離散最適化基礎論

第 6 回

最大流問題：容量スケーリング法

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2023 年 11 月 14 日

最終更新：2023 年 11 月 14 日　09:48

スケジュール (前半)

1. 最大流と最小費用流：定義 (10/3)
2. 最大流問題：増加道法 (10/10)

＊ 休み (10/17)

3. 線形計画法の復習 (10/24)
4. 最大流問題：線形計画問題として (10/31)
5. 最大流問題：Edmonds-Karp のアルゴリズム (11/7)
6. 最大流問題：容量スケーリング法 (11/14)
7. 最大流問題：Push-Relabel 法 (概要) (11/21)
8. 最大流問題：Push-Relabel 法 (計算量評価) (11/28)

スケジュール (後半)

＊ 休み (12/5)
9. 最小費用流問題：線形計画問題として (12/12)
10. 最小費用流問題：負閉路消去法 (12/19)
11. 最小費用流問題：正カット消去法 (12/26)
＊ 休み (1/2)
12. 最小費用流問題：逐次最短路法 (1/9)
13. 最小費用流問題：容量スケーリング法 (1/16)
14. 最小費用流問題：費用スケーリング法 (1/23)
＊ 休み (1/30)

ここまでの話 と 今日の話

最大流問題　　増加道法

工夫

Edmonds-Karp のアルゴリズム

弧の数が最小の増加道を選ぶ

ここまでの話 と 今日の話

最大流問題

増加道法

↓ 工夫

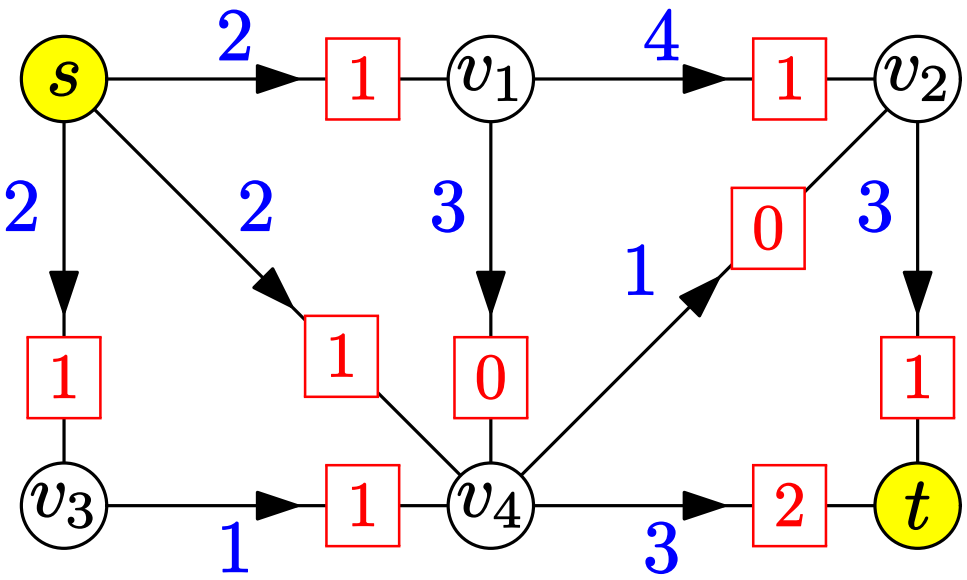
Edmonds-Karp のアルゴリズム

- 弧の数が最小の増加道を選ぶ
- 増加量が最大の増加道を選ぶ

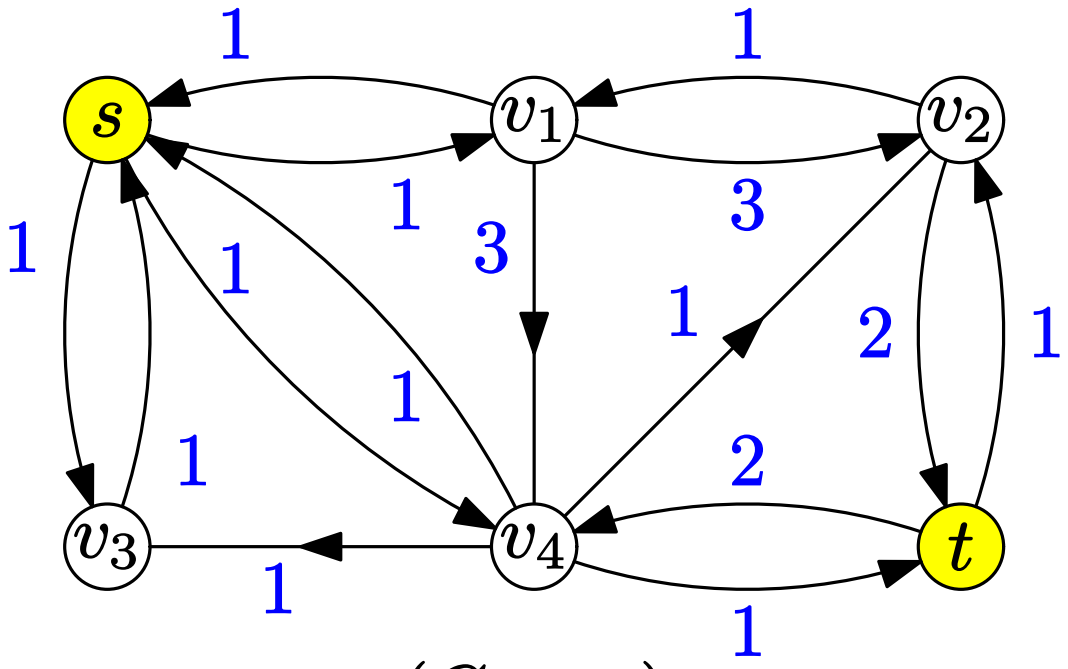
↓ 工夫

容量スケーリング法

[復習] 補助ネットワーク



$(G, u), f$

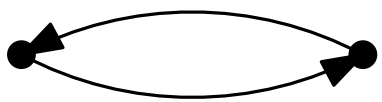


(G_f, u_f)

流　$f(a)$

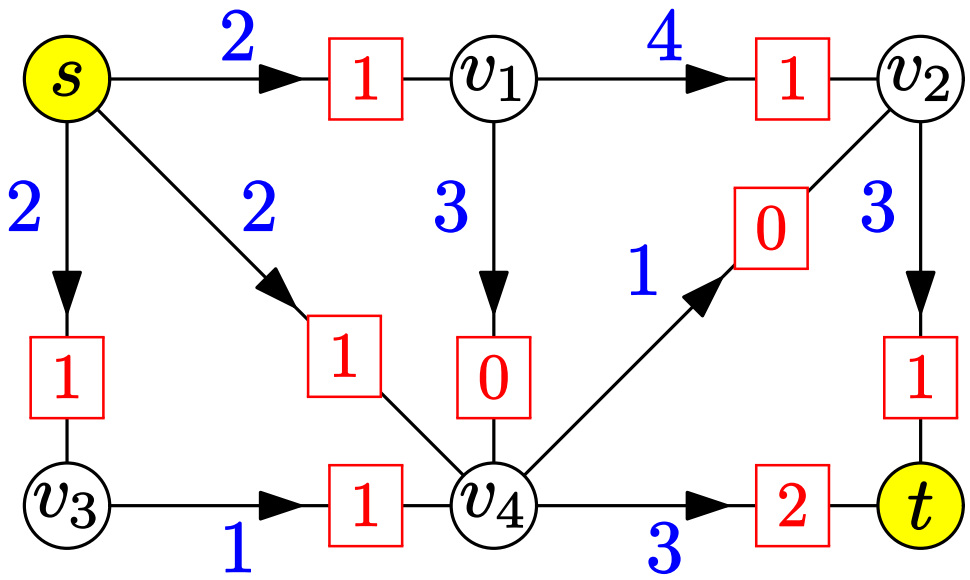
容量　$u(a)$

逆向き 容量　$f(a)$

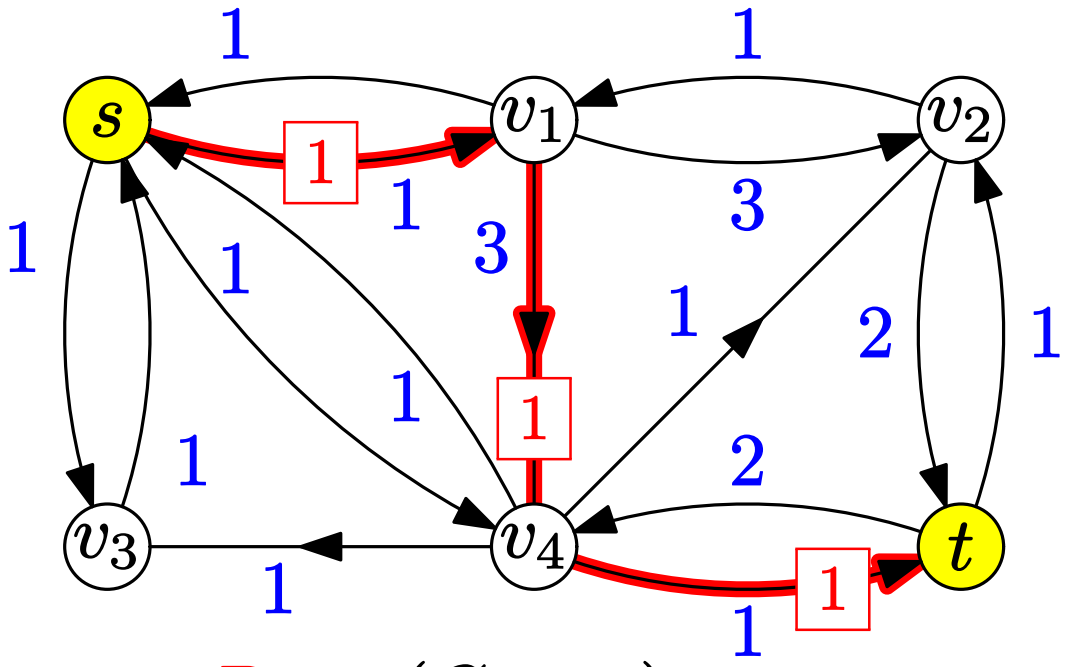


順向き 容量　$u(a) - f(a)$

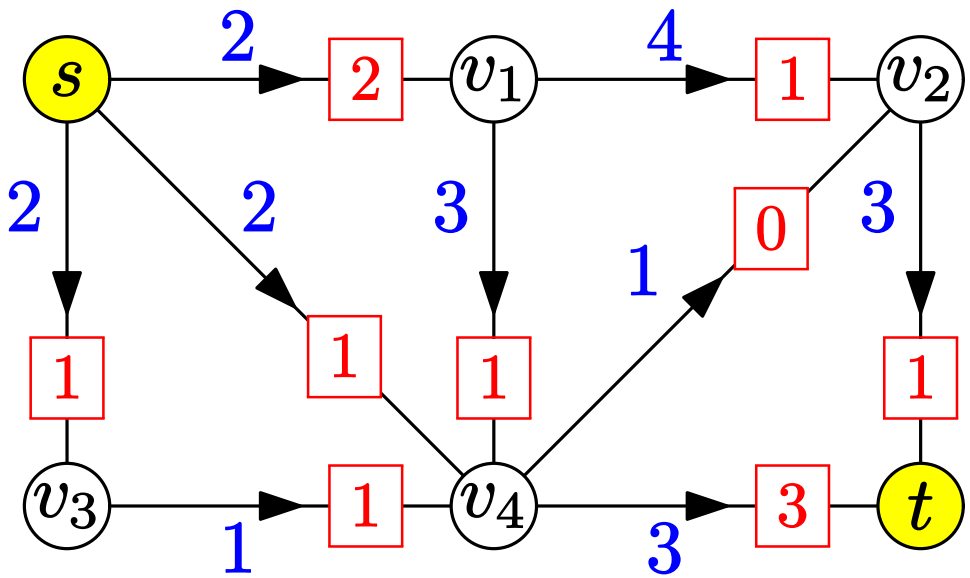
[復習] 増加道



$(G, u), f$



P (G_f, u_f)

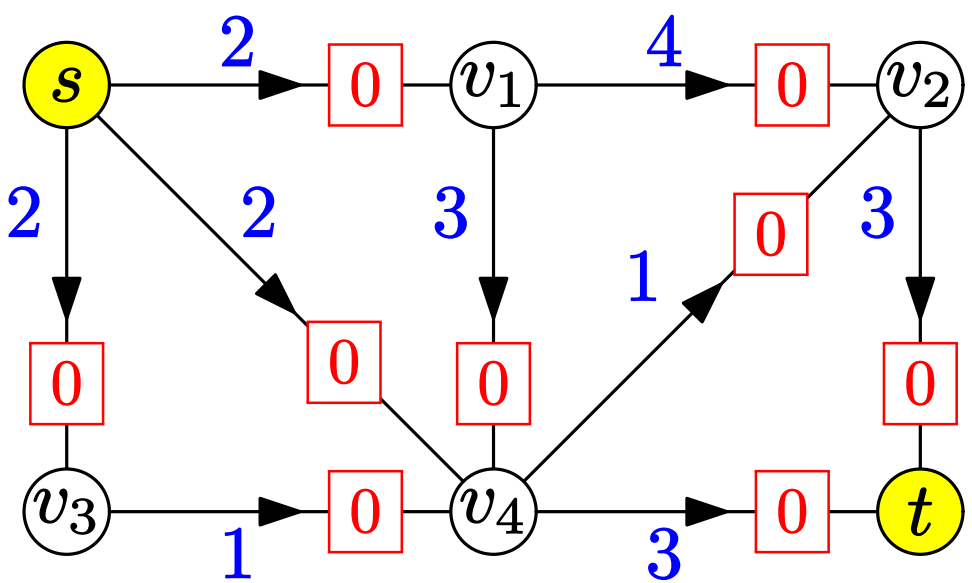


P に沿って f を増加

[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

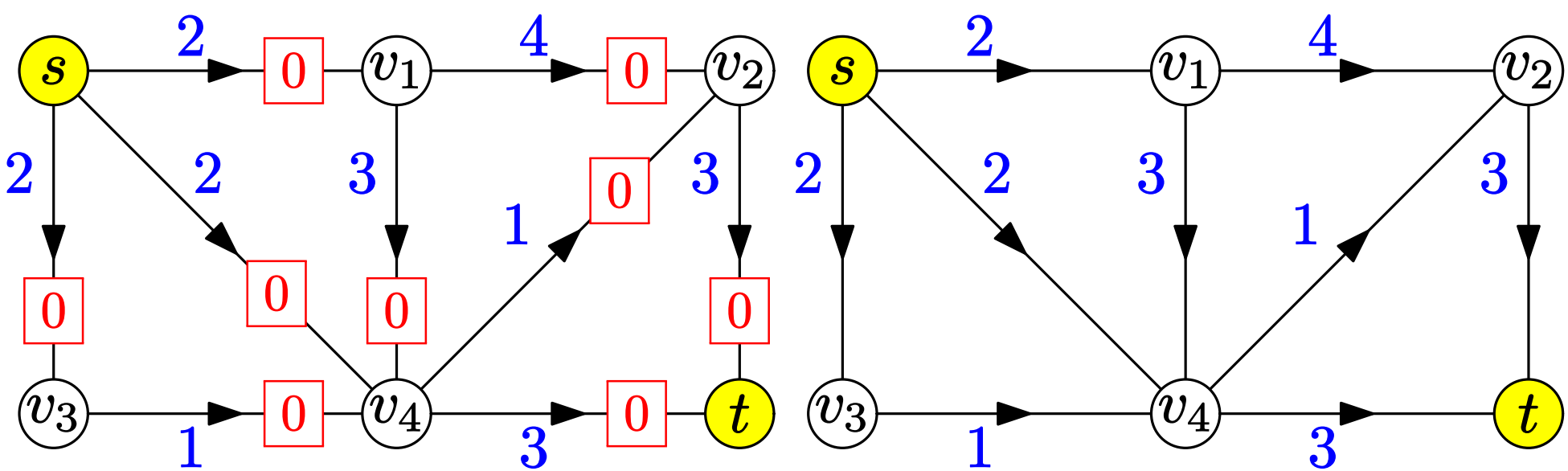
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

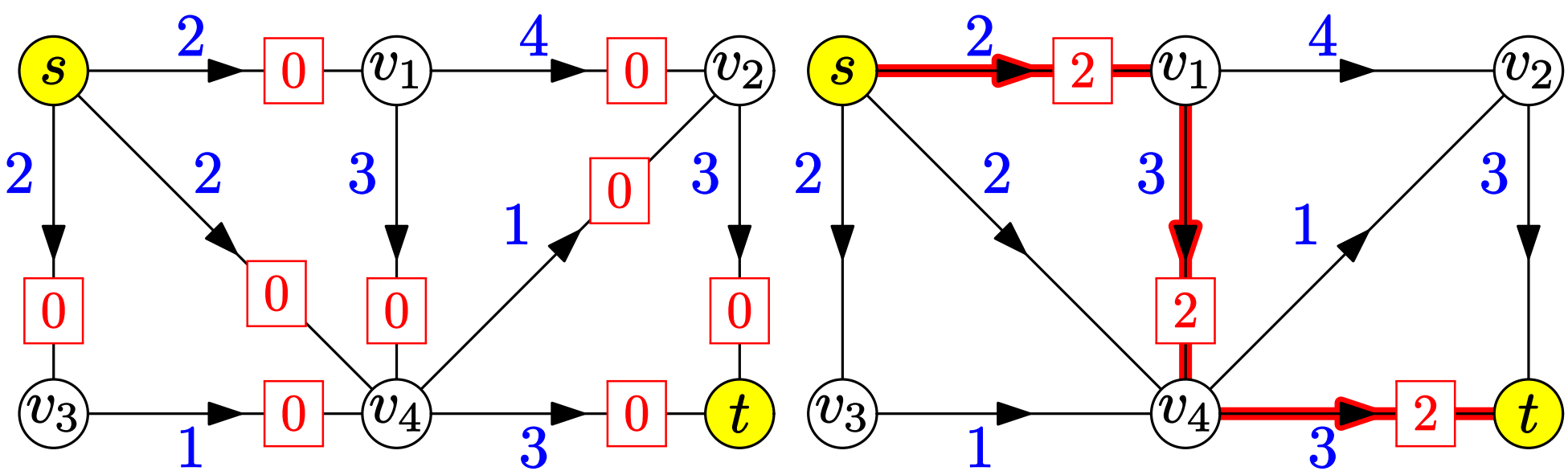
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

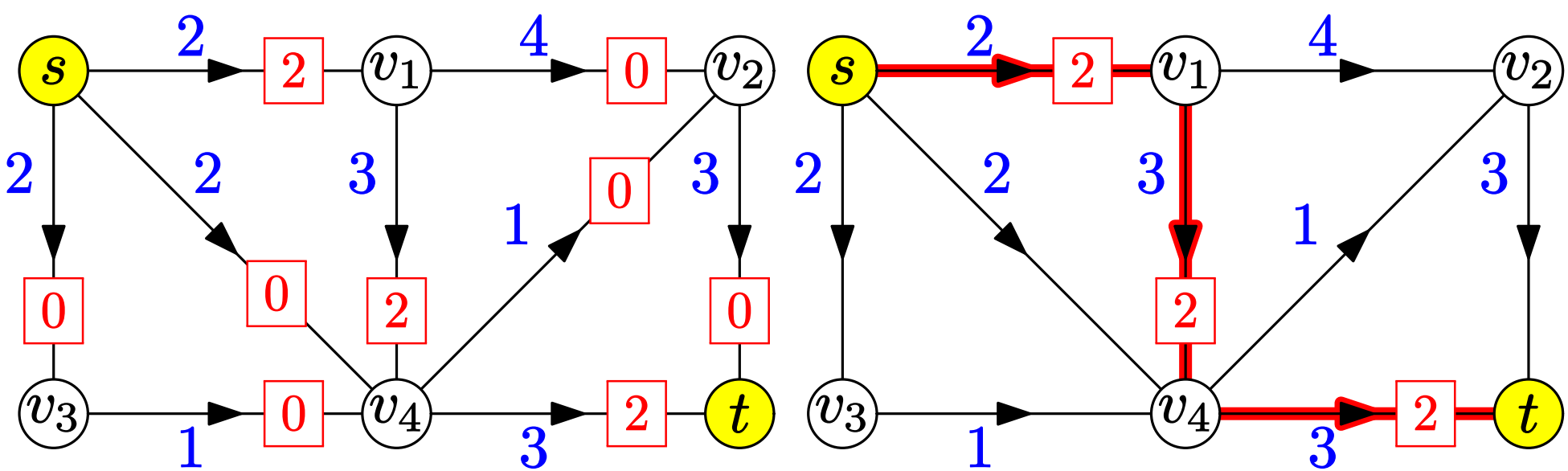
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

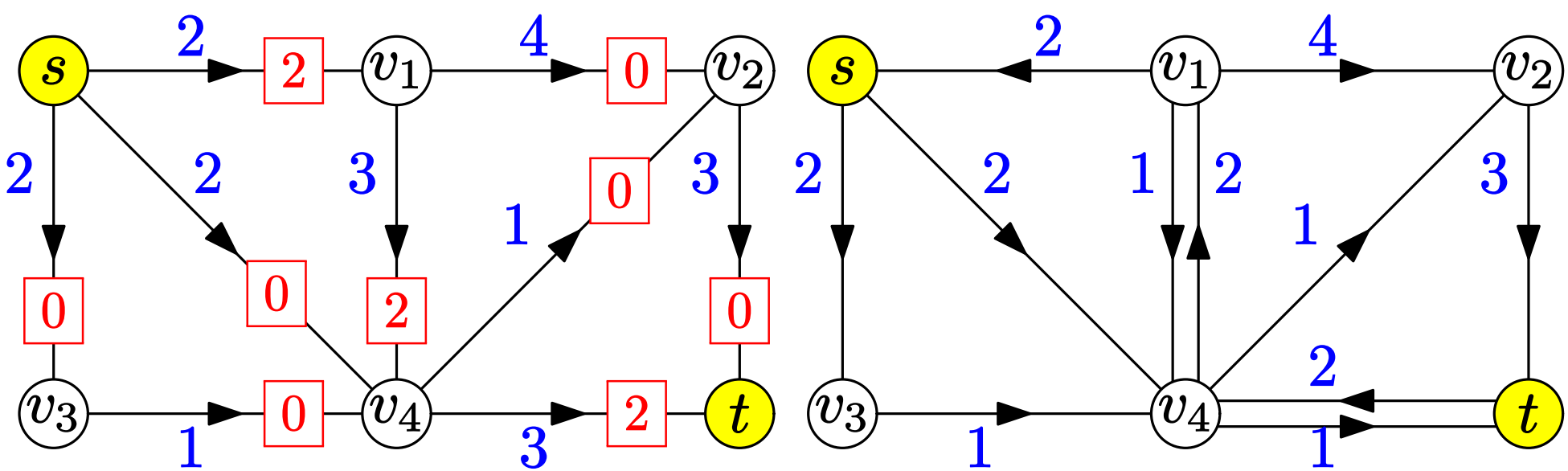
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

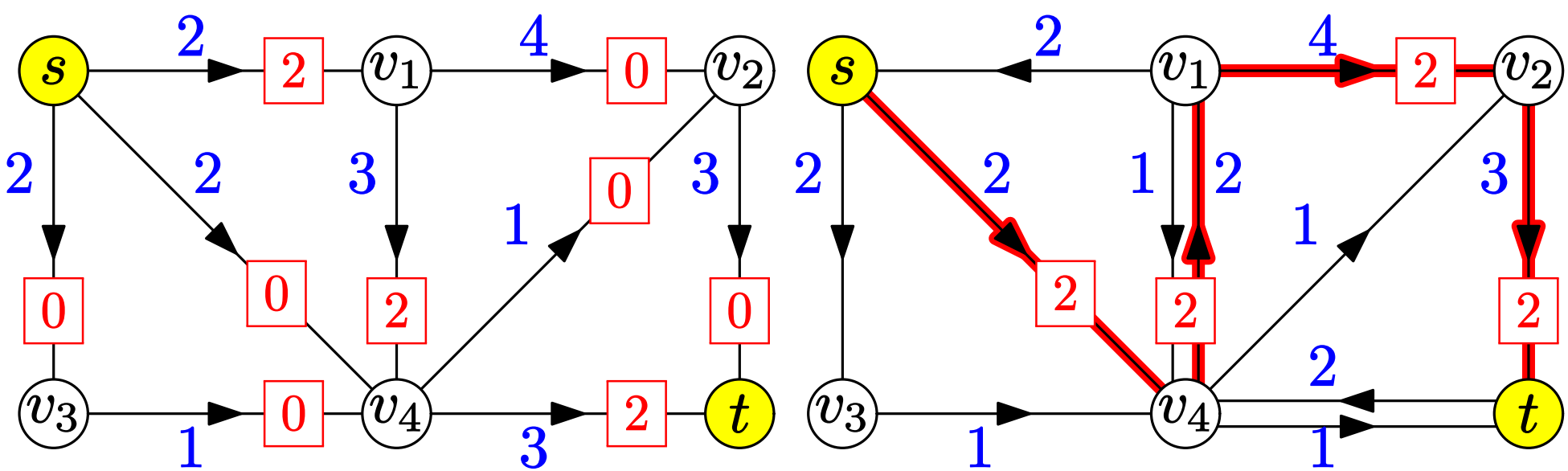
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

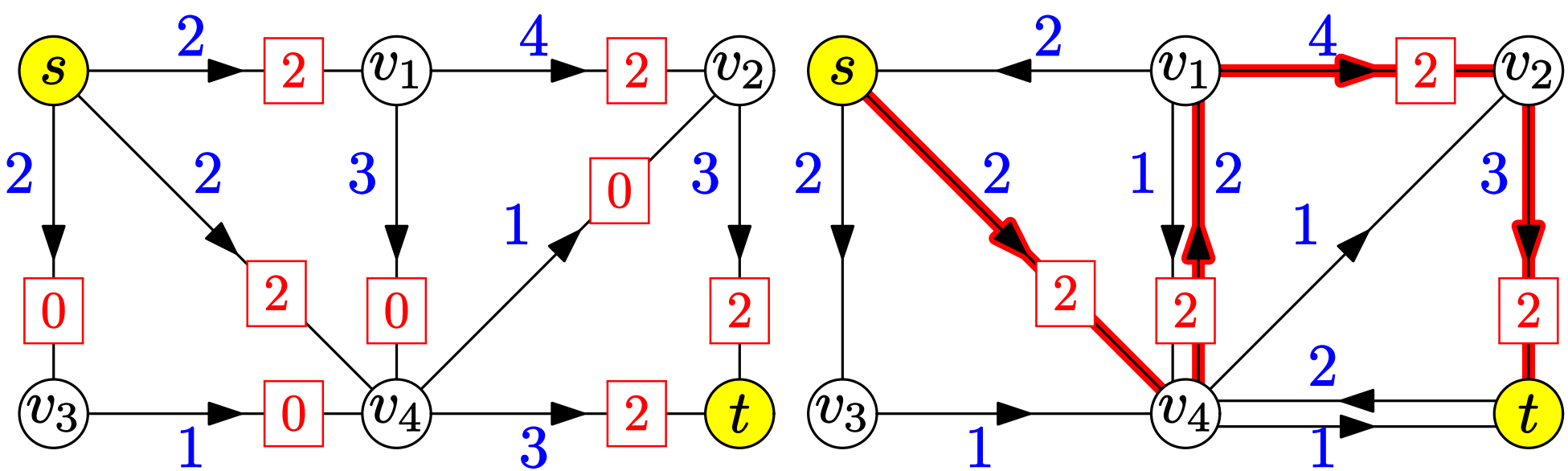
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

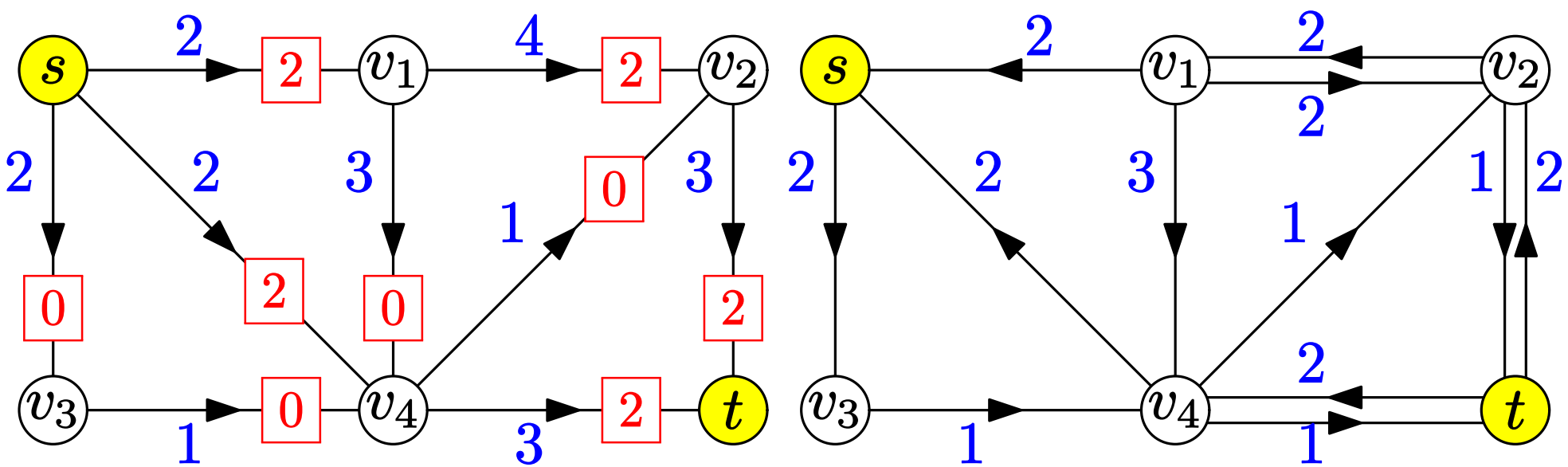
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

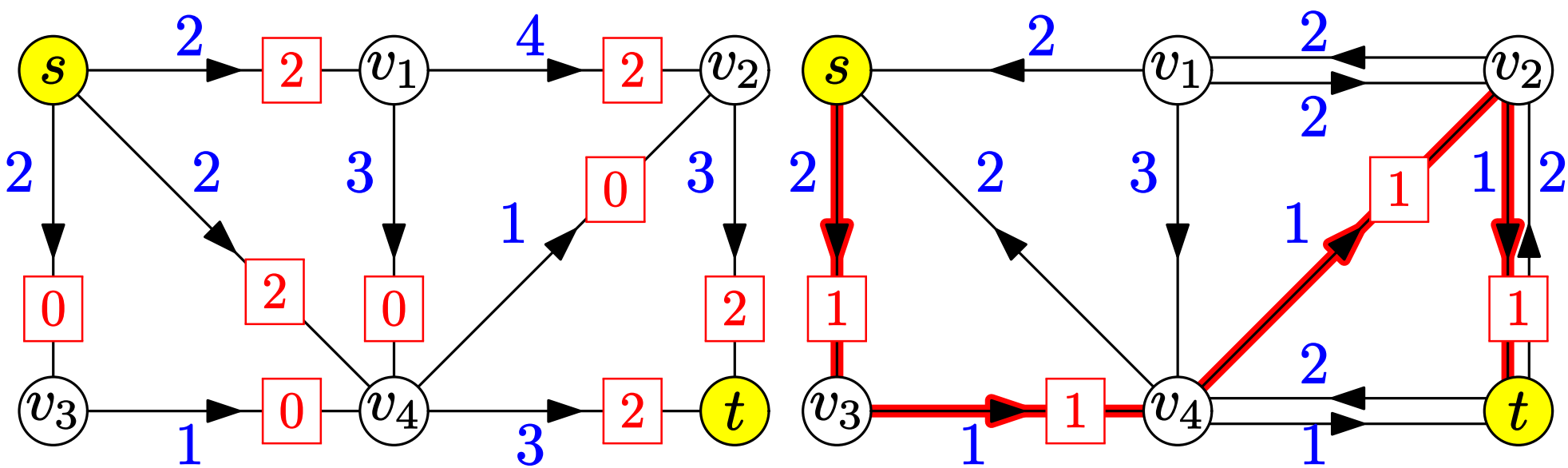
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

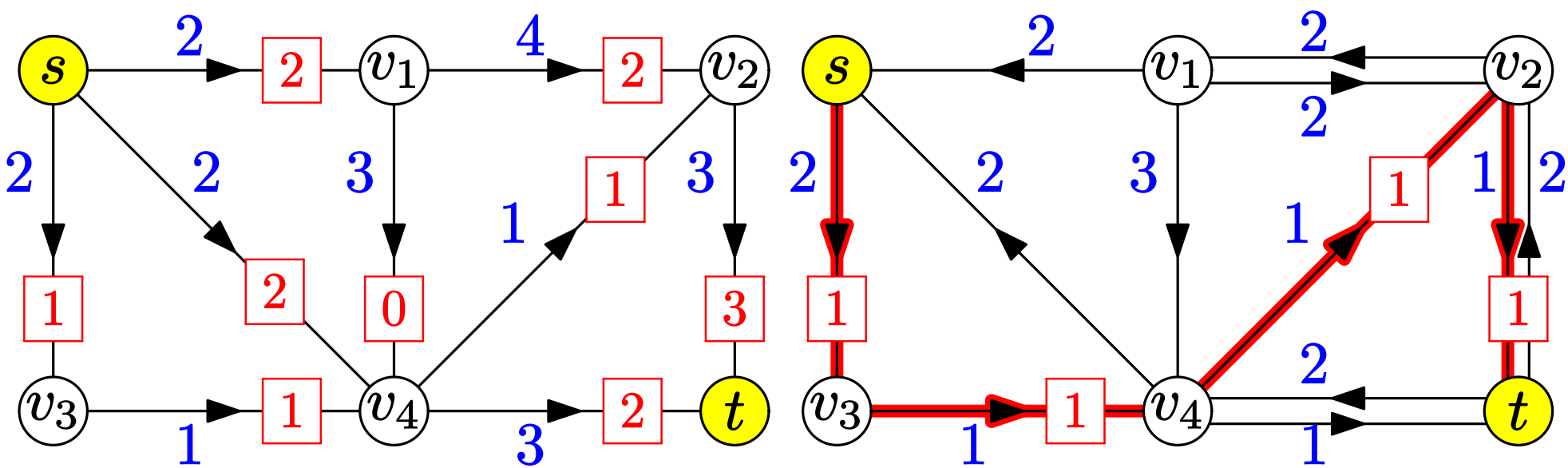
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

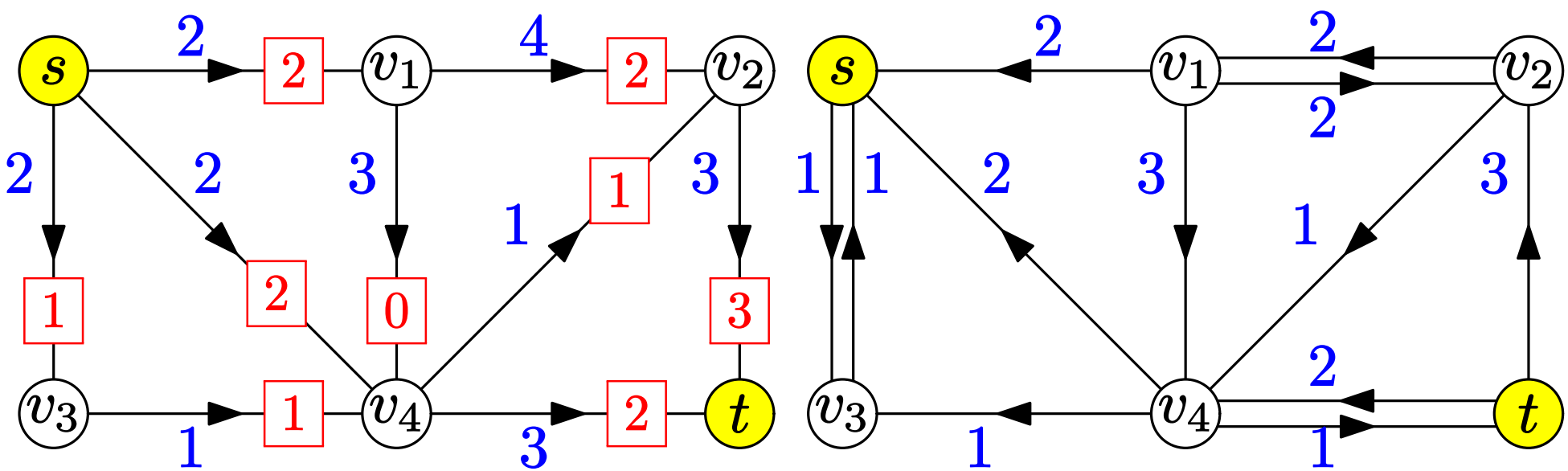
- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 増加道法

アルゴリズム：増加道法 (augmenting path method)

- 初期化：任意の s-t 流 f (例えば，$f = 0$)
- 反復：f に対する増加道がある限り，以下を実行
  1. 増加道 P を見つける
  2. P に沿って f を増加させる
- 出力：f



[復習] 最大流問題の最適性条件

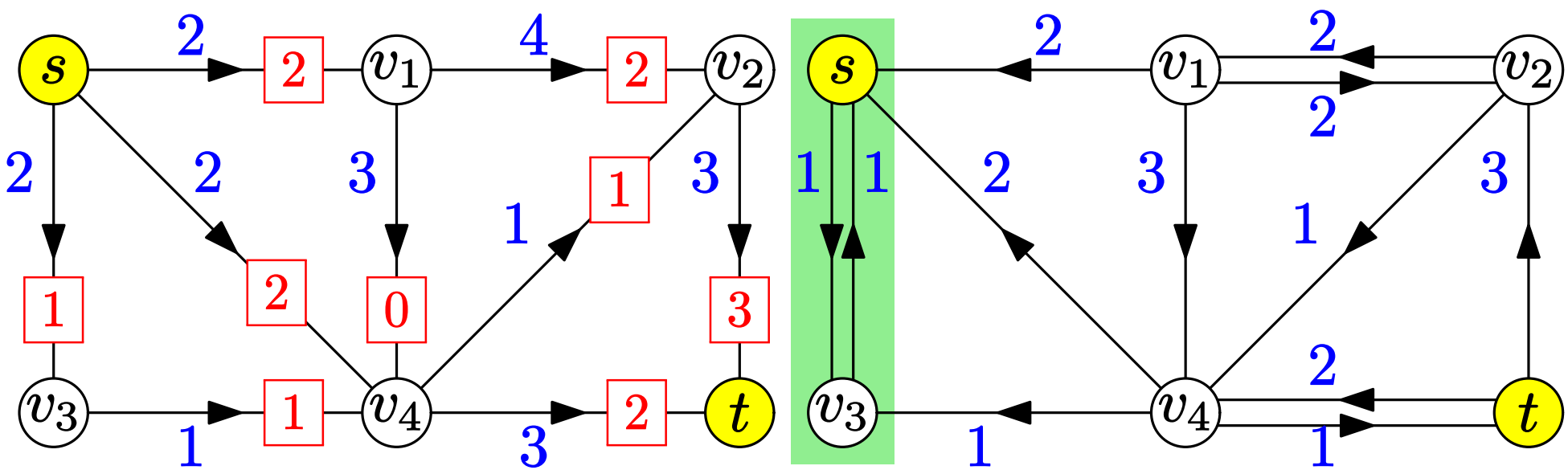
設定：有向グラフ $G = (V, A)$，2 頂点 $s, t \in V$，
弧容量関数 u: $A \to \mathbb{R}_+$，s-t 流 f: $A \to \mathbb{R}_+$

性質：最大流問題の最適性条件

f が最大 s-t 流 $\Leftrightarrow$ f に対する増加道が存在しない

特に，最大 s-t 流 f と最小容量の s-t カット S に対して

$$\mathsf{val}(f) = \mathsf{cap}(S)$$



[復習] 最適性条件と増加道法

設定：有向グラフ $G = (V, A)$，2 頂点 $s, t \in V$，
弧容量関数 $u\colon A \to \mathbb{R}_+$，$s$-$t$ 流 $f\colon A \to \mathbb{R}_+$

性質：最大流問題の最適性条件

f が最大 s-t 流 $\Leftrightarrow$ f に対する増加道が存在しない

性質：増加道法と最大 s-t 流

増加道法が停止する $\Rightarrow$
出力 f は (G, u) に対する最大 s-t 流

注：増加道法は停止しないかもしれない $\rightsquigarrow$ 工夫が必要

容量が整数のとき

性質：整数容量に対する増加道法の停止性

任意の $a \in A$ に対して，$u(a)$ が整数である $\Rightarrow$
増加道法は停止する

証明：s-t 流 f に対して，増加道 P に沿った増加を行い，
s-t 流 f' が得られたとする

- P に沿った増加の増加量は 1 以上 ($\because$ 容量が整数)
- $\mathrm{val}(f) + 1 \leq \mathrm{val}(f')$

- $\therefore$ 止まらないと s-t 流の値が発散してしまい，矛盾 □

容量が整数のとき

性質：整数容量に対する増加道法の停止性

任意の $a \in A$ に対して，$u(a)$ が整数である $\Rightarrow$
増加道法は停止する

証明：s-t 流 f に対して，増加道 P に沿った増加を行い，
s-t 流 f' が得られたとする

- P に沿った増加の増加量は 1 以上 ($\because$ 容量が整数)
- $\mathrm{val}(f) + 1 \leq \mathrm{val}(f')$

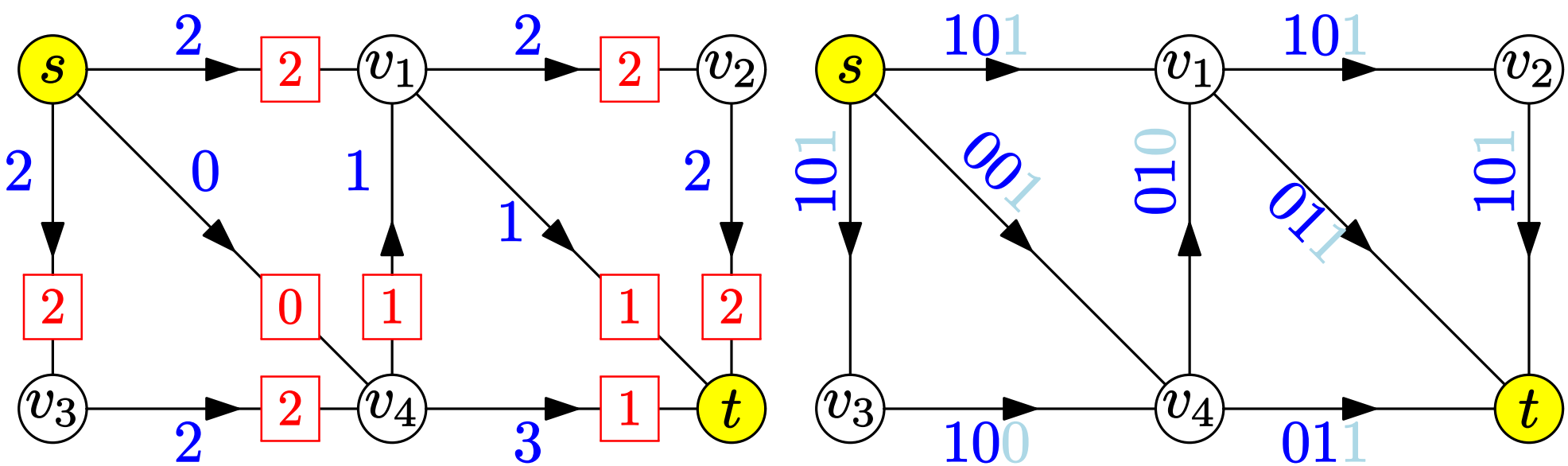
- $\therefore$ 止まらないと s-t 流の値が発散してしまい，矛盾 □

性質：整数容量に対する増加道法の反復回数

このとき，増加道法の反復回数は O(最適値)

本日の内容

1. **アルゴリズムの計算量：分類**
2. 容量スケーリング法：概要
3. 容量スケーリング法：計算量



入力のサイズ

最大流問題の入力

- 有向グラフ G の頂点集合 V
- 有向グラフ G の弧集合 A
- 2 頂点 $s, t \in V$

- 弧容量関数 $u\colon A \to \mathbb{Z}_+$

入力のサイズ

最大流問題の入力

- 有向グラフ G の頂点集合 V
- 有向グラフ G の弧集合 A
- 2 頂点 $s, t \in V$
- 弧容量関数 $u\colon A \to \mathbb{Z}_+$

サイズ (ビット)

- $O(|V| \log |V|)$
- $O(|A| \log |V|)$
- $O(\log |V|)$
- $O\left(\sum_{a \in A} \log u(a)\right)$

入力のサイズ

最大流問題の入力

- 有向グラフ G の頂点集合 V
- 有向グラフ G の弧集合 A
- 2 頂点 $s, t \in V$
- 弧容量関数 $u\colon A \to \mathbb{Z}_+$

サイズ (ビット)

- $O(|V| \log |V|)$
- $O(|A| \log |V|)$
- $O(\log |V|)$
- $O\left(\sum_{a \in A} \log u(a)\right)$

$= O(|A| \log U)$

ただし，$U = \max_{a \in A} u(a)$

入力のサイズ

最大流問題の入力

- 有向グラフ G の頂点集合 V
- 有向グラフ G の弧集合 A
- 2 頂点 $s, t \in V$
- 弧容量関数 $u\colon A \to \mathbb{Z}_+$

サイズ (ビット)

- $O(|V| \log |V|)$
- $O(|A| \log |V|)$
- $O(\log |V|)$
- $O\left(\sum_{a \in A} \log u(a)\right)$

$$= O(|A| \log U)$$

ただし，$U = \max_{a \in A} u(a)$

最大流問題のアルゴリズムの計算量は

$|V|, |A|, U$　　(以後，$n = |V|, m = |A|$ とする)

に対する依存性で測ることにする

3 種類の計算時間

最大流問題に対して

定義：強/弱/擬多項式時間アルゴリズム

{ **強多項式時間** / **弱多項式時間** / **擬多項式時間** } アルゴリズム とは 計算時間が

{ n, m / $n, m, \log U$ / n, m, U } の多項式で抑えられるアルゴリズム

注：強多項式時間 $\Rightarrow$ 弱多項式時間 $\Rightarrow$ 擬多項式時間

3 種類の計算時間：例

	強多項式時間	弱多項式時間	擬多項式時間
$O(nm^2)$			
$O(m^2 \log U)$			
$O(mU)$			

3 種類の計算時間：例

	強多項式時間	弱多項式時間	擬多項式時間
$O(nm^2)$	○	○	○
$O(m^2 \log U)$			
$O(mU)$			

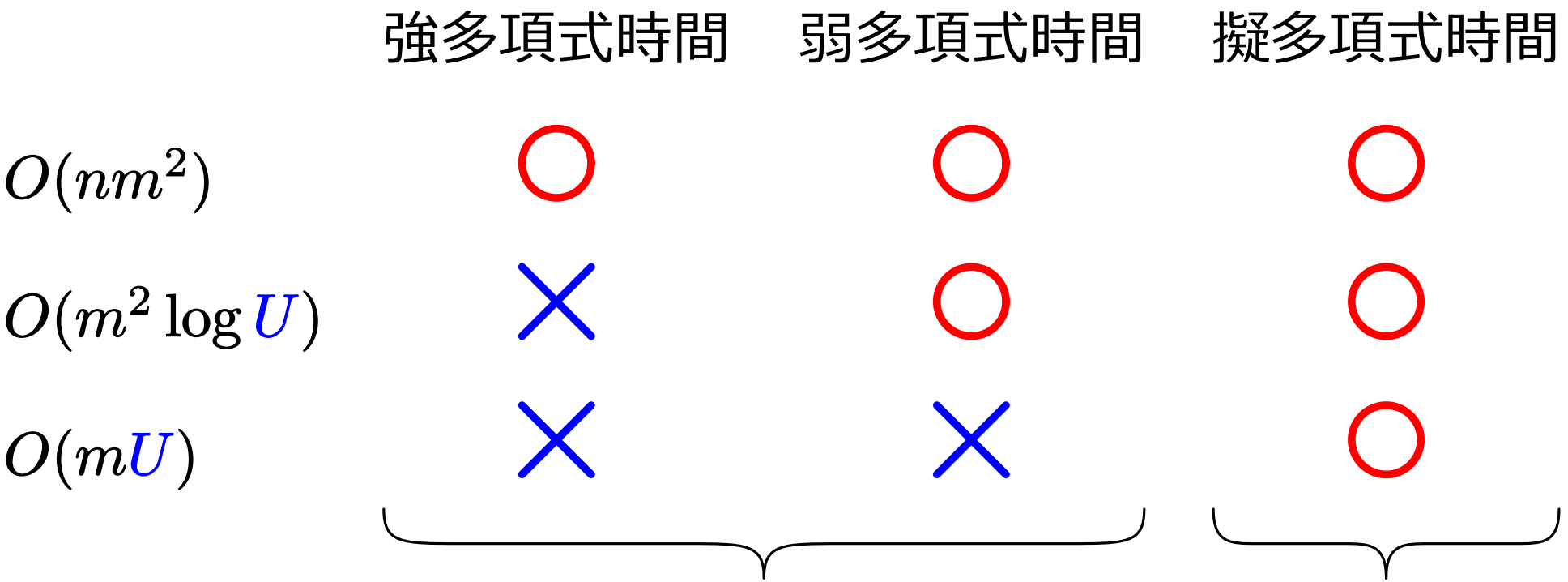
3 種類の計算時間：例

	強多項式時間	弱多項式時間	擬多項式時間
$O(nm^2)$	○	○	○
$O(m^2 \log U)$	×	○	○
$O(mU)$			

3 種類の計算時間：例

	強多項式時間	弱多項式時間	擬多項式時間
$O(nm^2)$	○	○	○
$O(m^2 \log U)$	×	○	○
$O(mU)$	×	×	○

3 種類の計算時間：例


強多項式時間
弱多項式時間
擬多項式時間
$O(nm^2)$
$O(m^2 \log U)$
$O(mU)$
入力のサイズに関する
多項式時間アルゴリズム
である
とは言えない

増加道法の計算量

増加道法の計算量

(工夫のない) 増加道法は

- 容量が実数のとき，止まらないことがある
- 容量が整数のとき，必ず止まり，計算量は $O(m^2U)$

計算量 = 反復回数 × 各反復の計算量

$O($最適値$)$　　$O(m)$

$= O(mU)$

つまり，(工夫のない) 増加道法は，

容量が整数のとき，**擬多項式時間** アルゴリズム

Edmonds-Karp のアルゴリズムの計算量

Edmonds-Karp のアルゴリズムの計算量

Edmonds-Karp のアルゴリズムは
容量が実数であっても必ずとまり，計算量は $O(nm^2)$

つまり，Edmonds-Karp のアルゴリズムは，

容量が実数であっても，**強多項式時間** アルゴリズム

容量スケーリング法：特徴

容量スケーリング法の特徴

- 容量が整数のときのみ，実行できる
- **弱多項式時間** アルゴリズムである
- アルゴリズムの解析が 簡単である

容量スケーリング法：特徴

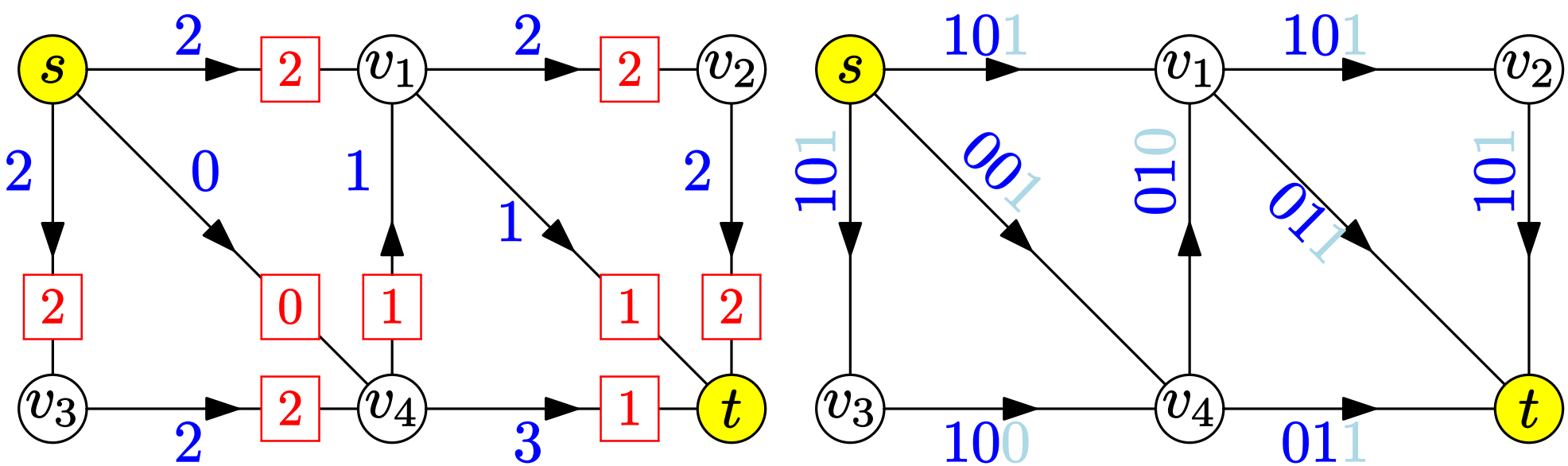
容量スケーリング法の特徴

- 容量が整数のときのみ，実行できる
- **弱多項式時間** アルゴリズムである
- アルゴリズムの解析が 簡単である

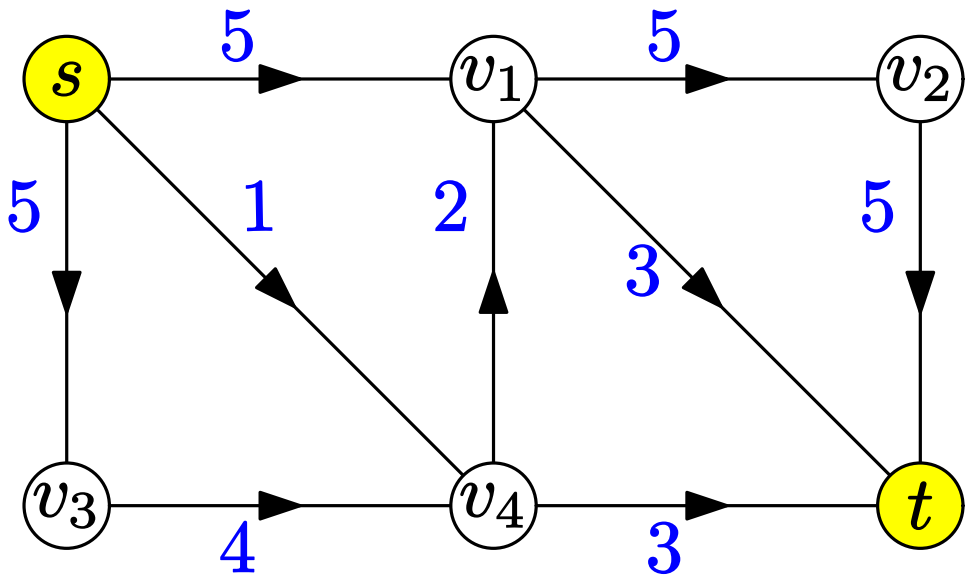
Edmonds-Karp のアルゴリズムに比べて

本日の内容

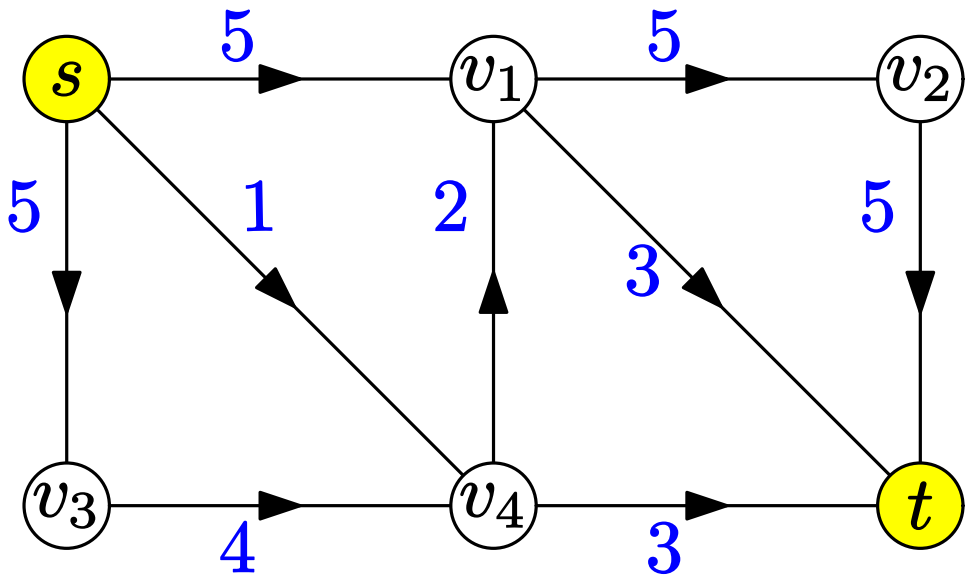
1. アルゴリズムの計算量：分類
2. **容量スケーリング法：概要**
3. 容量スケーリング法：計算量

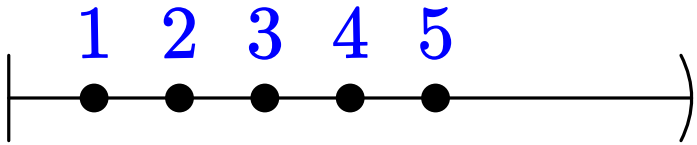


容量スケーリング法：例

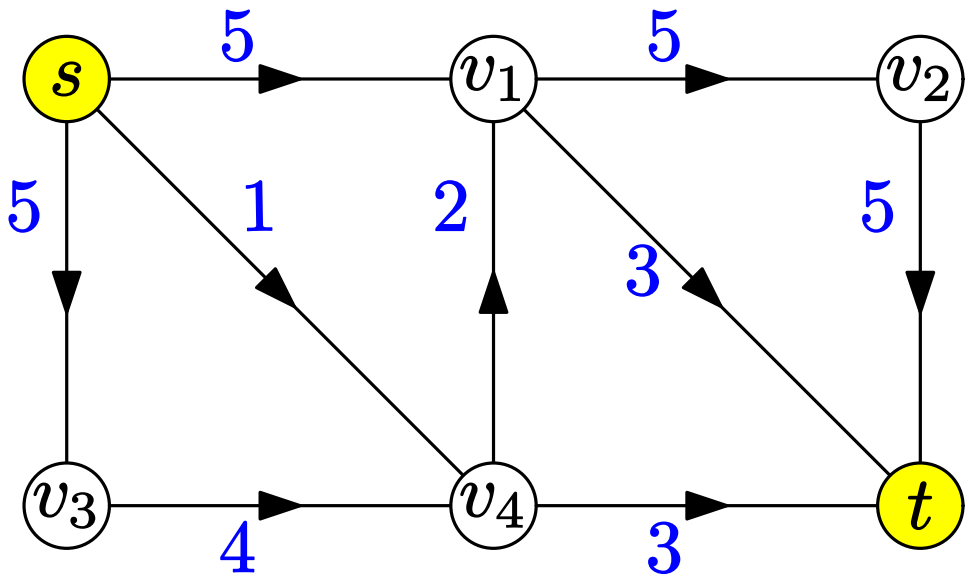


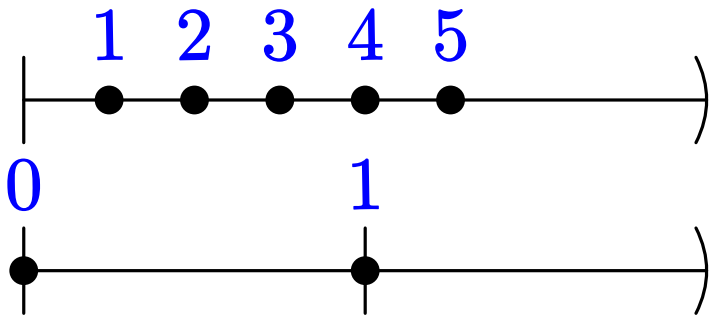
容量スケーリング法：例

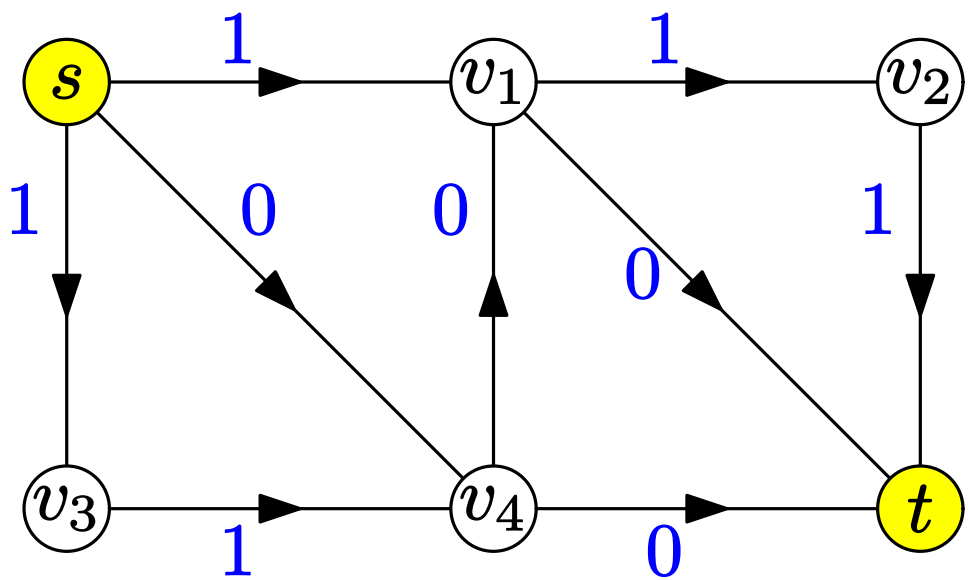




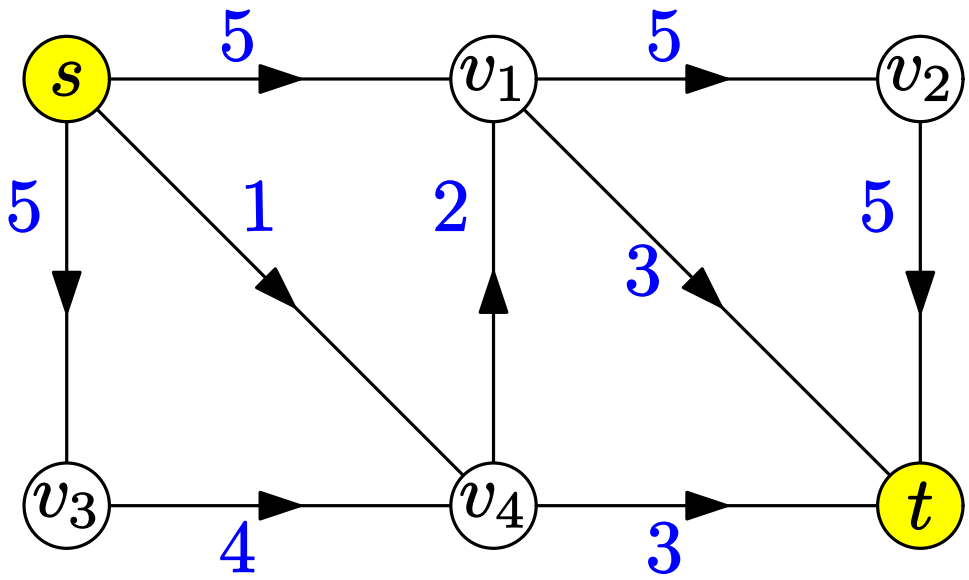
容量スケーリング法：例

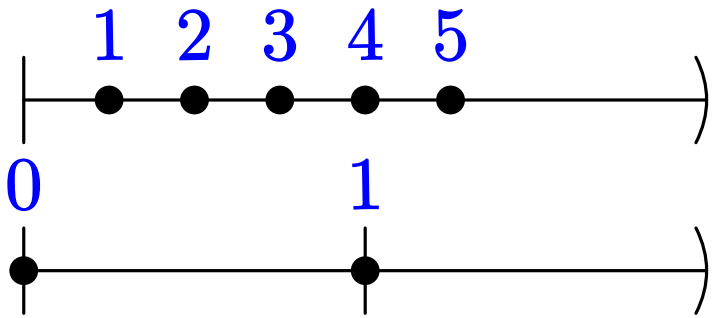


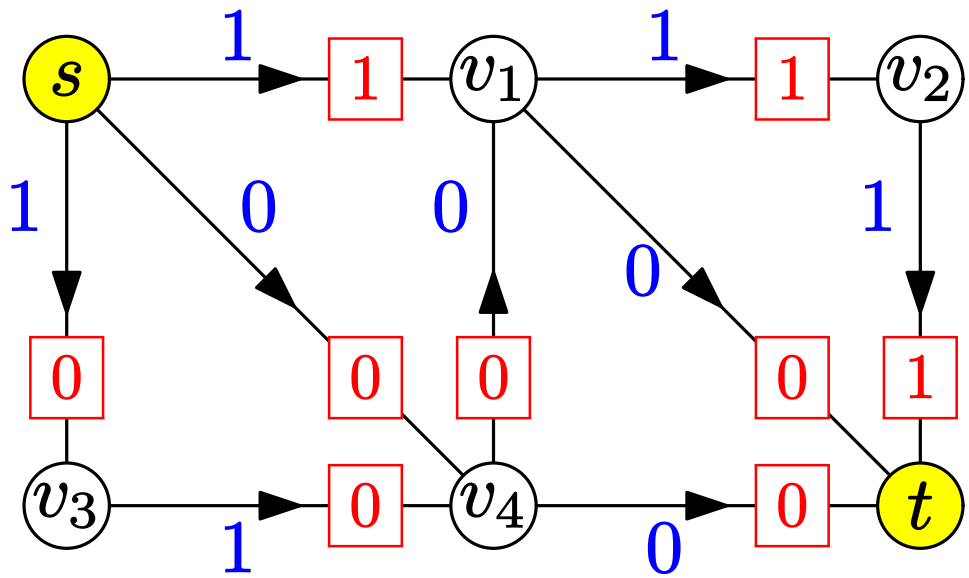




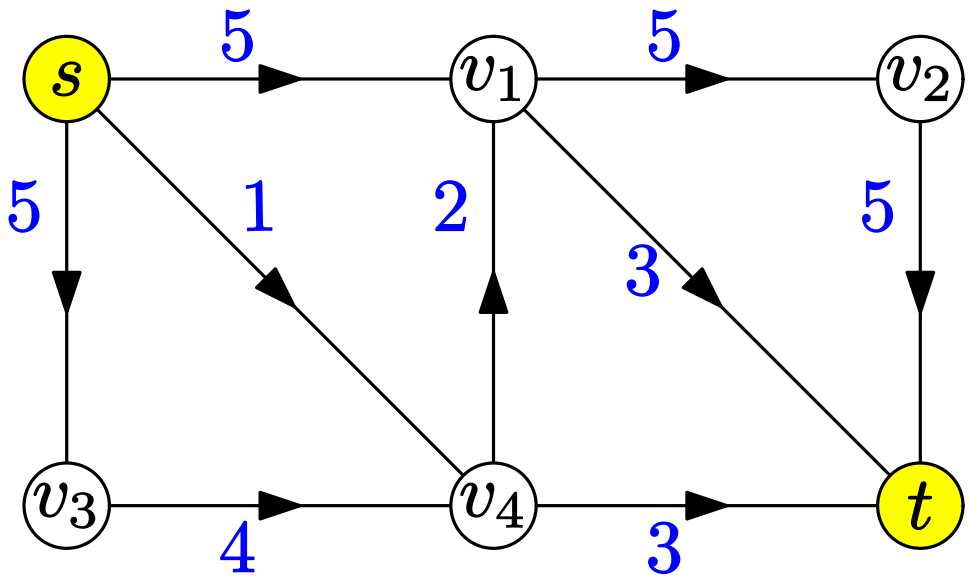
容量スケーリング法：例

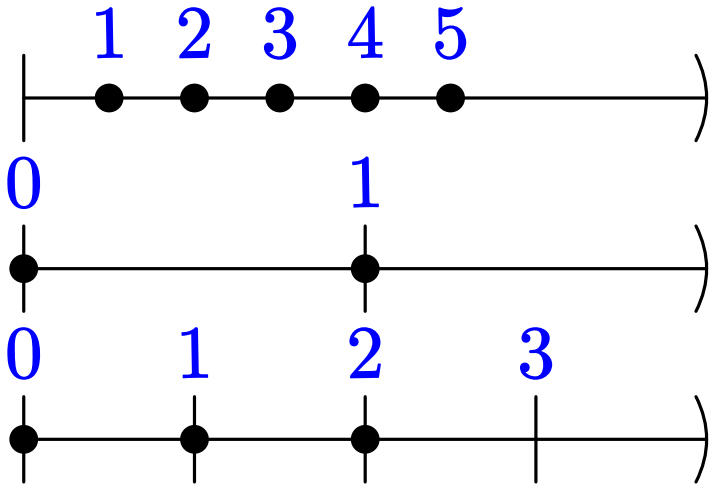

s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
3

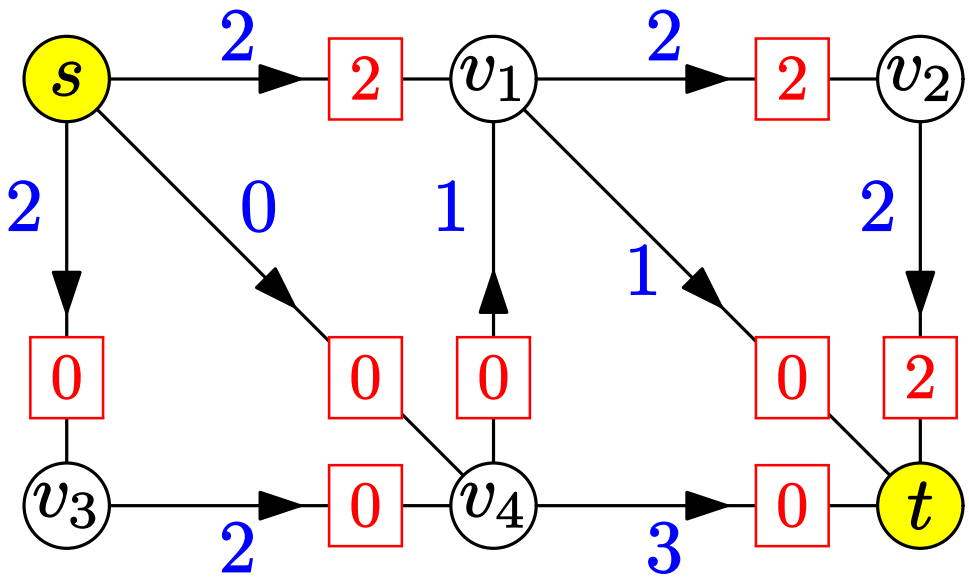

1
2
3
4
5
0
1


s
v1
v2
v3
v4
t
1
1
1
1
0
0
0
1
0
0
0
0
1
0
0
1

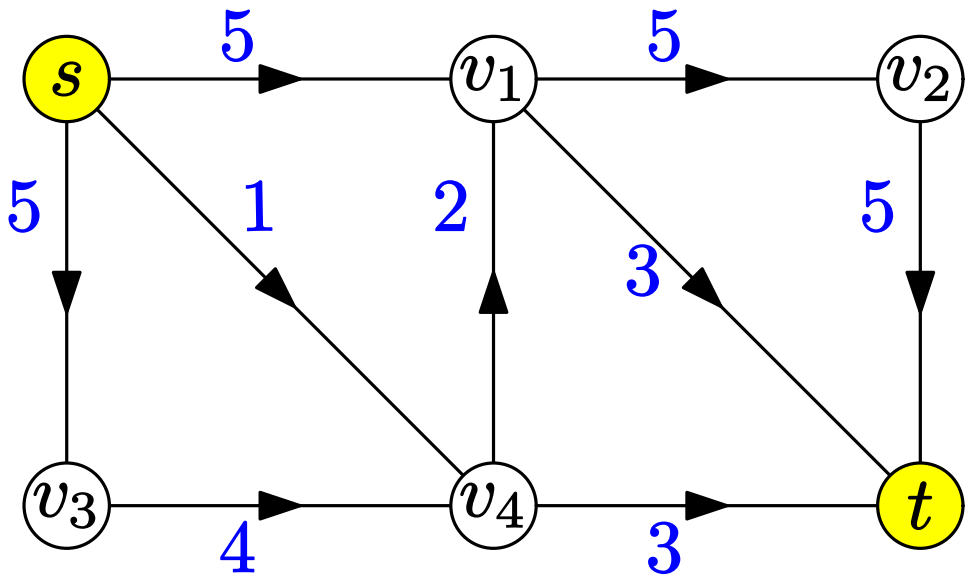
容量スケーリング法：例

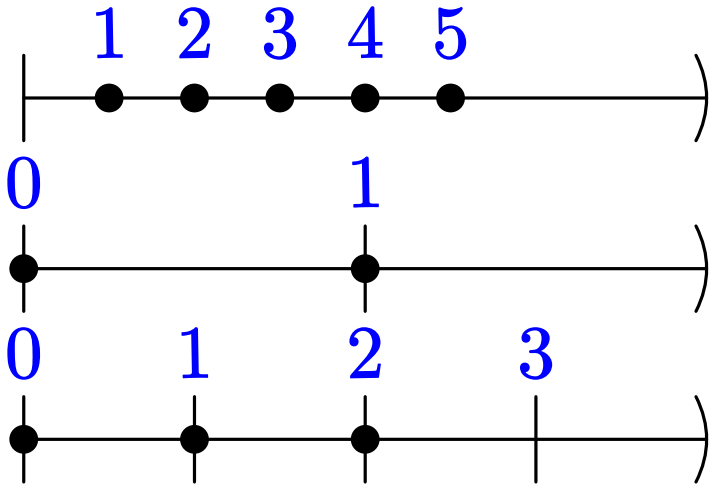

s
v1
v2
v3
v4
t
5
5
5
1
2
5
3
4
3

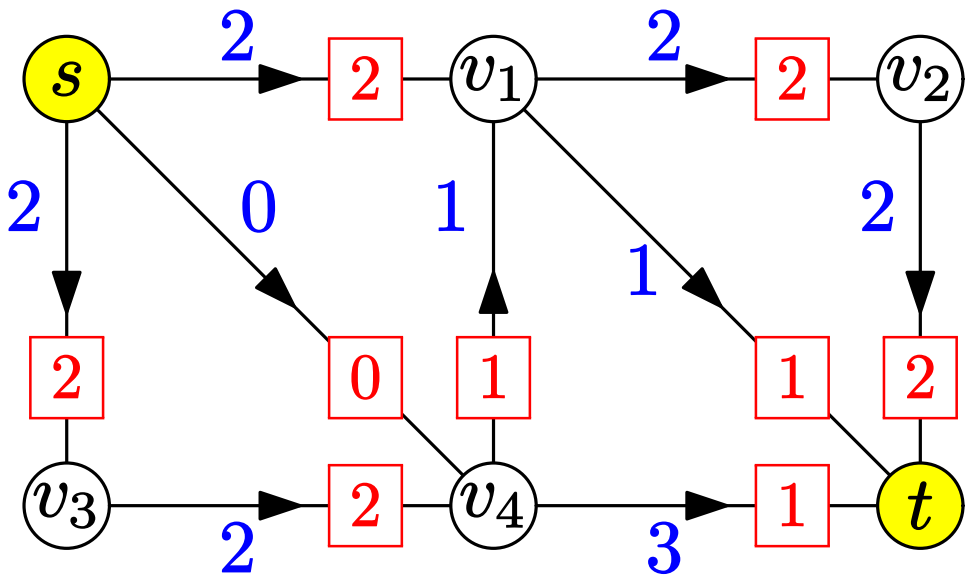

1 2 3 4 5
0 1
0 1 2 3


s
v1
v2
v3
v4
t
2
2
2
2
0
1
2
1
0
0
0
0
2
0
0
2
3

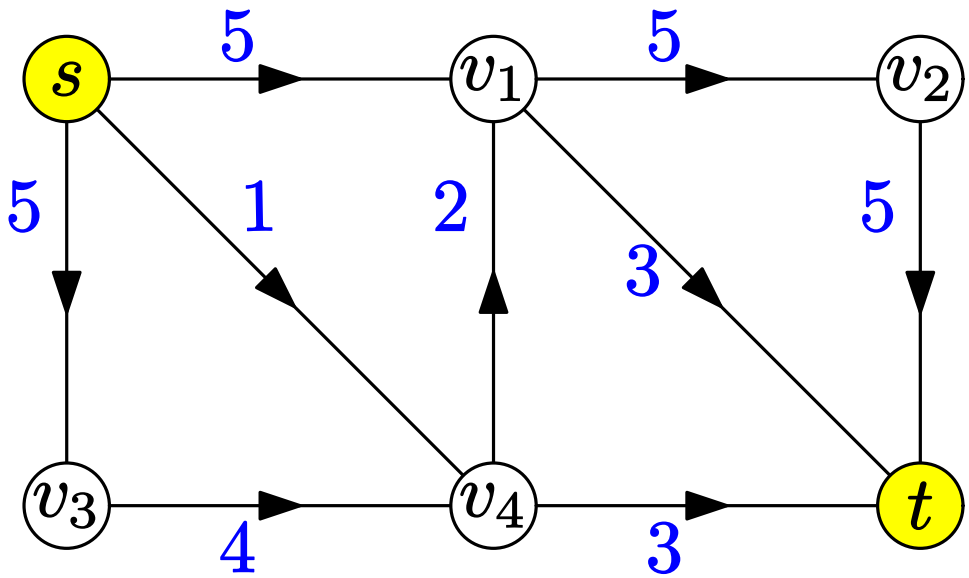
容量スケーリング法：例

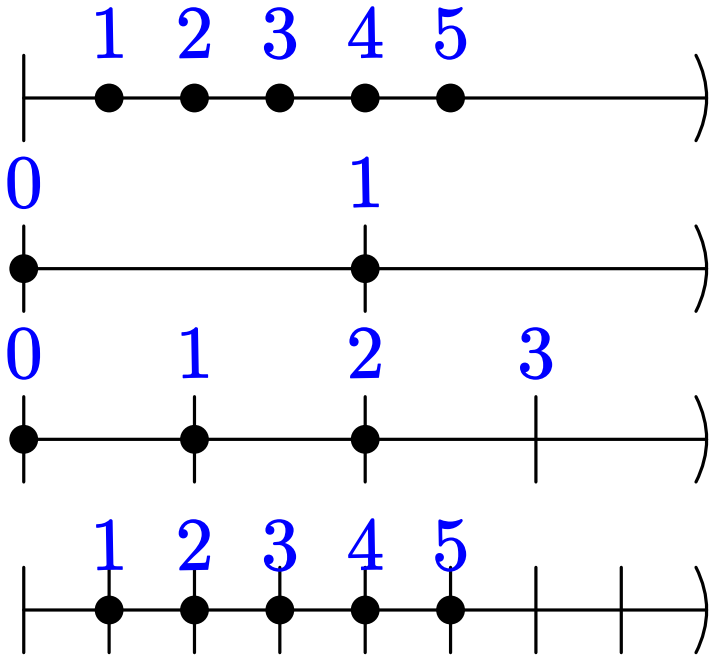

s
v1
v2
v3
v4
t
5
5
5
1
2
5
3
4
3

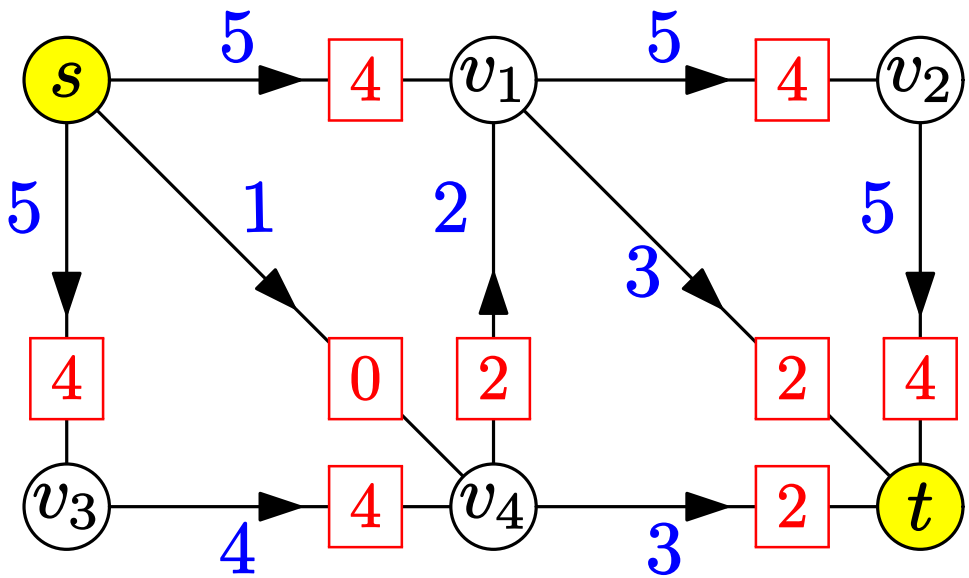

1 2 3 4 5
0 1
0 1 2 3


s
v1
v2
v3
v4
t
2
2
2
2
0
1
1
2
2
0
1
1
2
2
3

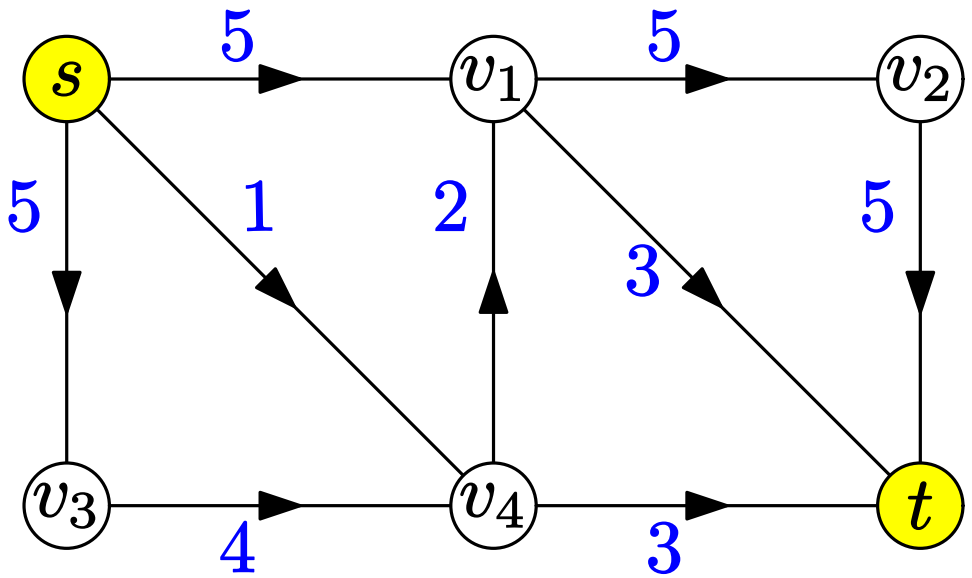
容量スケーリング法：例

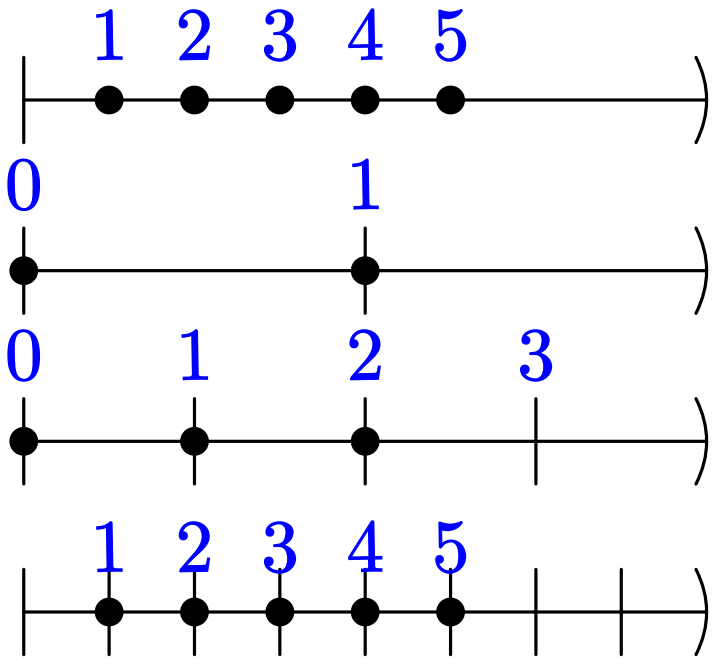

s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
3

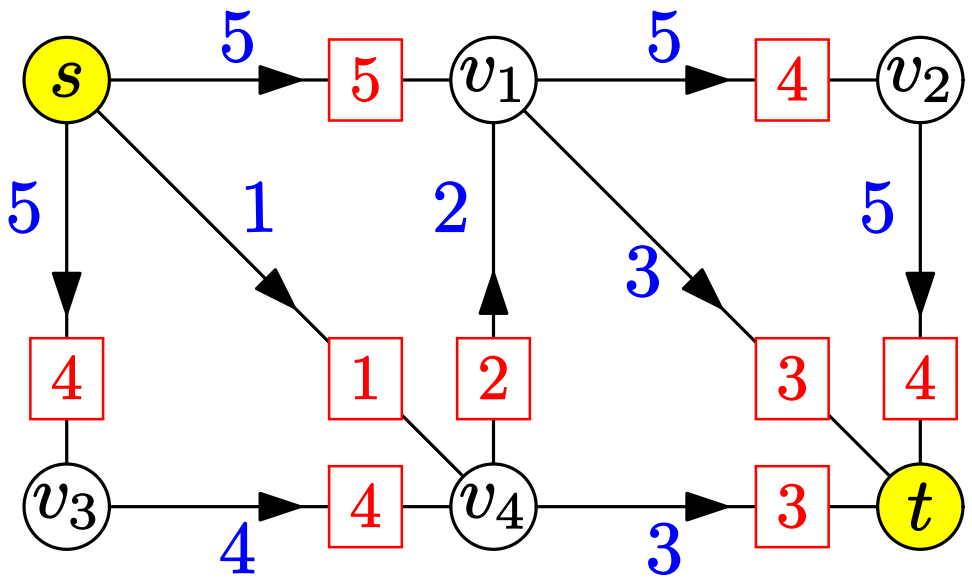

1 2 3 4 5
0 1
0 1 2 3
1 2 3 4 5


s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
4
0
2
2
4
4
4
2
4
3

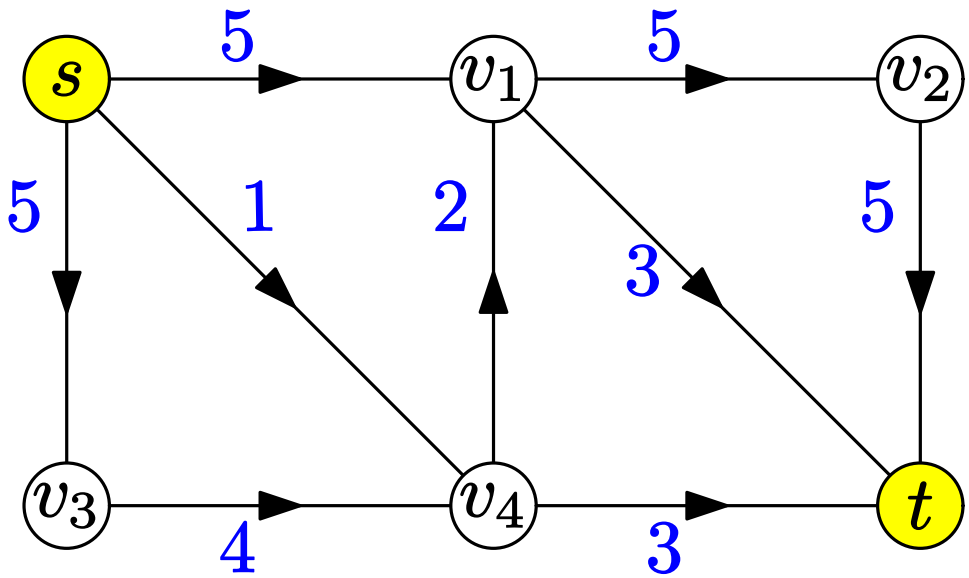
容量スケーリング法：例

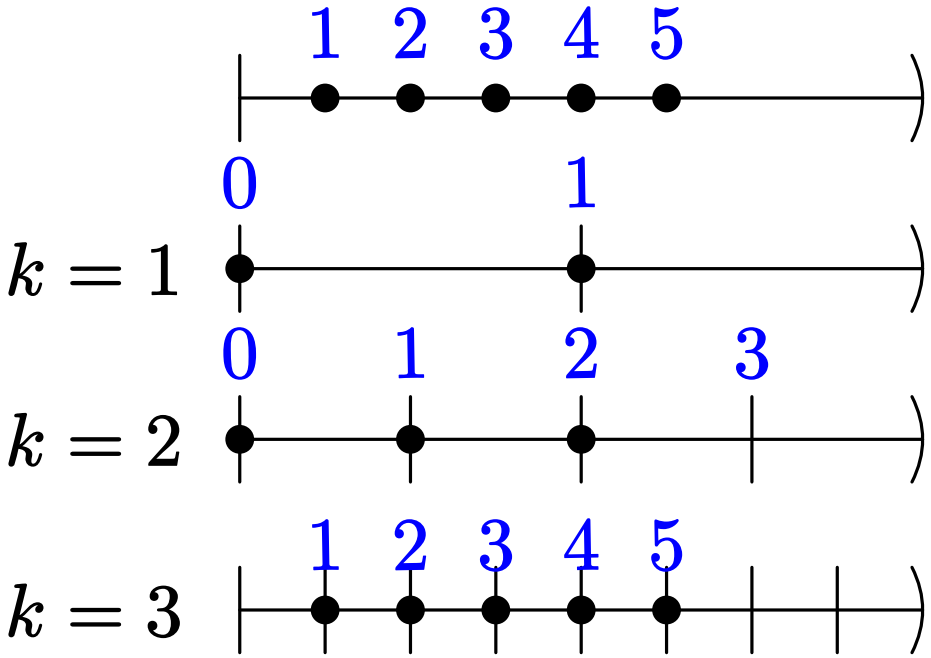

s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
3

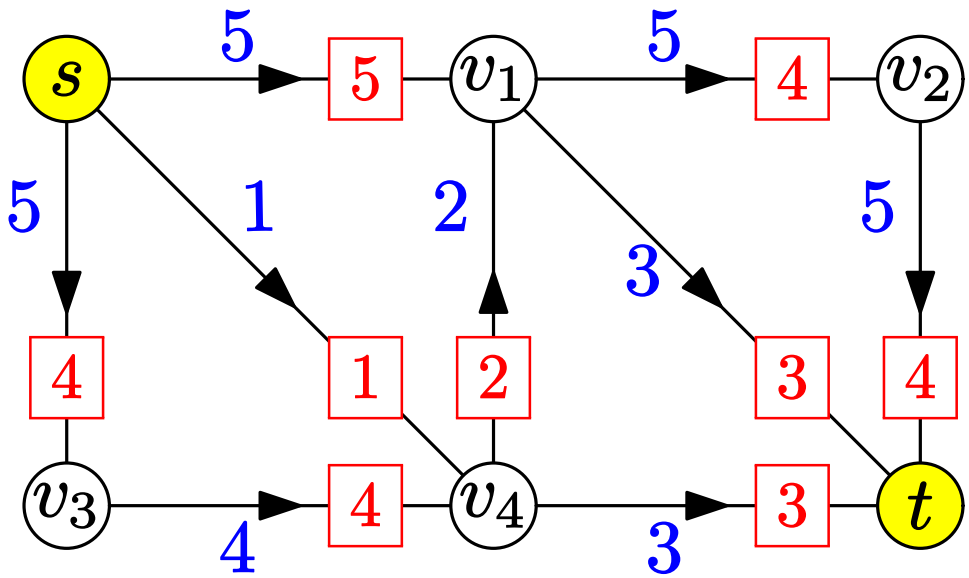

1 2 3 4 5
0 1
0 1 2 3
1 2 3 4 5


s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
3
5
4
1
2
3
4
4
3

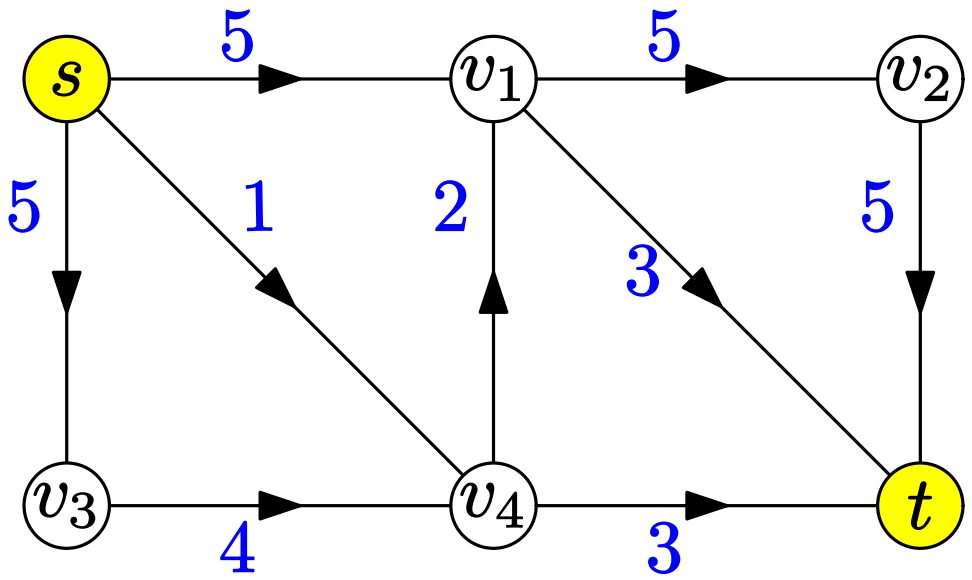
容量スケーリング法：例

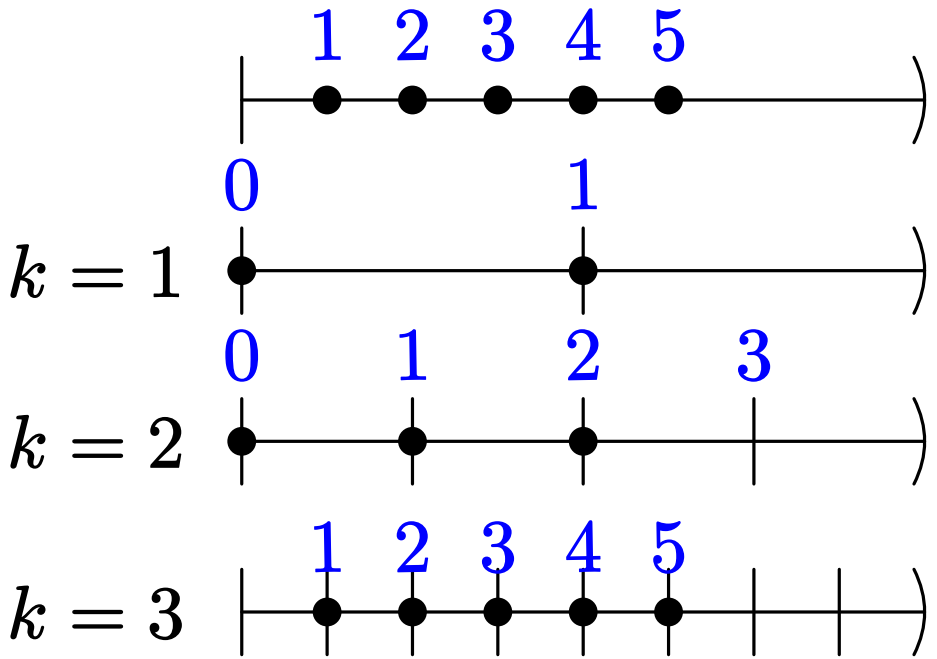

s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
3

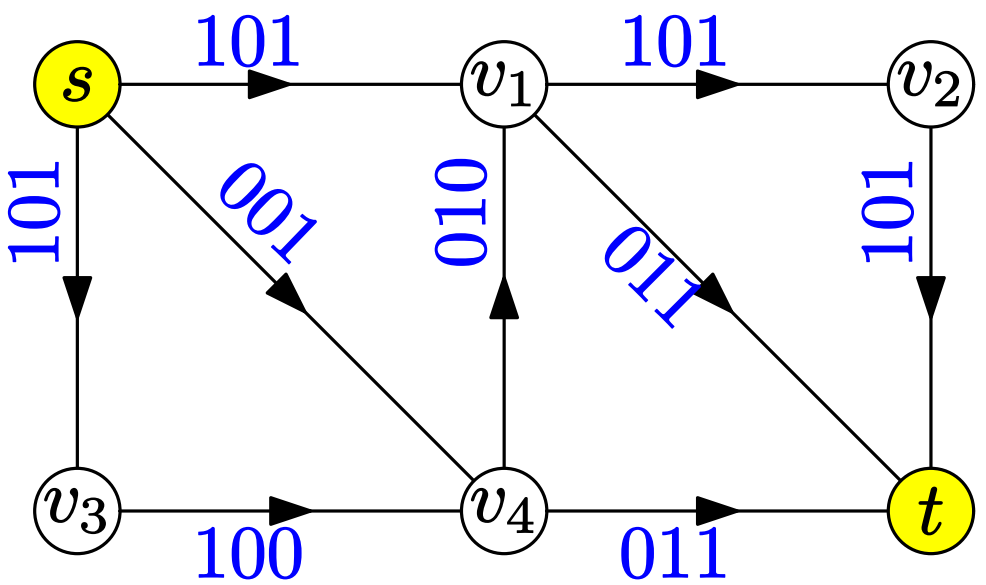

1 2 3 4 5
0 1
k = 1
0 1 2 3
k = 2
1 2 3 4 5
k = 3


s
v1
v2
v3
v4
t
5
5
5
1
2
3
5
4
3
5
4
1
2
3
4
4
3

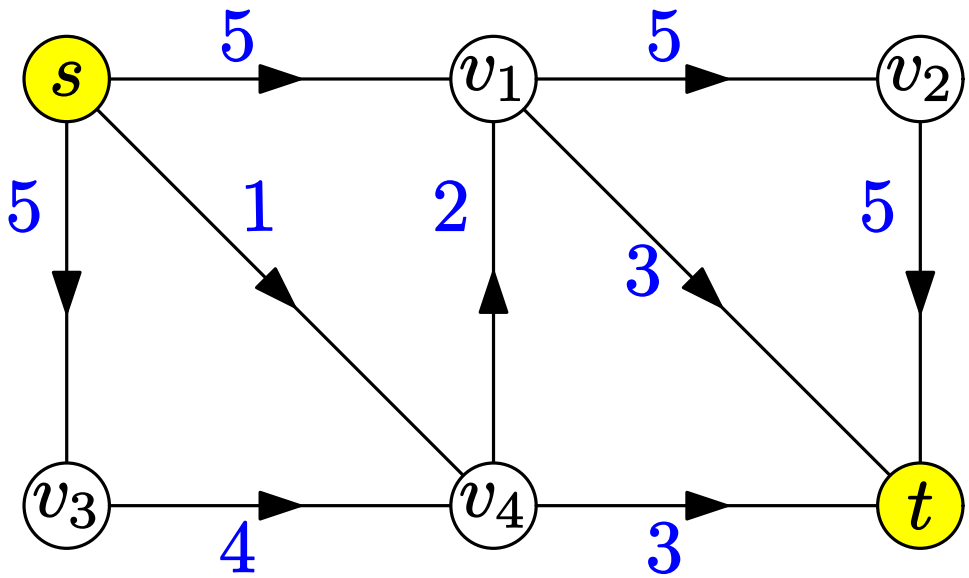
容量スケーリング法：別の見方

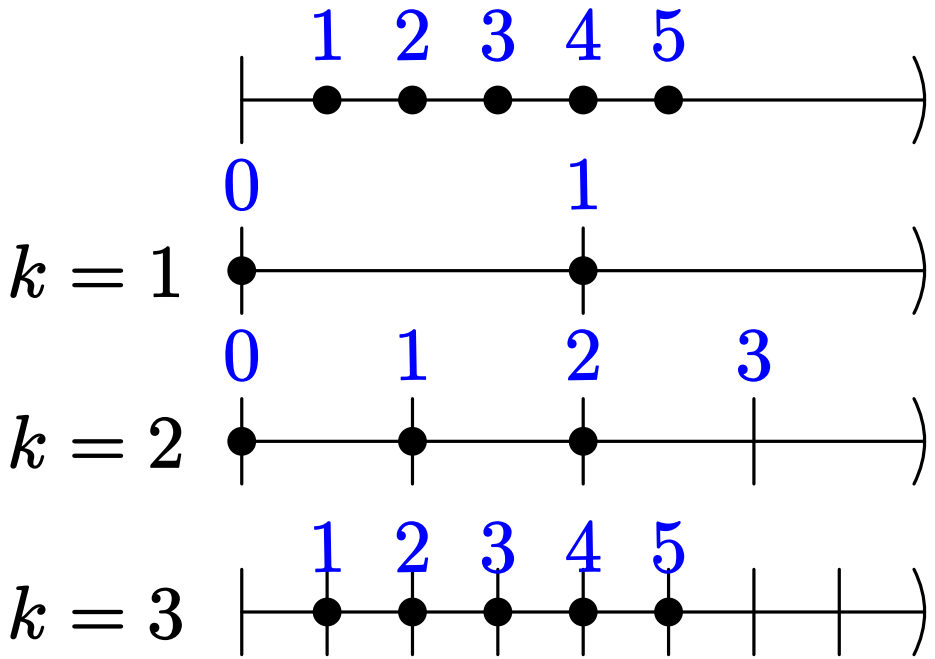


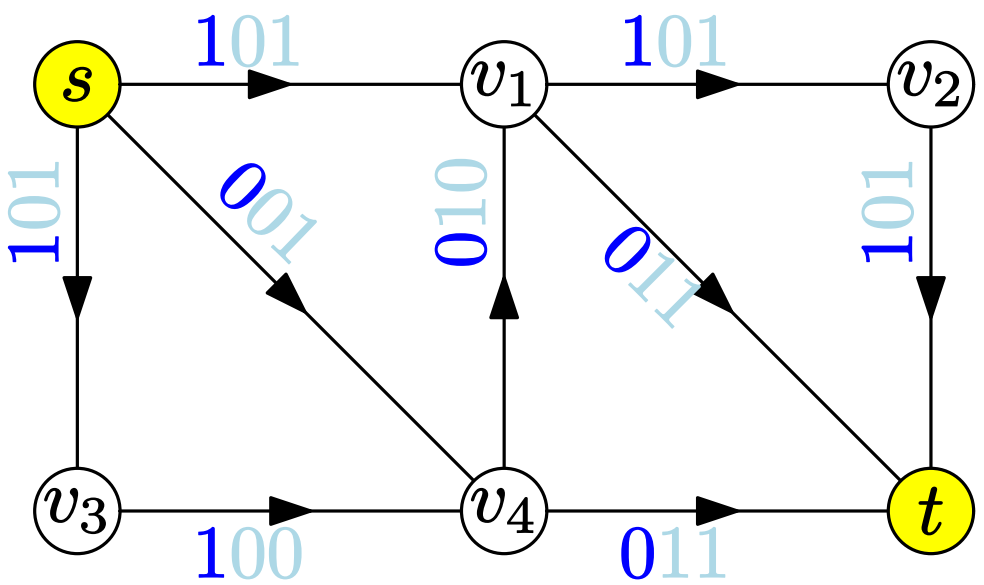




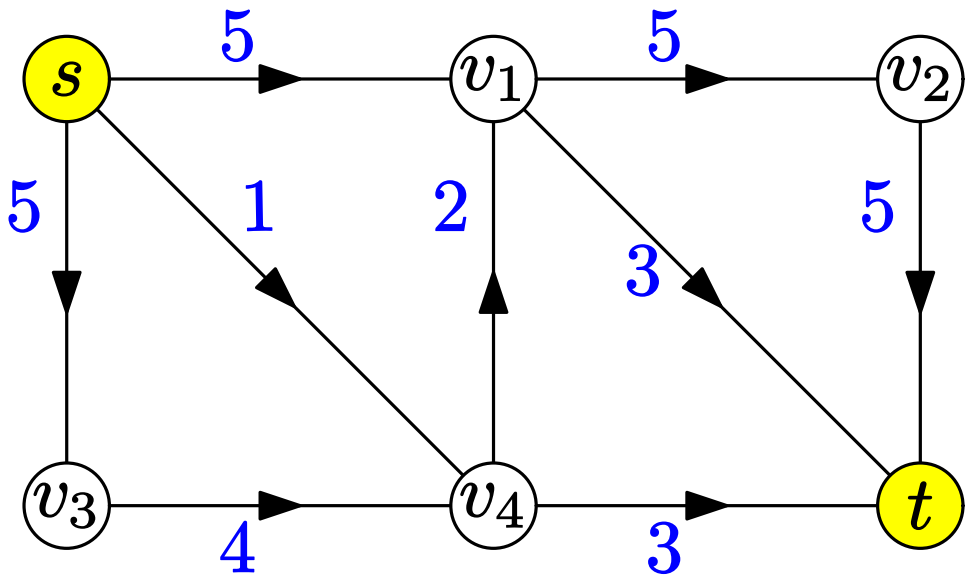
容量スケーリング法：別の見方

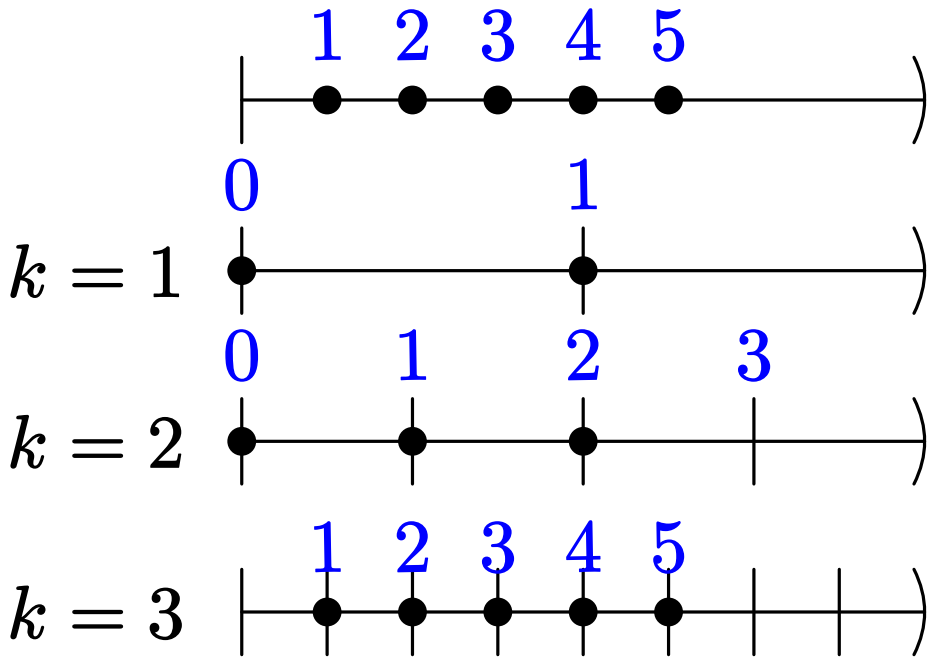


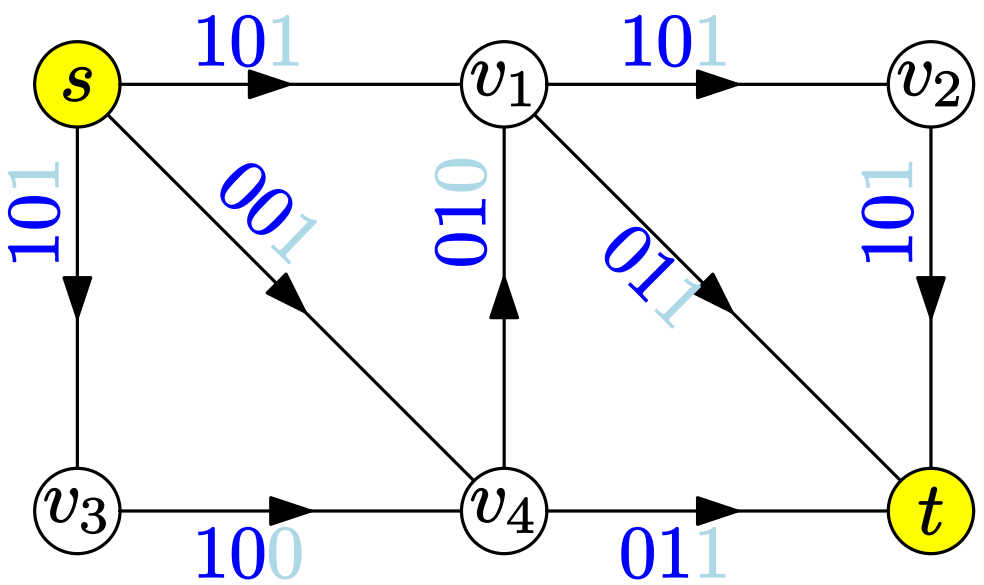




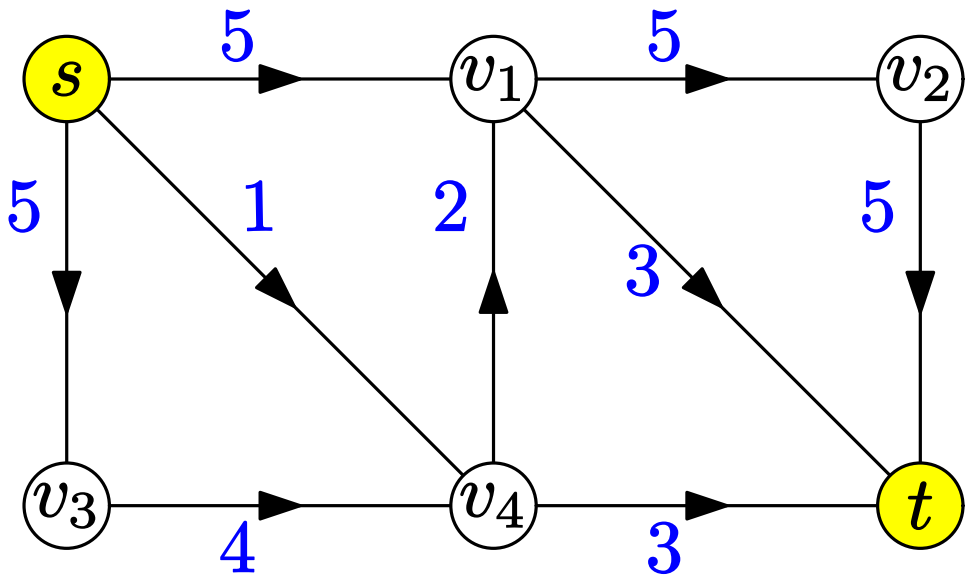
容量スケーリング法：別の見方

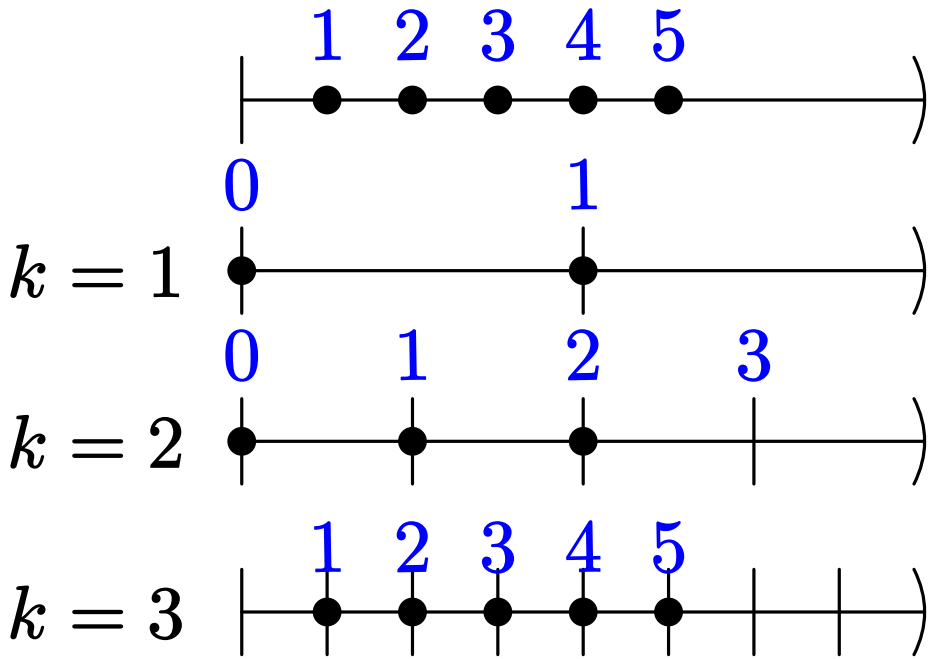


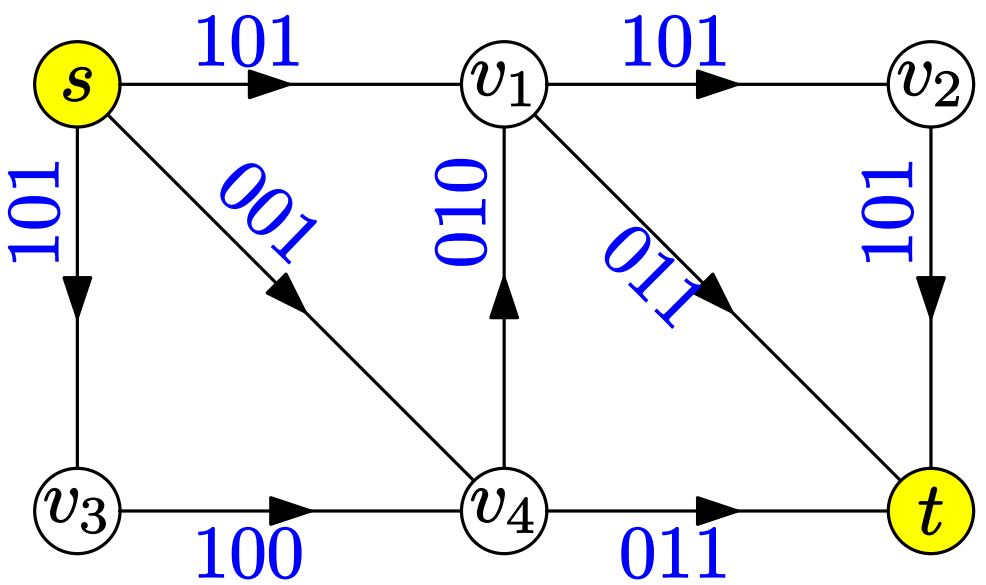




容量スケーリング法：別の見方







容量スケーリング法

(capacity scaling algorithm)

アルゴリズム：容量スケーリング (Gabow '85)

- 初期化：s-t 流 $f_0 = 0$，$K = \lfloor 1 + \log_2 U \rfloor$
- 反復：$k = 1, 2, \ldots, K$ に対して順に以下を実行する
  1. 各 $a \in A$ に対して $u_k(a) = \lfloor u(a)/2^{K-k} \rfloor$ とする
  2. $2f_{k-1}$ を初期 s-t 流として，増加道法で
     ネットワーク (G, u_k) の最大 s-t 流 f_k を見つける
- 出力：f_K

一般的に，スケーリング法といったら

数値を見る「目盛り」を順に変えることで
最適解に「近い」解を順に得ていく手法

容量スケーリング法：正当性

性質：容量スケーリング法の正当性

容量スケーリング法が停止する ⇒
その出力 f_K は，(G, u) の最大 s-t 流である

アルゴリズム：容量スケーリング (Gabow '85)

- 初期化：s-t 流 $f_0 = 0$，$K = \lfloor 1 + \log_2 U \rfloor$
- 反復：$k = 1, 2, \ldots, K$ に対して順に以下を実行する
  1. 各 $a \in A$ に対して $u_k(a) = \lfloor u(a)/2^{K-k} \rfloor$ とする
  2. $2f_{k-1}$ を初期 s-t 流として，増加道法で
     ネットワーク (G, u_k) の最大 s-t 流 f_k を見つける
- 出力：f_K

容量スケーリング法：正当性

性質：容量スケーリング法の正当性

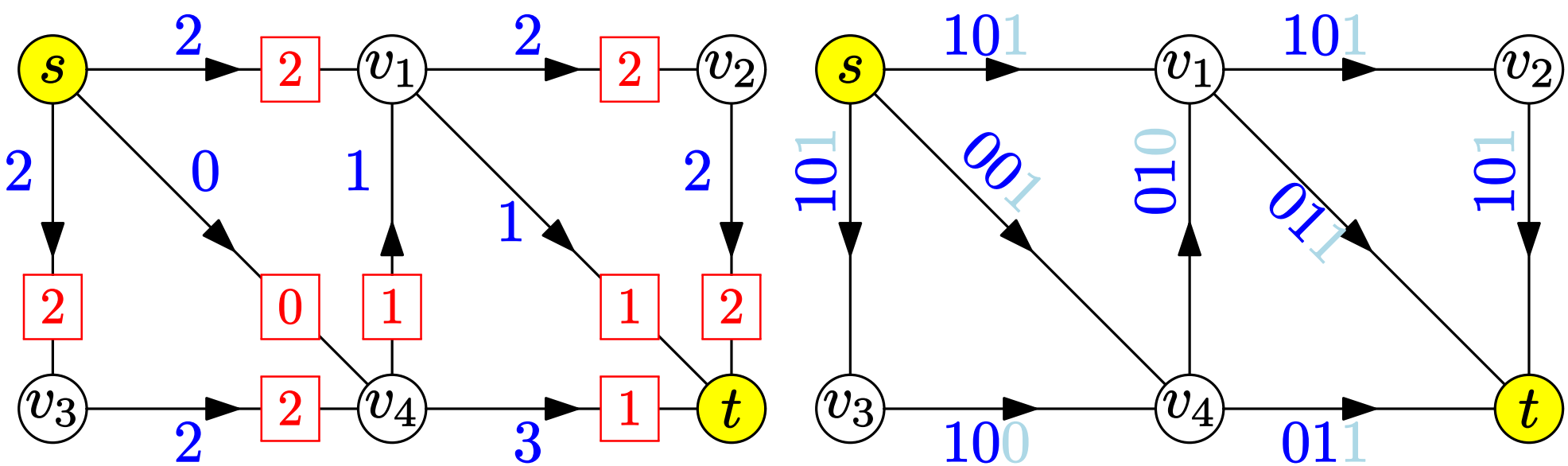
容量スケーリング法が停止する ⇒
その出力 f_K は，(G, u) の最大 s-t 流である

証明：

- f_K は (G, u_K) の最大 s-t 流
- ここで，$u_K(a) = \lfloor u(a)/2^{K-K} \rfloor = u(a)$
- $\therefore f_K$ は (G, u) の最大 s-t 流 □

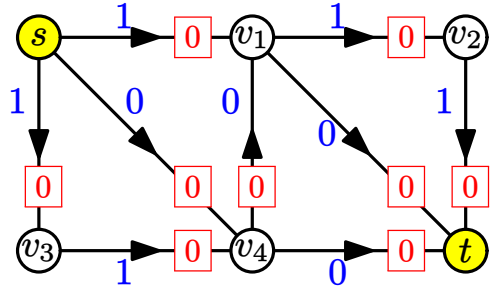
本日の内容

1. アルゴリズムの計算量：分類
2. 容量スケーリング法：概要
3. **容量スケーリング法：計算量**

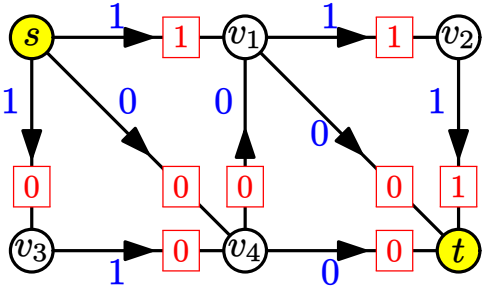


計算量に向けて：記法

$k = 1 \quad (G, u_1)$

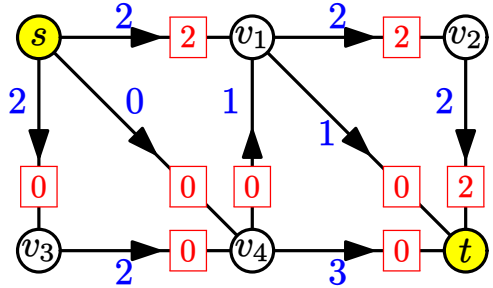


$2f_0$

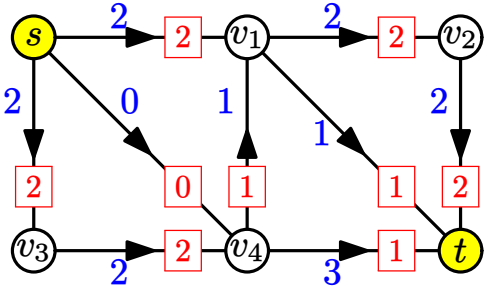


f_1

$k = 2 \quad (G, u_2)$

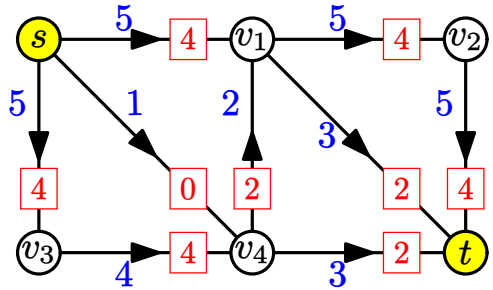


$2f_1$

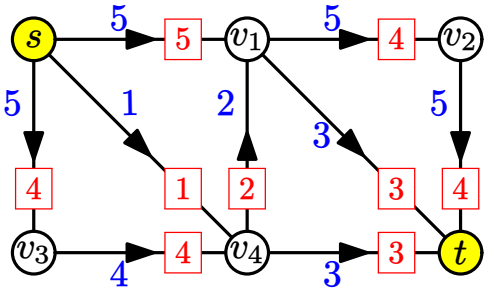


f_2

$k = 3 \quad (G, u_3)$



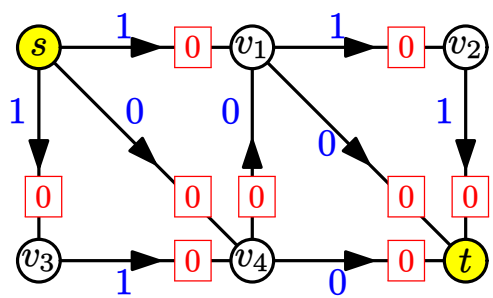
$2f_2$



f_3

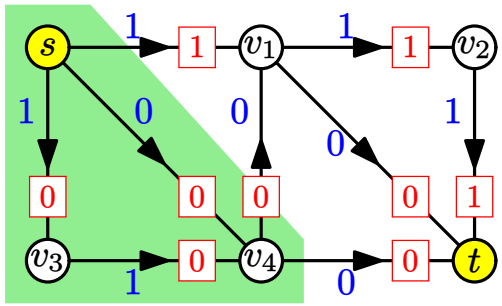
計算量に向けて：記法

$k = 1 \quad (G, u_1)$



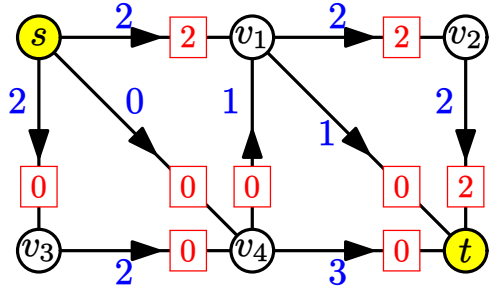
$2f_0$

S_1



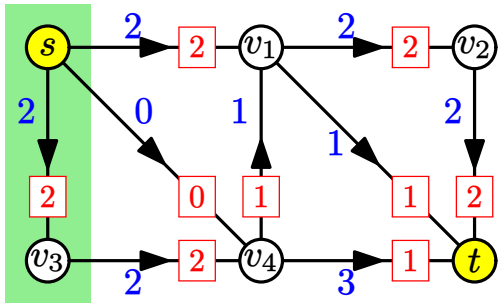
f_1

$k = 2 \quad (G, u_2)$



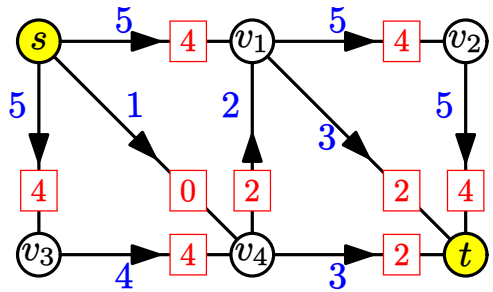
$2f_1$

S_2



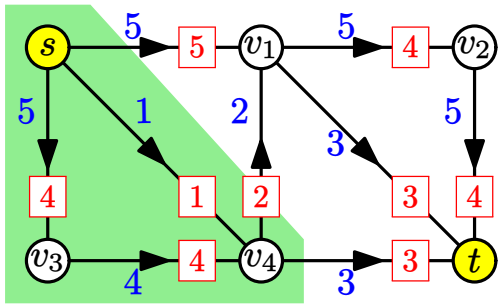
f_2

$k = 3 \quad (G, u_3)$



$2f_2$

S_3



f_3

第１反復の容量

性質：第１反復の容量

任意の弧 $a \in A$ に対して，$u_1(a) \in \{0, 1\}$

証明：$K = \lfloor 1 + \log_2 U \rfloor, U = \max_{a \in A} u(a)$ を思い出す

$$2^{K-1} = 2^{\lfloor 1+\log_2 U \rfloor - 1} > 2^{(\log_2 U)-1} = \frac{1}{2}U$$

$\lfloor 1 + x \rfloor > x$

$$\therefore \quad 2^K \geq U + 1$$

したがって，

$$u_1(a) = \lfloor u(a)/2^{K-1} \rfloor \geq 0,$$

$$u_1(a) = \lfloor u(a)/2^{K-1} \rfloor \leq \lfloor 2u(a)/(U+1) \rfloor$$

$$\leq \lfloor 2u(a)/(u(a)+1) \rfloor = \lfloor 2 - (2/(u(a)+1)) \rfloor \leq 1$$

□

容量の変化

性質：容量の変化

任意の $k = 2, \ldots, K$ と任意の弧 $a \in A$ に対して，

$$2u_{k-1}(a) \leq u_k(a) \leq 2u_{k-1}(a) + 1$$

証明：任意の非負実数 α に対して，次を証明すれば十分

$$2\lfloor \alpha \rfloor \leq \lfloor 2\alpha \rfloor \leq 2\lfloor \alpha \rfloor + 1$$

(復習)：$u_k(a) = \lfloor u(a)/2^{K-k} \rfloor$

容量の変化

性質：容量の変化

任意の $k = 2, \ldots, K$ と任意の弧 $a \in A$ に対して，

$$2u_{k-1}(a) \leq u_k(a) \leq 2u_{k-1}(a) + 1$$

証明：任意の非負実数 α に対して，次を証明すれば十分

$$2\lfloor \alpha \rfloor \leq \lfloor 2\alpha \rfloor \leq 2\lfloor \alpha \rfloor + 1$$

(復習)：$u_k(a) = \lfloor u(a)/2^{K-k} \rfloor$

- $\lfloor \alpha \rfloor \leq \alpha$ より，$2\lfloor \alpha \rfloor \leq 2\alpha$
- $\lfloor 2\alpha \rfloor$ は 2α 以下の最大整数なので，$2\lfloor \alpha \rfloor \leq \lfloor 2\alpha \rfloor$
- $\alpha < \lfloor \alpha \rfloor + 1$ より，$2\alpha < 2\lfloor \alpha \rfloor + 2$
- $\therefore \lfloor 2\alpha \rfloor < 2\lfloor \alpha \rfloor + 2$
- $\lfloor 2\alpha \rfloor, 2\lfloor \alpha \rfloor + 2$ は整数なので，$\lfloor 2\alpha \rfloor \leq 2\lfloor \alpha \rfloor + 1$ □

初期 s-t 流の許容性

性質：初期 s-t 流の許容性

任意の $k = 2, \ldots, K$ に対して，
$2f_{k-1}$ は (G, u_k) における s-t 流である

証明：f_{k-1} は (G, u_{k-1}) における s-t 流である

- $\therefore (G, u_{k-1})$ において，容量制約と流量保存制約を満たす

$$0 \leq f_{k-1}(a) \leq u_{k-1}(a) \quad \forall\, a \in A$$
$$f^-_{k-1}(v) = f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$

初期 s-t 流の許容性

性質：初期 s-t 流の許容性

任意の $k = 2, \ldots, K$ に対して，
$2f_{k-1}$ は (G, u_k) における s-t 流である

証明：f_{k-1} は (G, u_{k-1}) における s-t 流である

- $\therefore (G, u_{k-1})$ において，容量制約と流量保存制約を満たす

$$0 \leq f_{k-1}(a) \leq u_{k-1}(a) \quad \forall\, a \in A$$
$$f^-_{k-1}(v) = f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$

- $\therefore (G, u_k)$ において，

$$2f^-_{k-1}(v) = 2f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$

初期 s-t 流の許容性

性質：初期 s-t 流の許容性

任意の $k = 2, \ldots, K$ に対して，
$2f_{k-1}$ は (G, u_k) における s-t 流である

証明：f_{k-1} は (G, u_{k-1}) における s-t 流である

- $\therefore (G, u_{k-1})$ において，容量制約と流量保存制約を満たす

$$0 \leq f_{k-1}(a) \leq u_{k-1}(a) \quad \forall\, a \in A$$
$$f^-_{k-1}(v) = f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$

- $\therefore (G, u_k)$ において，

$$2f^-_{k-1}(v) = 2f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$
$$0 \leq 2f_{k-1}(a) \leq 2u_{k-1}(a) \leq u_k(a) \quad \forall\, a \in A$$

初期 s-t 流の許容性

性質：初期 s-t 流の許容性

任意の $k = 2, \ldots, K$ に対して，
$2f_{k-1}$ は (G, u_k) における s-t 流である

証明：f_{k-1} は (G, u_{k-1}) における s-t 流である

- $\therefore (G, u_{k-1})$ において，容量制約と流量保存制約を満たす

$$0 \leq f_{k-1}(a) \leq u_{k-1}(a) \quad \forall\, a \in A$$
$$f^-_{k-1}(v) = f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$

- $\therefore (G, u_k)$ において，

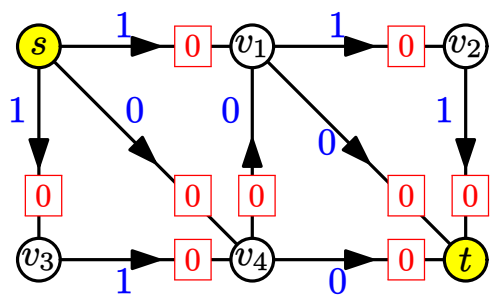
$$2f^-_{k-1}(v) = 2f^+_{k-1}(v) \quad \forall\, v \in V - \{s, t\}$$
$$0 \leq 2f_{k-1}(a) \leq 2u_{k-1}(a) \leq u_k(a) \quad \forall\, a \in A$$

- $\therefore (G, u_k)$ において，$2f_{k-1}$ は s-t 流である

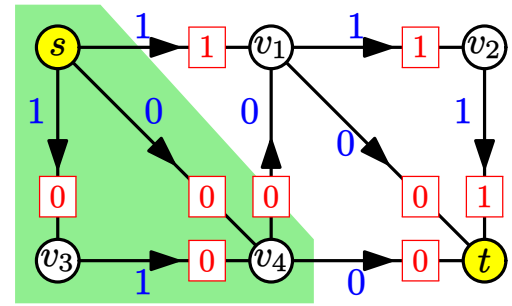
□

計算量に向けて：記法 (再掲)

$k = 1 \quad (G, u_1)$

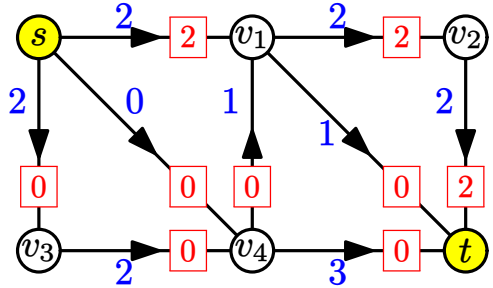


$2f_0$

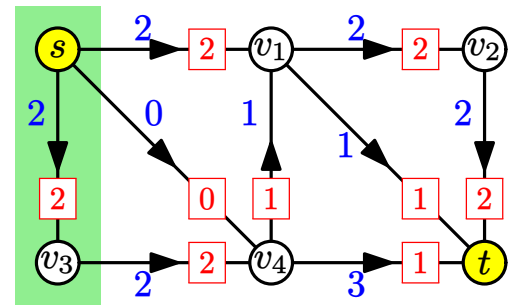


S_1 f_1

$k = 2 \quad (G, u_2)$

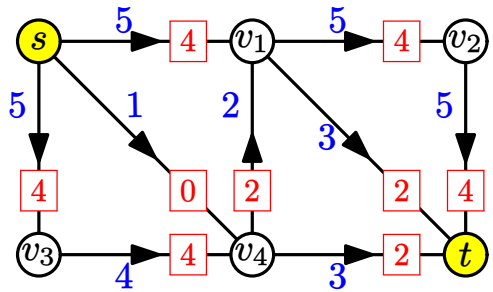


$2f_1$

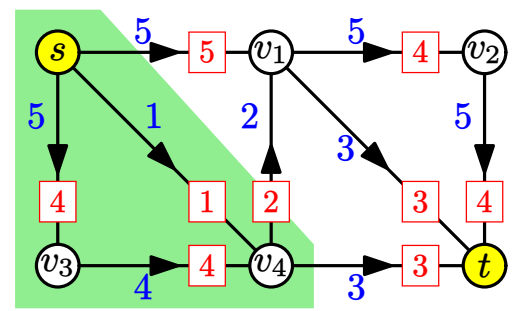


S_2 f_2

$k = 3 \quad (G, u_3)$



$2f_2$



S_3 f_3

s-t カットの容量の変化

第 k 反復における容量最小の s-t カットを S_k とし，

s-t カット S に対して，$\mathrm{cap}_k(S) = \sum_{a \in \delta^+(S)} u_k(a)$ とする

性質：s-t カットの容量の変化

任意の $k = 2, \ldots, K$ に対して，

$$\mathrm{cap}_k(S_{k-1}) \leq 2\mathrm{cap}_{k-1}(S_{k-1}) + m$$

証明：弧の容量の変化に従うと

$$\mathrm{cap}_k(S_{k-1}) = \sum_{a \in \delta^+(S_{k-1})} u_k(a) \leq \sum_{a \in \delta^+(S_{k-1})} (2u_{k-1}(a) + 1)$$

$$\leq 2\mathrm{cap}_{k-1}(S_{k-1}) + m$$

□

s-t 流の値の変化

性質：s-t 流の値の変化

任意の $k = 2, \ldots, K$ に対して

$$\mathsf{val}(f_k) - \mathsf{val}(2f_{k-1}) \leq m$$

証明：最大流最小カット定理を思い出しながら

$$\begin{aligned}
\mathsf{val}(f_k) - \mathsf{val}(2f_{k-1}) &= \mathsf{cap}_k(S_k) - 2\mathsf{cap}_{k-1}(S_{k-1}) \\
&\leq \mathsf{cap}_k(S_{k-1}) - 2\mathsf{cap}_{k-1}(S_{k-1}) \\
&\leq m
\end{aligned}$$

□

s-t 流の値の変化

性質：s-t 流の値の変化

任意の $k = 2, \ldots, K$ に対して

$$\mathsf{val}(f_k) - \mathsf{val}(2f_{k-1}) \leq m$$

証明：最大流最小カット定理を思い出しながら

$$\begin{aligned}
\mathsf{val}(f_k) - \mathsf{val}(2f_{k-1}) &= \mathsf{cap}_k(S_k) - 2\mathsf{cap}_{k-1}(S_{k-1}) \\
&\leq \mathsf{cap}_k(S_{k-1}) - 2\mathsf{cap}_{k-1}(S_{k-1}) \\
&\leq m
\end{aligned}$$

□

s-t 流の値の変化

性質：s-t 流の値の変化

任意の $k = 2, \ldots, K$ に対して

$$\mathsf{val}(f_k) - \mathsf{val}(2f_{k-1}) \leq m$$

証明：最大流最小カット定理を思い出しながら

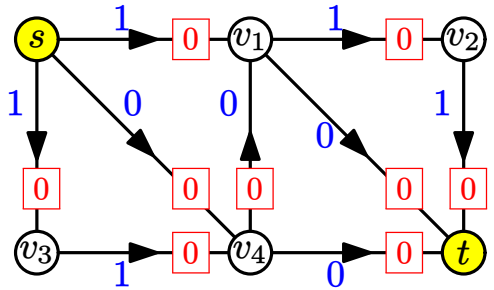
$$\begin{aligned}
\mathsf{val}(f_k) - \mathsf{val}(2f_{k-1}) &= \mathsf{cap}_k(S_k) - 2\mathsf{cap}_{k-1}(S_{k-1}) \\
&\leq \mathsf{cap}_k(S_{k-1}) - 2\mathsf{cap}_{k-1}(S_{k-1}) \\
&\leq m
\end{aligned}$$

□

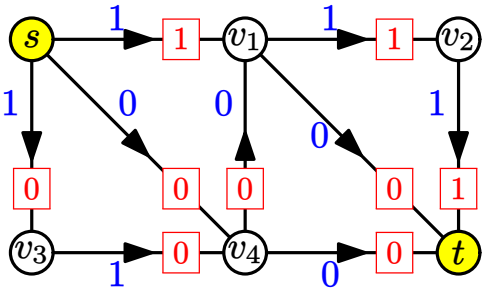
S_k は (G, u_k) の容量最小 s-t カット

反復回数

$k = 1 \quad (G, u_1)$

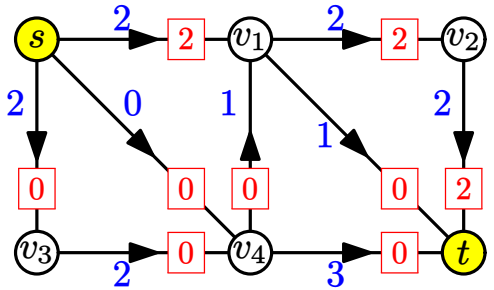


$2f_0$

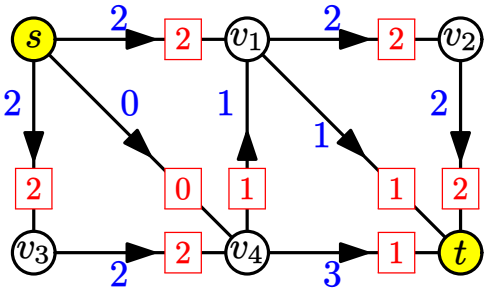


f_1

$k = 2 \quad (G, u_2)$



$2f_1$

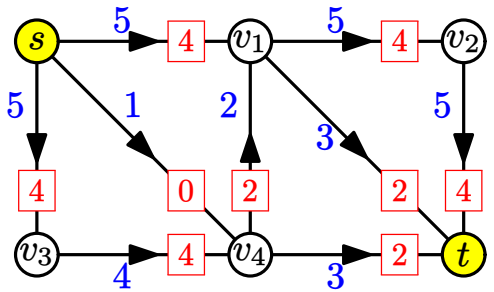


f_2

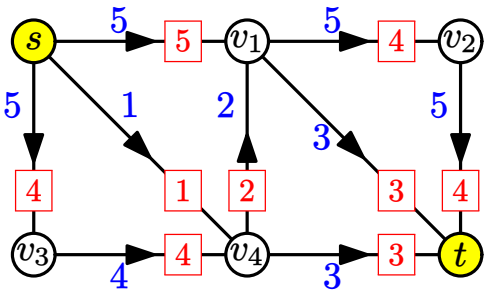
$\leq m$ 回

$k = 3 \quad (G, u_3)$

$O(\log U)$ 回



$2f_2$

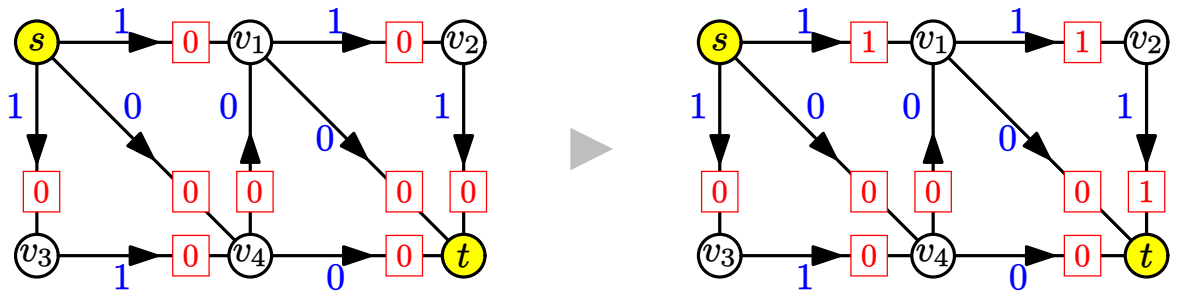


f_3

$\leq m$ 回

反復回数

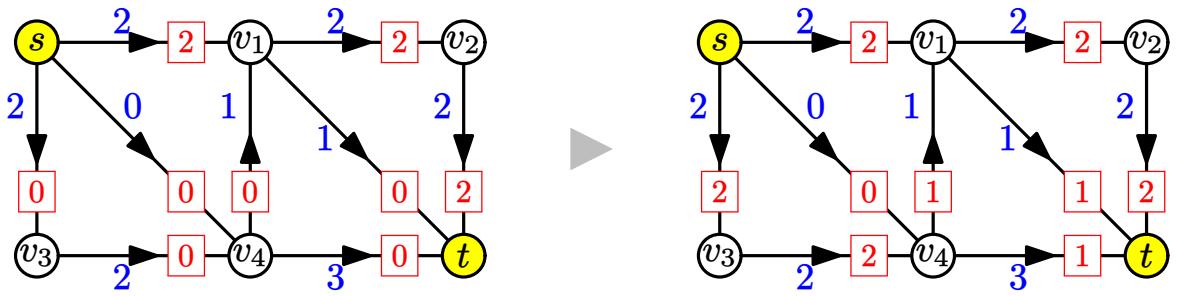
$k = 1 \quad (G, u_1)$



$2f_0 \le m$ 回 $\rightarrow f_1$

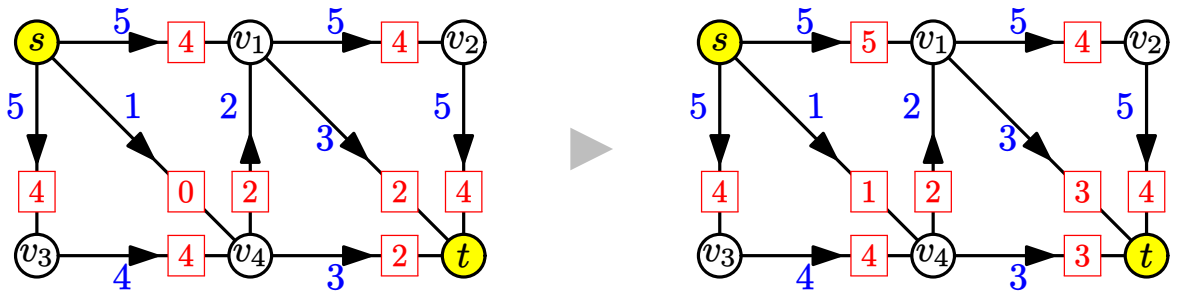
$mU = m$

$k = 2 \quad (G, u_2)$



$2f_1 \le m$ 回 $\rightarrow f_2$

$k = 3 \quad (G, u_3)$



$2f_2 \le m$ 回 $\rightarrow f_3$

$O(\log U)$ 回

容量スケーリング法：計算量

容量スケーリング法の計算量

容量スケーリング法は
容量が整数のとき必ず止まり，計算量は $O(m^2 \log U)$

各 $k = 1, 2, \ldots, K$ に対して

増加操作を行う回数 $= O(m)$ ← 工夫のない増加道法

増加操作 1 回の計算量 $= O(m)$

$\therefore$ 各 k に対する計算量 $= O(m^2)$

$K = O(\log U)$

$\therefore$ 全体の計算量 $= O(m^2 K) = O(m^2 \log U)$

つまり，容量スケーリング法は **弱多項式時間** アルゴリズム

アルゴリズムの比較

	計算量	分類	注意
増加道法	$O(m^2U)$	擬多項式時間	整数のみ
容量スケーリング	$O(m^2 \log U)$	弱多項式時間	整数のみ
Edmonds-Karp	$O(nm^2)$	強多項式時間	実数 OK

アルゴリズムの比較

	計算量	分類	注意
増加道法	$O(m^2U)$	擬多項式時間	整数のみ
容量スケーリング	$O(m^2 \log U)$	弱多項式時間	整数のみ
Edmonds-Karp	$O(nm^2)$	強多項式時間	実数 OK
Push-Relabel （次回，次々回）	$O(n^2m)$ （工夫すればもっと高速）	強多項式時間	実数 OK

まとめ

最大流問題

増加道法

↓ 工夫

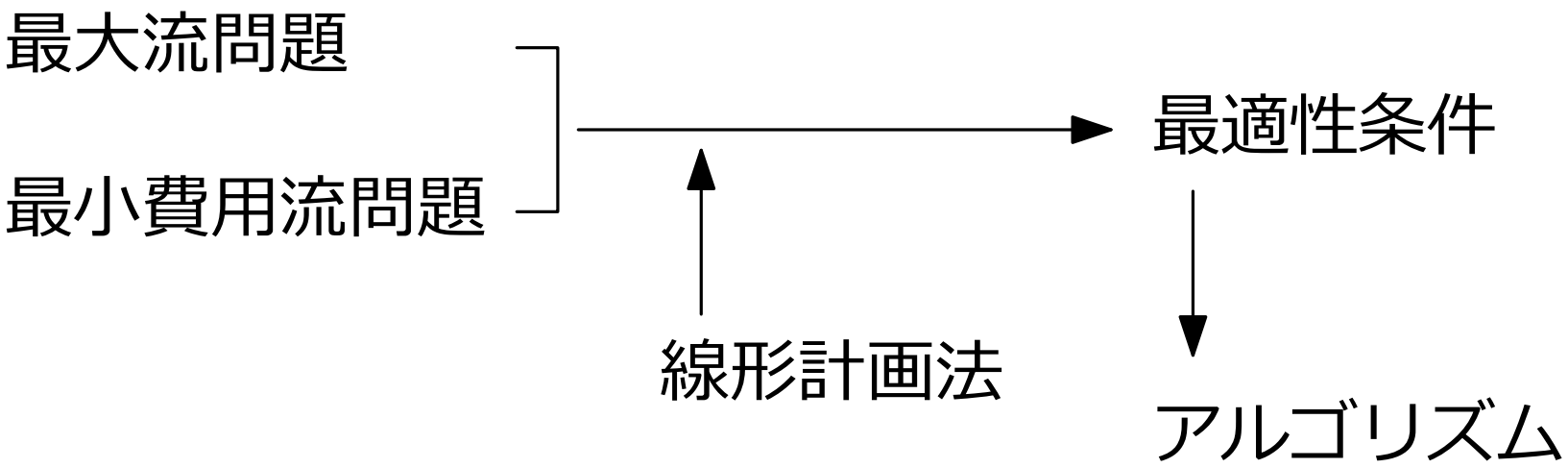
Edmonds-Karp のアルゴリズム

- 弧の数が最小の増加道を選ぶ
- 増加量が最大の増加道を選ぶ

↓ 工夫

容量スケーリング法

次回予告



次回の予告

Push-Relabel 法

- 増加道法に基づかない最大流アルゴリズム
- 線形計画法の視点が再度登場