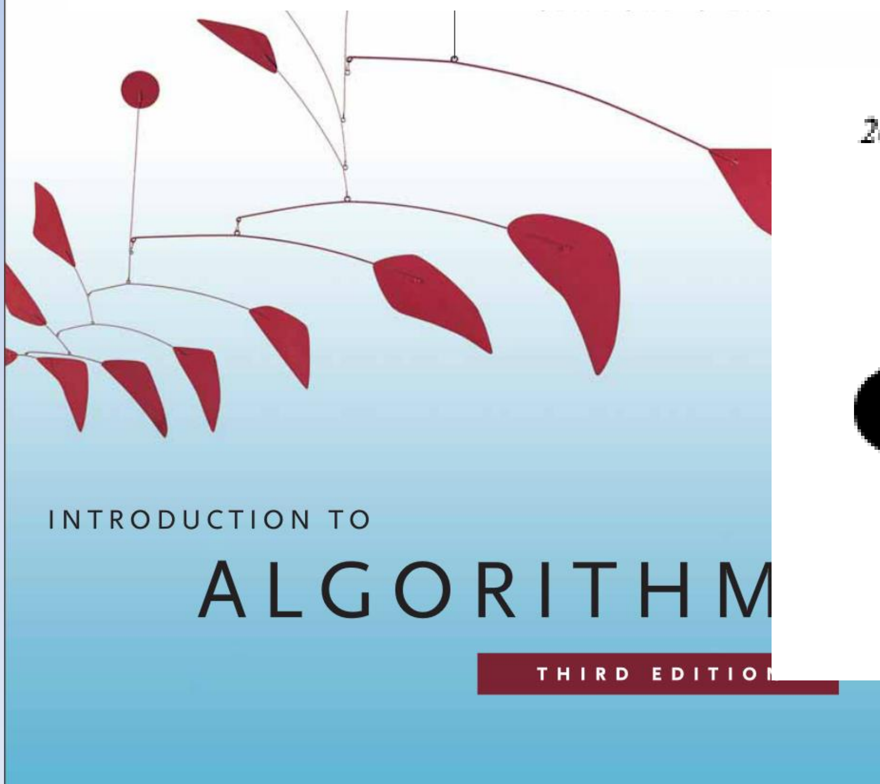
Design and Analysis of Algorithms

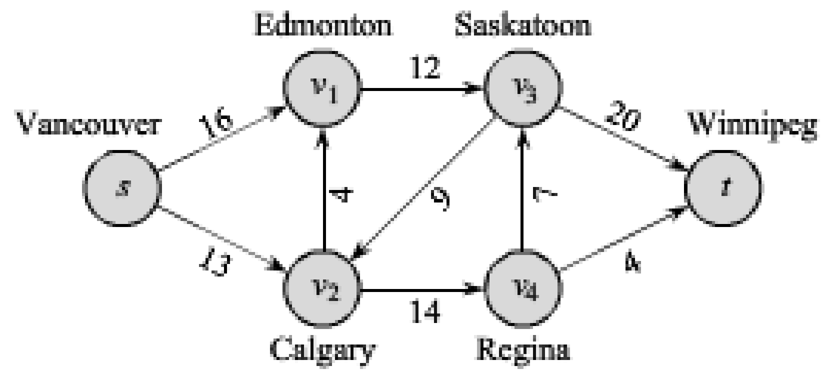
Section VI : Graph Algorithms
Chapter 26: Maximum Flow

VI ***Graph Algorithms***

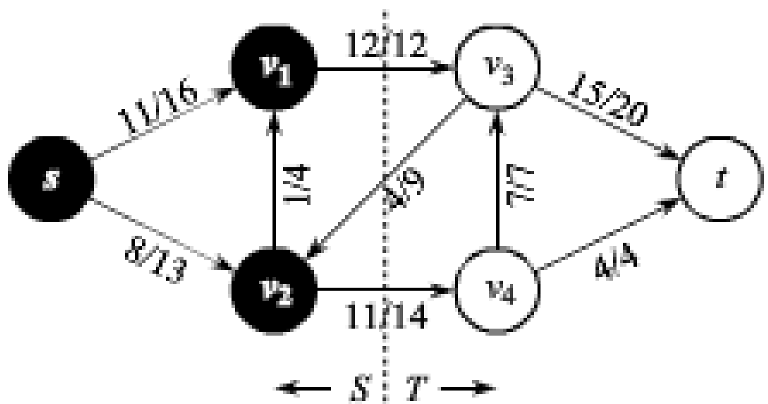
26 Maximum Flow



710 Chapter 26 Maximum Flow

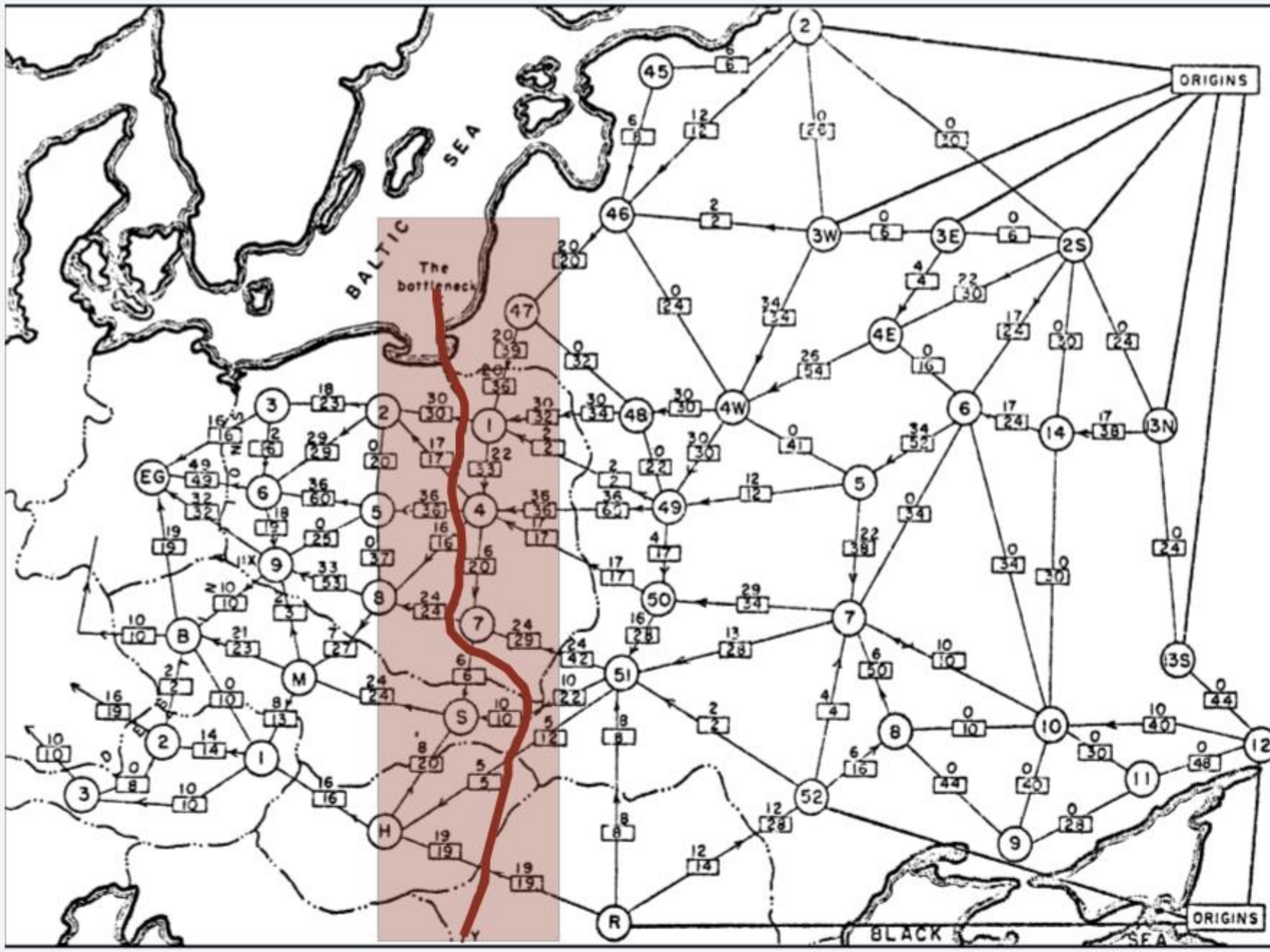


26.2 The Ford-Fulkerson method



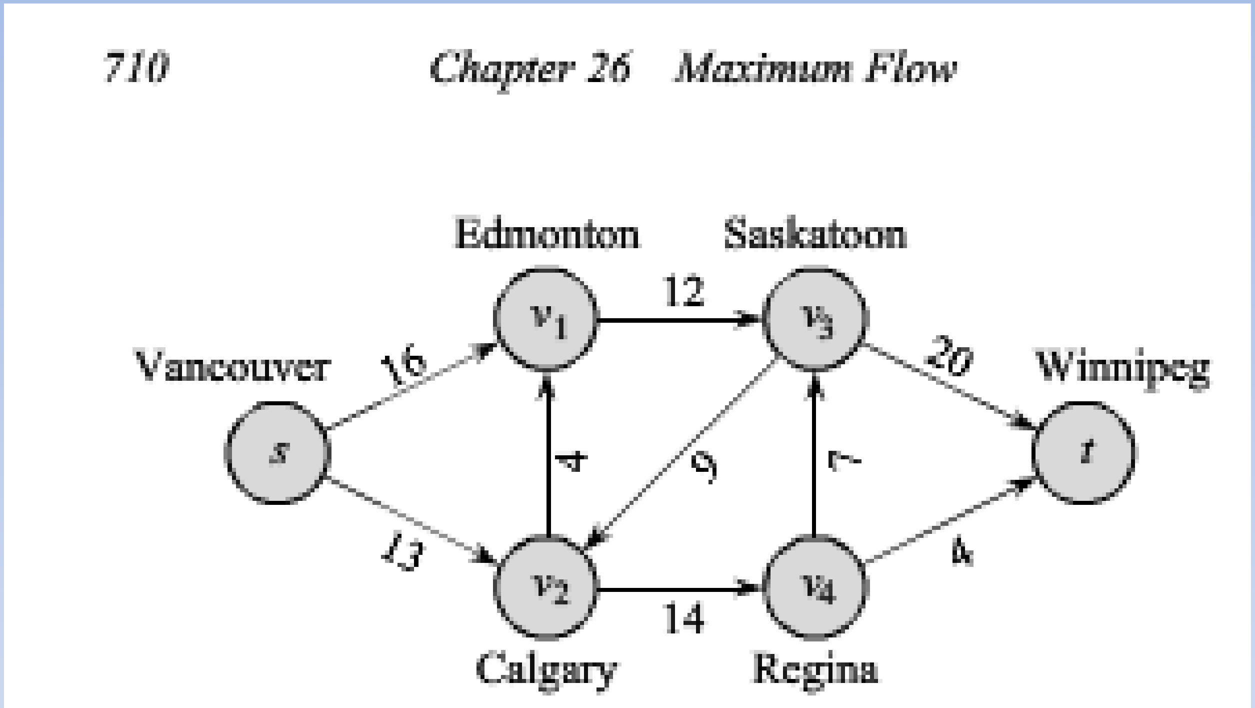
Soviet rail network (1950s)

"Free world" goal. Cut supplies (if cold war turns into real war).



Reference: *On the history of the transportation and maximum flow problems.* Alexander Schrijver in Math Programming, 91: 3, 2002.

Shipping Example

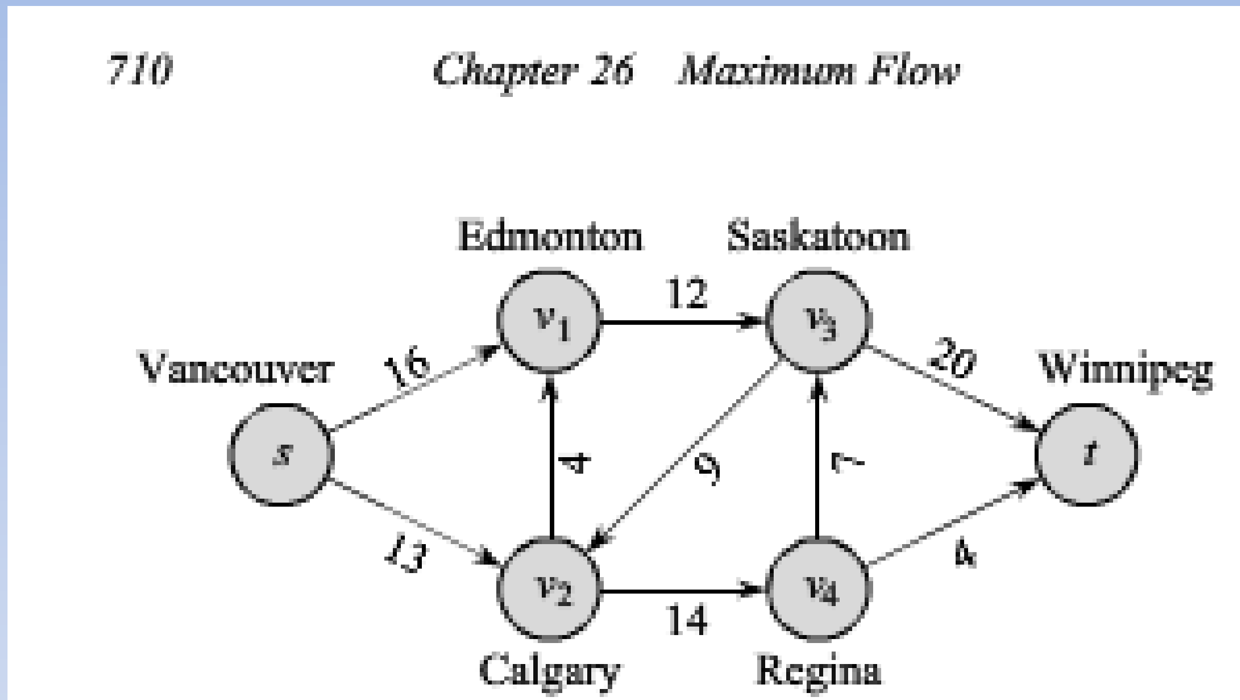


- Lucky Puck Company's Trucking Problem
- Vancouver Factory
- Winnipeg Warehouse

Flow Network

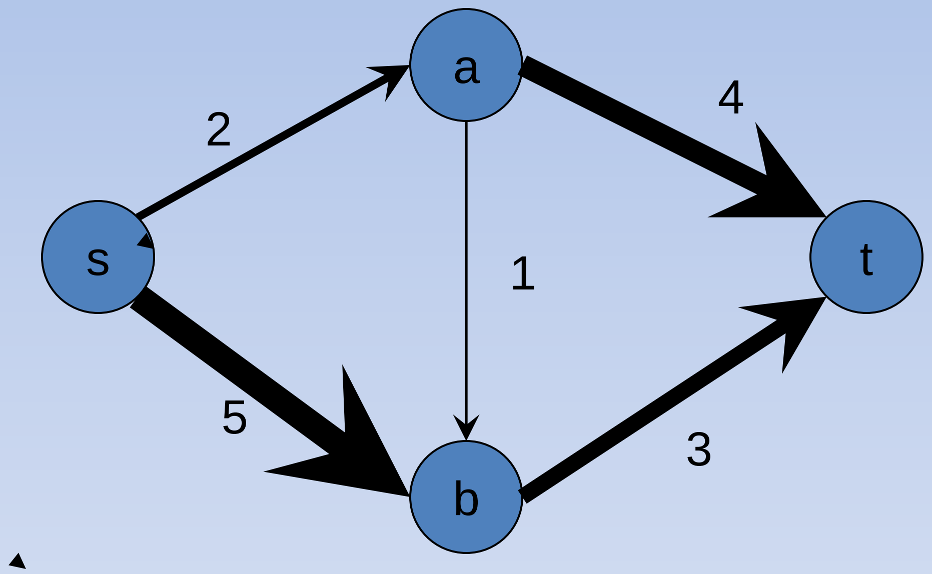
- Directed graph $G = (V, E)$ with
  - edge capacities $c(u,v) \geq 0$
  - a designated source node s
  - a designated target/sink node t
  - flows on edges $f(u,v)$

Shipping Example



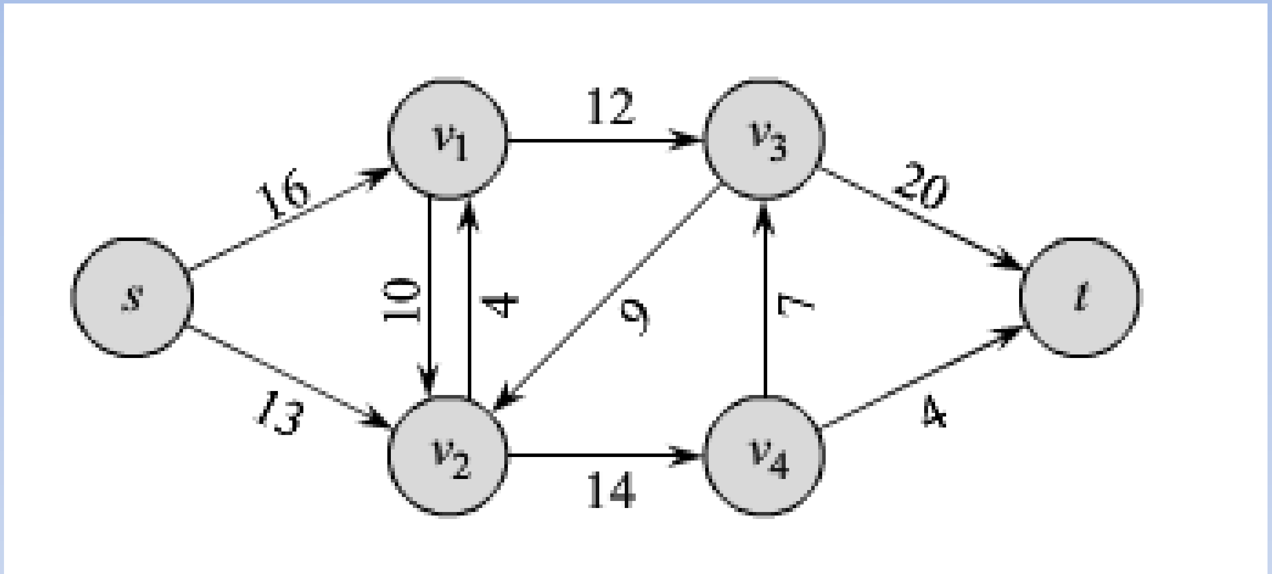
- Lucky Puck Company’s Trucking Problem
- Vancouver Factory
- Winnipeg Warehouse

Flow Network

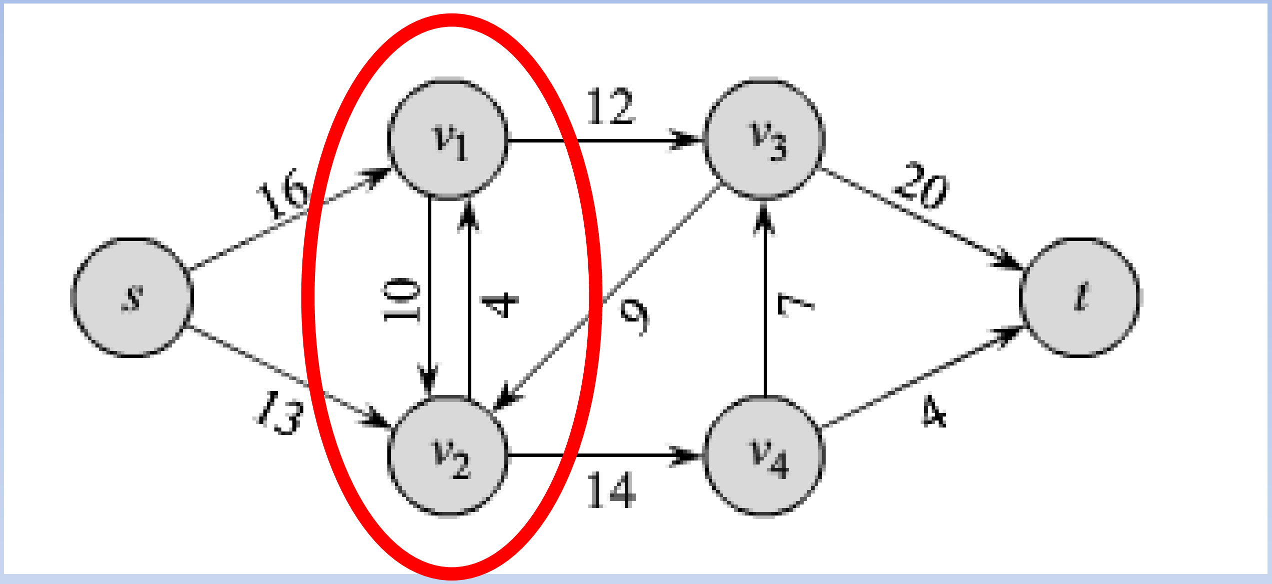


$c(s,a) = 2$
$c(s,b) = 5$
$c(a,b) = 1$
$c(a,t) = 4$
$c(b,t) = 3$

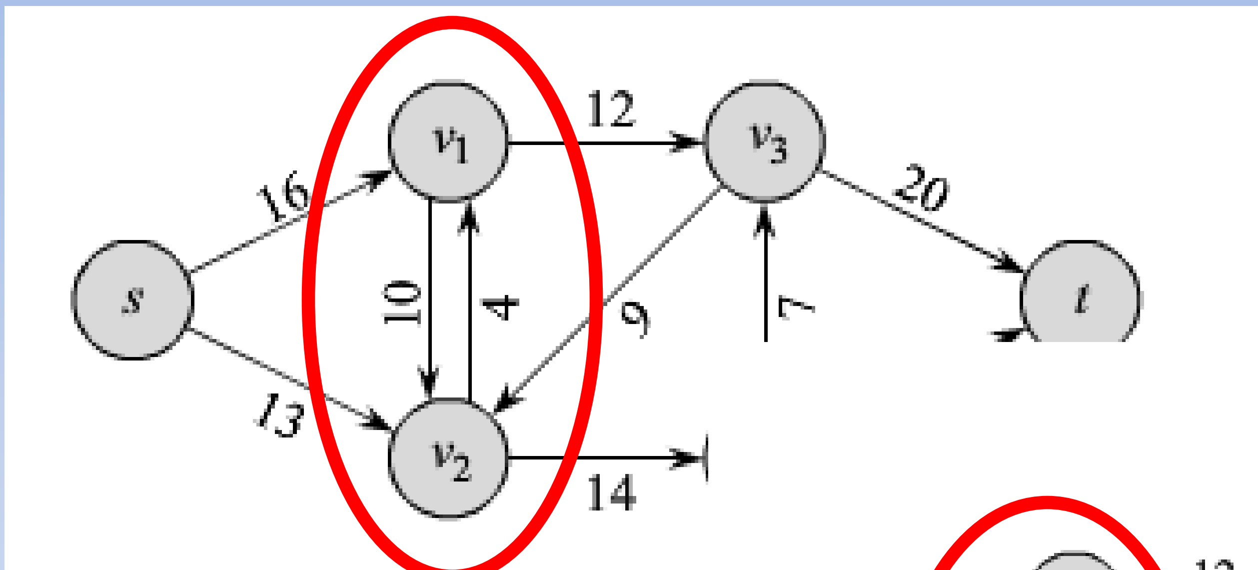
Antiparallel Edges Not Allowed

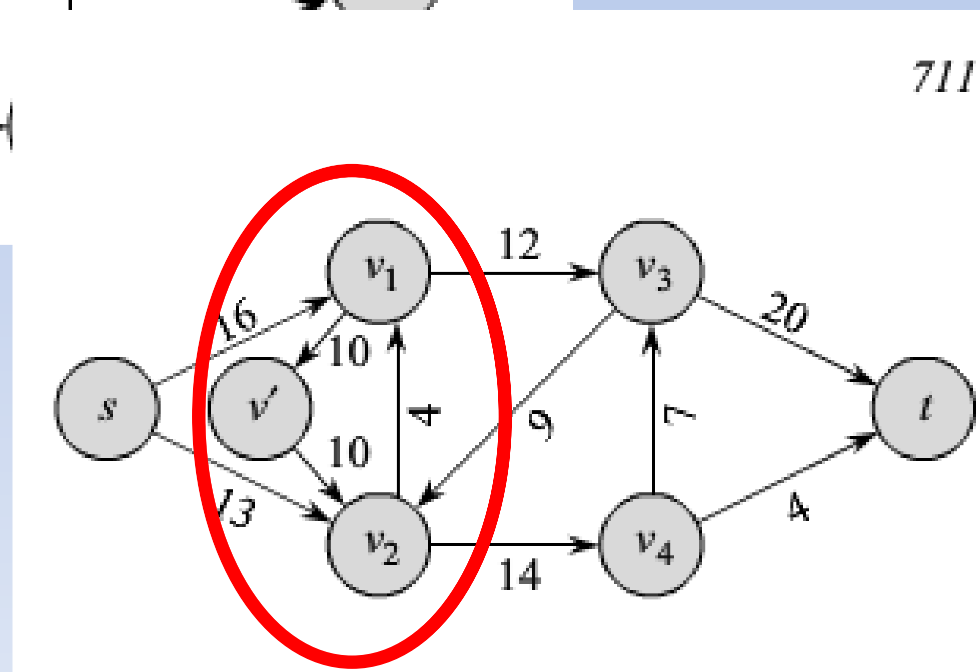


Antiparallel Edges Not Allowed



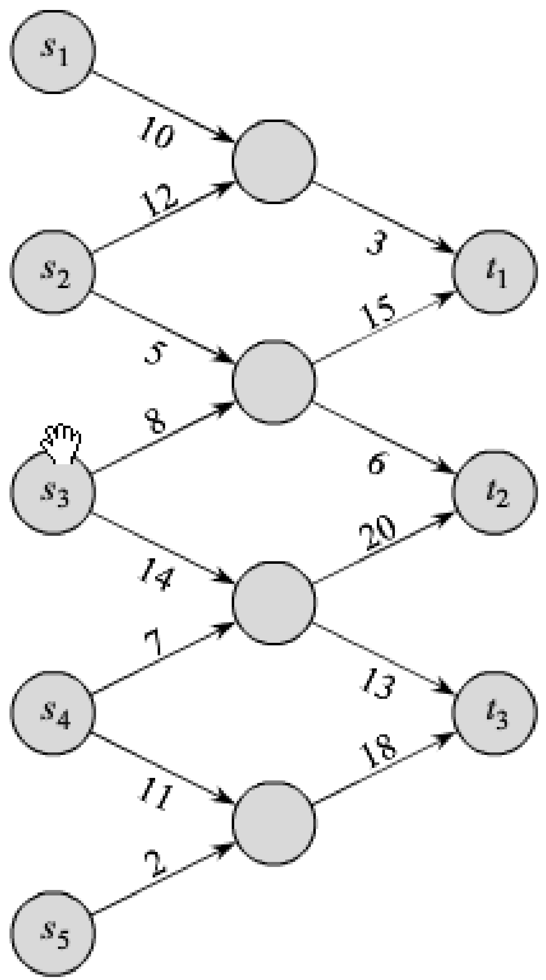
Antiparallel Edges
Not Allowed/Easily Converted





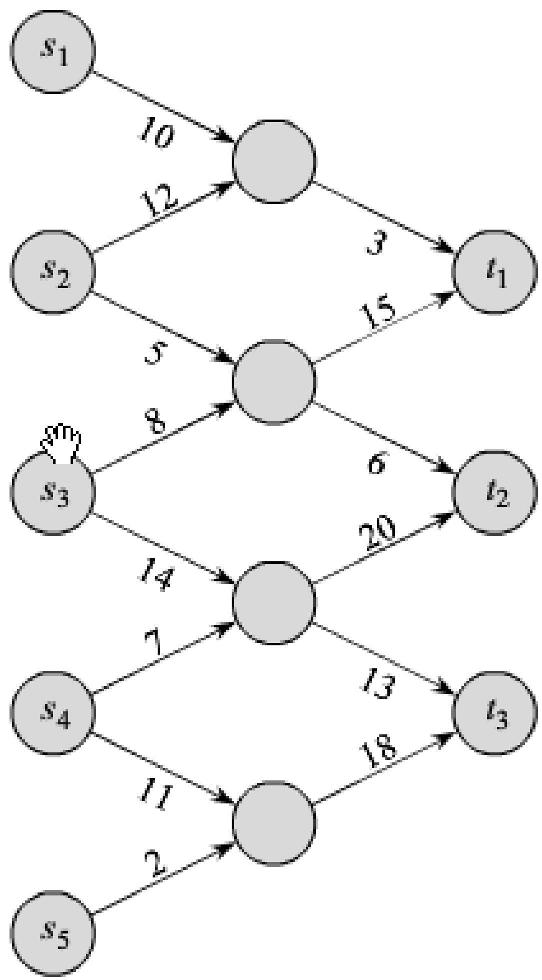
(b)

Multiple-Source/ Multiple-Sink

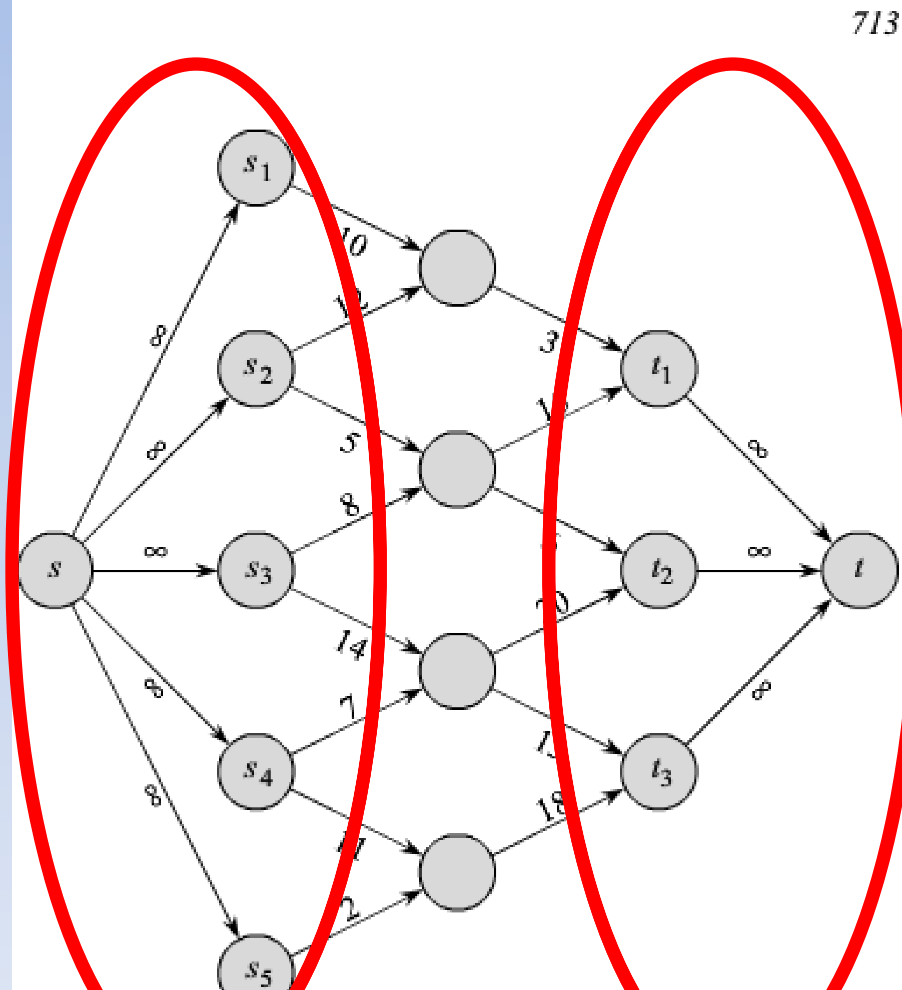


(a)

Multiple-Source/Multiple-Sink Easily Converted



(a)



(b)

Flow Network

- Directed graph G = (V, E) with
  - edge capacities $c(u,v) \geq 0$
  - flows on edges $f(u,v)$
- Capacity Constraint:
  - For all u, v ∋ V, $0 \leq f(u,v) \leq c(u,v)$
  - Flow is greater than 0
  - Flow is less than capacity

Flow Network

- Directed graph G = (V, E) with
  - edge capacities $c(u,v) \geq 0$
  - flows on edges $f(u,v)$
- Capacity Constraint:
  - For all u, v ∋ V, $0 \leq f(u,v) \leq c(u,v)$
  - Flow is greater than 0
  - Flow is less than capacity
- Flow Conservation:

$$\forall_{u \in V - \{s,t\}} \sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$$

  - Flow into Vertex equals Flow out

Applications

- fluid in pipes
- current in an electrical circuit
- traffic on roads
- data flow in a computer network
- money flow in an economy
- etc.

Maximum Flow Problem

- Assuming
  - Source produces the material at a steady rate
  - Sink consumes the material at a steady rate

- What is the maximum net flow from s to t?

Residual Capacity

- Given a flow f in network $G = (V, E)$
- Consider a pair of vertices $u, v \in V$
- Residual capacity =

  amount of additional flow we can push from u to v

  ○ $c_f\,(u, v) = c(u, v) - f\,(u, v) \geq 0$ | $IF\ (u, v) \in E$

  Since $f\,(u, v) \leq c(u, v)$

  ○ $c_f\,(u, v)$ = f(v, u) | IF (v, u) ϵ E

  Represents potential reduction in opposite flow.

  AND no parallel edges

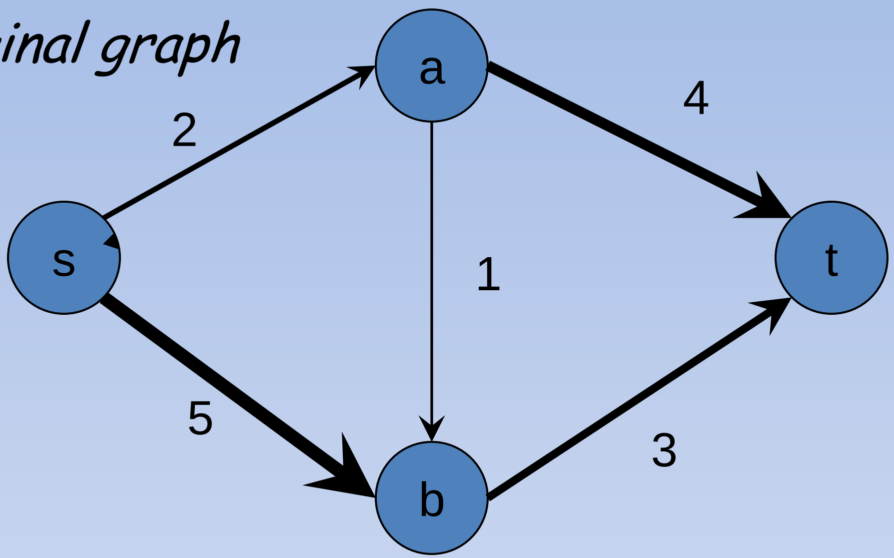
Residual Capacity

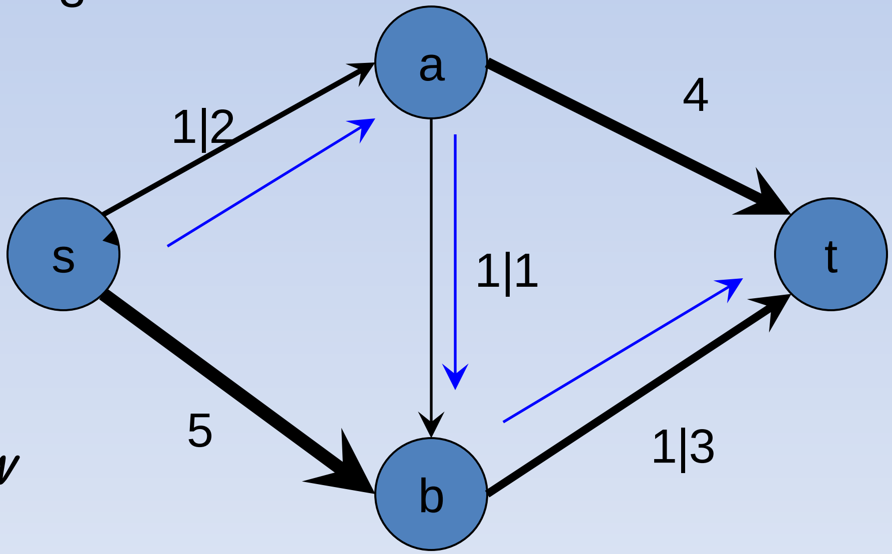
- Residual network induced by flow f is defined as

- $G_f = (V, E_f)$
  $E_f = \{ (u, v) \in V \times V \;\; | \;\; c_f(u, v) > 0 \}$

Example (1)

original graph

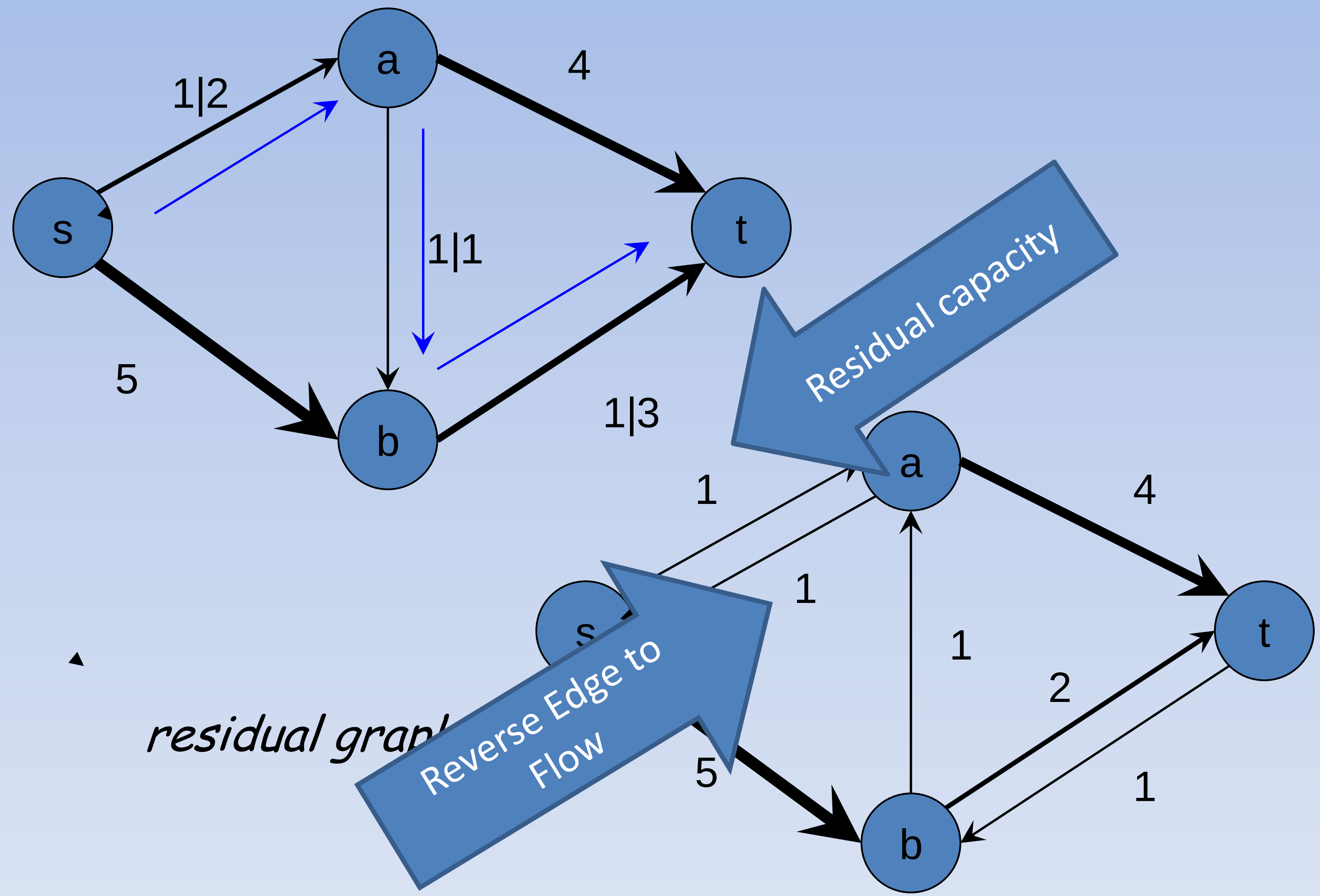




graph with flow

Example (2)

graph with flow



Augmentation

- Given:
  - Flow f in G
  - Flow f' in G_f (the Residual Graph from G w/ f)

$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + f'(u,v) - f'(v,u) & \text{if } (u,v) \in E \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{26.4}$$

Augmented Flow is a Flow

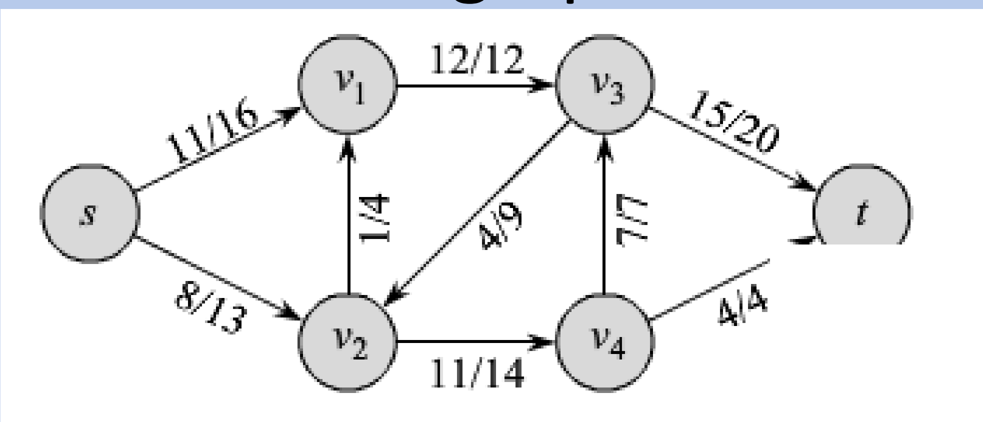
- Given:
  - Flow f in G
  - Flow f' in G_f (the Residual Graph from G w/ f)

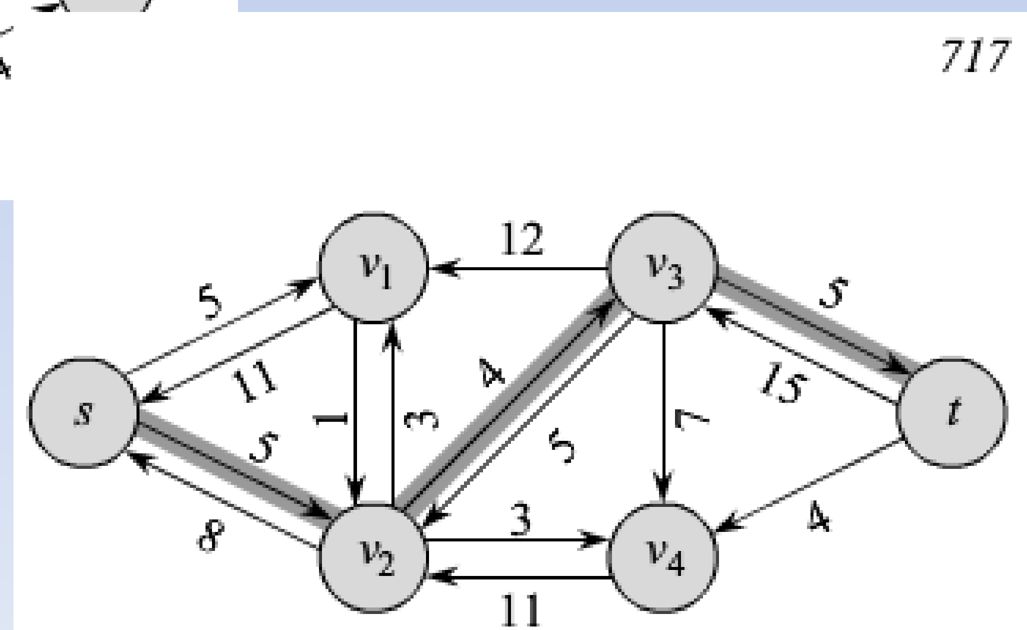
$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + f'(u,v) - f'(v,u) & \text{if } (u,v) \in E\,, \\ 0 & \text{otherwise}\,. \end{cases} \qquad (26.4)$$

Lemma 26.1
Let $G = (V, E)$ be a flow network with source s and sink t, and let f be a flow in G. Let G_f be the residual network of G induced by f, and let f' be a flow in G_f. Then the function $f \uparrow f'$ defined in equation (26.4) is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

Augmentation Path

- Augmentation Path is a simple path in the residual graph





Augmentation Path

- Augmentation Path is a simple path in the residual graph

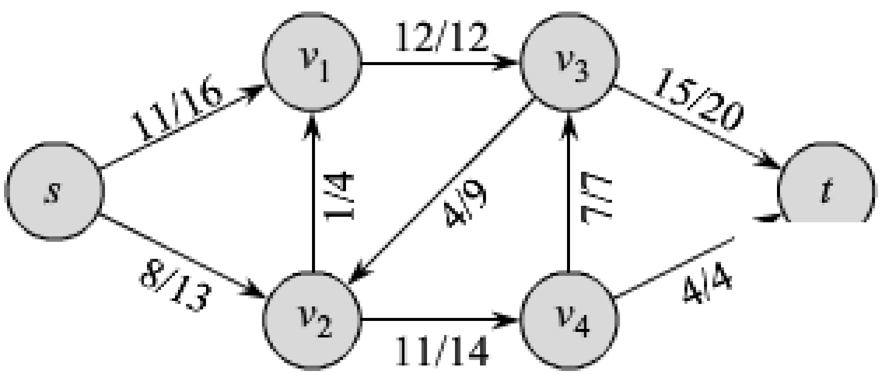
- Residual Capacity is the min residual capacity of any link in the Augmentation Path

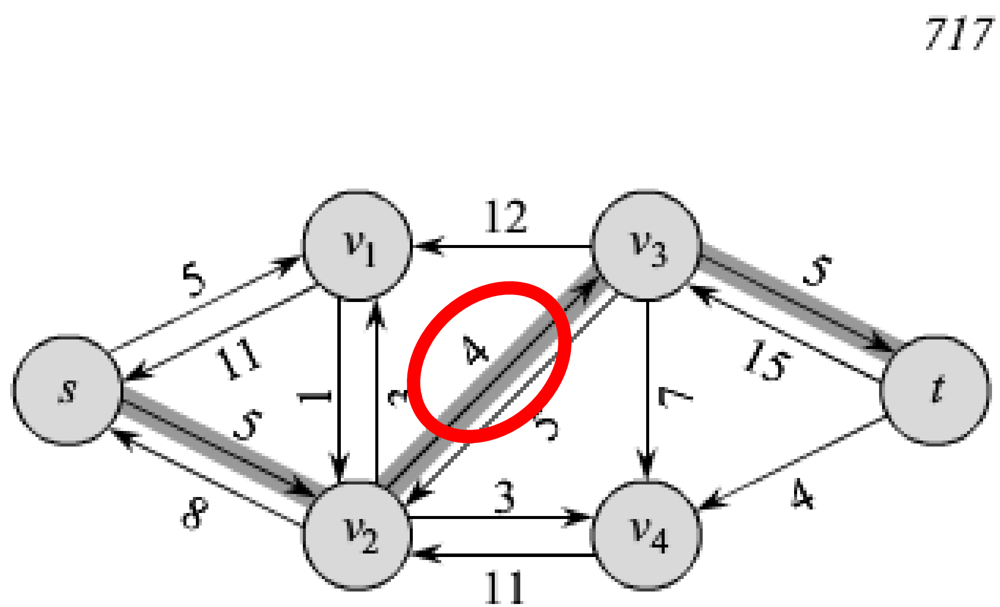
$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$$

Augmentation Path

- Residual Capacity is the min residual capacity of any link in the Augmentation Path

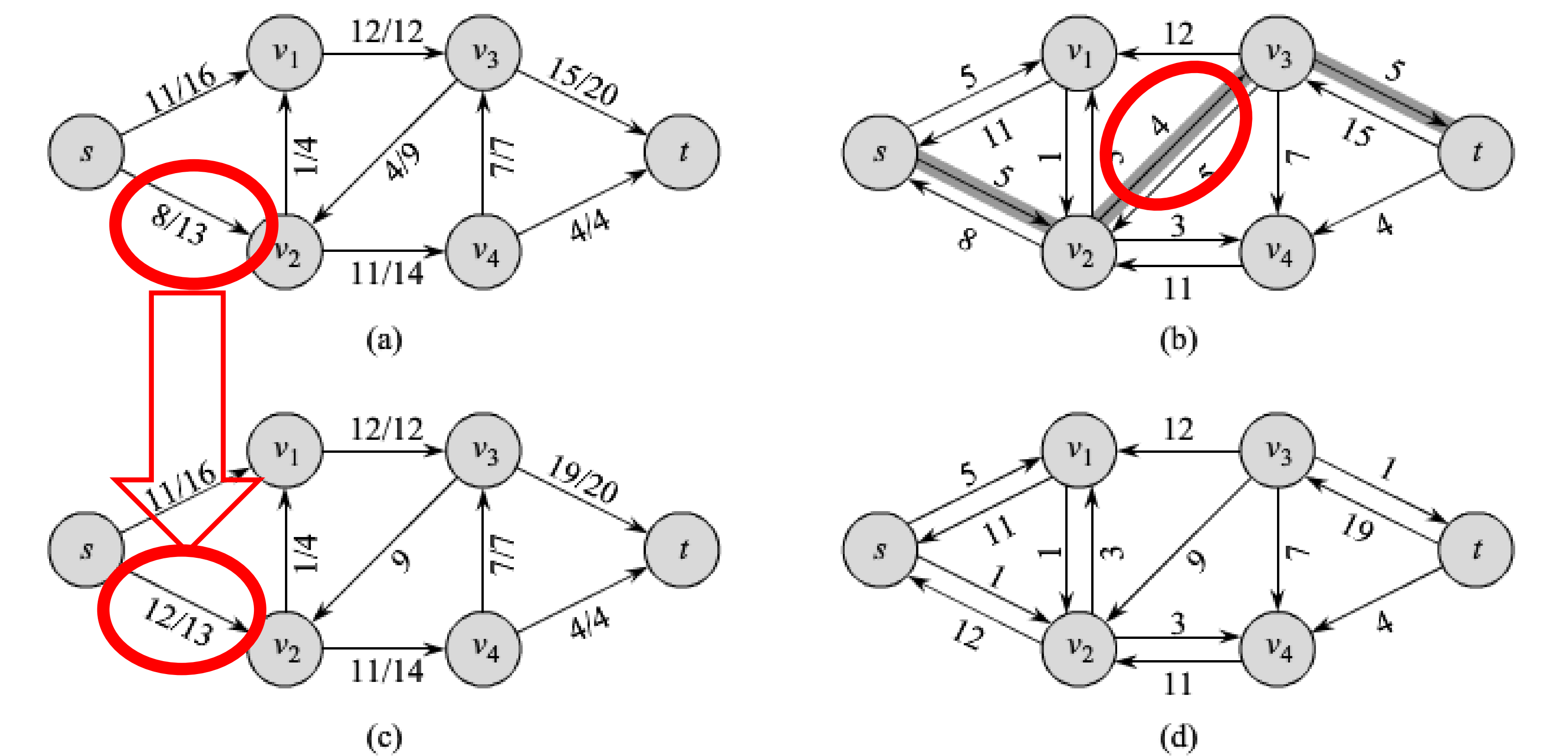
$$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on } p\}$$





Augmentation

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{if } (u, v) \in E , \\ 0 & \text{otherwise} . \end{cases} \tag{26.4}$$

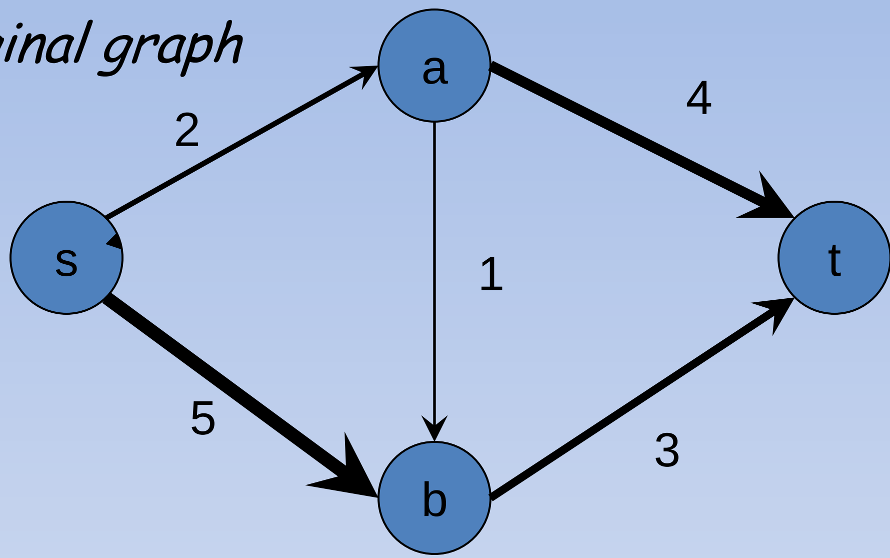


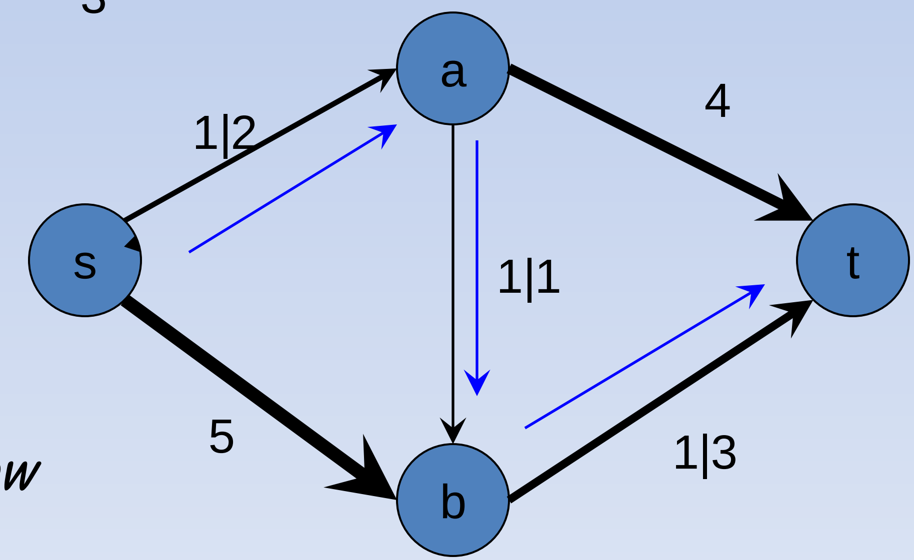
Ford-Fulkerson Algorithm

- Start with zero flow
- Repeat until convergence:
  - Find an augmenting path, from s to t along which we can push more flow
  - Augment flow along this path

Example (1)

original graph

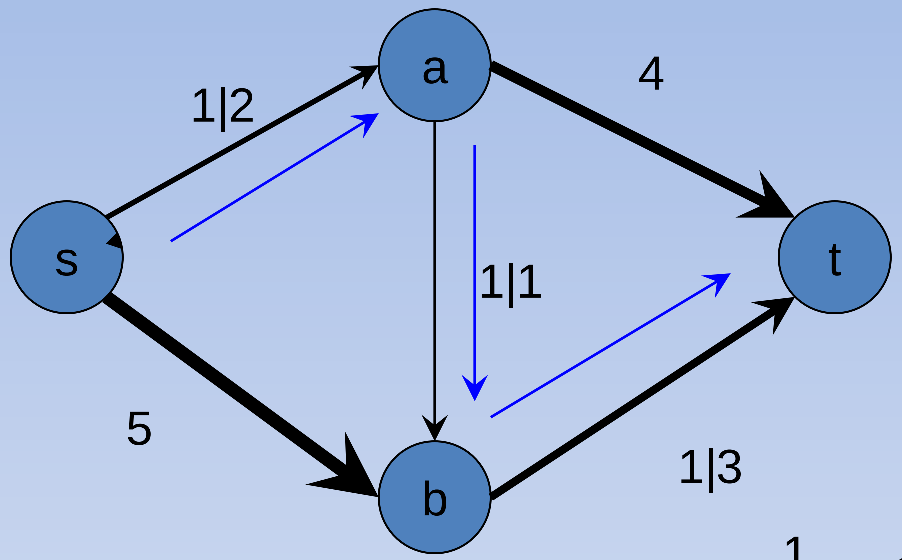




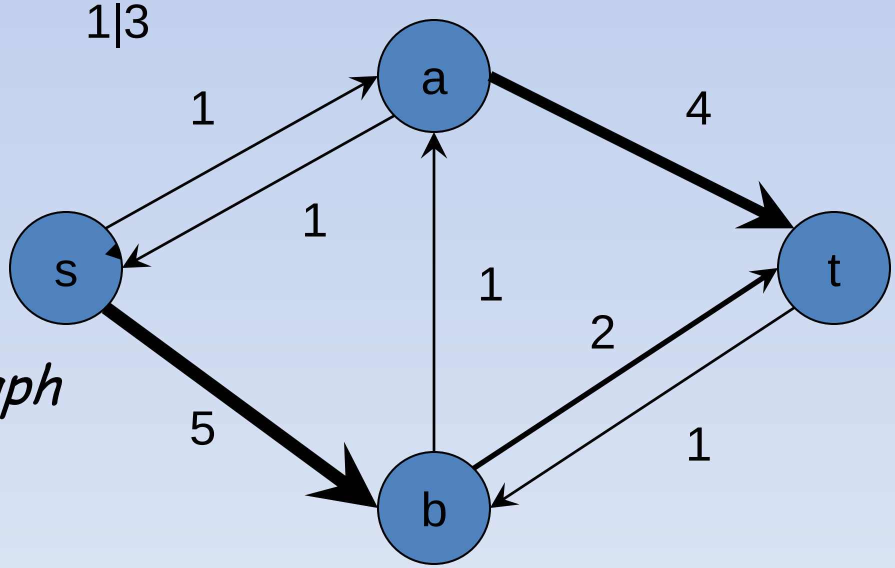
graph with flow

Example (2)

graph with flow

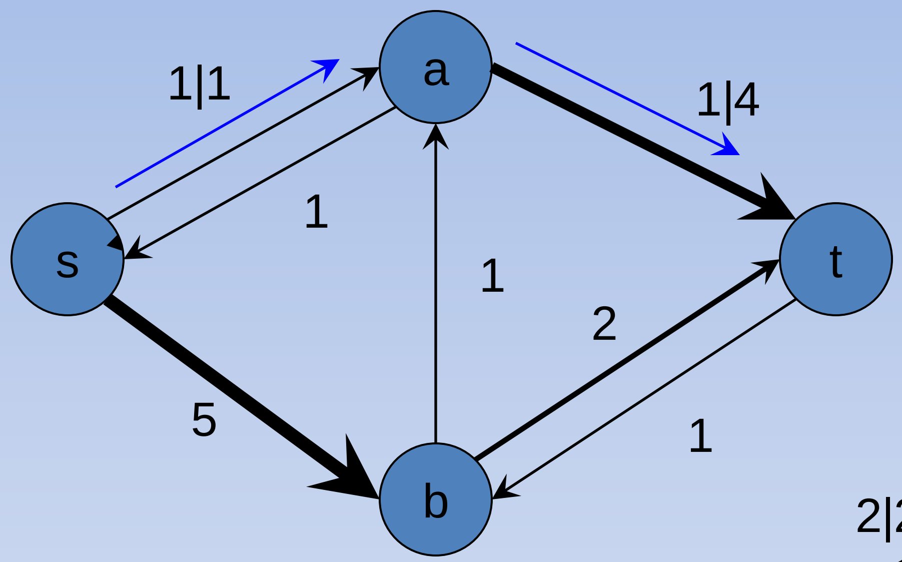


residual graph

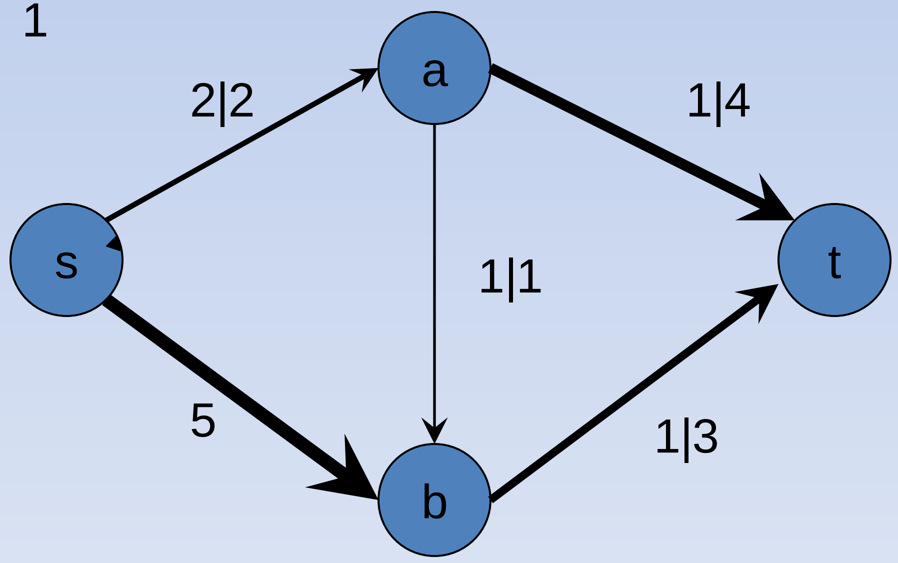


Example (3)

residual graph, with flow-augmenting path

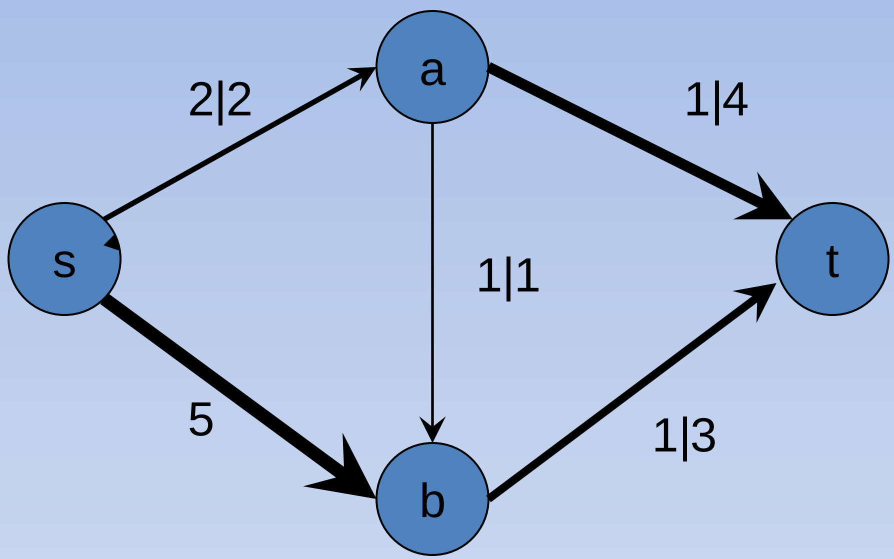


original graph
with new flow

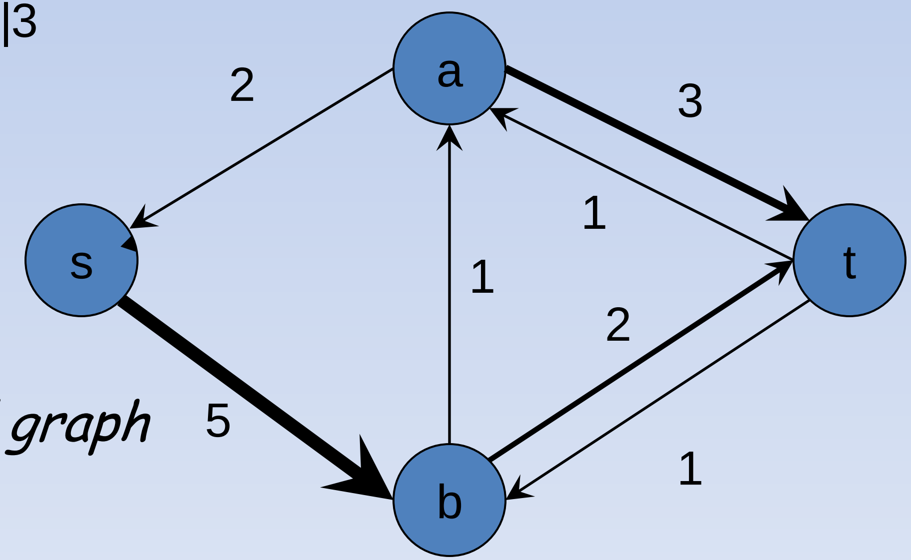


Example (4)

original graph with new flow

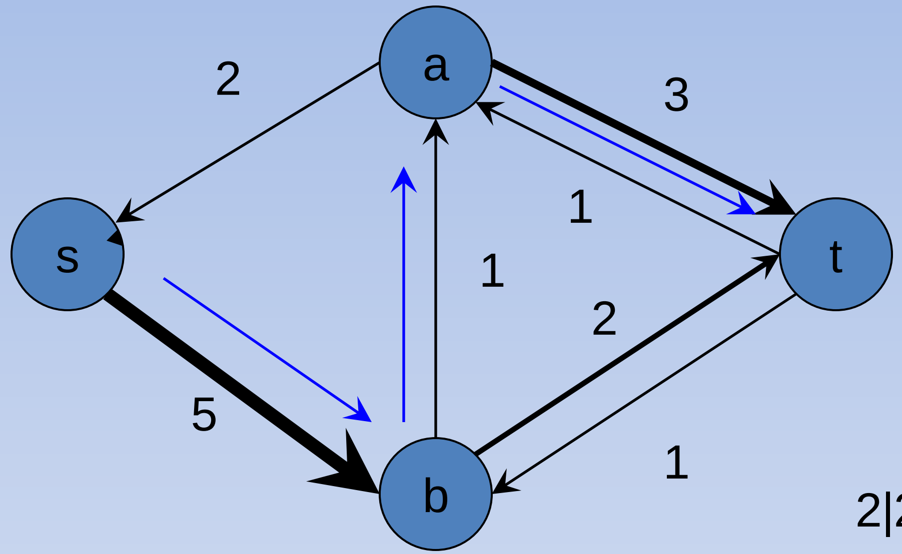


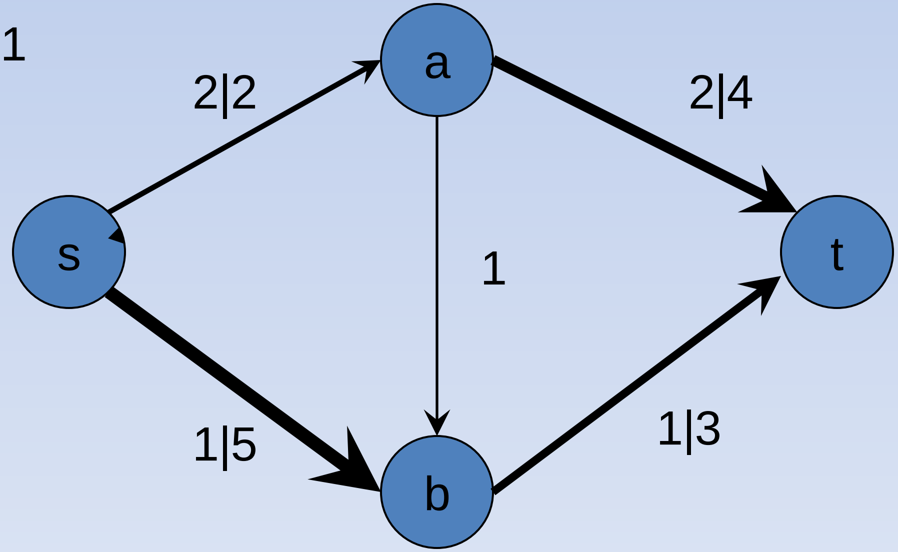
new residual graph



Example (5)

new residual graph, with augmenting path

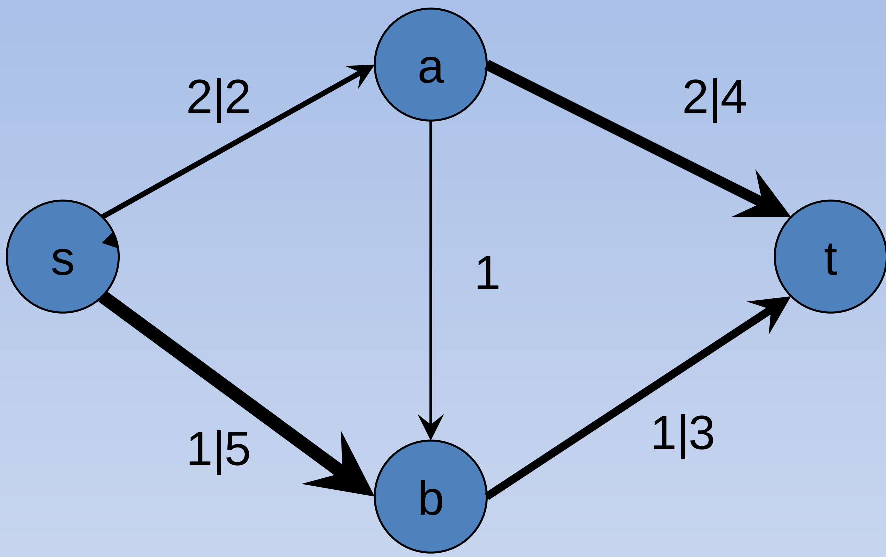




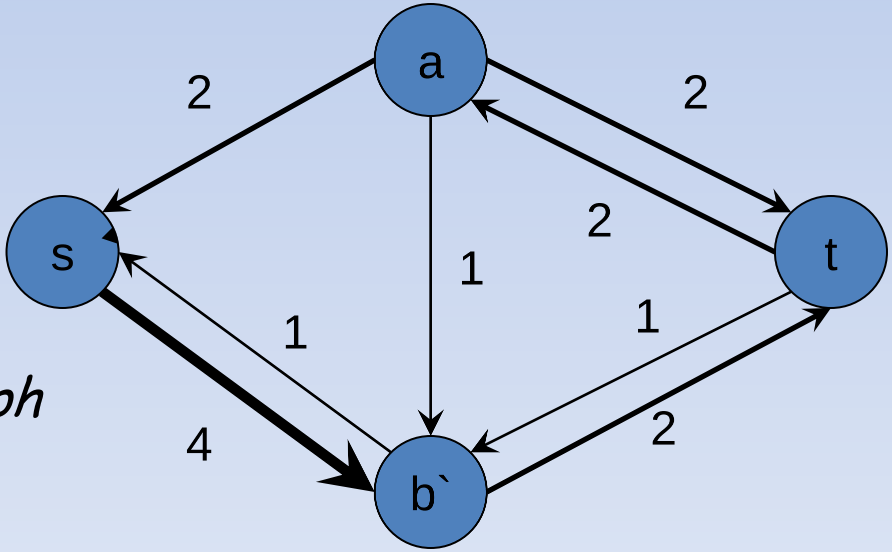
original graph with new flow

Example (6)

original graph with new flow

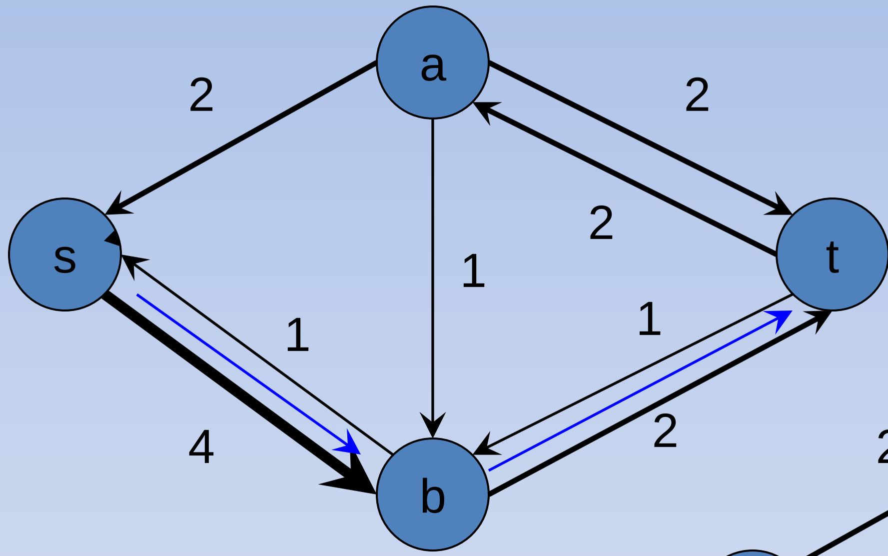


new residual graph

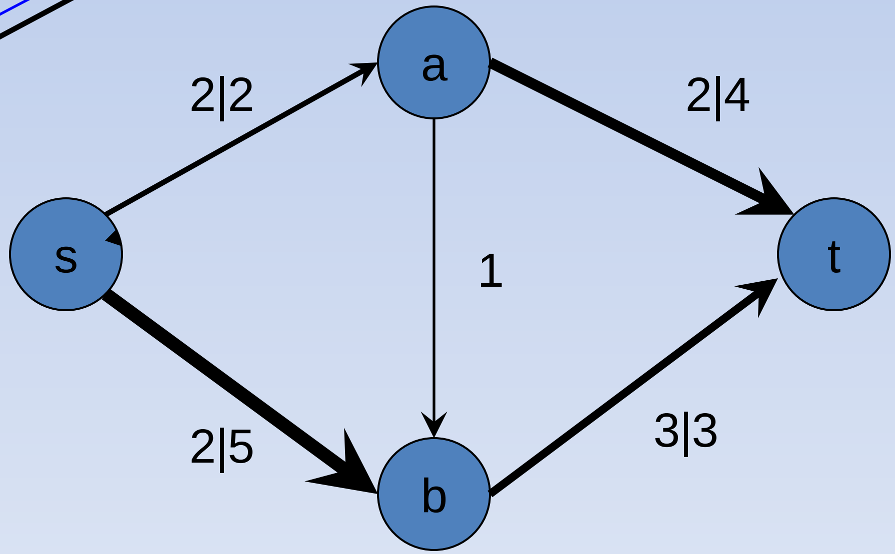


Example (7)

new residual graph, with augmenting path

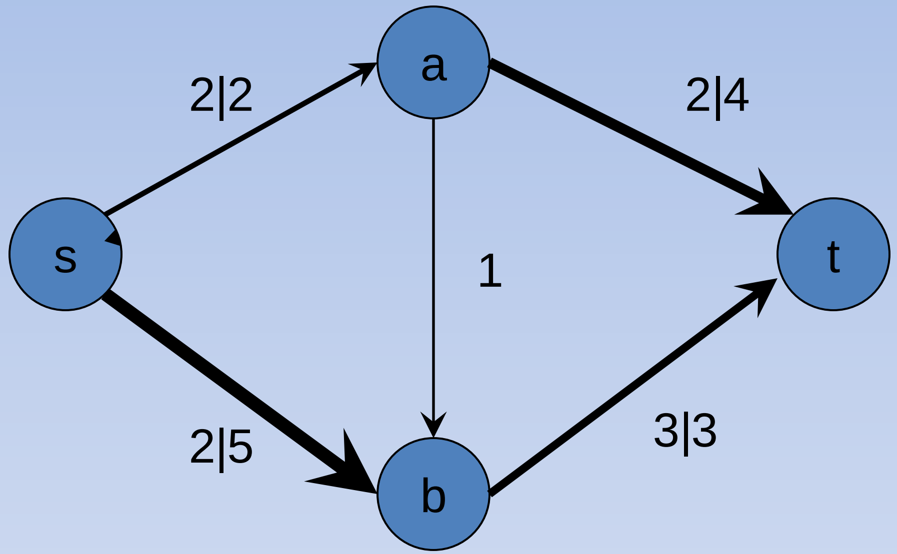


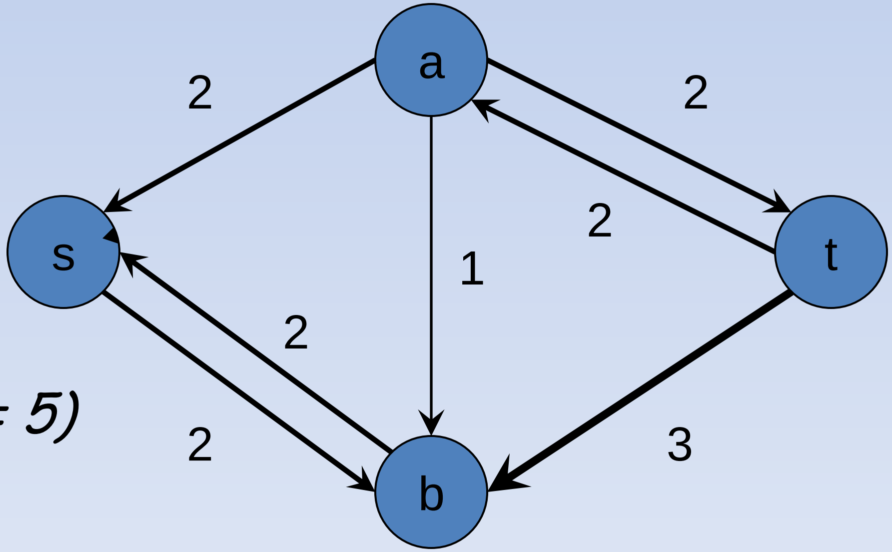
original graph
with new flow



Example (8)

original graph, with new flow

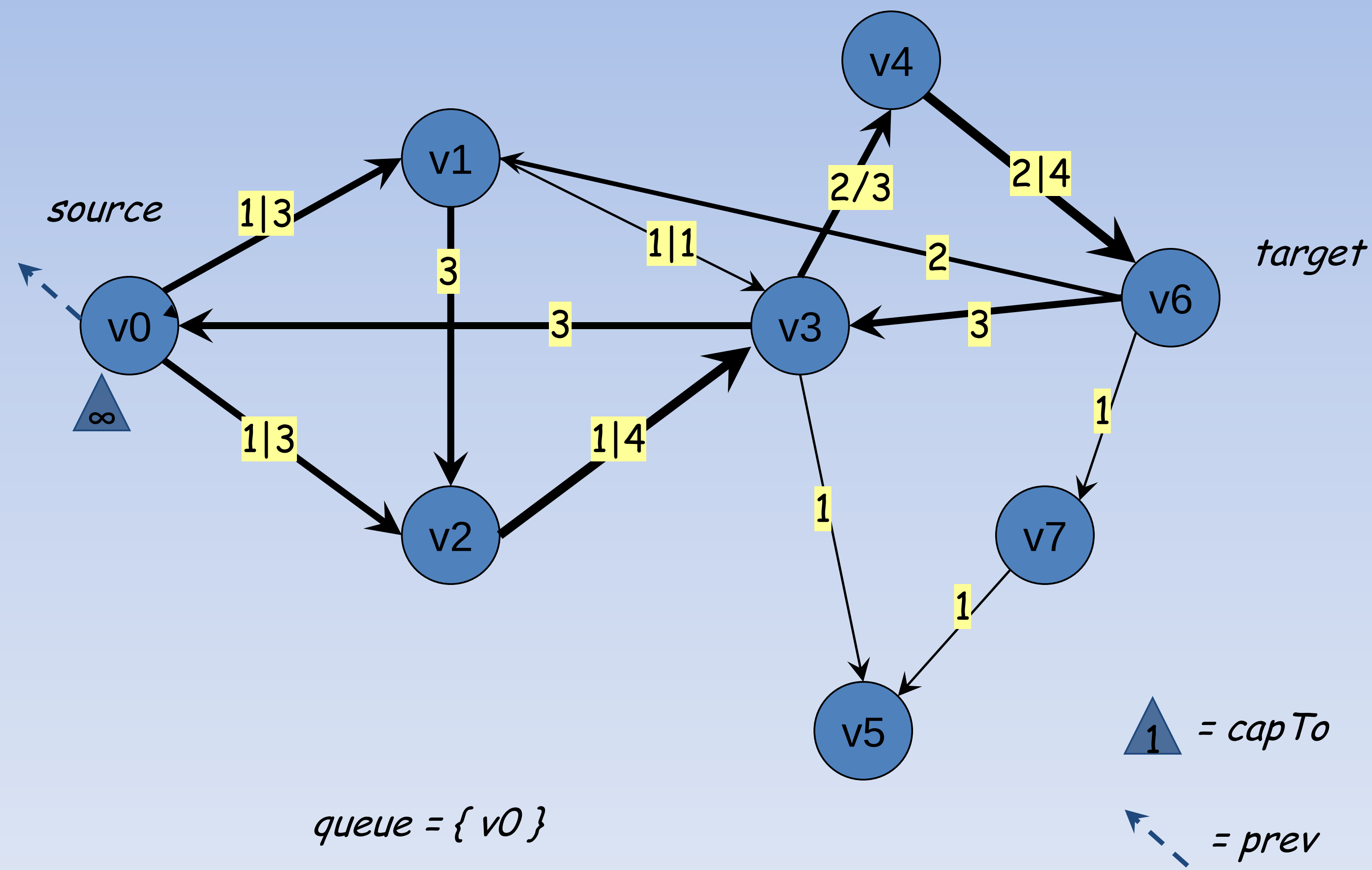




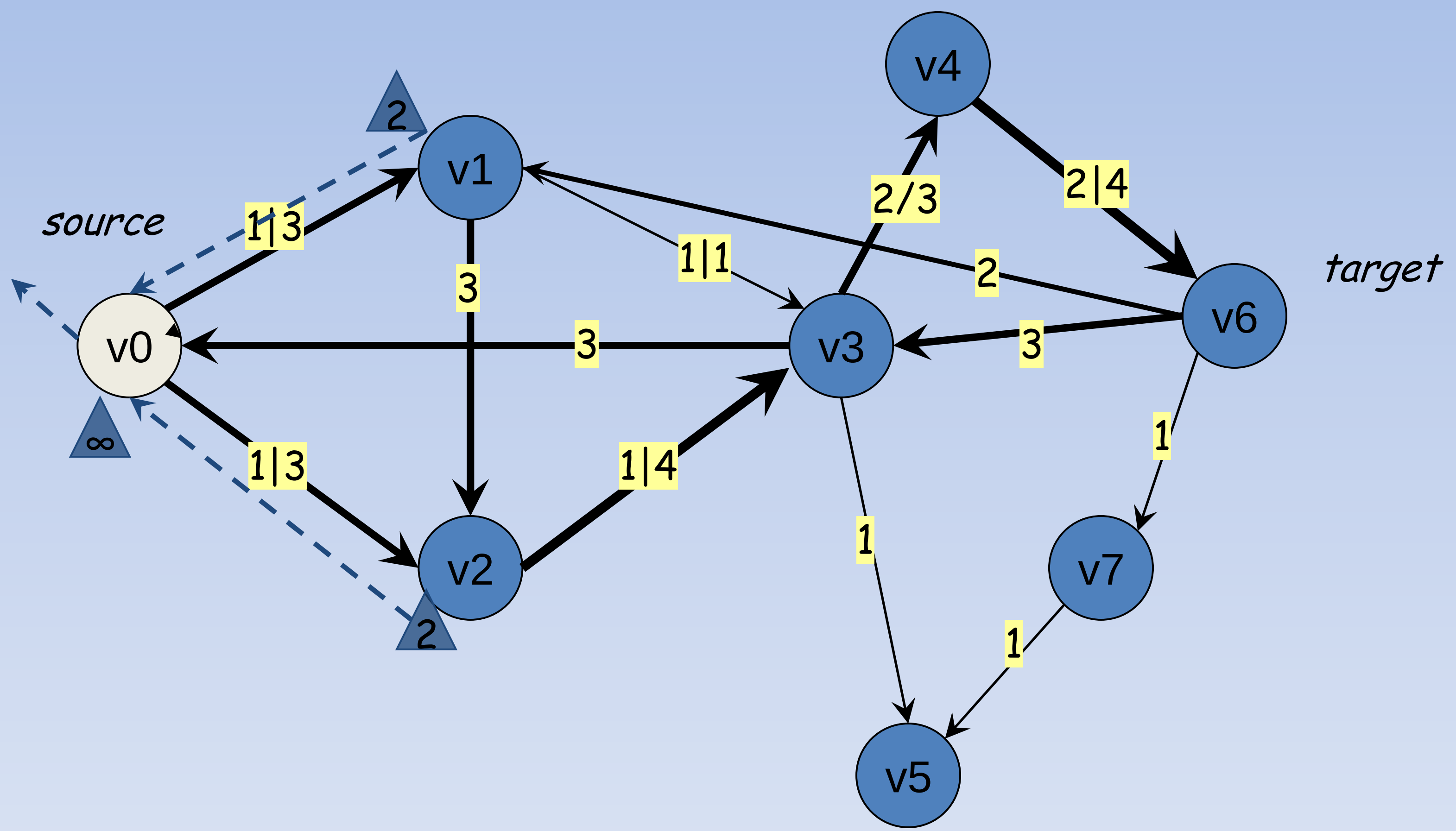
residual graph
(maximum flow = 5)

```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min{c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

Example: Finding Augmenting Path

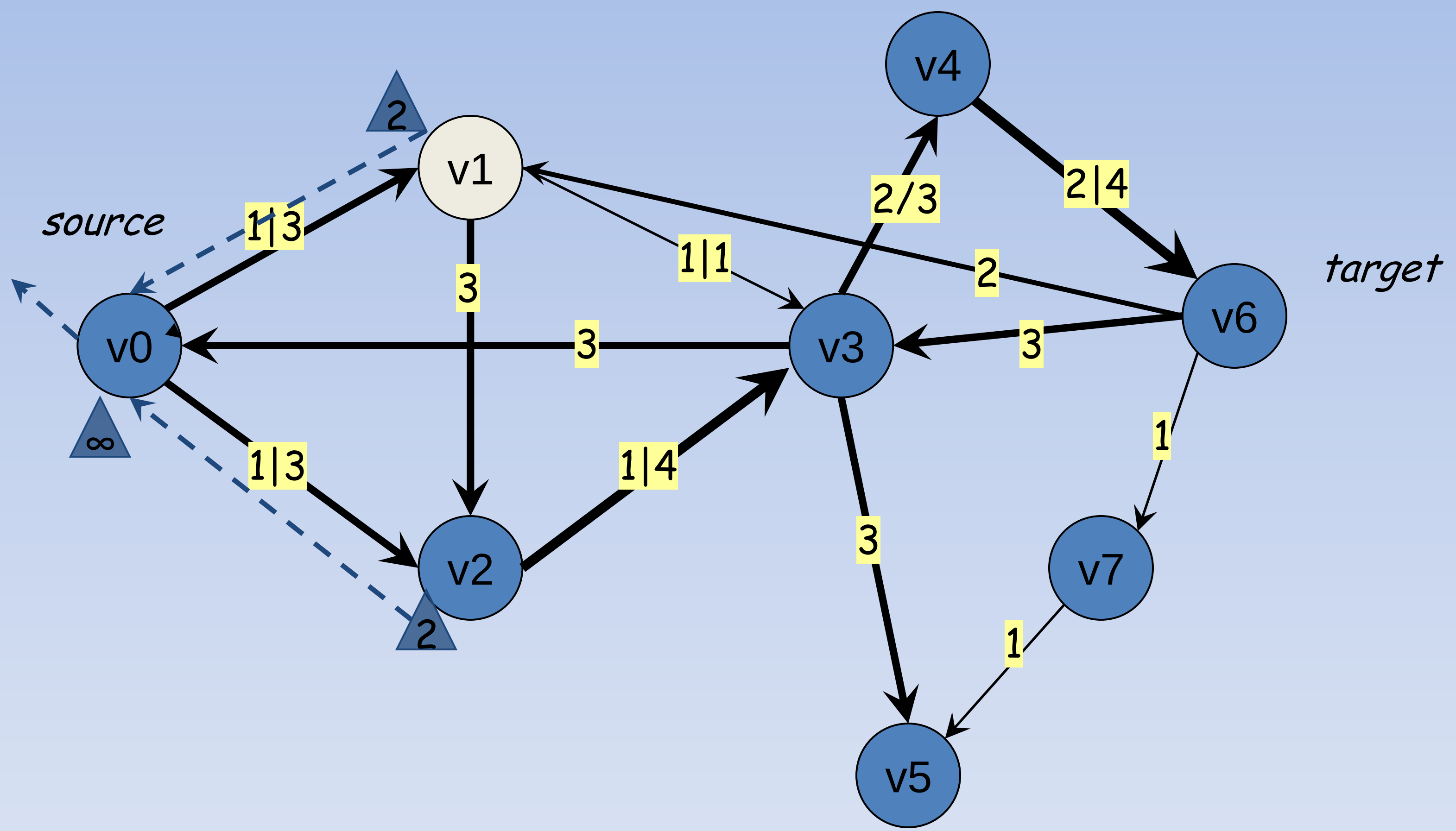


Application to Augmenting Path



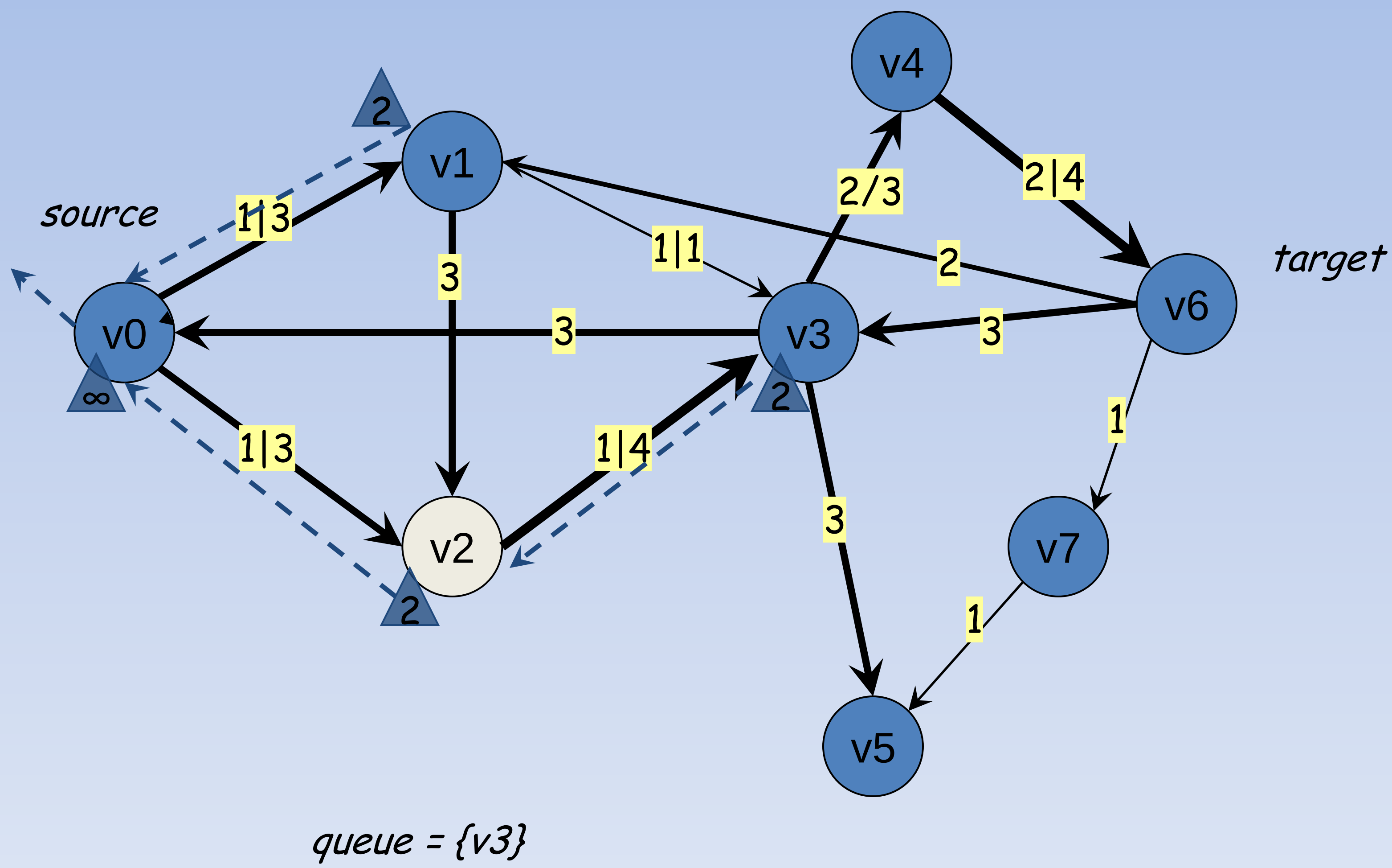
queue = { v1, v2 }

Application to Augmenting Path

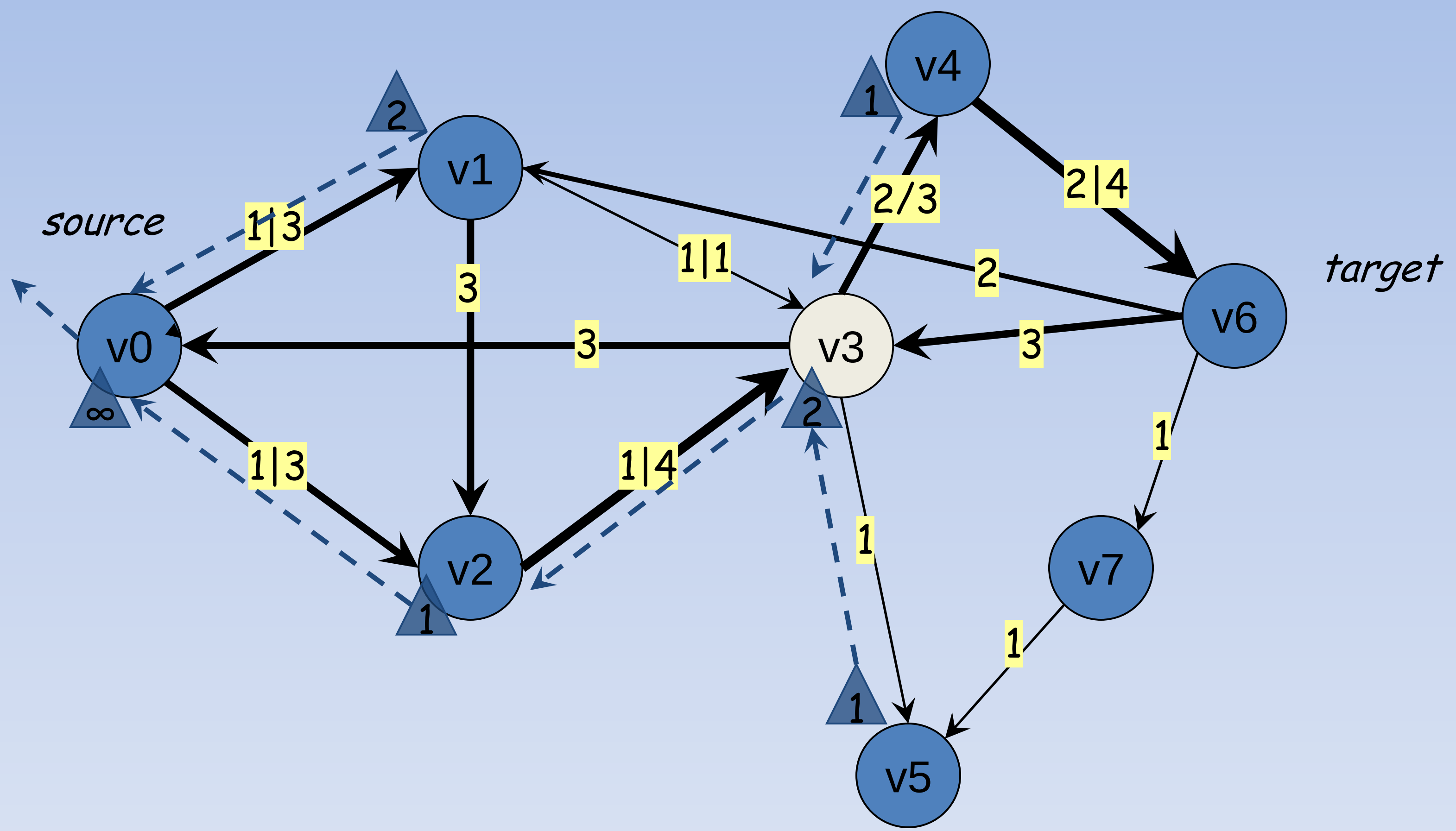


queue = {v2}

Application to Augmenting Path

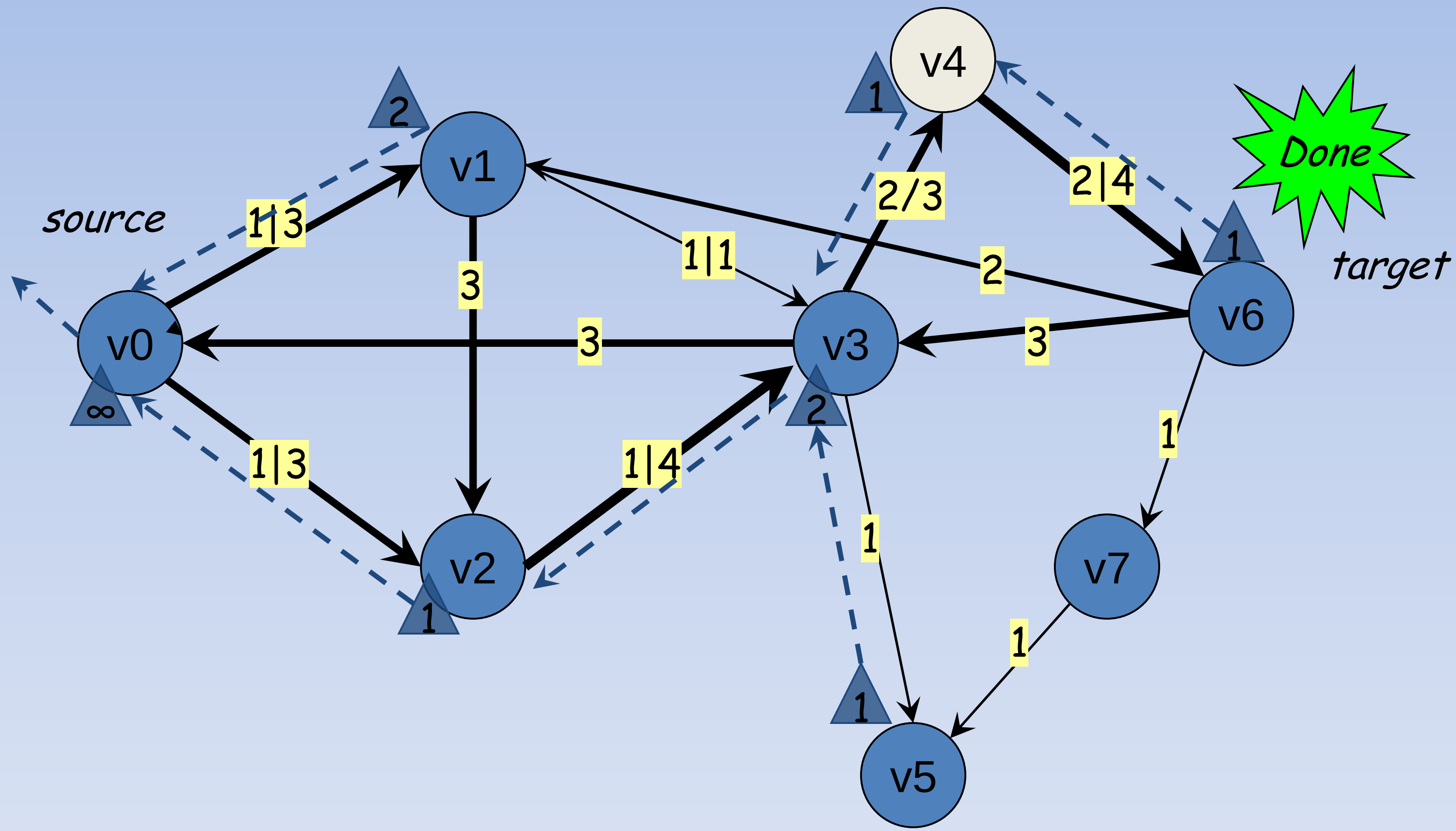


Application to Augmenting Path



queue = {v4, v5}

Application to Augmenting Path

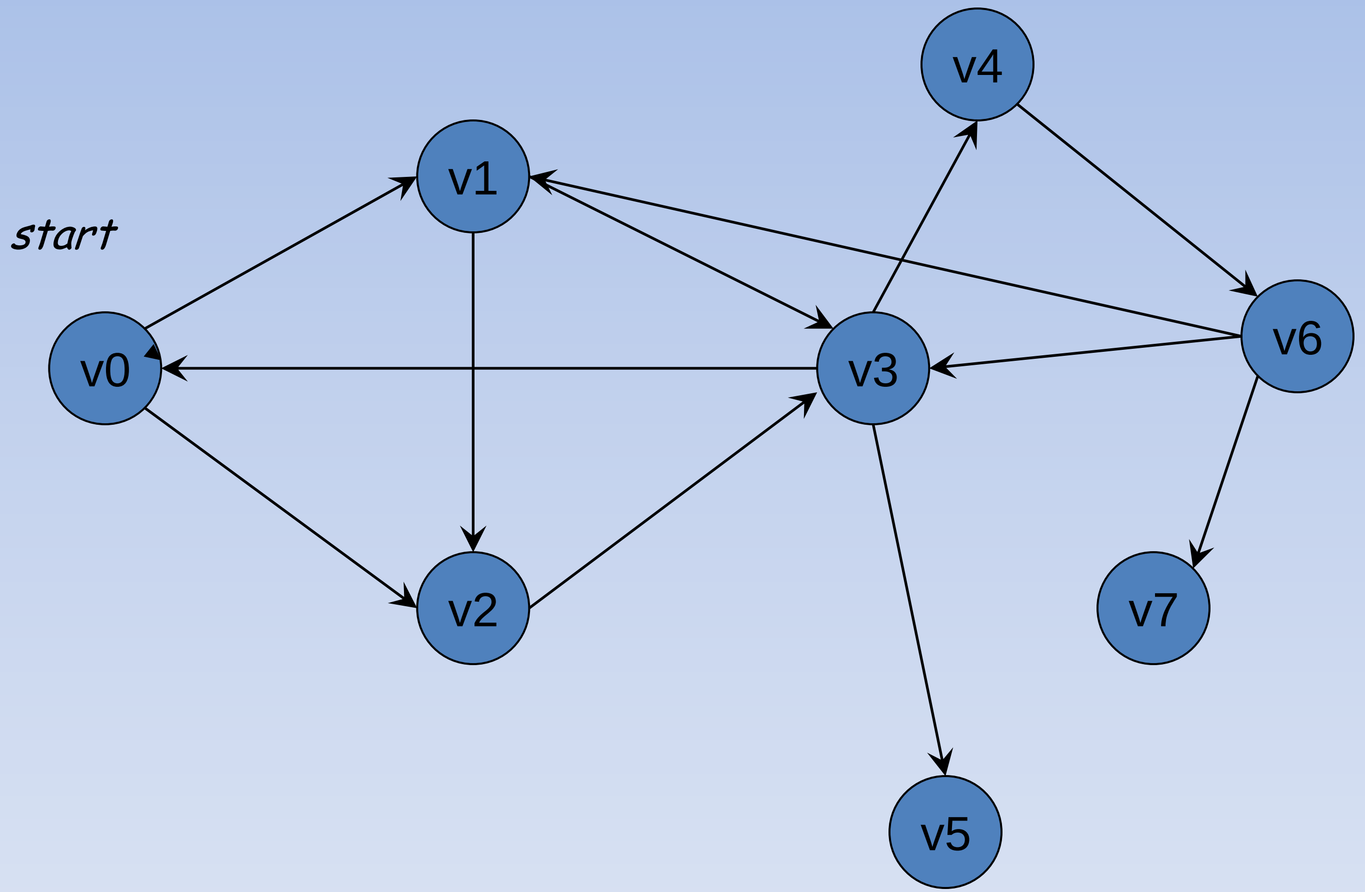


queue = { v5, v6 }

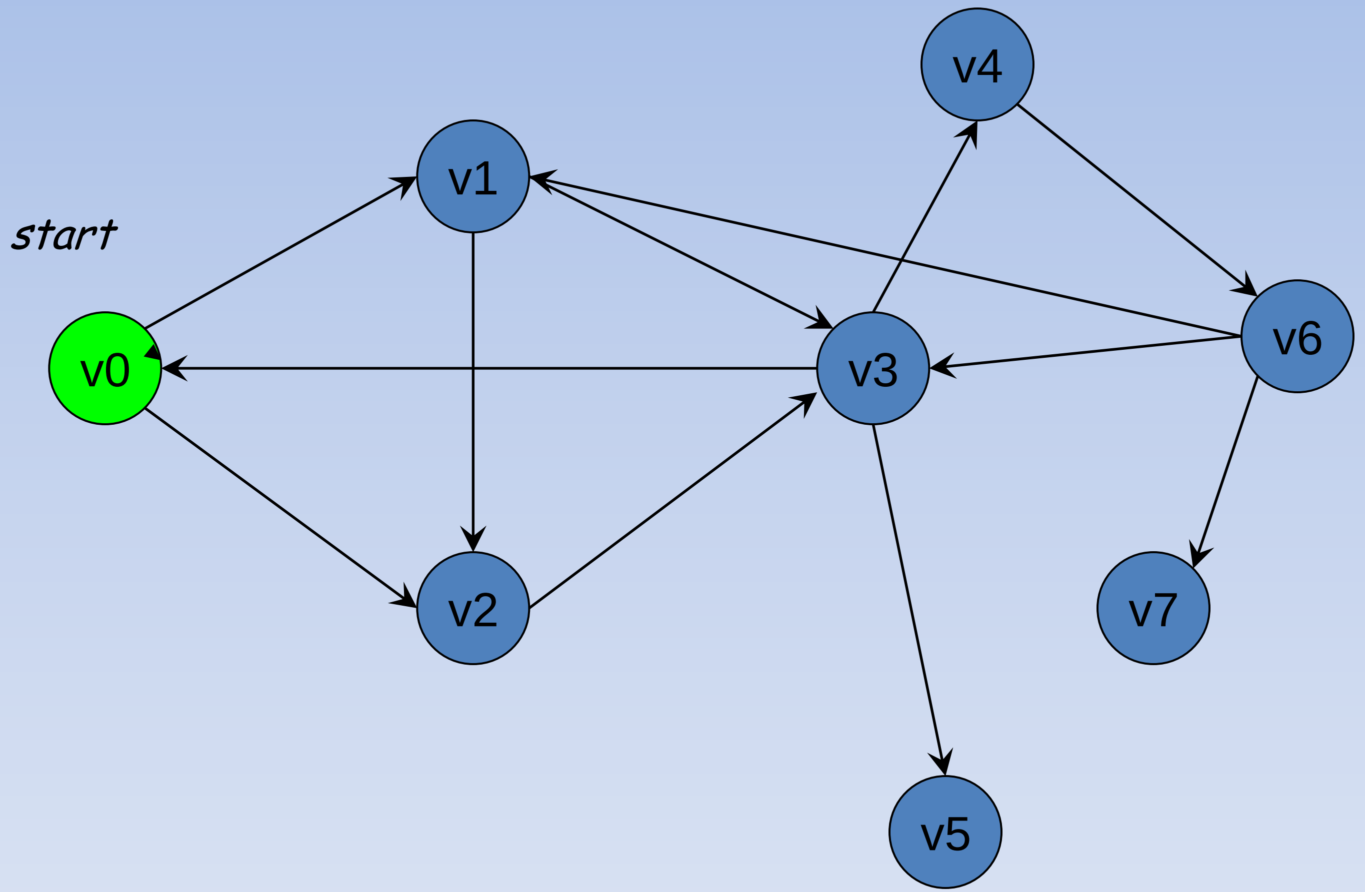
Breadth-first search

- The previous is an example
- Depth-first search is an alternative
- The code is nearly the same
- Only the queuing order differs

Breadth-first Search

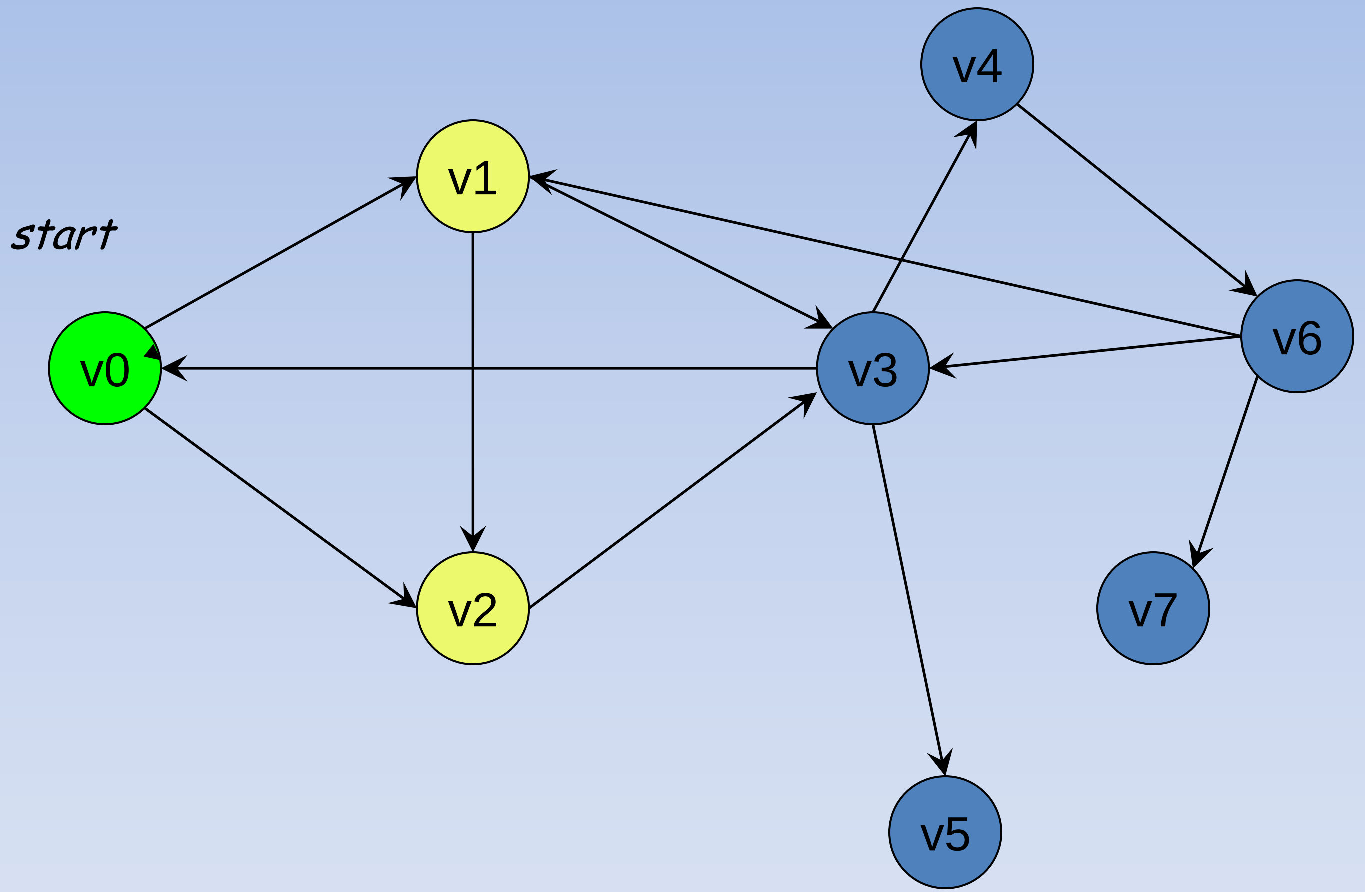


Breadth-first Search



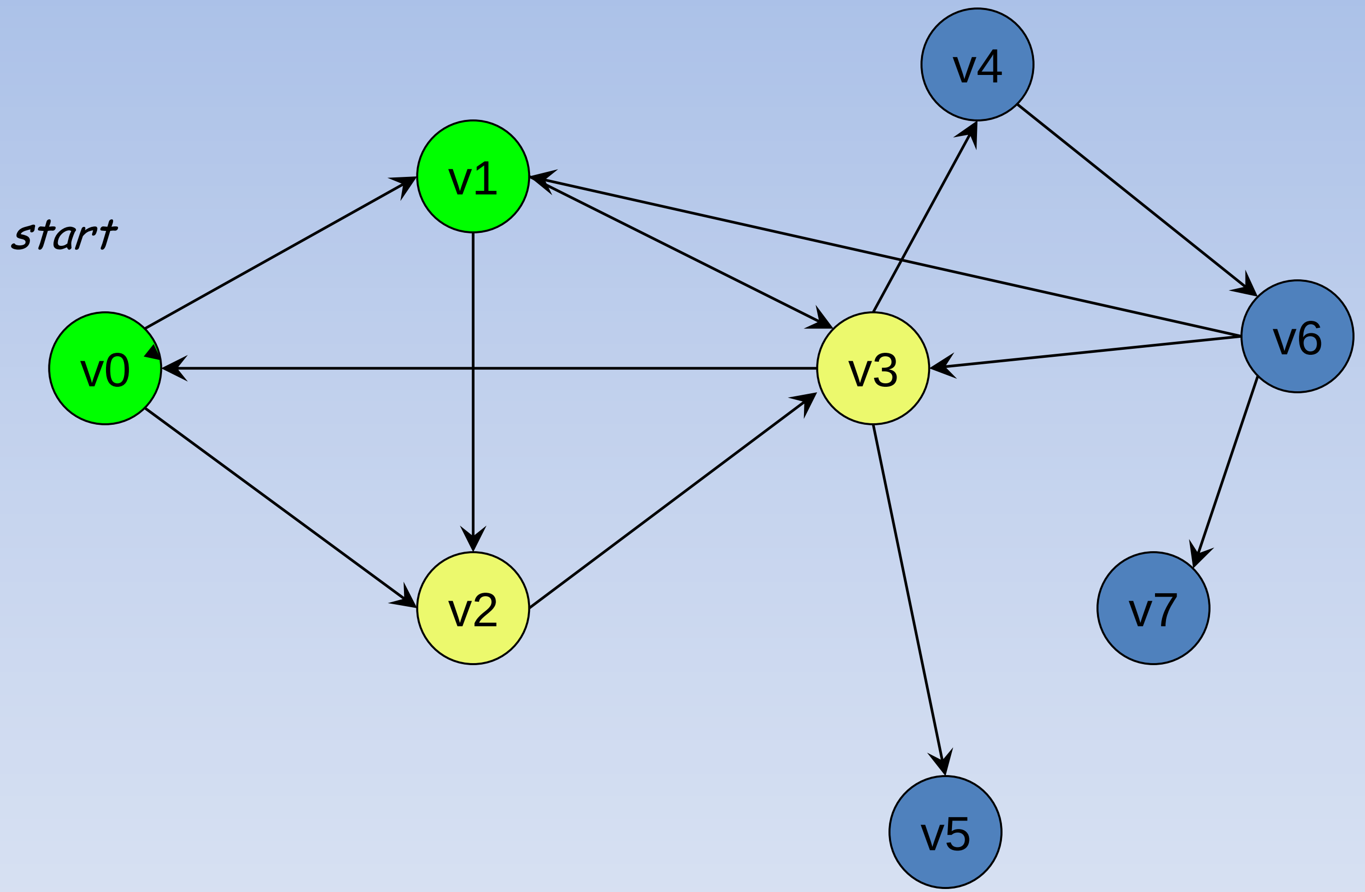
ToVisit = { v0 }

Breadth-first Search



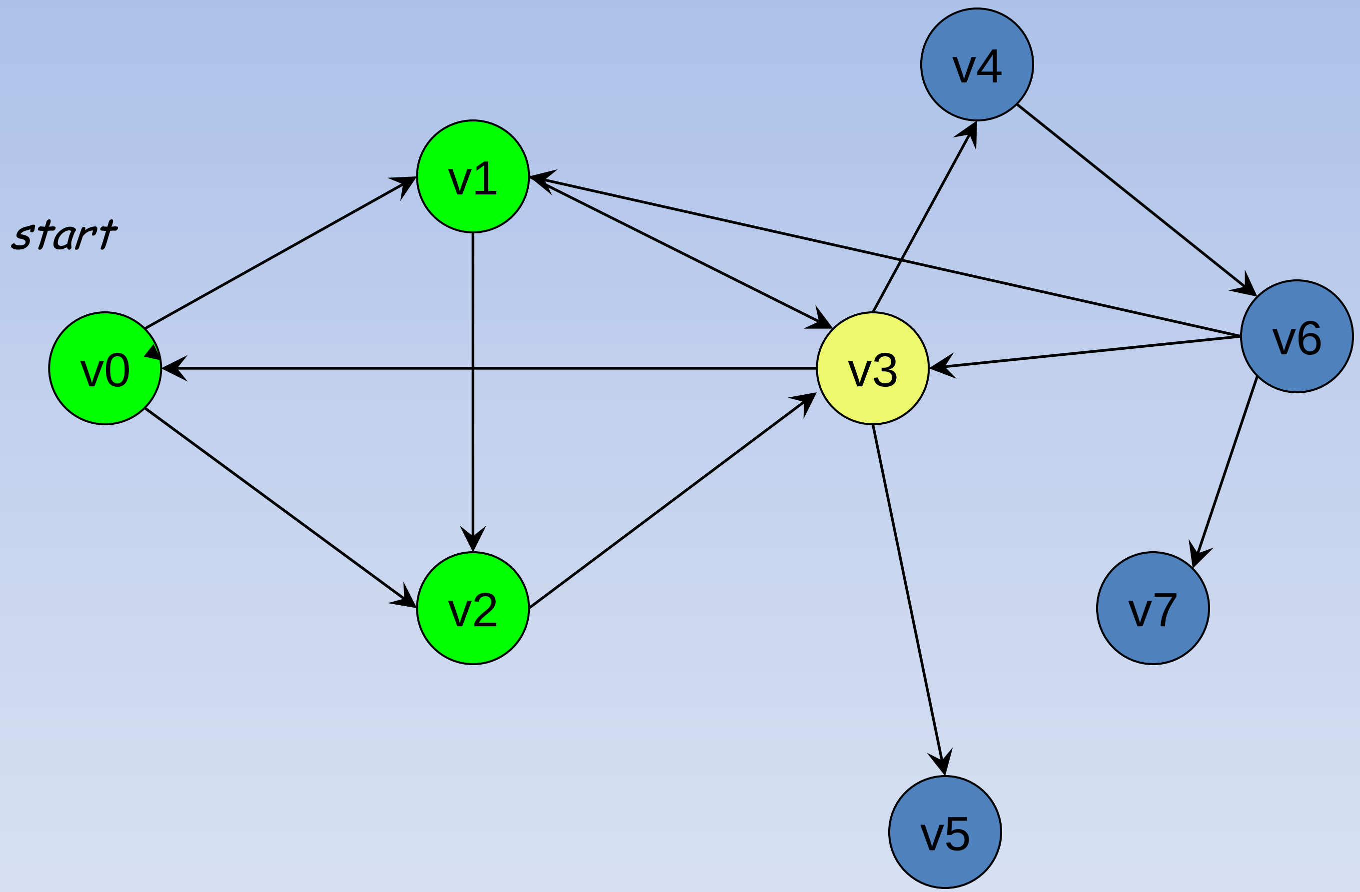
ToVisit = { v1, v2 }

Breadth-first Search



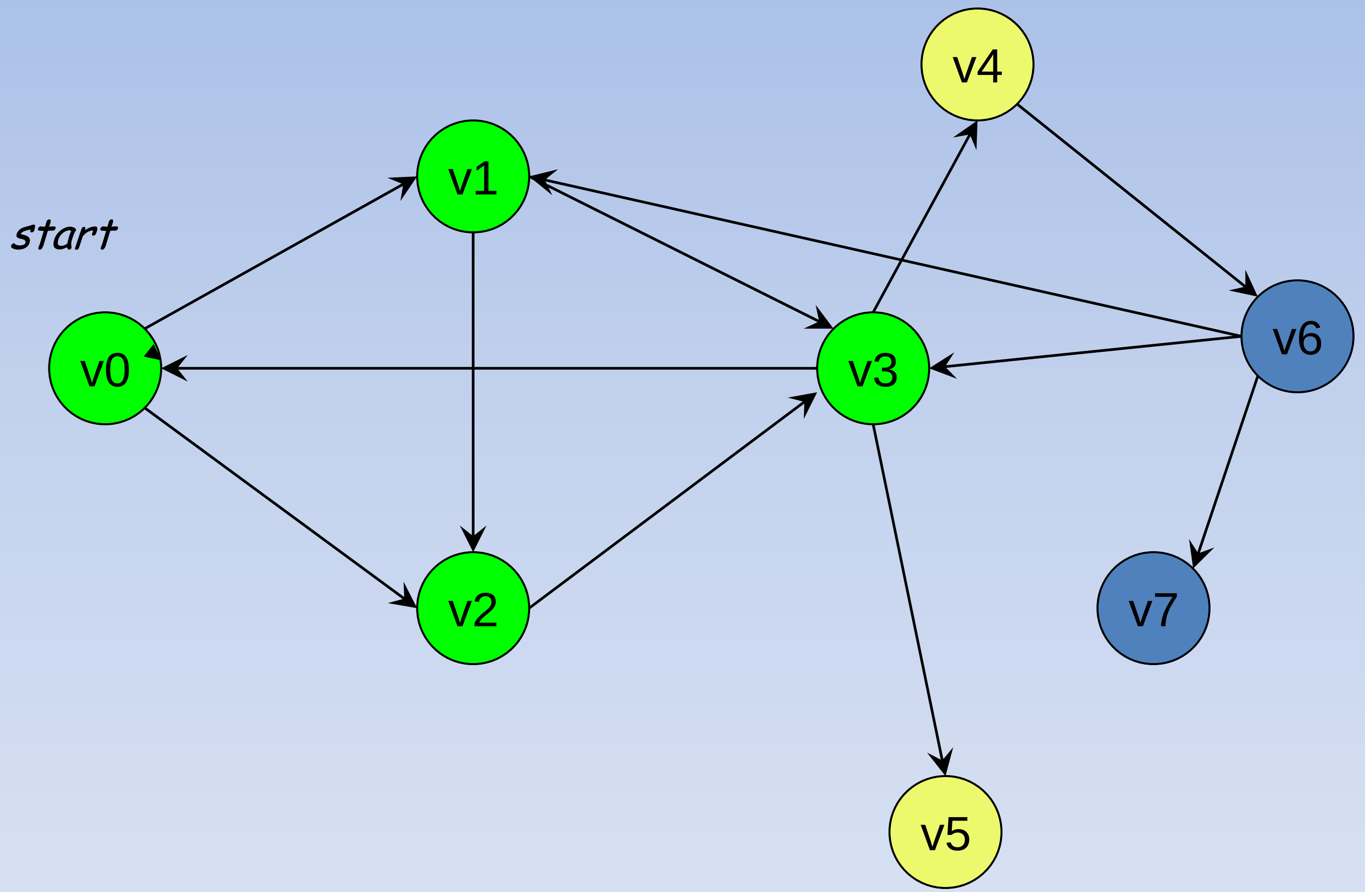
ToVisit = { v2, v3 }

Breadth-first Search



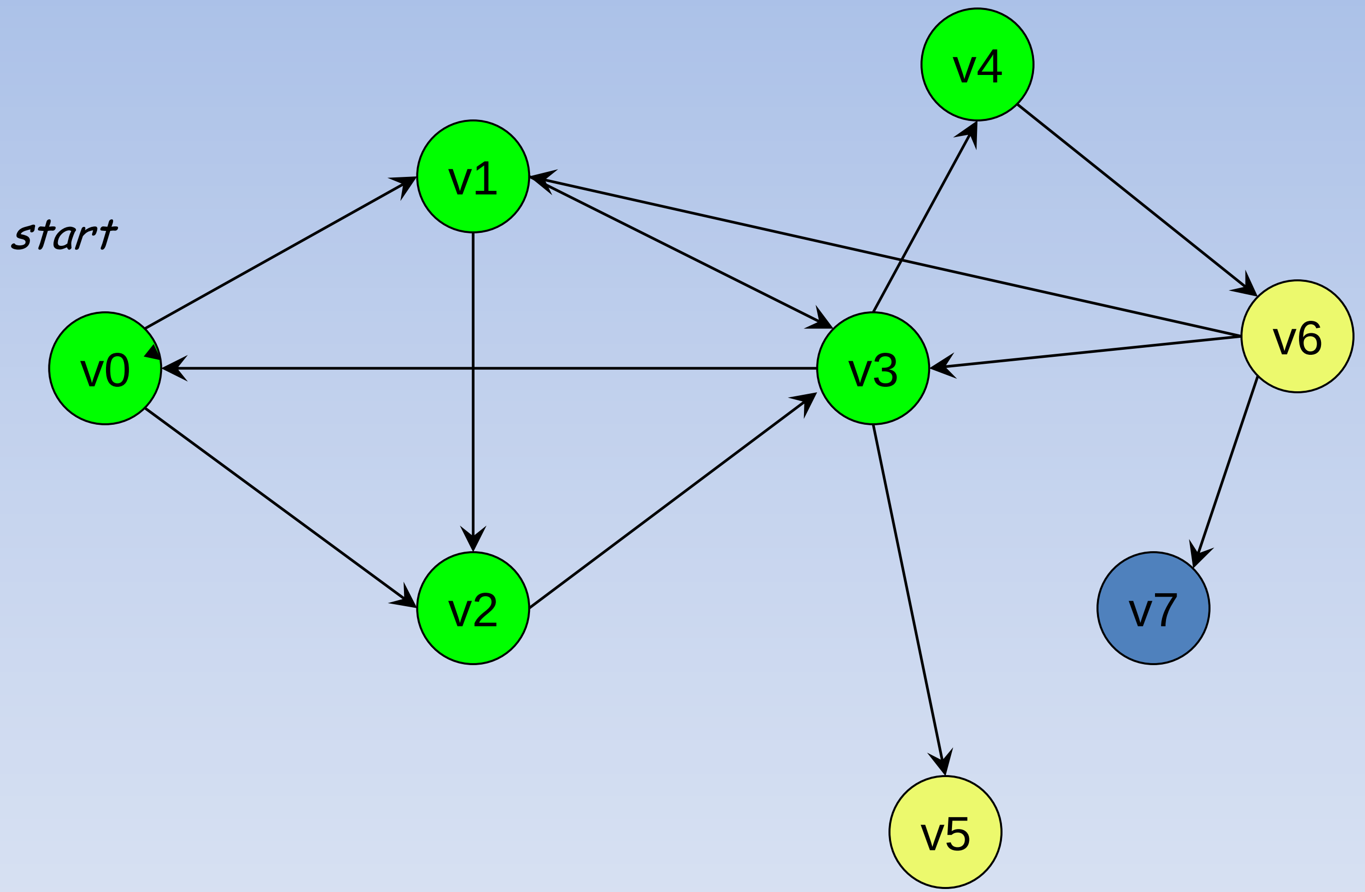
ToVisit = { v2, v3 }

Breadth-first Search



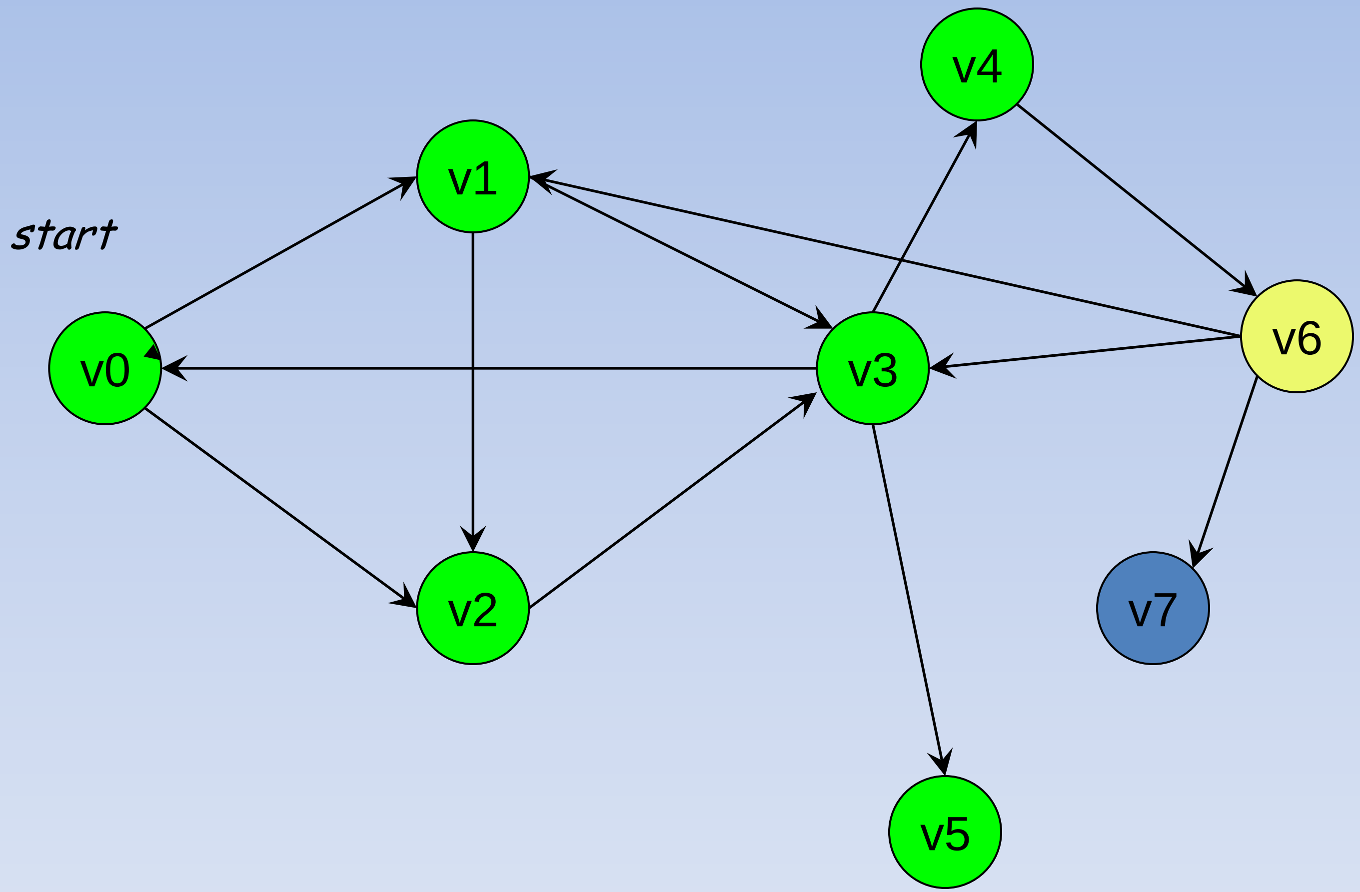
ToVisit = { v4, v5 }

Breadth-first Search



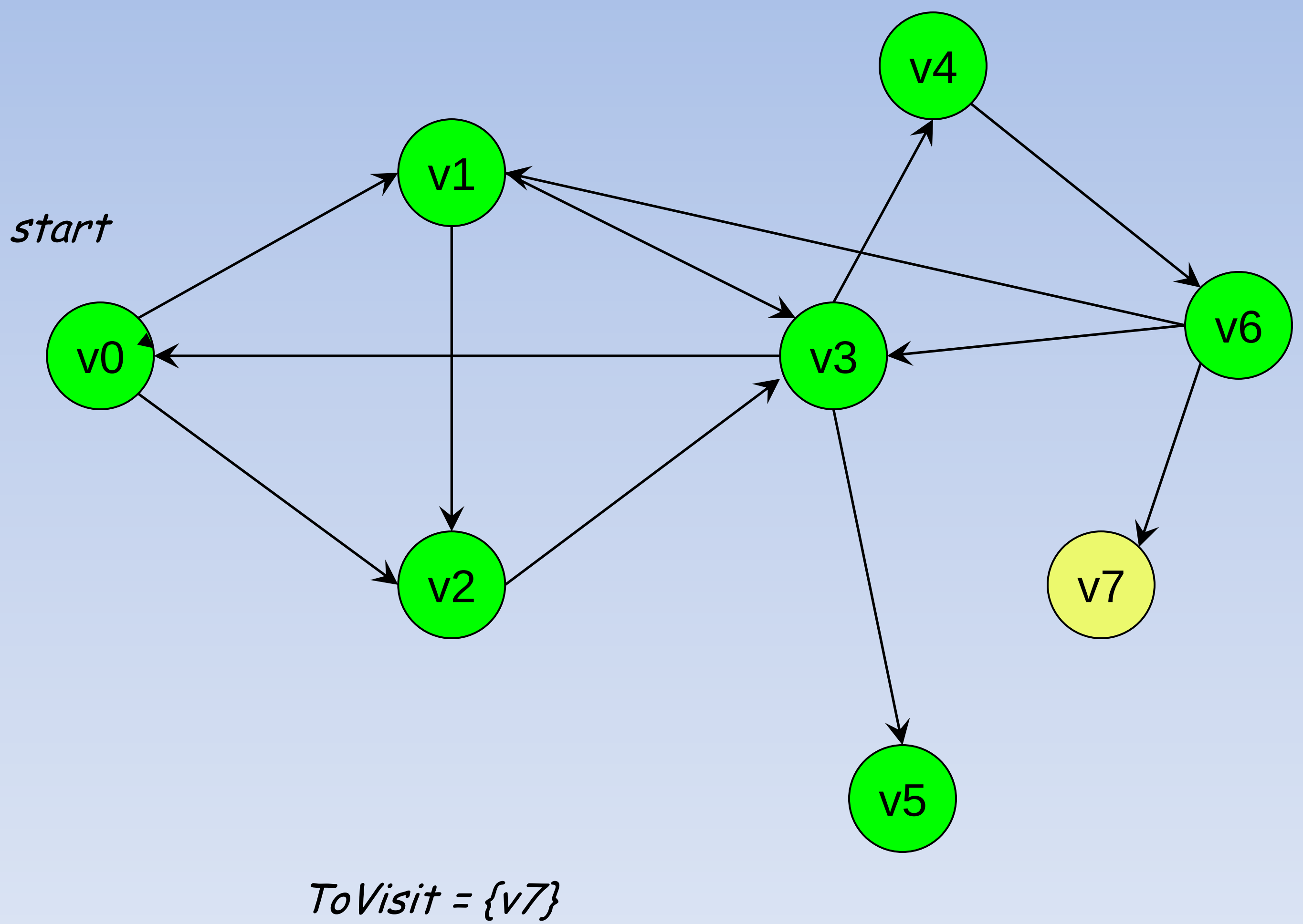
ToVisit = { v5 , v6}

Breadth-first Search

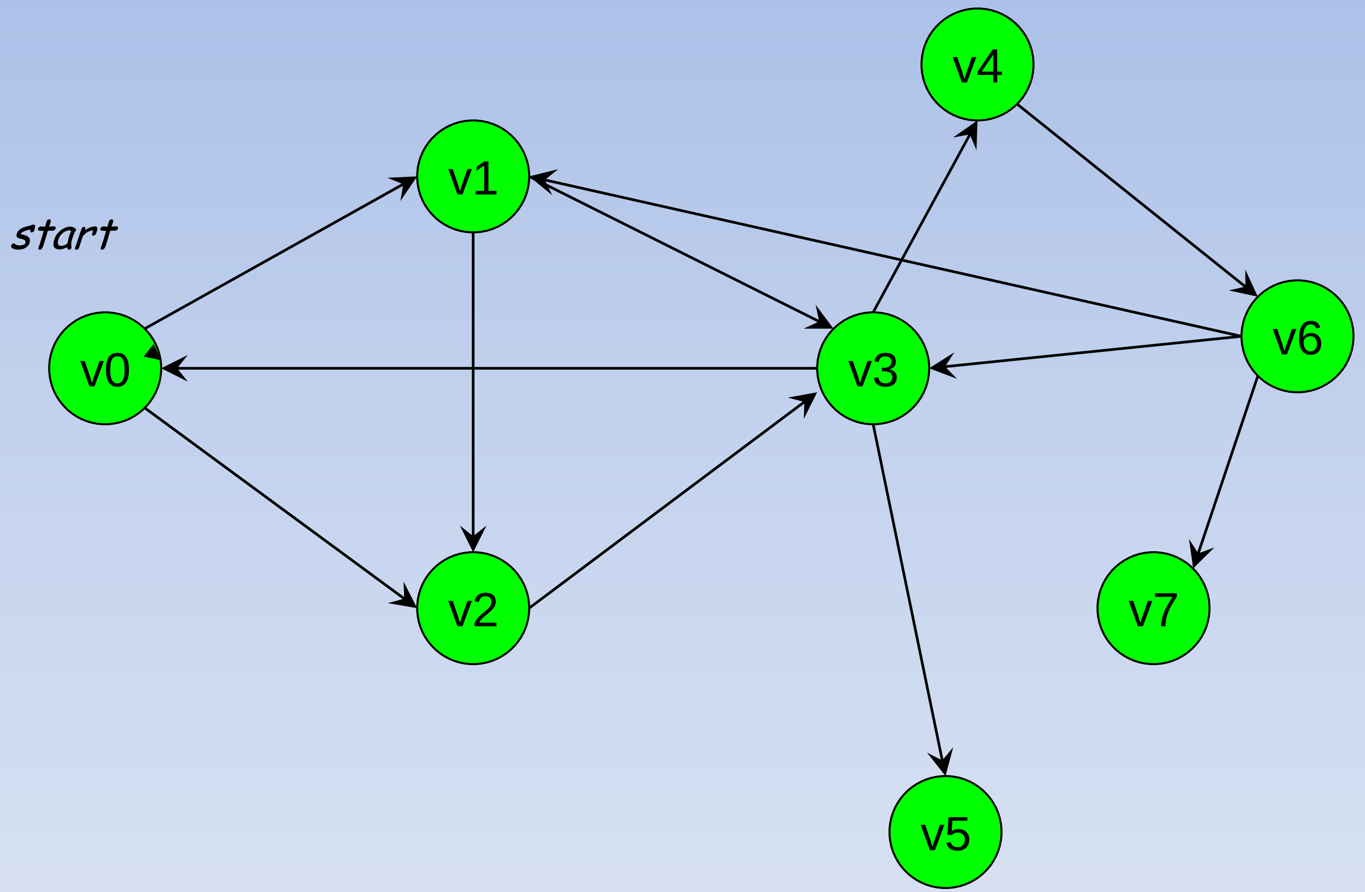


ToVisit = {v6}

Breadth-first Search



Breadth-first Search

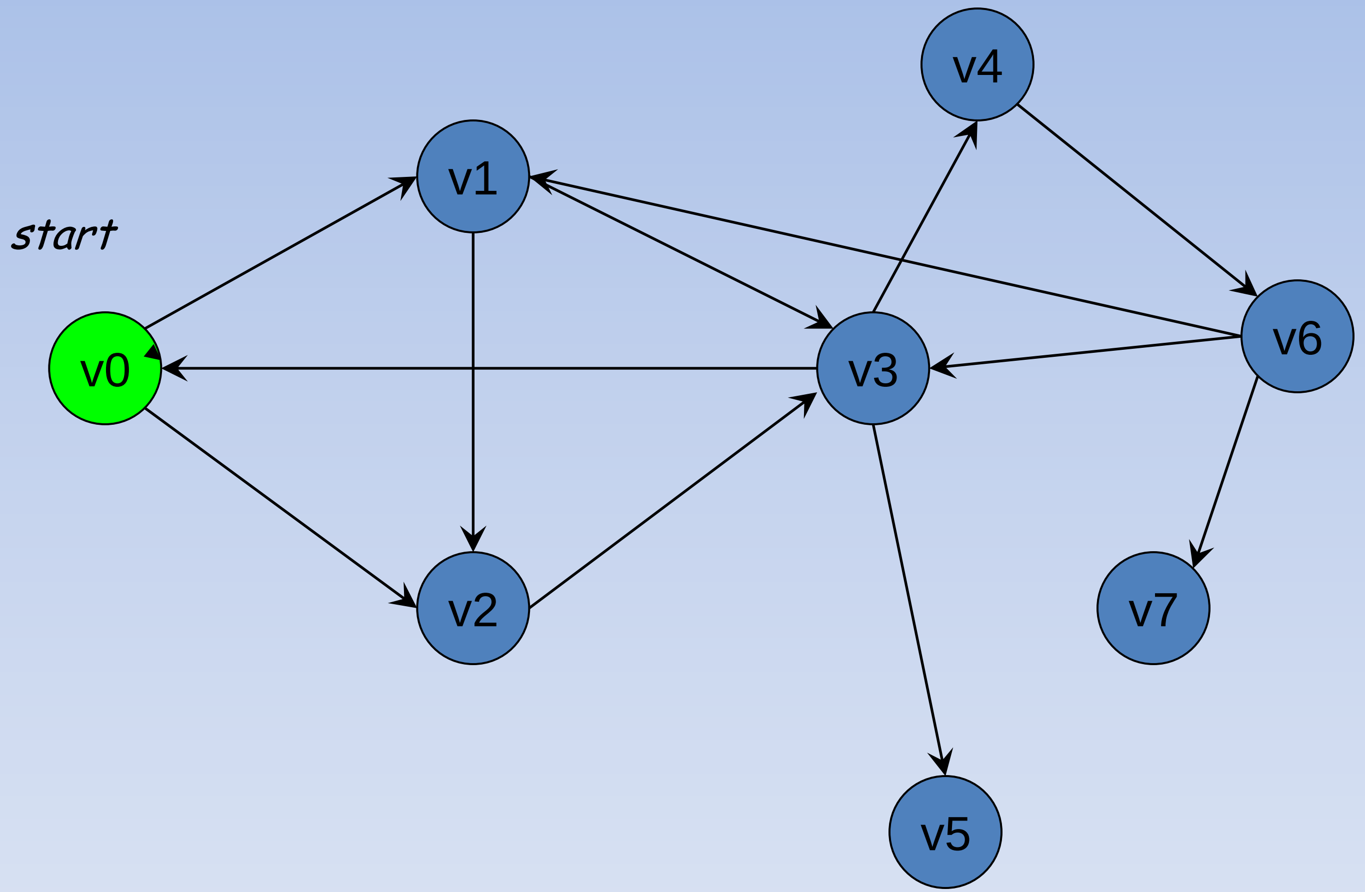


ToVisit = {}

Depth-first Search

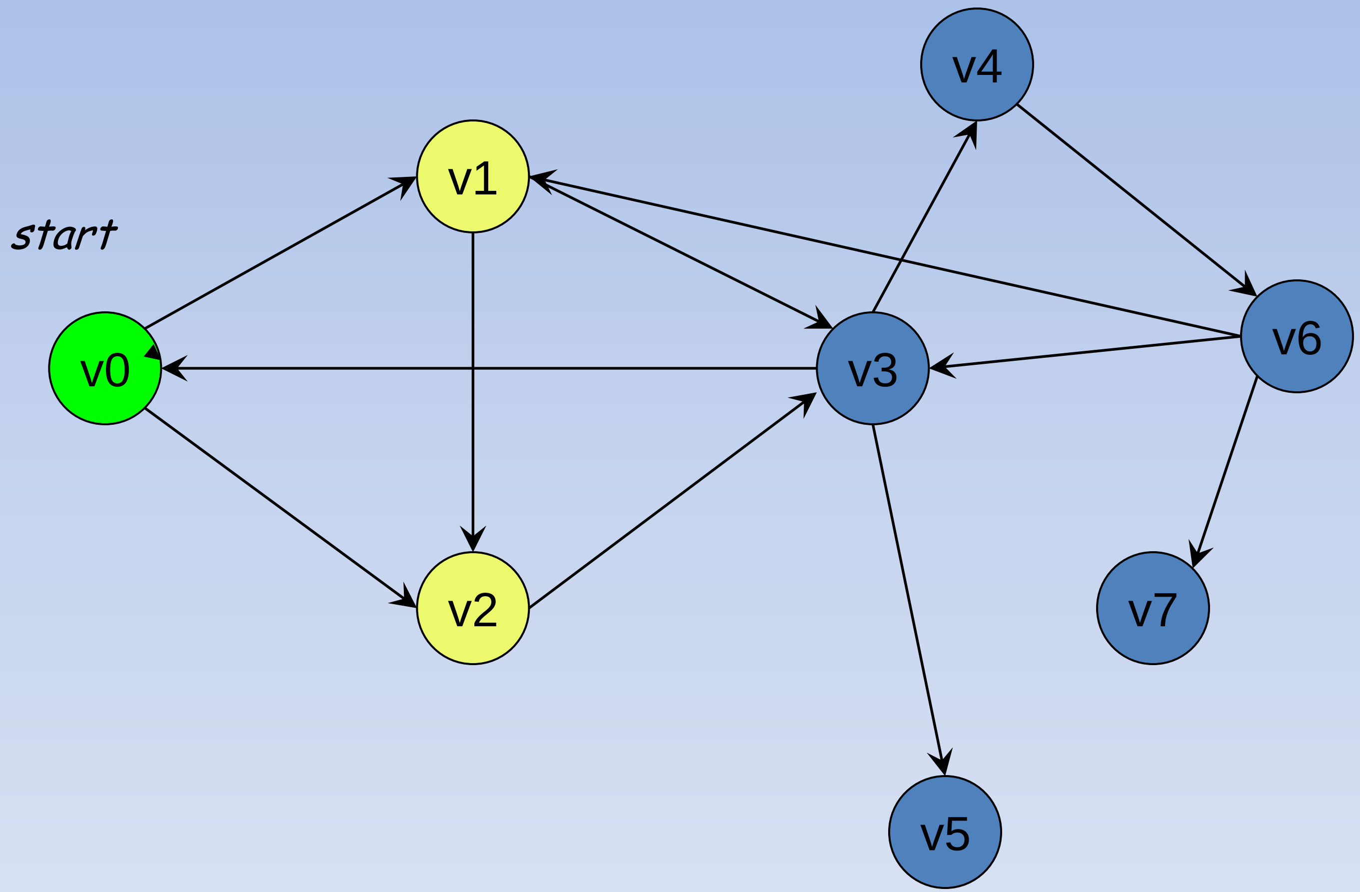
- Now see what happens if ToVisit is implemented as a stack (LIFO).

Depth-first Search



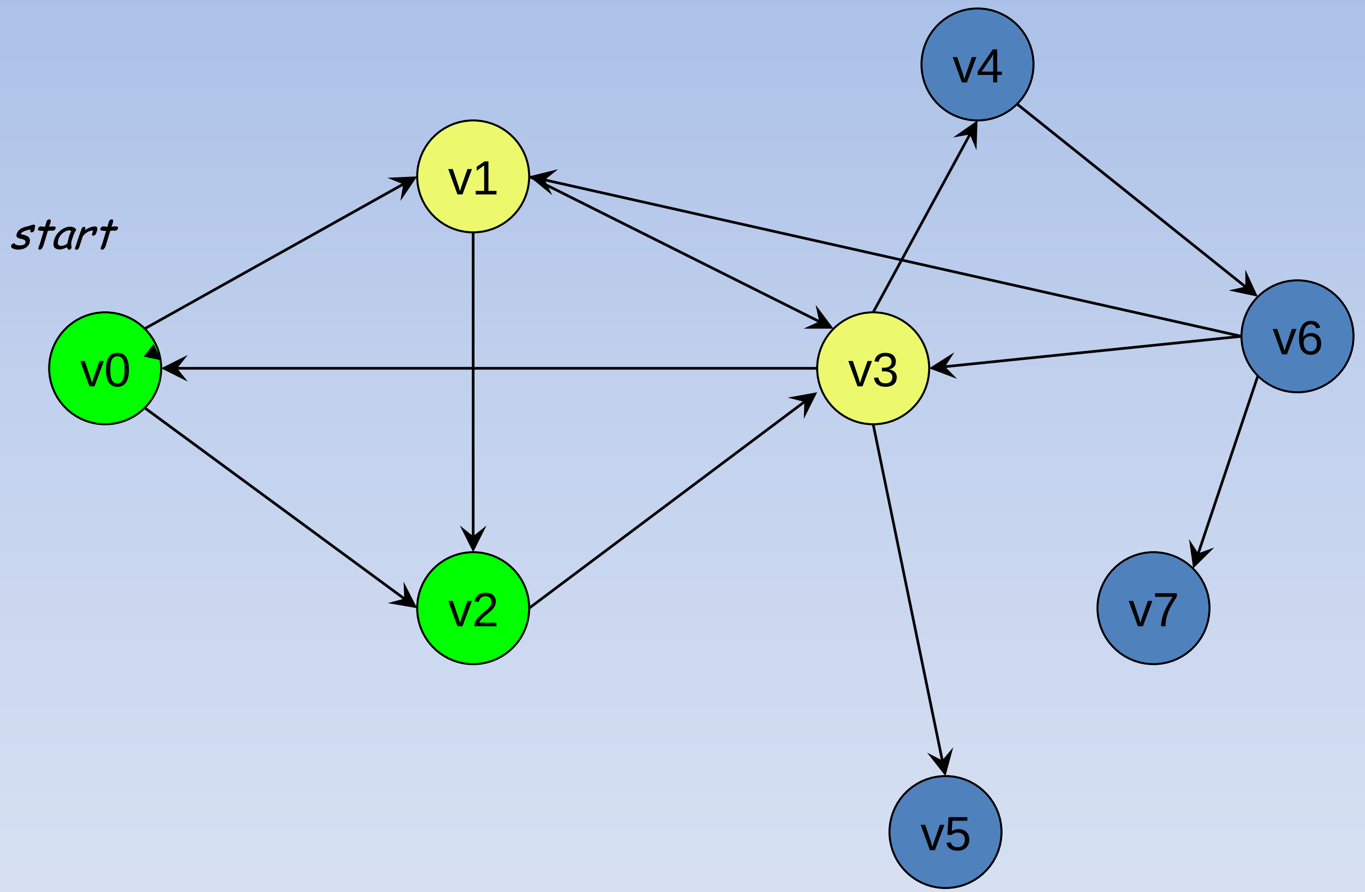
ToVisit = { v0 }

Depth-first Search



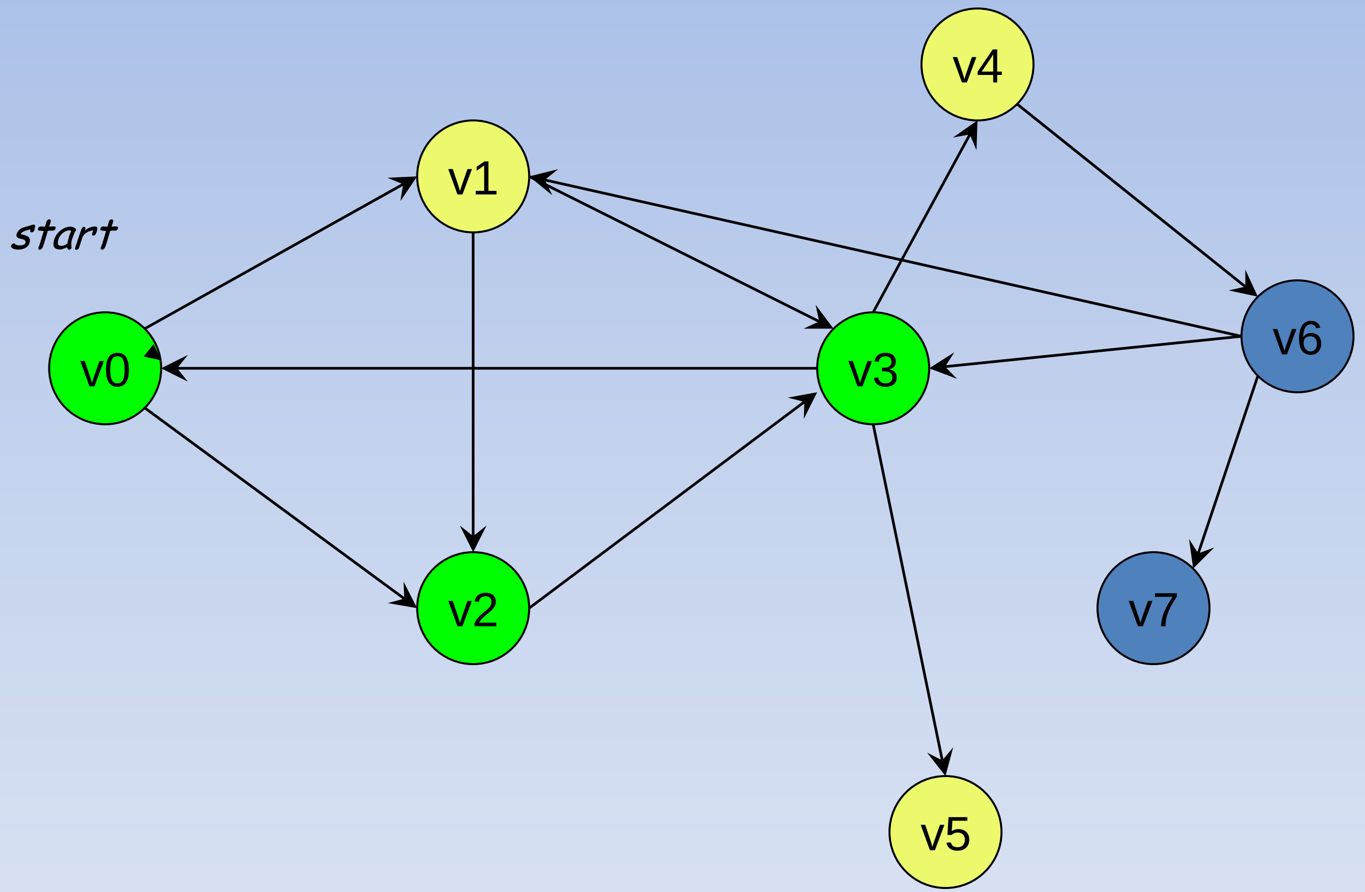
ToVisit = { v1, v2 }

Depth-first Search



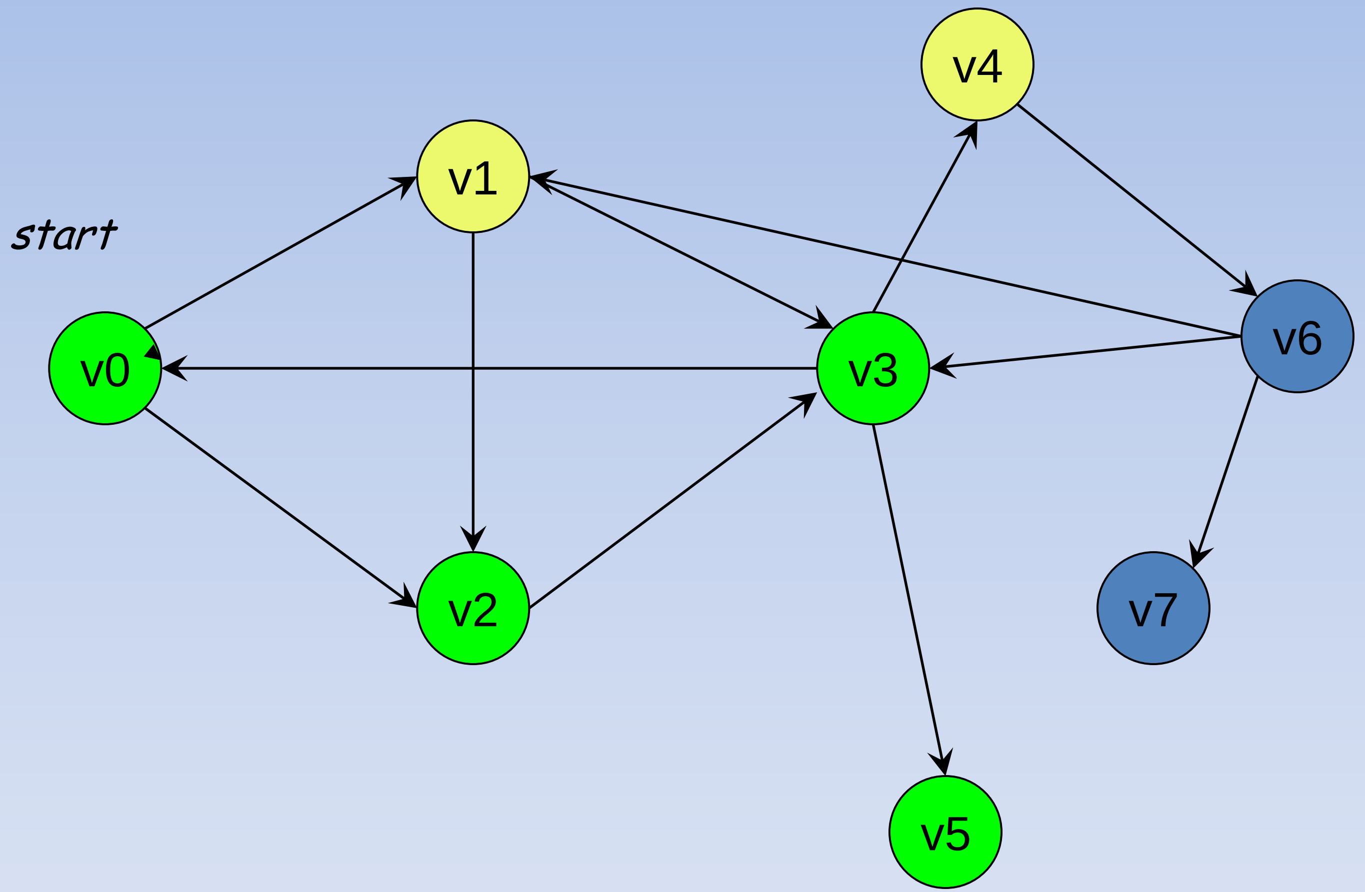
ToVisit = { v1, v3 }

Depth-first Search



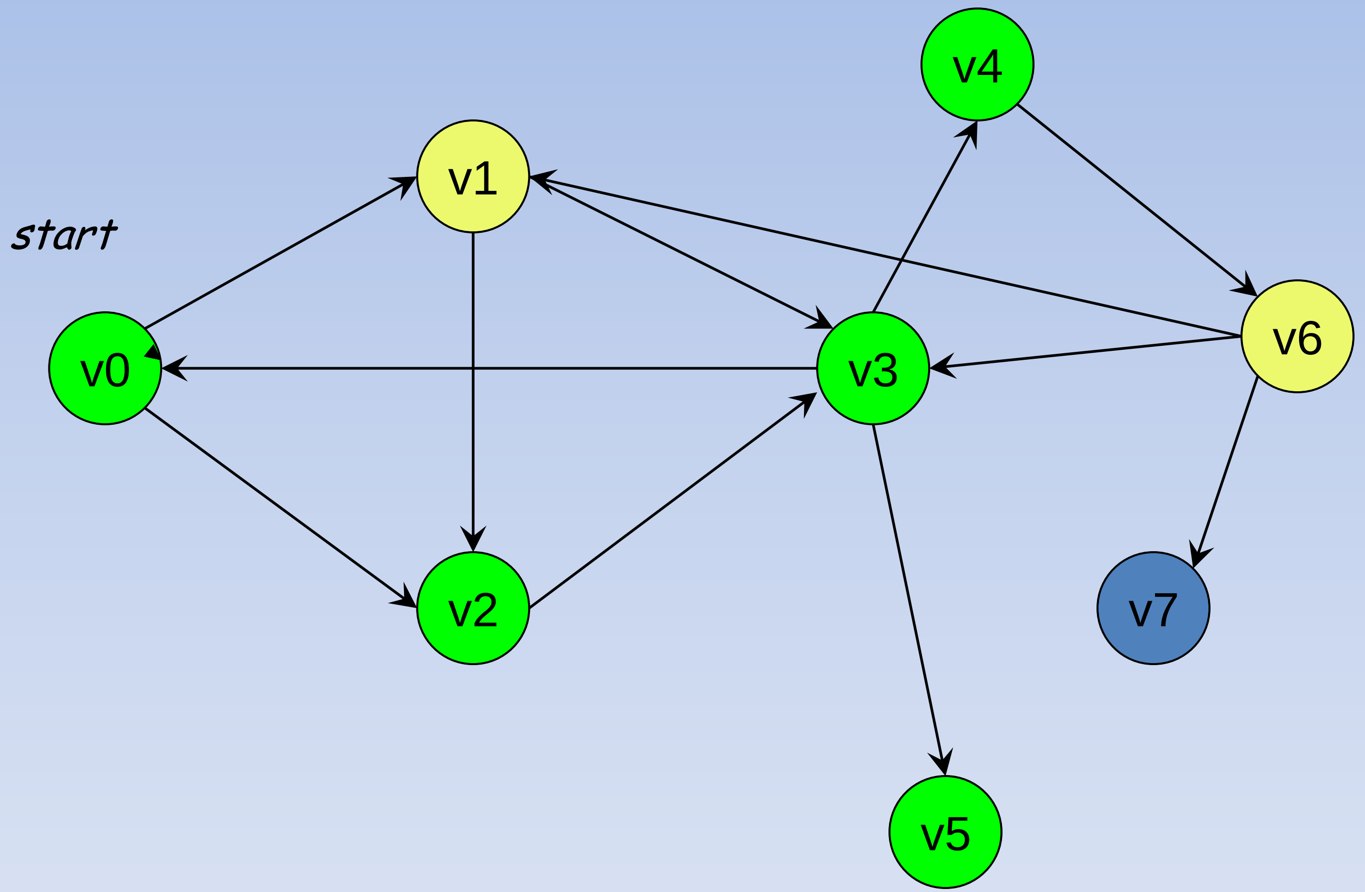
ToVisit = { v1, v4, v5}

Depth-first Search



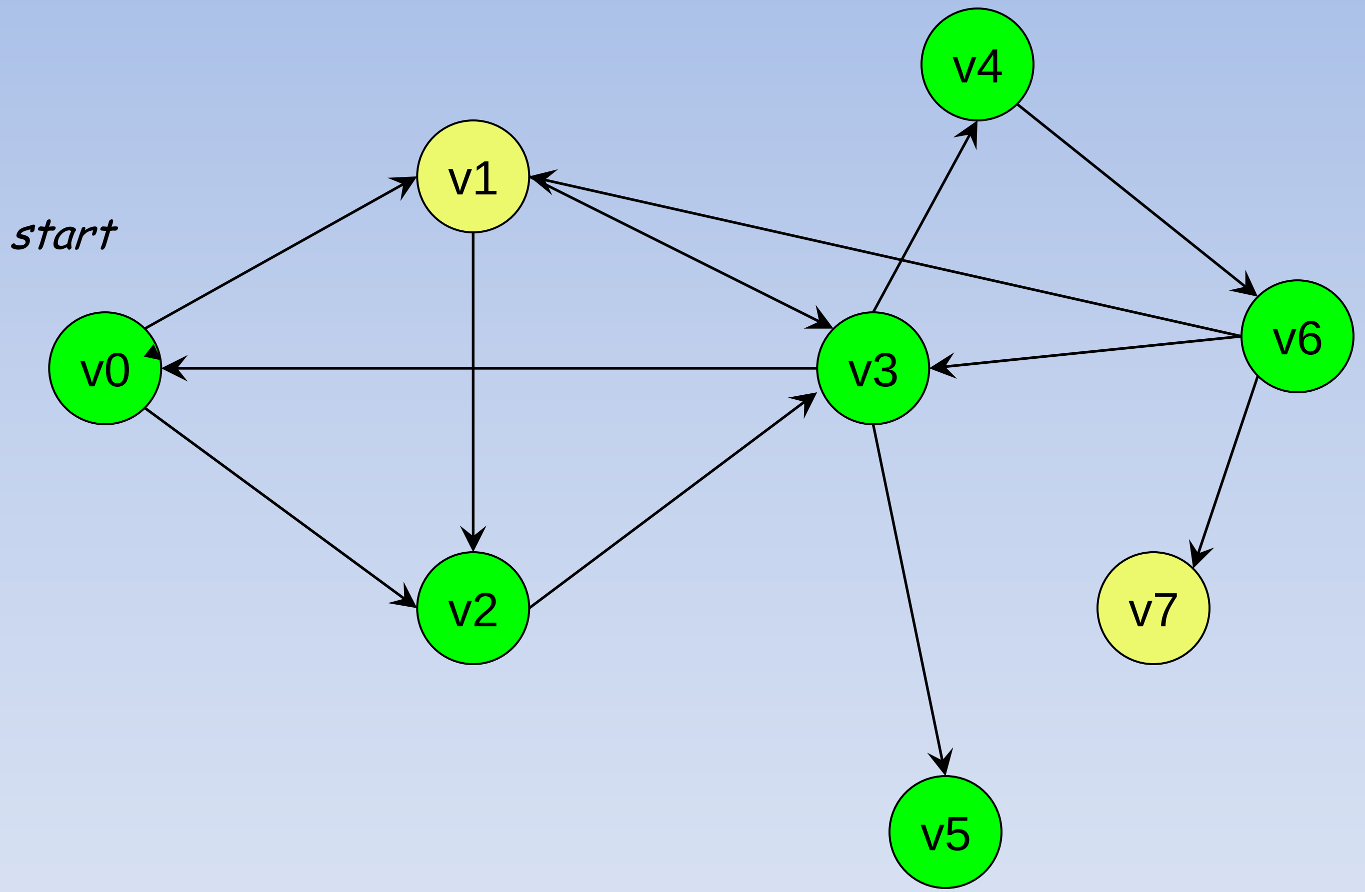
ToVisit = { v1, v4 }

Depth-first Search



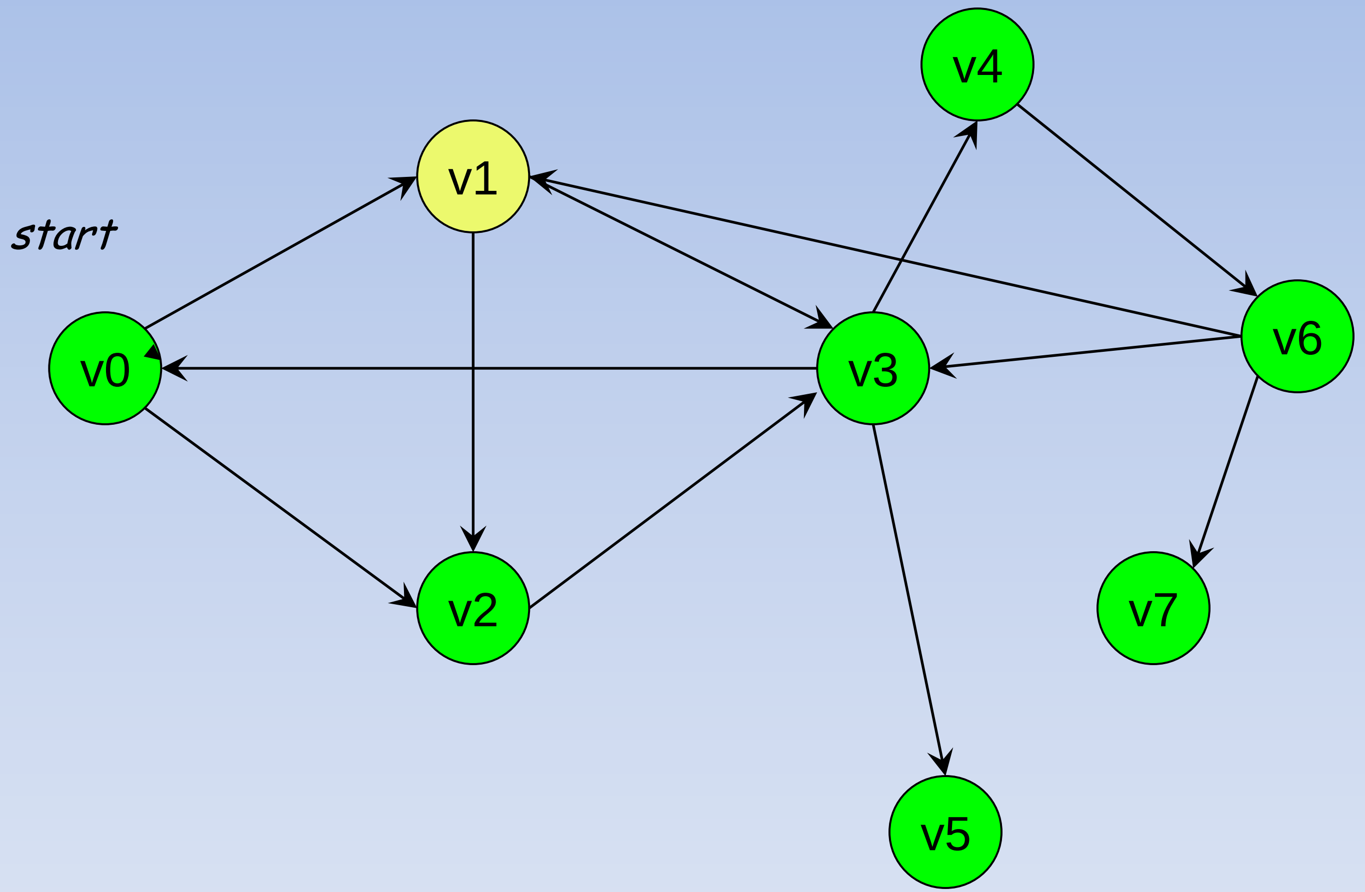
ToVisit = { v1, v6}

Depth-first Search



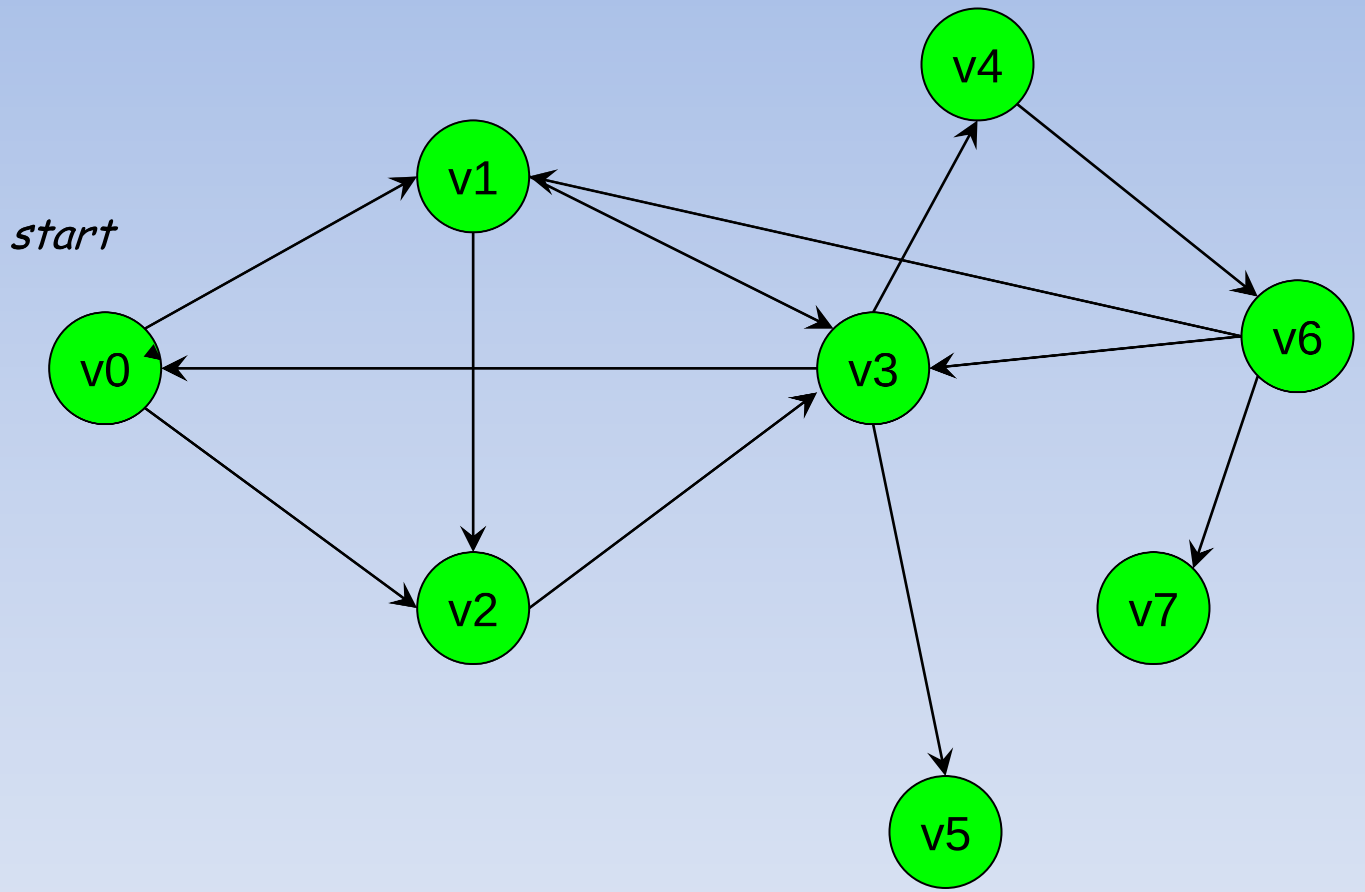
ToVisit = { v1, v7}

Depth-first Search



ToVisit = { v1 }

Depth-first Search

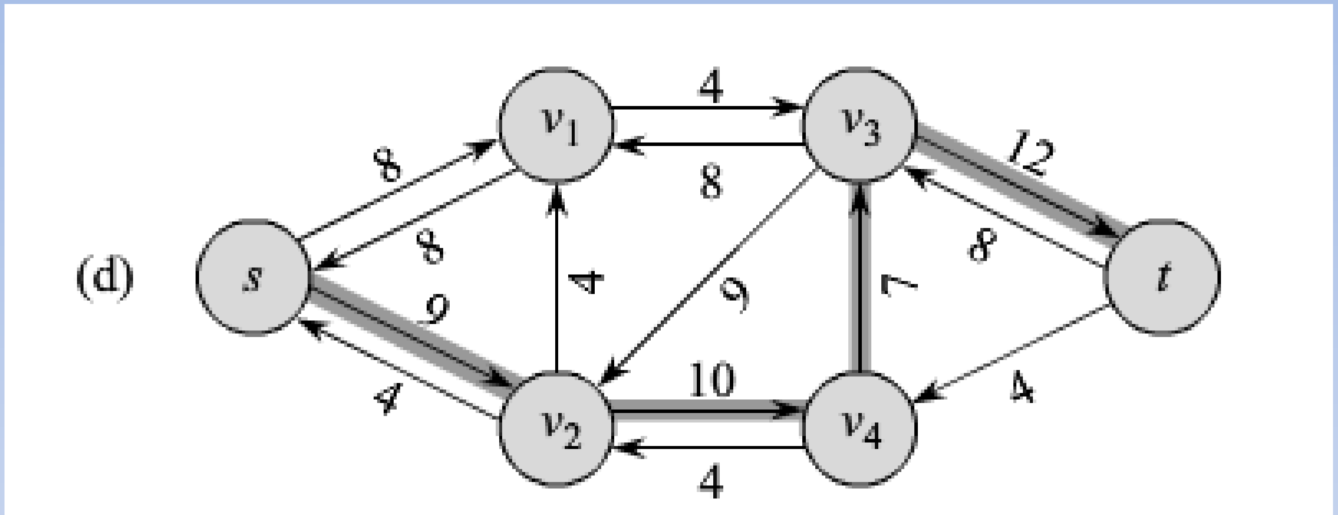


ToVisit = { }

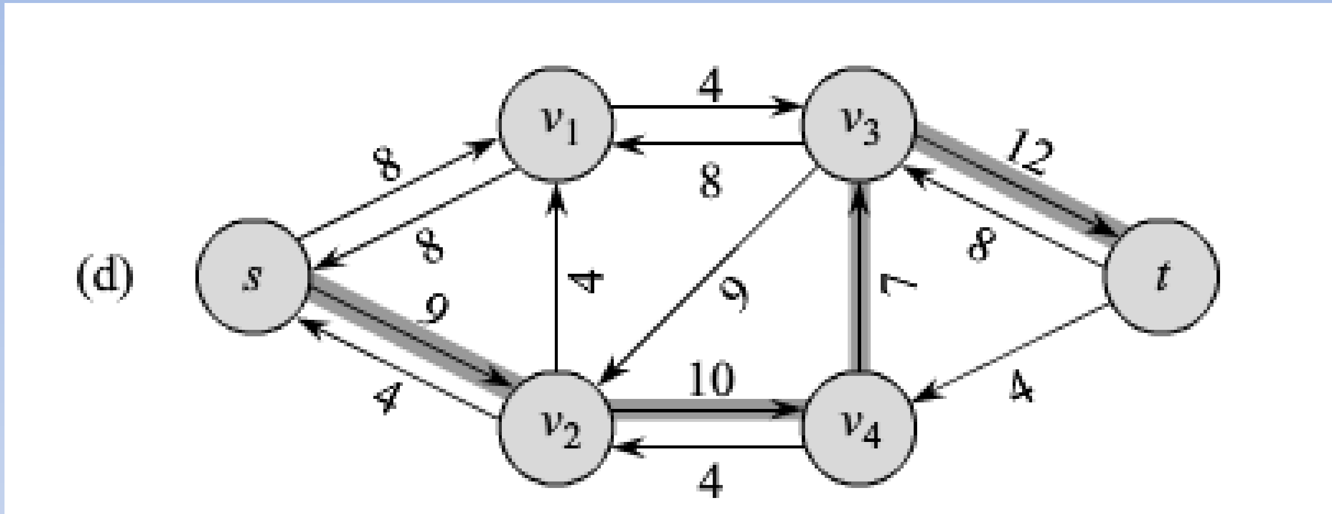
Analysis

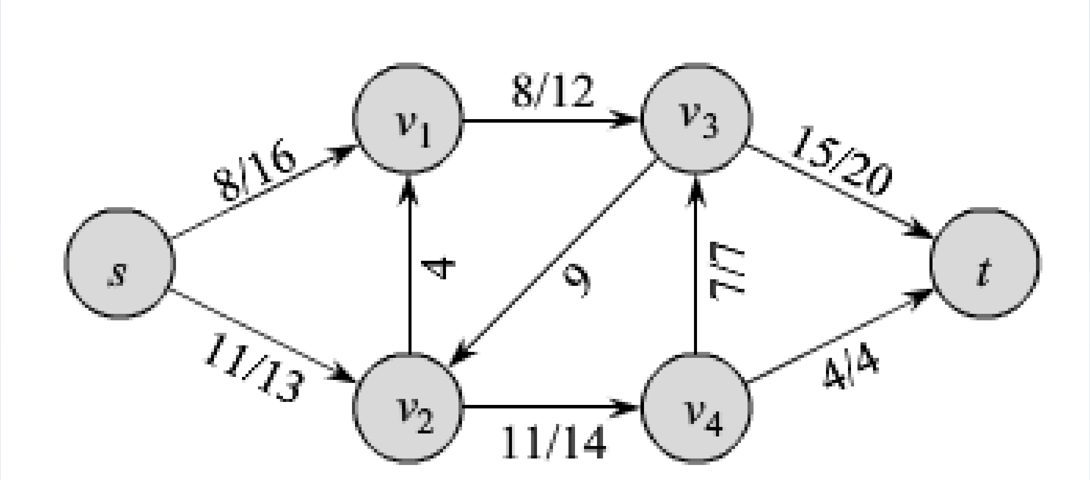
- Each iteration adjusts flow by at least 1
- $O(E\,|f^*|)$

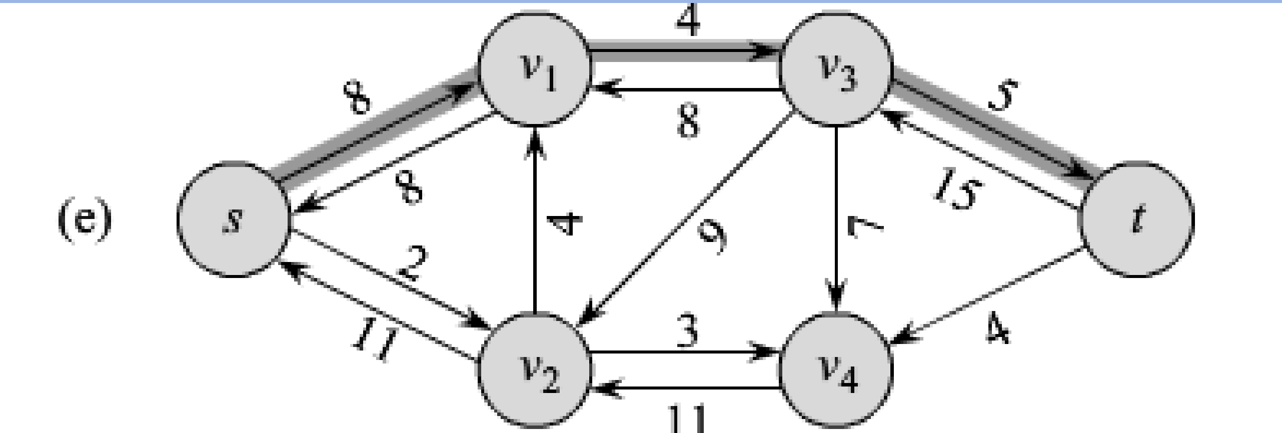
Example

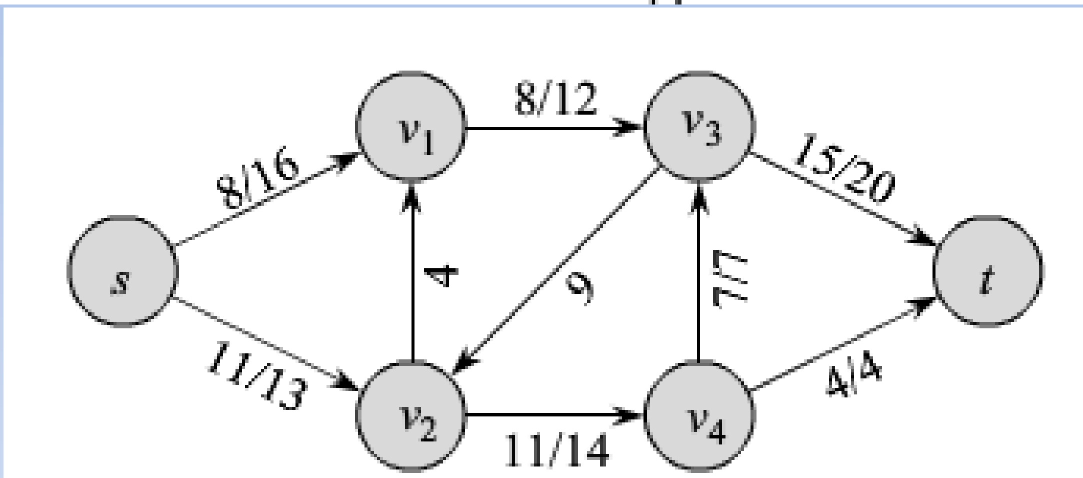


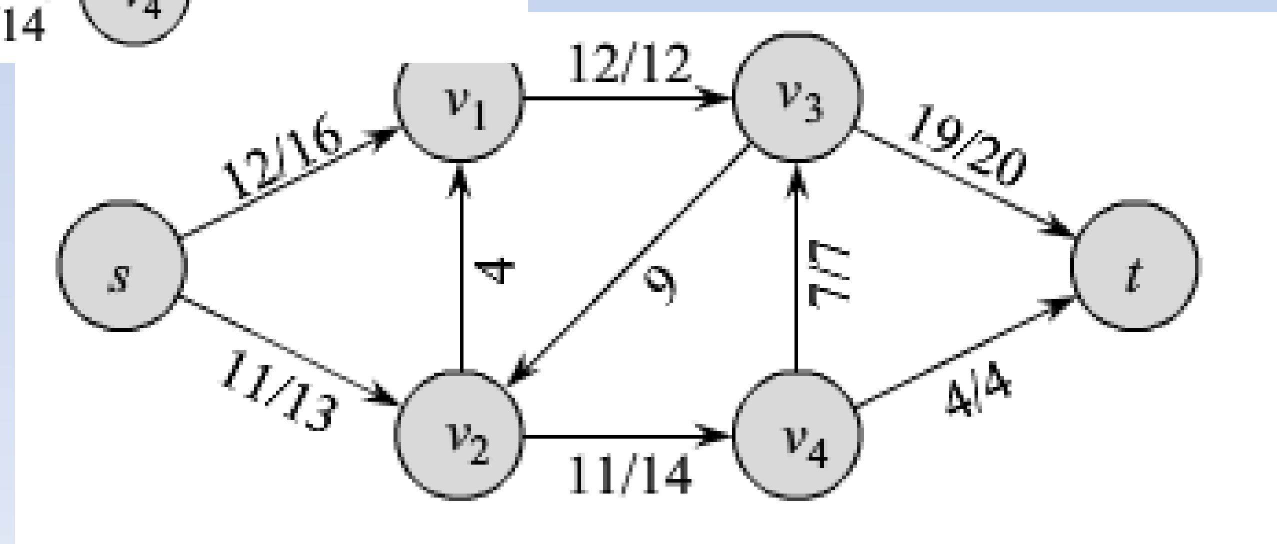
Example


(d)
s
v1
v2
v3
v4
t
4
8
12
8
8
9
4
4
9
7
8
10
4
4


s
v1
v2
v3
v4
t
8/12
8/16
15/20
4
9
7/7
11/13
11/14
4/4


(e)
s
v1
v2
v3
v4
t
8
8
4
8
4
2
11
9
7
5
15
3
11
4


s
v1
v2
v3
v4
t
8/16
8/12
15/20
11/13
4
9
7/7
11/14
4/4


s
v1
v2
v3
v4
t
12/16
12/12
19/20
11/13
4
9
7/7
11/14
4/4

No Augmentation Paths