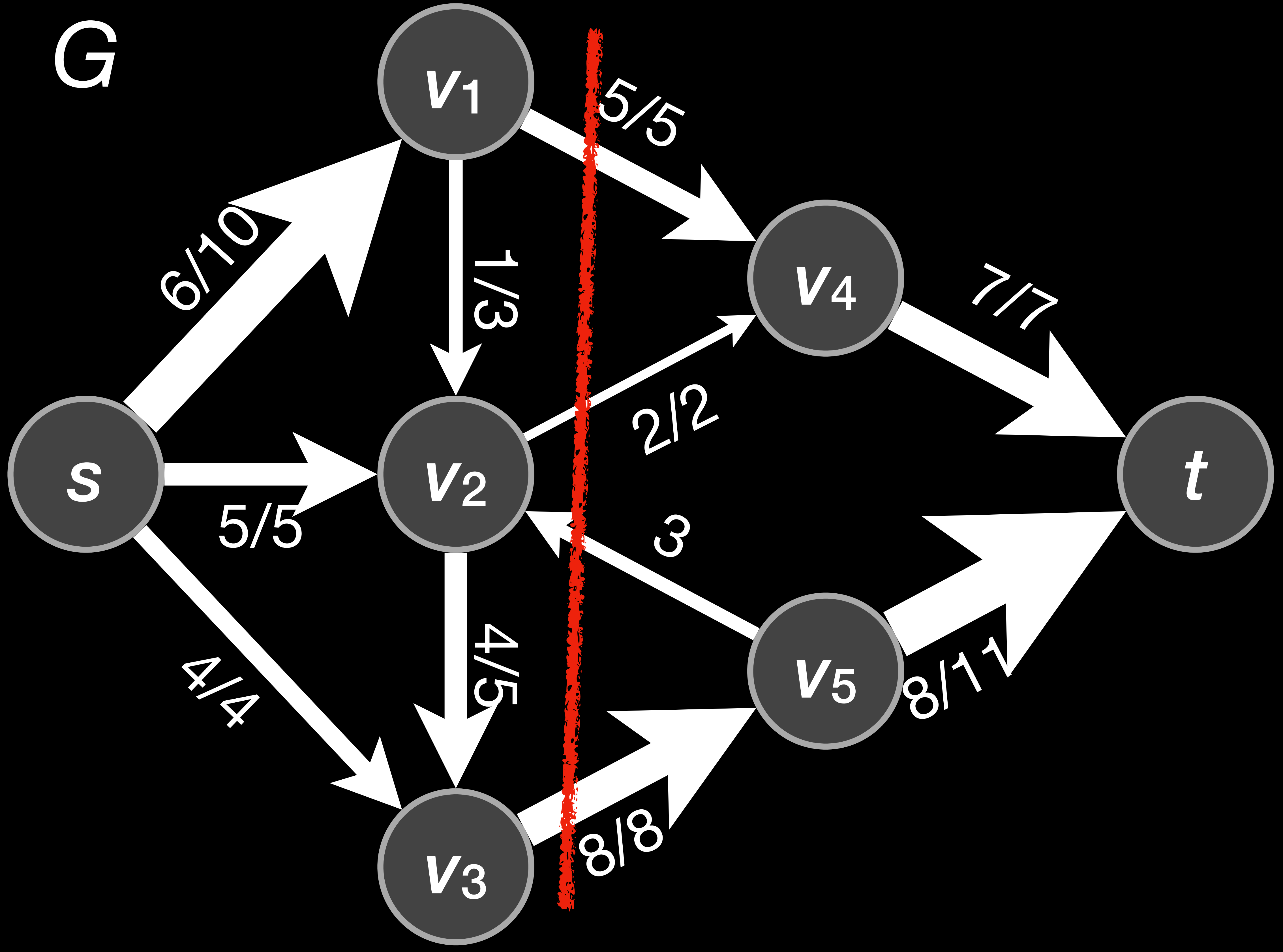


G
v1
v2
v3
v4
v5
s
t
5/10
5/5
3
5/5
2/2
7/7
3
3/5
4/4
7/8
7/11

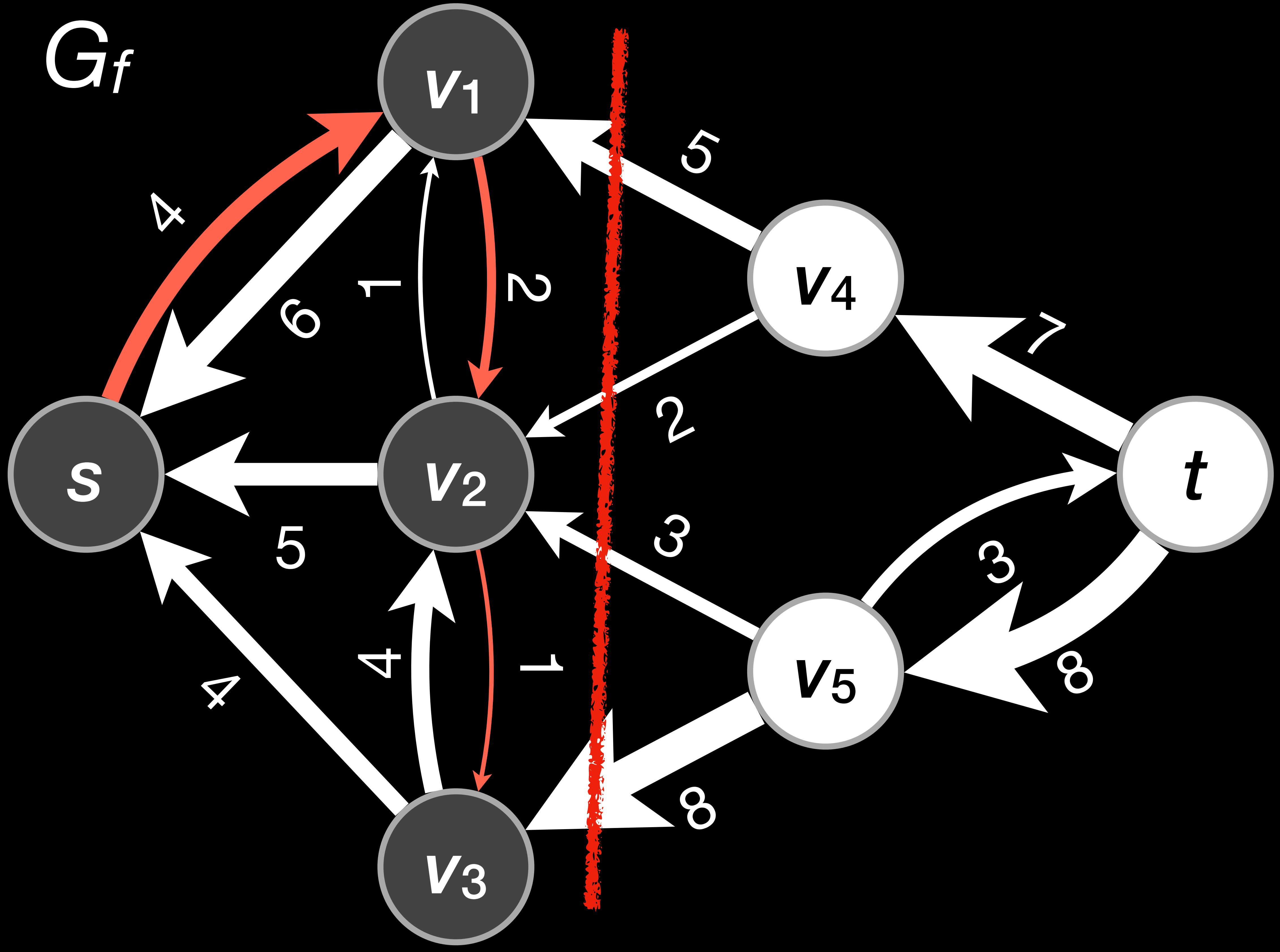


Gf
v1
v2
v3
v4
v5
s
t
5
5
3
5
2
7
5
3
4
3
2
1
7
4
7

Example

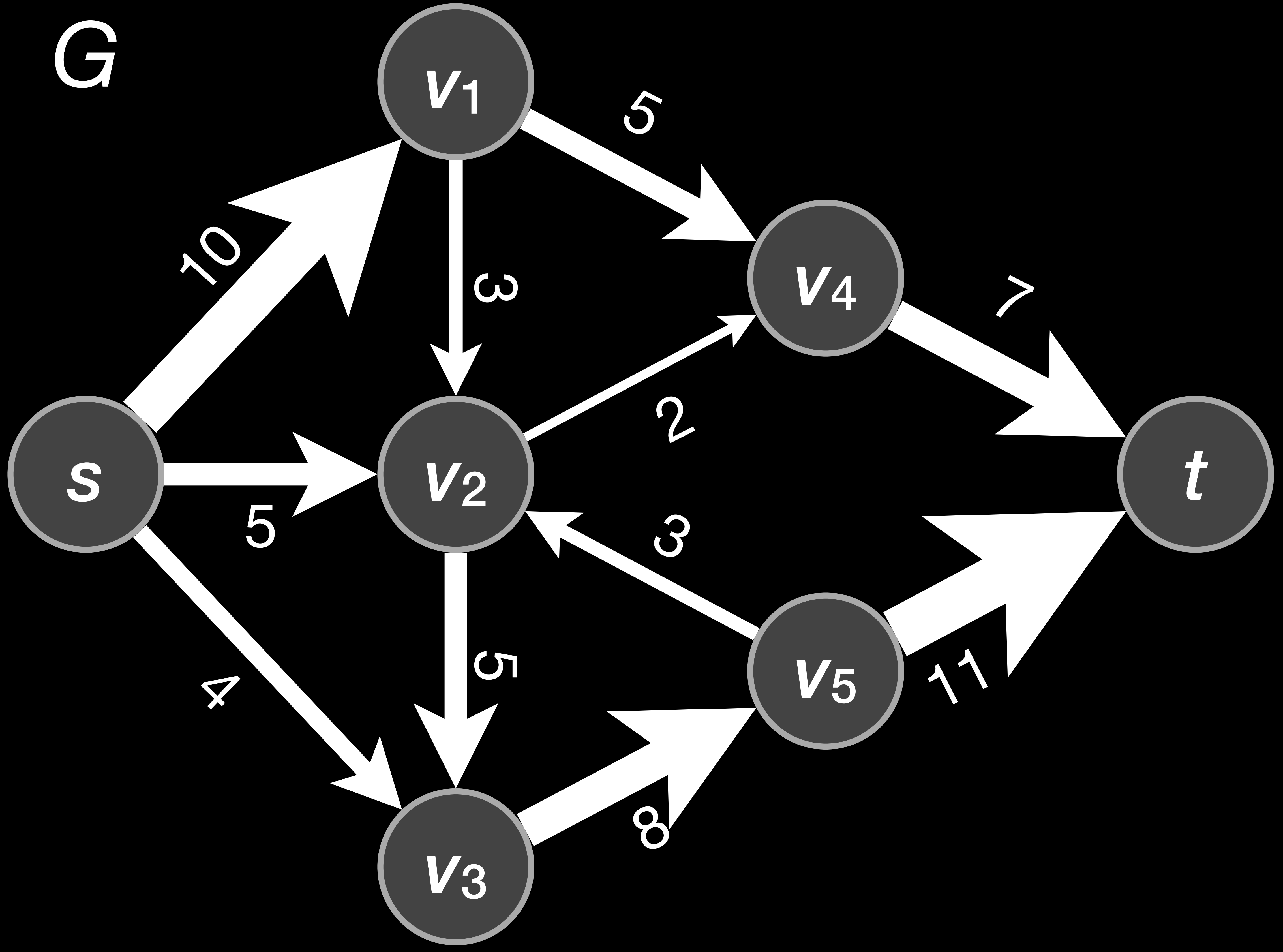


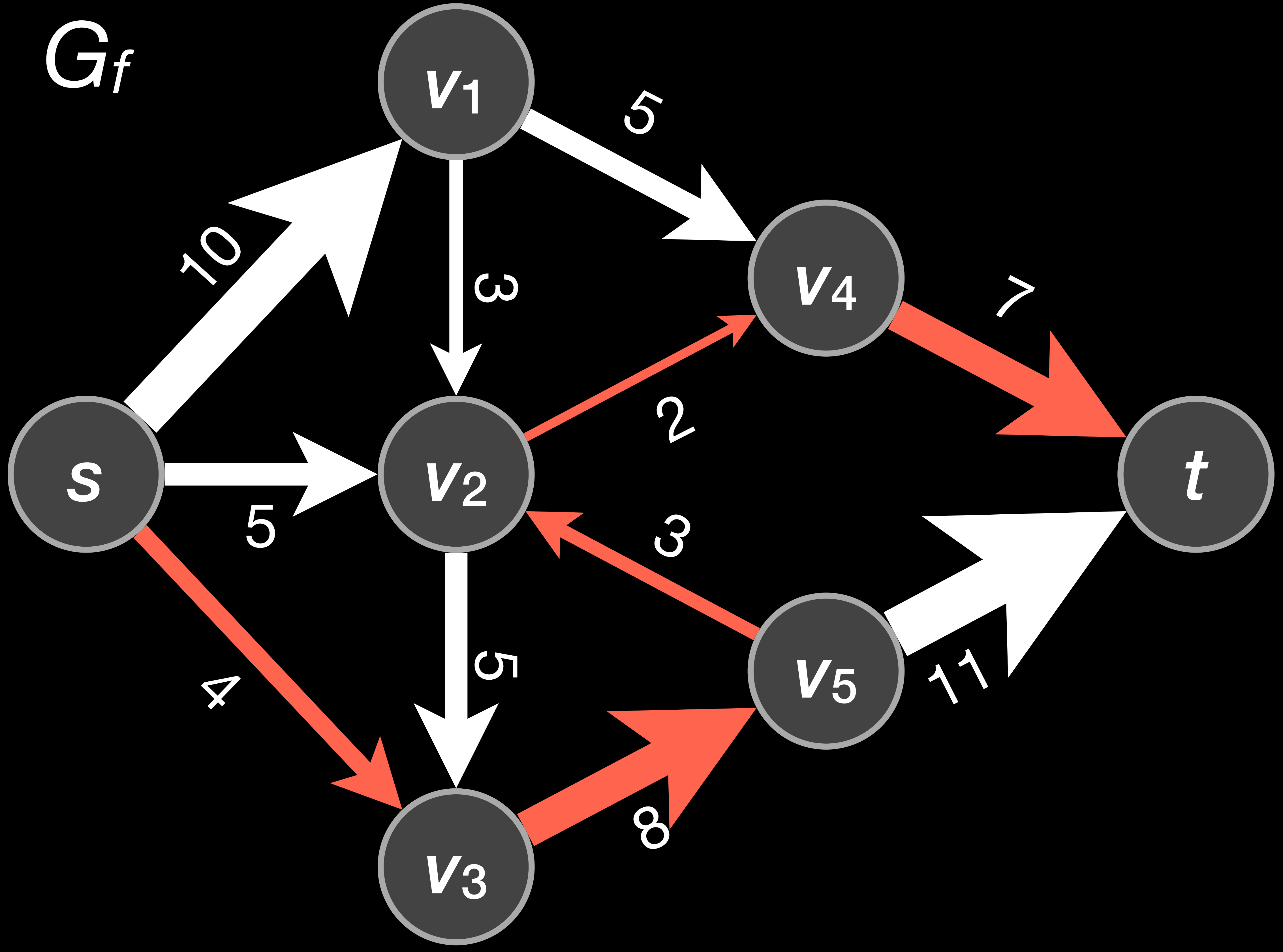
G
v1
v2
v3
v4
v5
s
t
6/10
5/5
4/4
1/3
4/5
5/5
2/2
3
8/8
7/7
8/11



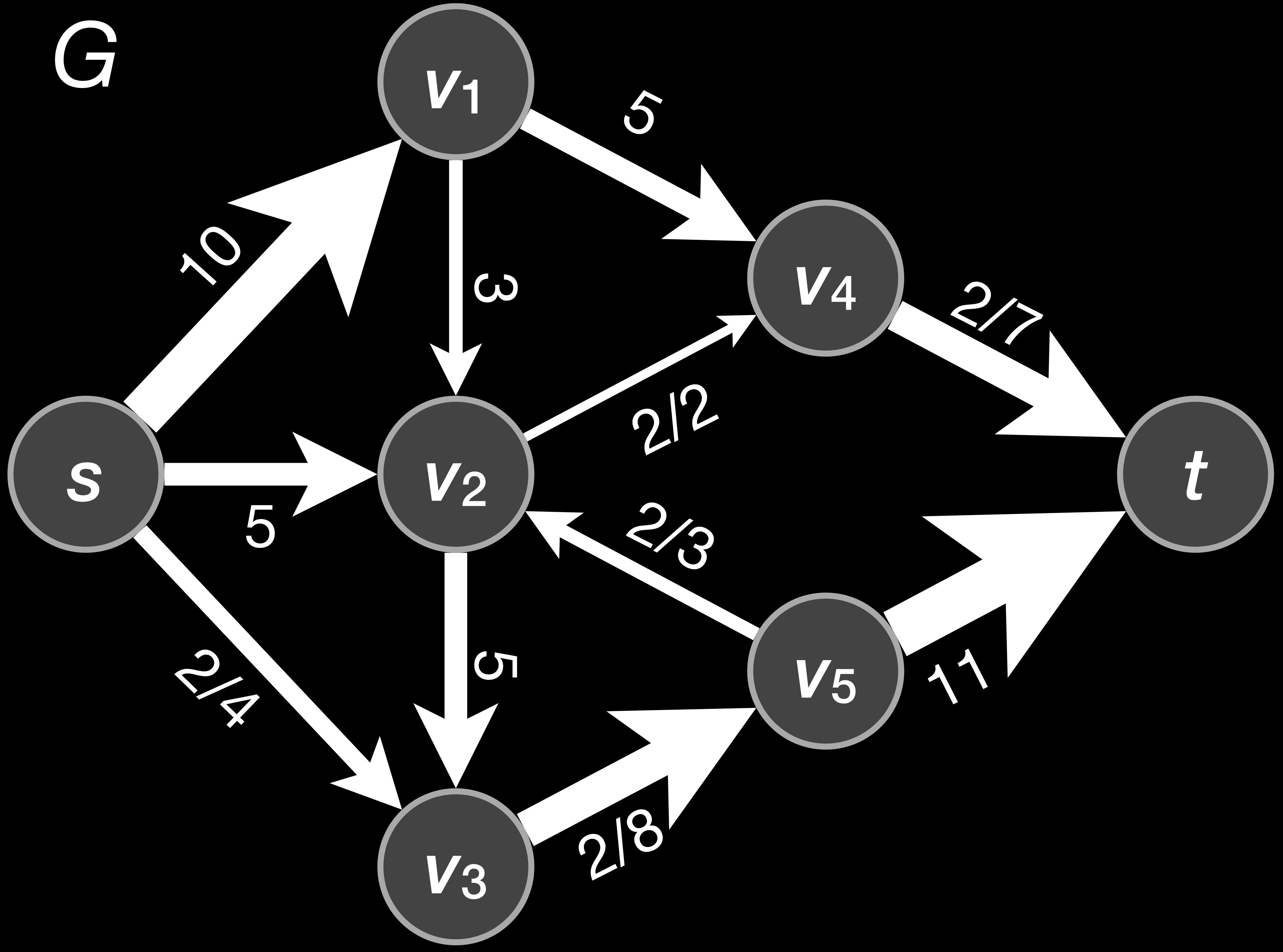
Gf
v1
v2
v3
v4
v5
s
t
4
6
5
4
1
2
4
1
5
2
3
8
7
3
8

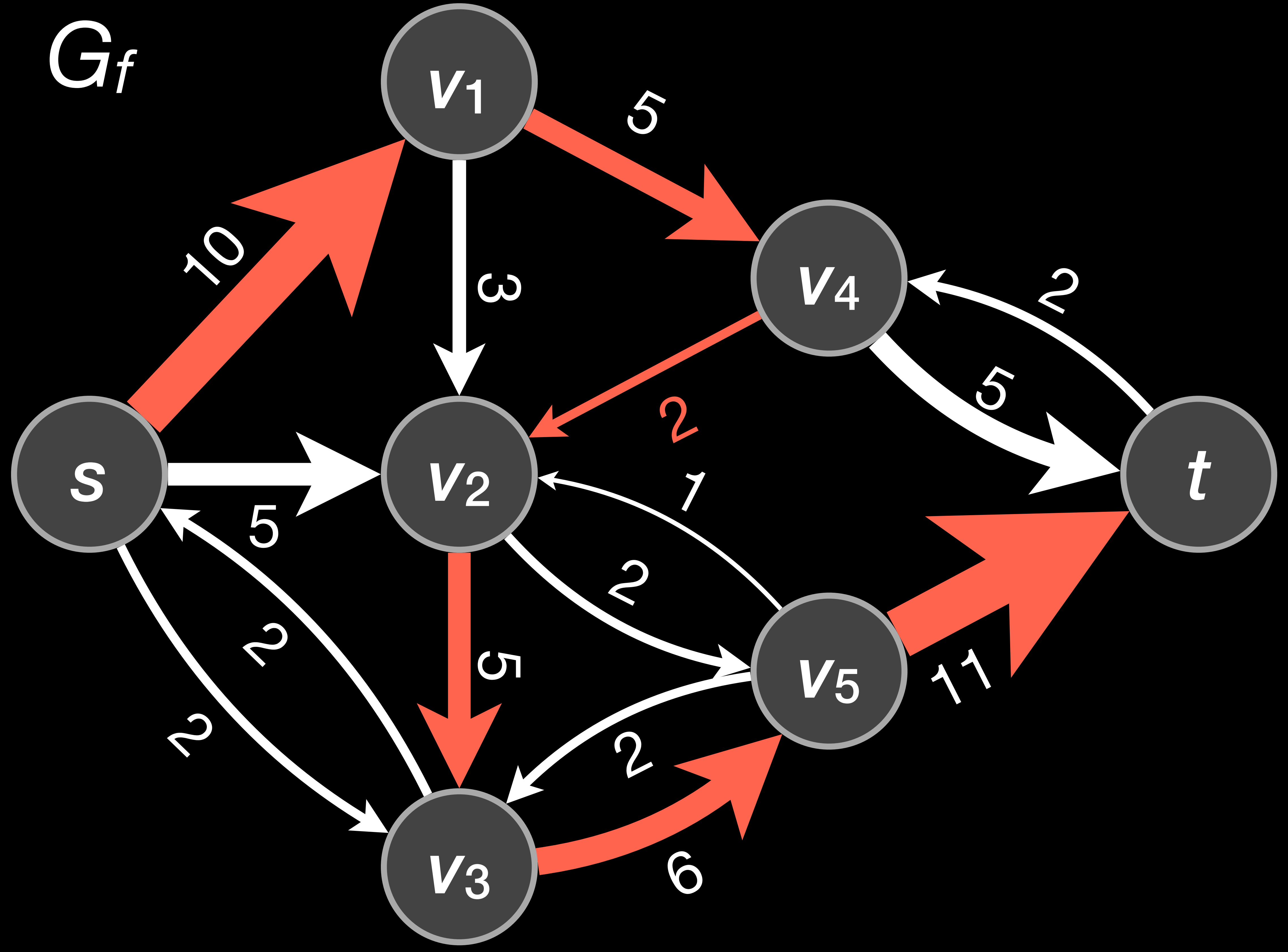
Example: bad choices



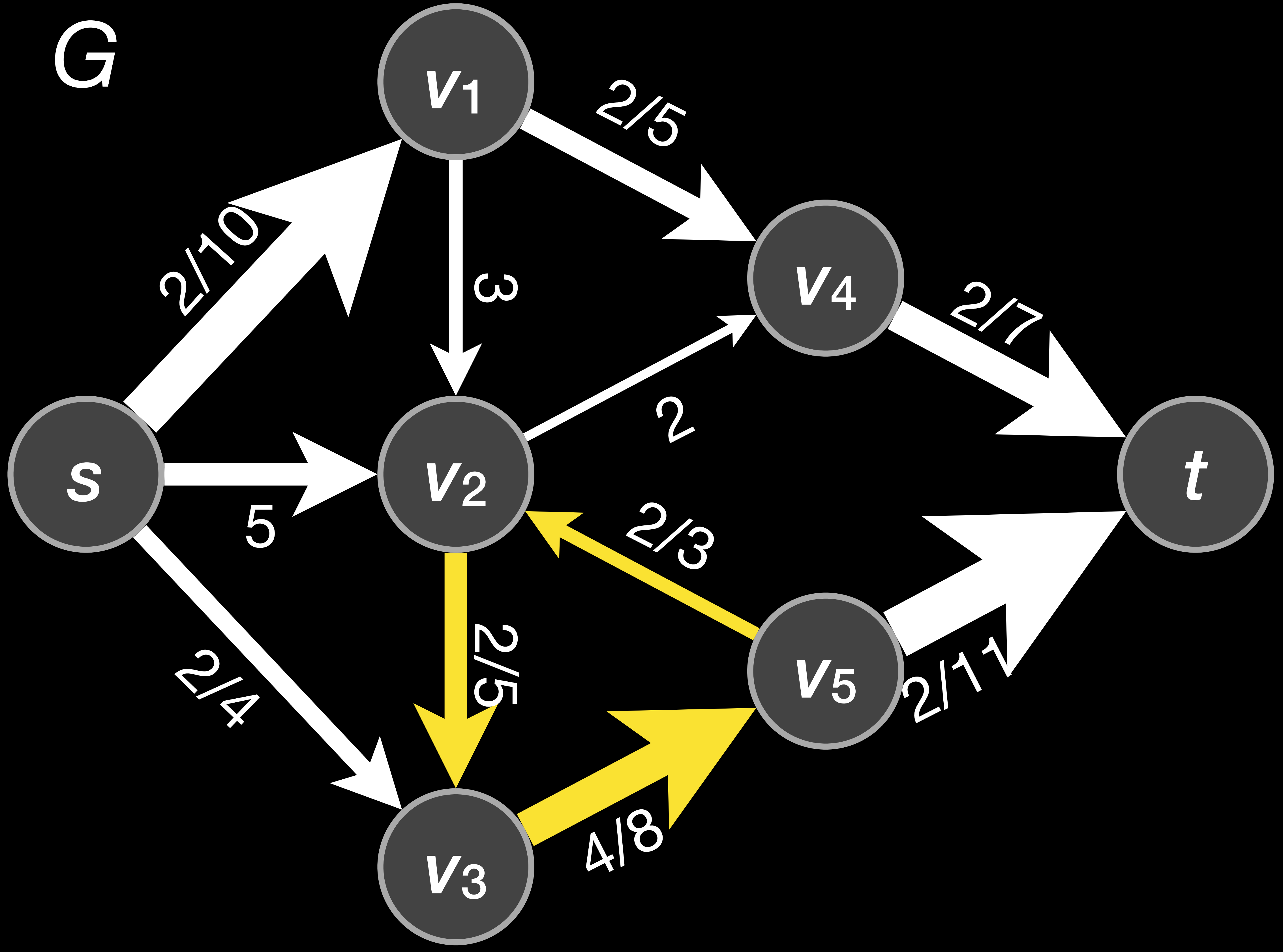


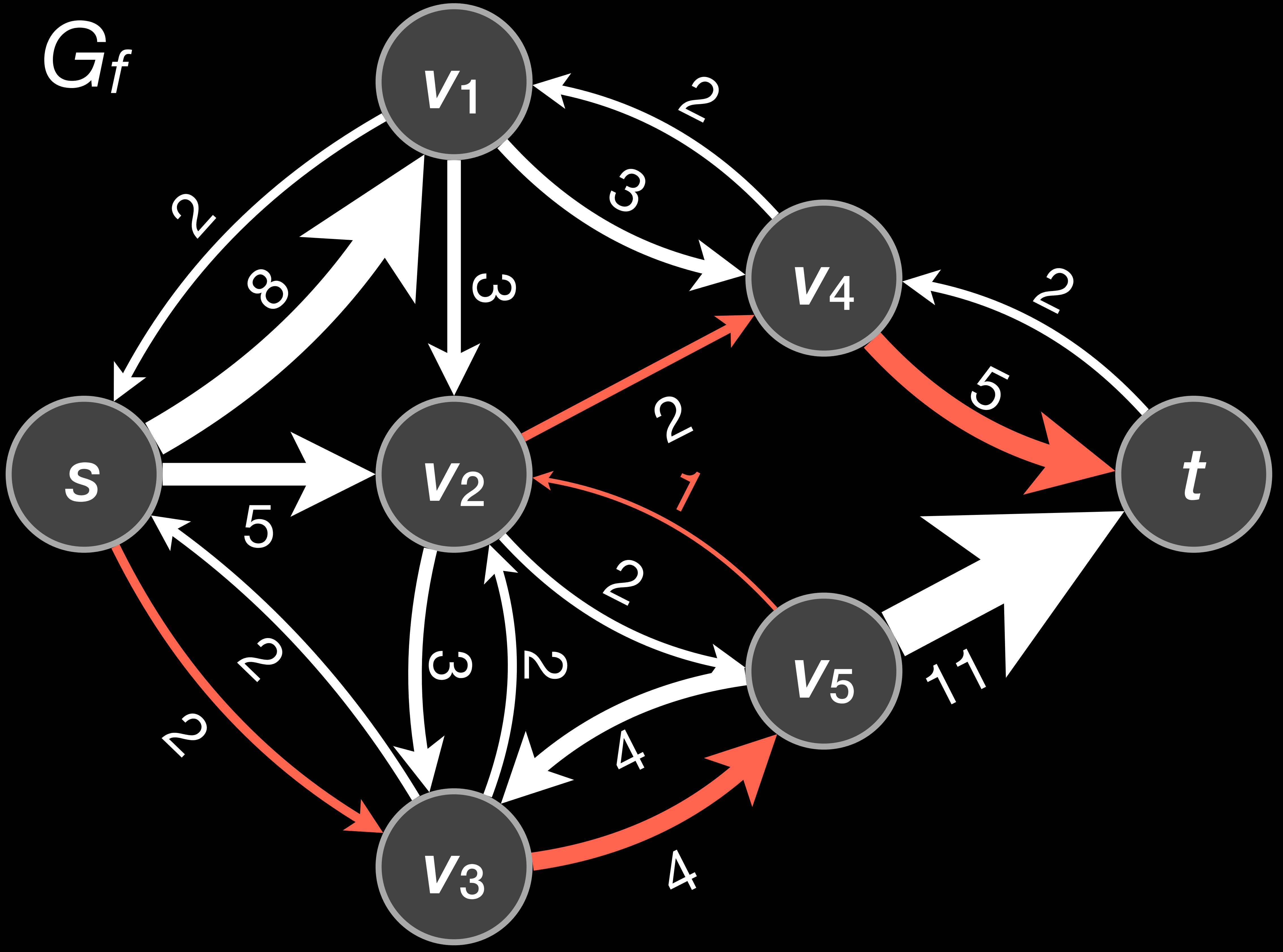
Example: bad choices





Example: bad choices





Maximum Bipartite Matching 最大二分匹配

- one application of maximum flow
- illustration that maximum flow problems appear where one may not expect them.

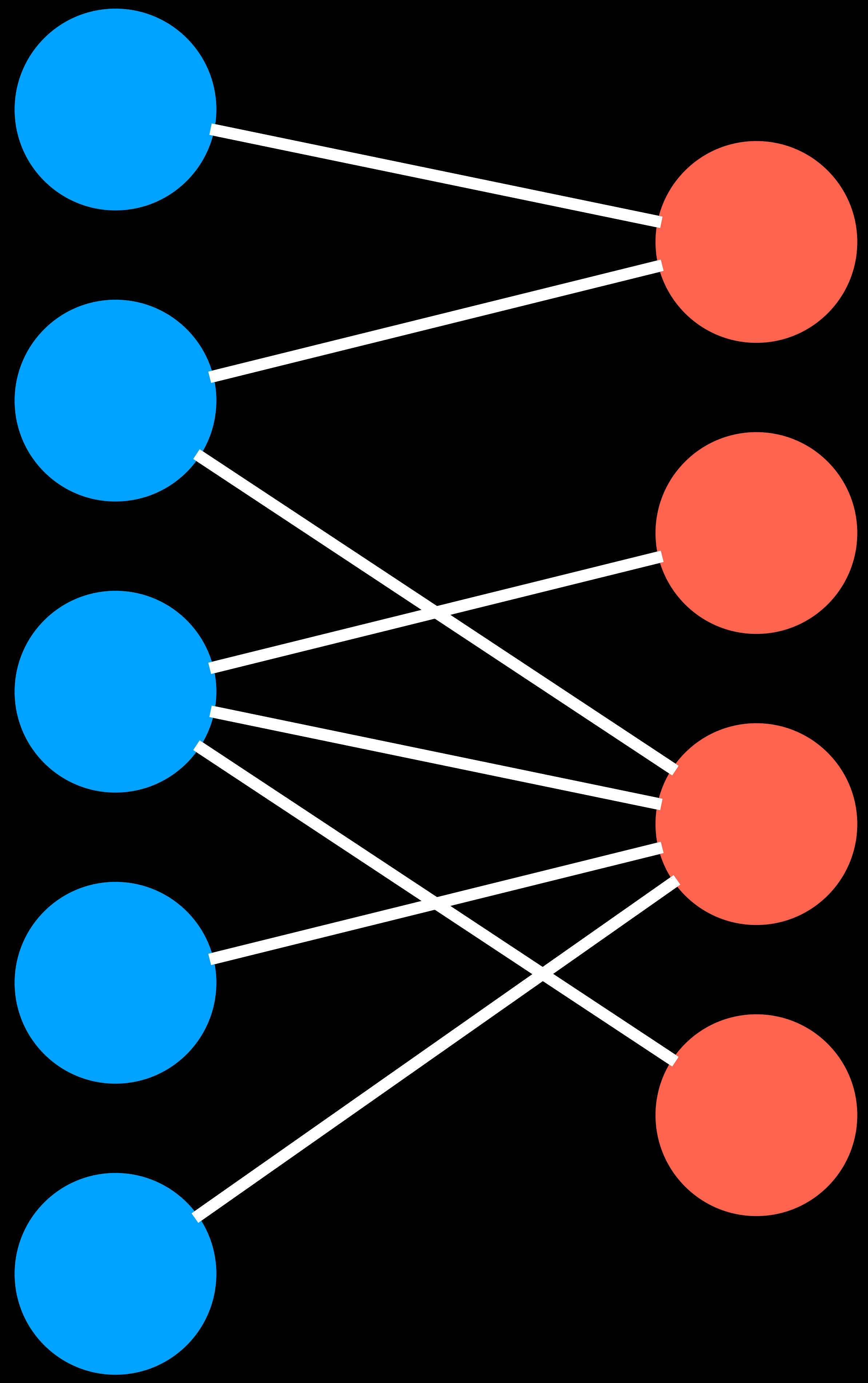
Maximum Bipartite Matching 最大二分匹配

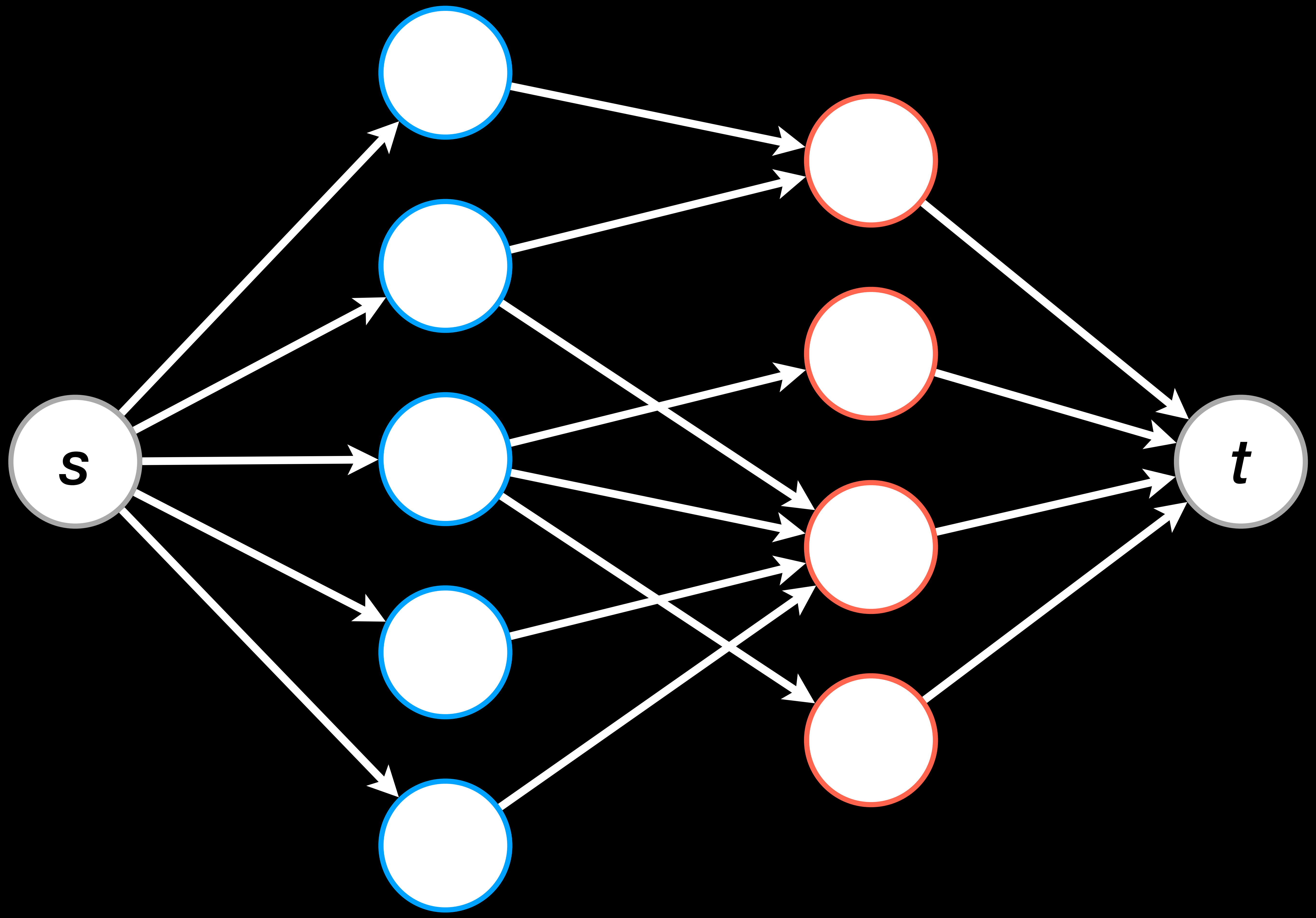
- Example: We have a group of students and a number of apartments.
  Every student is allowed to indicate which apartment(s) s/he likes.
  The housing agency assigns as many students to apartments they like as possible.

- Bipartite graph 二分图 $G = (V,E)$ where $V = L \cup R$. Edges $E \subseteq L \times R$.
  A matching 匹配 M is a subset of E that contains at most one edge for every vertex.

- The problem to find a maximum matching can be solved using maximum flow.

Maximum Bipartite Matching 最大二分匹配



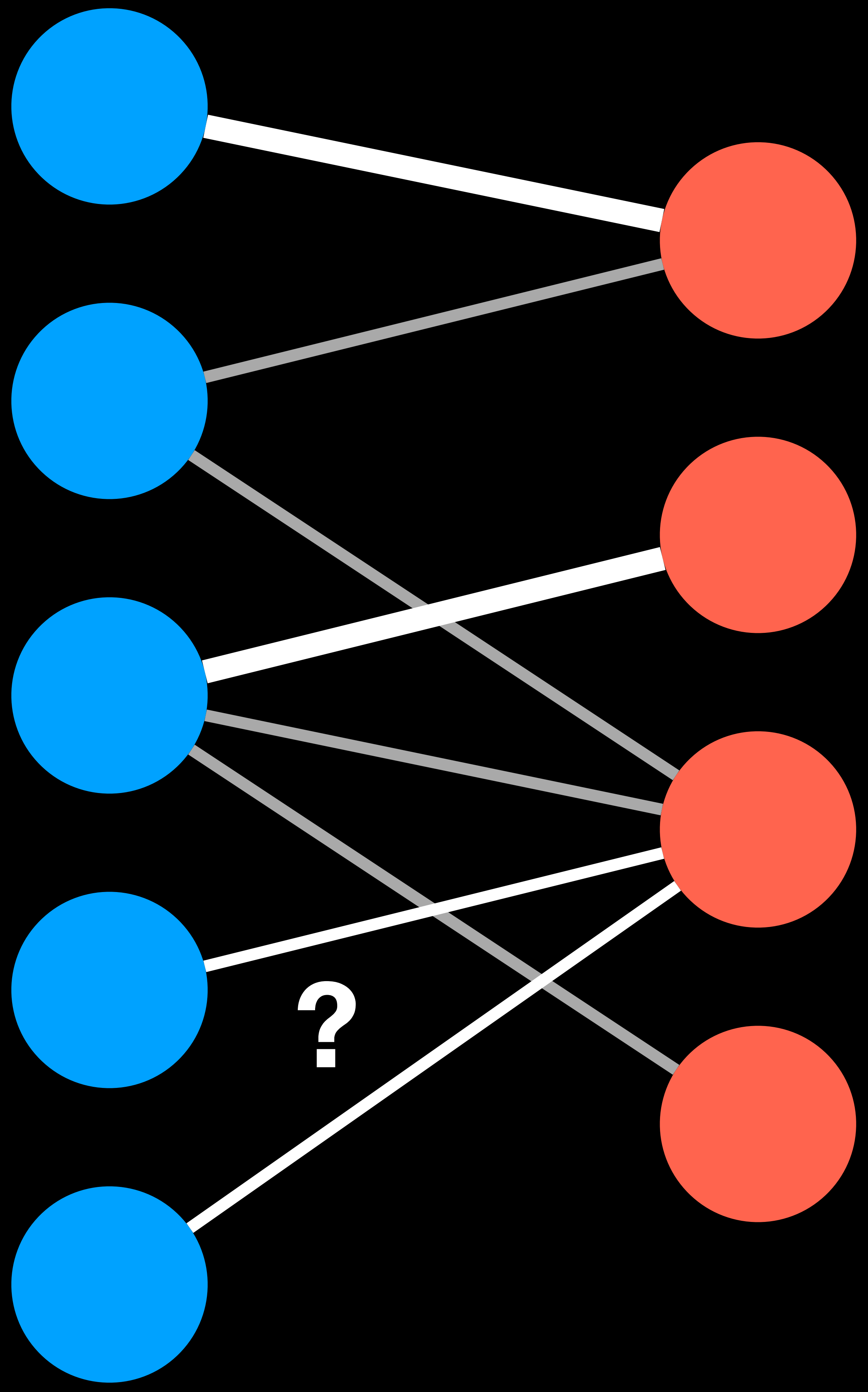


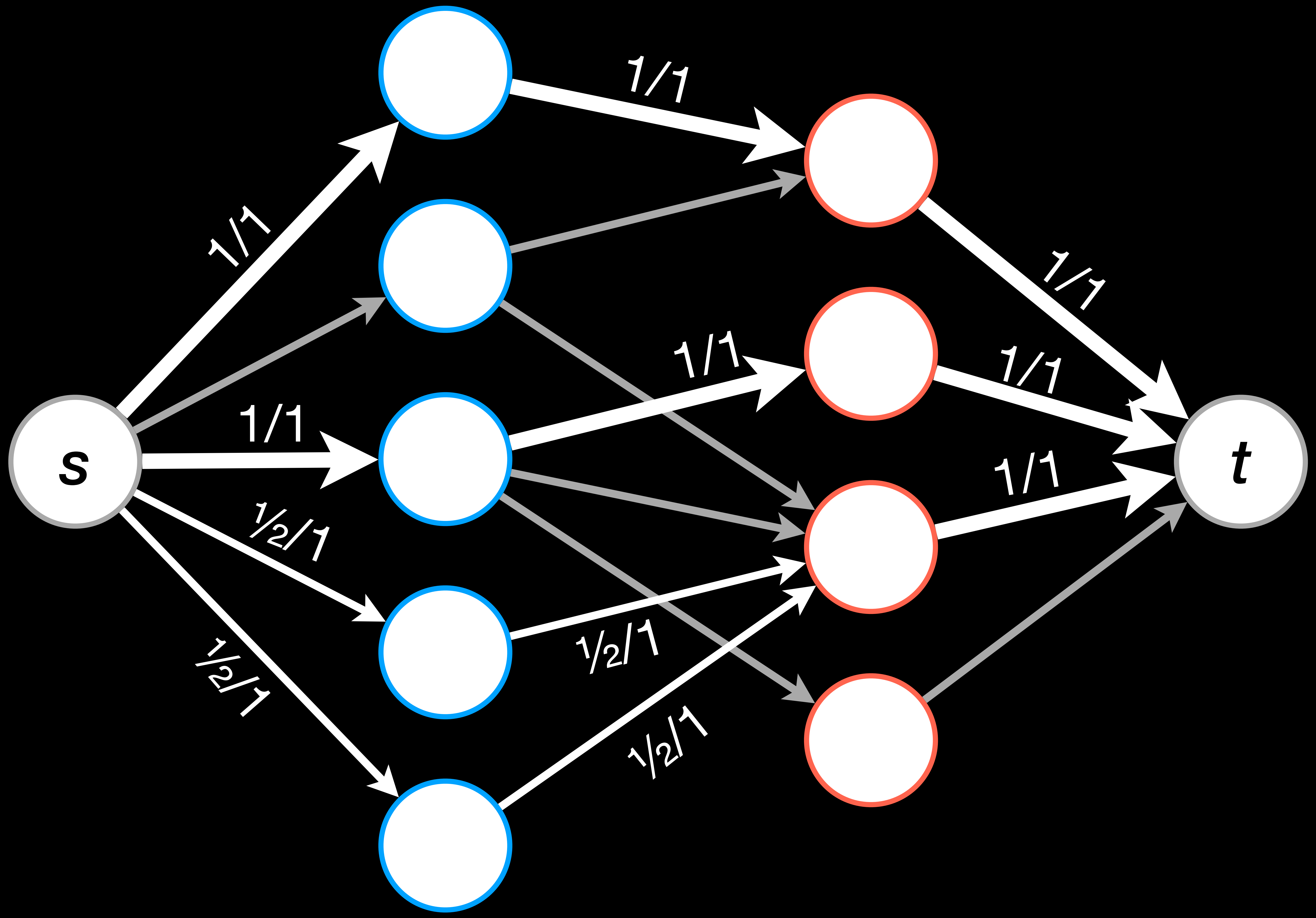
Maximum Bipartite Matching 最大二分匹配

- Lemma 26.9: Let G = (V,E) be a bipartite graph (where $V = L \cup R$ and $E \subseteq L \times R$). Let $G' = (V',E')$ be its corresponding flow network ($V' = V \cup \{s,t\}$ and $E' = E \cup \{s\} \times L \cup R \times \{t\}$). If there is a matching $M \subseteq E$ in G,
  then there is an integer-valued flow f in G' with $|M| = |f|$.
  If there is an integer-valued flow f in G',
  then there is a matching $M \subseteq E$ in G with $|M| = |f|$.

- Note: "integer-valued"—required for a matching to exist. The Ford–Fulkerson method would normally produce an integer solution (see the discussion about running time).

Maximum Bipartite Matching





Summary

总结

- Maximum flow problems pose the question “How much material/people/liquid/... can be transported from the source to the sink?”

- **Ford–Fulkerson** uses augmenting paths to approach the maximum flow.

- **Edmonds–Karp** improve timing by requiring the shortest augmenting paths to be selected.

- 最大流问题提出了问题：“有多少材料/人/液体/...可以从源点运输到汇点？”

- **Ford-Fulkerson** 使用增广路径来接近最大流。

- **Edmonds–Karp** 选择最短的增广路径，来改进运行时间。

Exercise 29.5-6

Solve the following linear program using SIMPLEX:

用SIMPLEX求解下面的线性规划：

$$
\begin{array}{lrcr}
\text{maximize 最大化} & x_1 - 2x_2 & & \\
\text{subject to 满足约束} & x_1 + 2x_2 & \le & 4 \\
 & -2x_1 - 6x_2 & \le & -12 \\
 & x_2 & \le & 1 \\
 & x_1 & \ge & 0 \\
 & x_2 & \ge & 0
\end{array}
$$

Exercise 26.2-3

Show the execution of the Edmonds–Karp algorithm on the flow network of Figure 26.1(a).

在图 26.1(a) 所示的流网络上演示 Edmonds–Karp 算法的执行过程。