実践的幾何アルゴリズム
Advanced Algorithms for Computational Geometry

アルゴリズムの設計と解析の基礎

担当:上原隆平

Advanced Algorithms for Computational Geometry
実践的幾何アルゴリズム

Foundation of Design and Analysis of Algorithms

Ryuhei Uehara

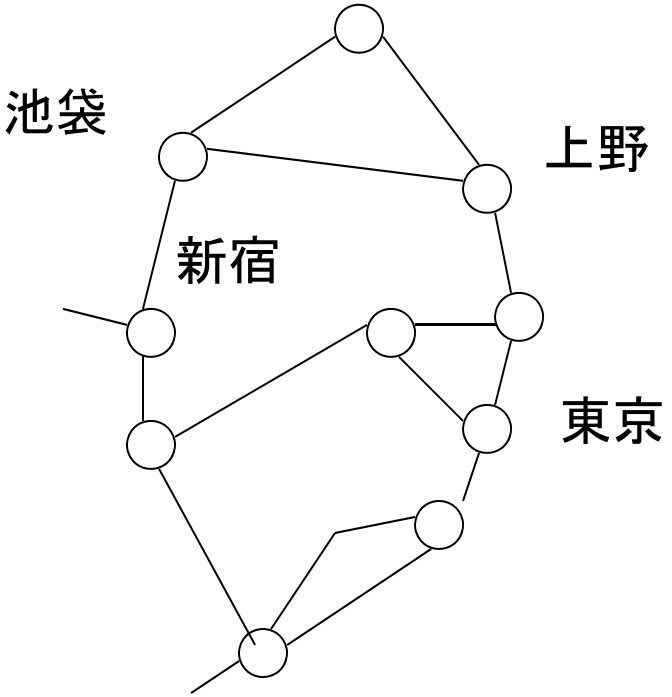
基礎的なデータ構造

- グラフに関する基礎知識
- グラフを表現するデータ構造
  1. 隣接行列
  2. 隣接リスト
- 平面グラフの幾何構造を表現するデータ構造
  - 2重連結辺リスト

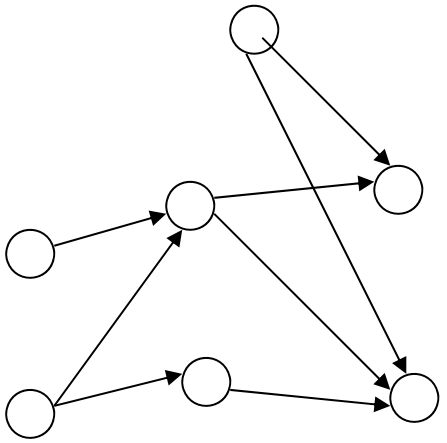
グラフ (graph)

- 頂点を辺で結んだもの
  - 有向グラフ: 辺に方向がある
  - 無向グラフ: 辺に方向がない

例: 路線図

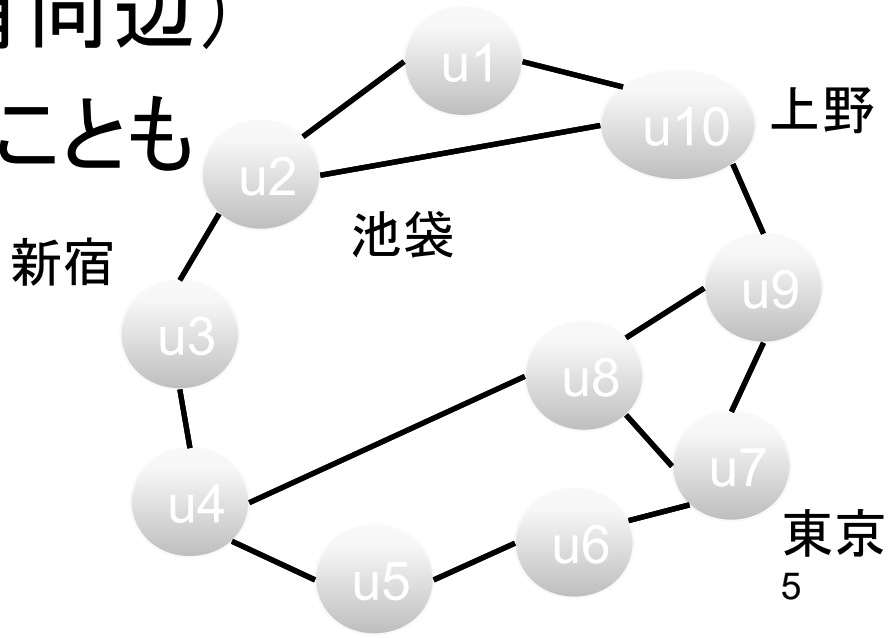


例: 講義の履修順序



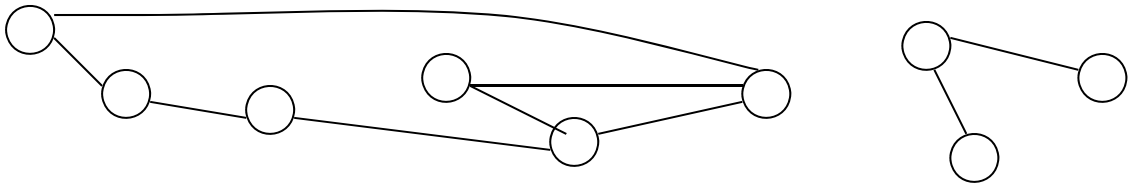
グラフ: 表記

- グラフ: $G = (V, E)$
  - V: 頂点集合, E: 辺集合
- 頂点: $u, v, \ldots \in V$
- 辺: $e = \{u, v\} \in E$ (無向辺)
  $a = (u, v) \in E$ (有向辺)
- 頂点や辺は重みを持つことも
  - $w(u), w(e)$
  - 距離, 金額, 時間など

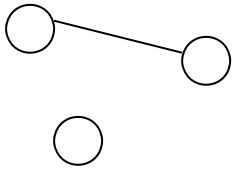


グラフ: 基本的な用語 (1/2)

- 路 (path): 辺で隣り合う頂点を繋いだもの
  - 単純路 (simple path): 同じ頂点を複数回通らない

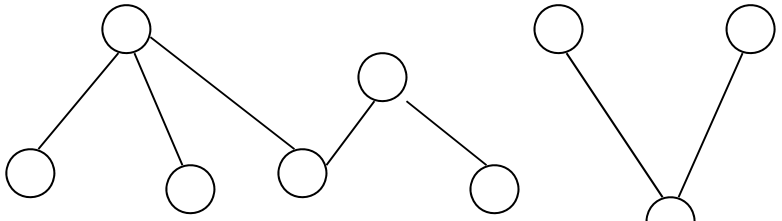


- 閉路 (cycle, closed path): v から v への路
- 連結グラフ (connected graph): どの2頂点間にも路が存在するグラフ

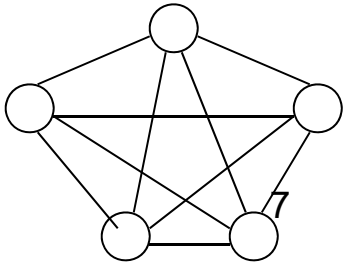


グラフ: 基本的な用語 (2/2)

- 森 (forest): 閉路を含まないグラフ
- 木 (tree): 連結で閉路を含まないグラフ



- 平面的グラフ (planar graph): 辺の交差なしで平面に描画できるグラフ
- 完全グラフ (complete graph): 全ての頂点対を辺で隣接させたグラフ
  - 完全グラフ K_5 は極小非平面的グラフ



グラフアルゴリズムの計算量

- 頂点数 n, 辺数 $m \in O(n^2)$
  - 無向グラフ: $m \leqq n(n\text{-}1)/2$
  - 有向グラフ: $m \leqq n(n\text{-}1)$

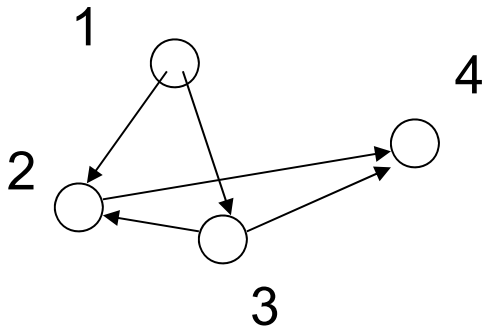
- 平面グラフでは $m \in O(n)$

- グラフアルゴリズムの計算量は
  n や m の式で表す

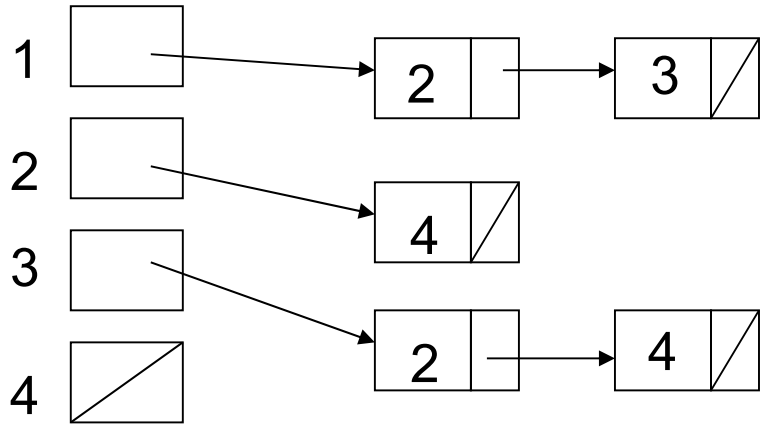
グラフの表現方法

- 隣接行列

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



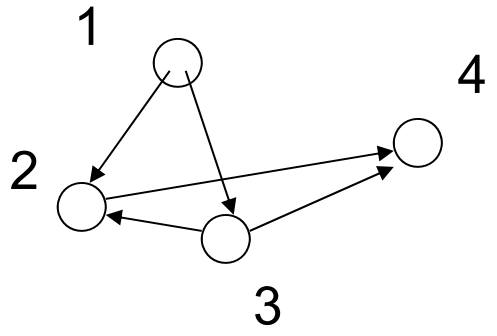
- 隣接リスト



頂点　　隣接頂点のリスト

グラフの表現方法:
行列表現(隣接行列)

- $(u, v) \in E \Rightarrow M[u, v] = 1$
- $(u, v) \notin E \Rightarrow M[u, v] = 0$

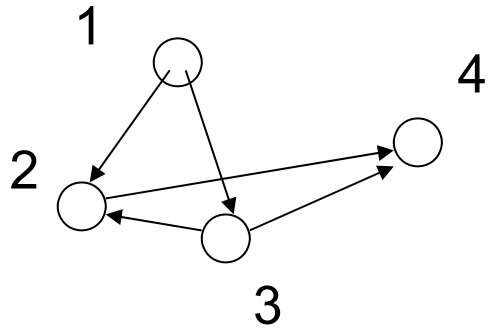


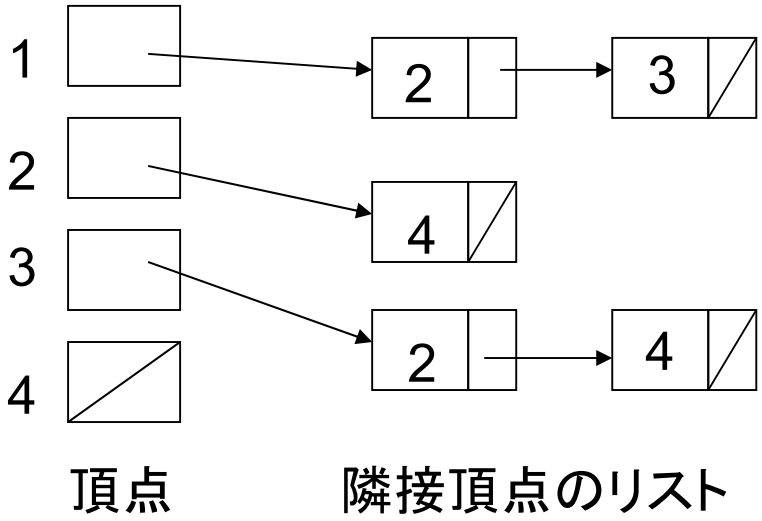
$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

グラフの表現方法: リスト表現(隣接リスト)

- $(u, v) \in E \Leftrightarrow v \in T(u)$
  - $T(u)$ は u の隣接点のリスト

- 例:





隣接行列 vs 隣接リスト

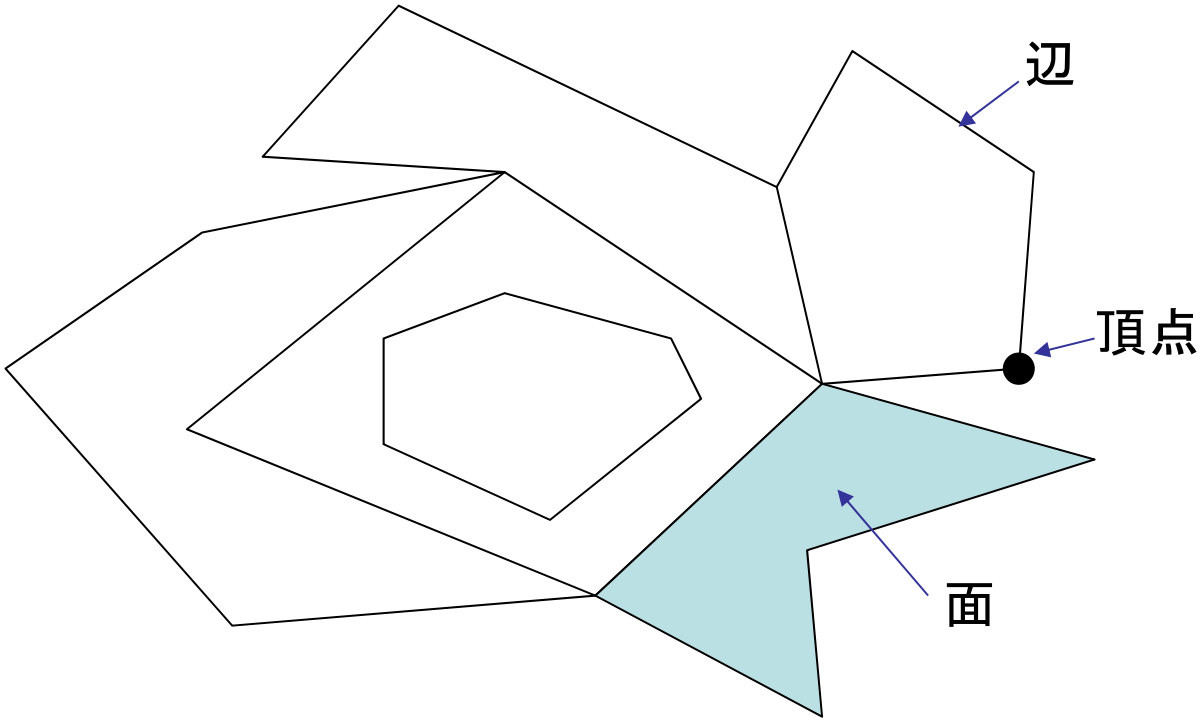
- メモリ量
  - 隣接行列: $\Theta(n^2)$
  - 隣接リスト: $\Theta(m \log n)$
- 隣接チェックにかかる時間: $(u, v) \in E$?
  - 隣接行列: $\Theta(1)$
  - 隣接リスト: $\Theta(n)$

Q. グラフの更新 (e.g., 頂点・辺の追加・削除) は?

二重連結辺リスト

平面グラフ

グラフ:　頂点と頂点間を結ぶ辺によって表現される関係
平面グラフ:　辺が互いに交差しないように頂点の場所を決めて
　　描かれたグラフのこと，あるいは，そのように描けるグラフ．



二重連結辺リスト

二重連結辺リスト　(DCEL: Doubly-Connected Edge List)

　平面に描かれた平面グラフを表現するのに適したデータ構造

平面グラフ G = (V, E),

　頂点集合　V = {v[1], v[2], ... , v[n]},

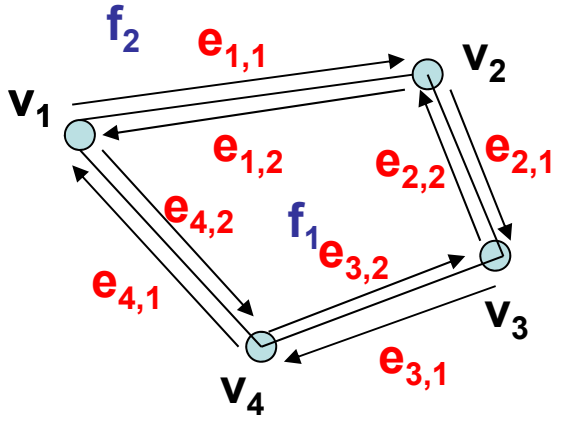
　辺集合　　E = {e[1], e[2], ... , e[m]}

　面集合　　F = {f[1], f[2], ... f[k]}

各頂点vに関する情報

　vの座標(x(v), y(v)),

　vから出る一つの辺 out_edge(v)（任意）



各面fに関する情報

　その外部境界上の一つの辺 outer_component(f)

　面の中のそれぞれの穴について，その境界上の一つの辺のリスト

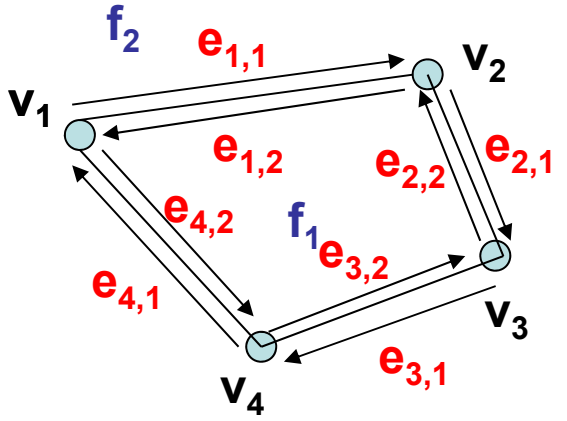
　　inner_component(f)

	座標	接続辺
v1	(x(v1), y(v1))	$e_{1,1}$
v2	(x(v2), y(v2))	$e_{2,1}$
v3	(x(v3), y(v3))	$e_{3,1}$
v4	(x(v4), y(v4))	$e_{4,1}$

二重連結辺リスト

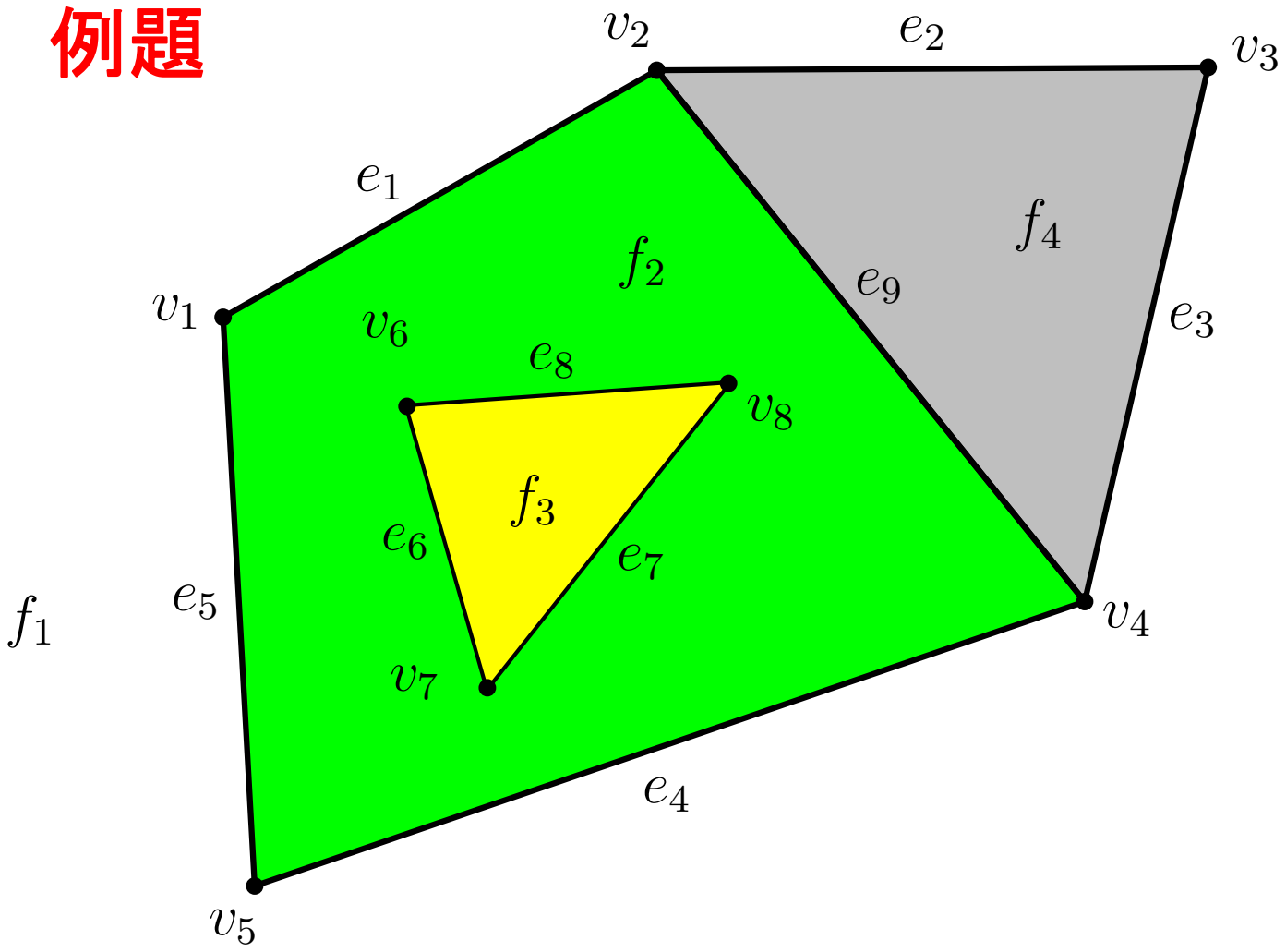
辺に関する情報

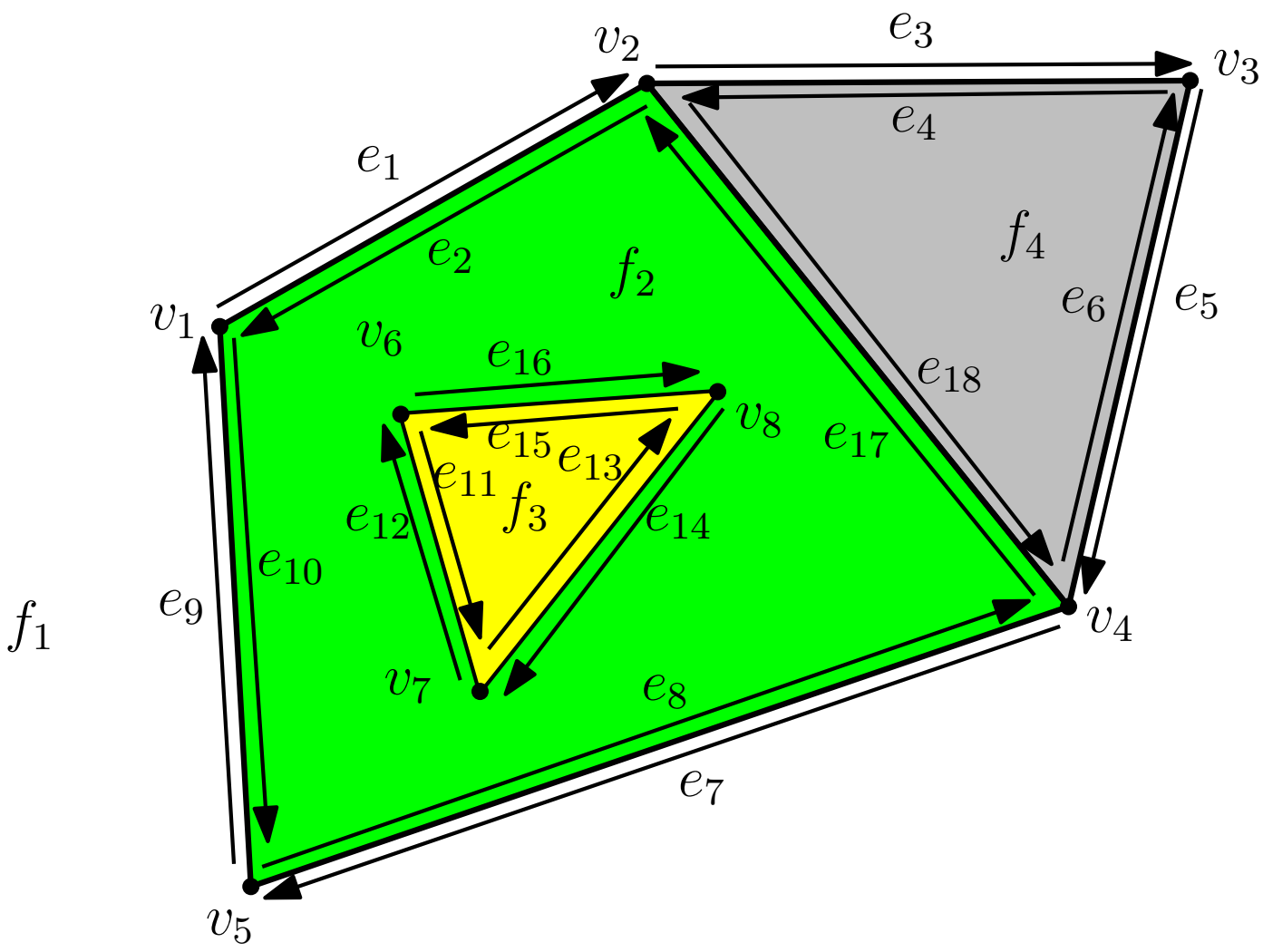
各辺について，
- 異なる方向をもつ2つの有向辺を仮定.
- 各辺の左にある領域の名前: face(e).
- 各辺の始点と終点の頂点名.
- 各辺について，同じ面での次の辺へのポインタをもつ：NextEdge(e).
- 各辺について，反対方向の辺へのポインタをもつ：TwinEdge(e).



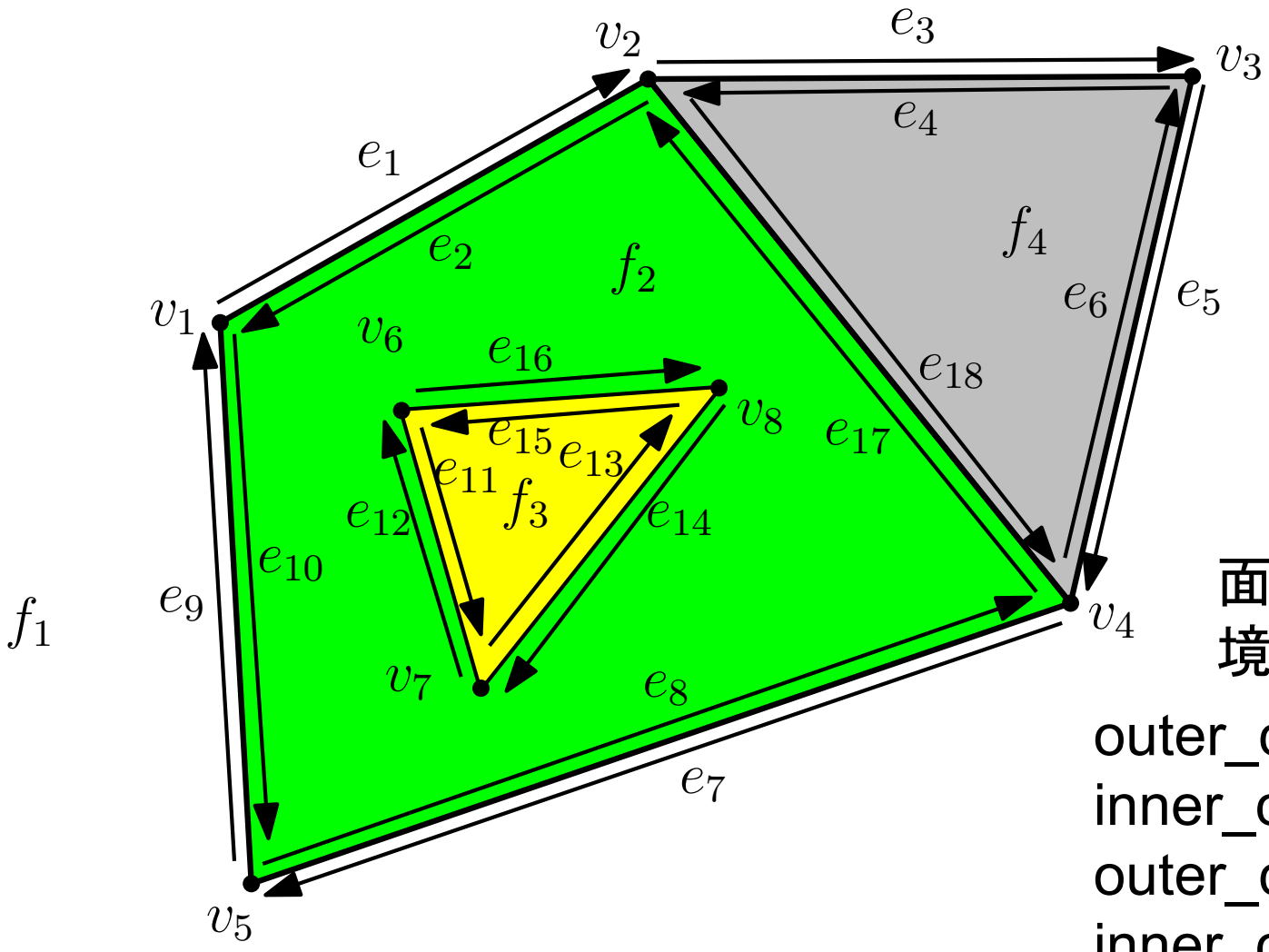
辺	左の領域名	始点	終点	次の辺	反対方向辺
$e_{1,1}$	f_2	v_1	v_2	$e_{2,1}$	$e_{1,2}$
$e_{1,2}$	f_1	v_2	v_1	$e_{4,2}$	$e_{1,1}$
$e_{2,1}$	f_2	v_2	v_3	$e_{3,1}$	$e_{2,2}$
$e_{2,2}$	f_1	v_3	v_2	$e_{1,2}$	$e_{1,2}$
$e_{3,1}$	f_2	v_3	v_4	$e_{4,1}$	$e_{3,2}$
.....	.....	.....	...		

例題





各辺を異なる方向をもつ2辺で置き換える.



面f2のみ内部
境界をもつ.

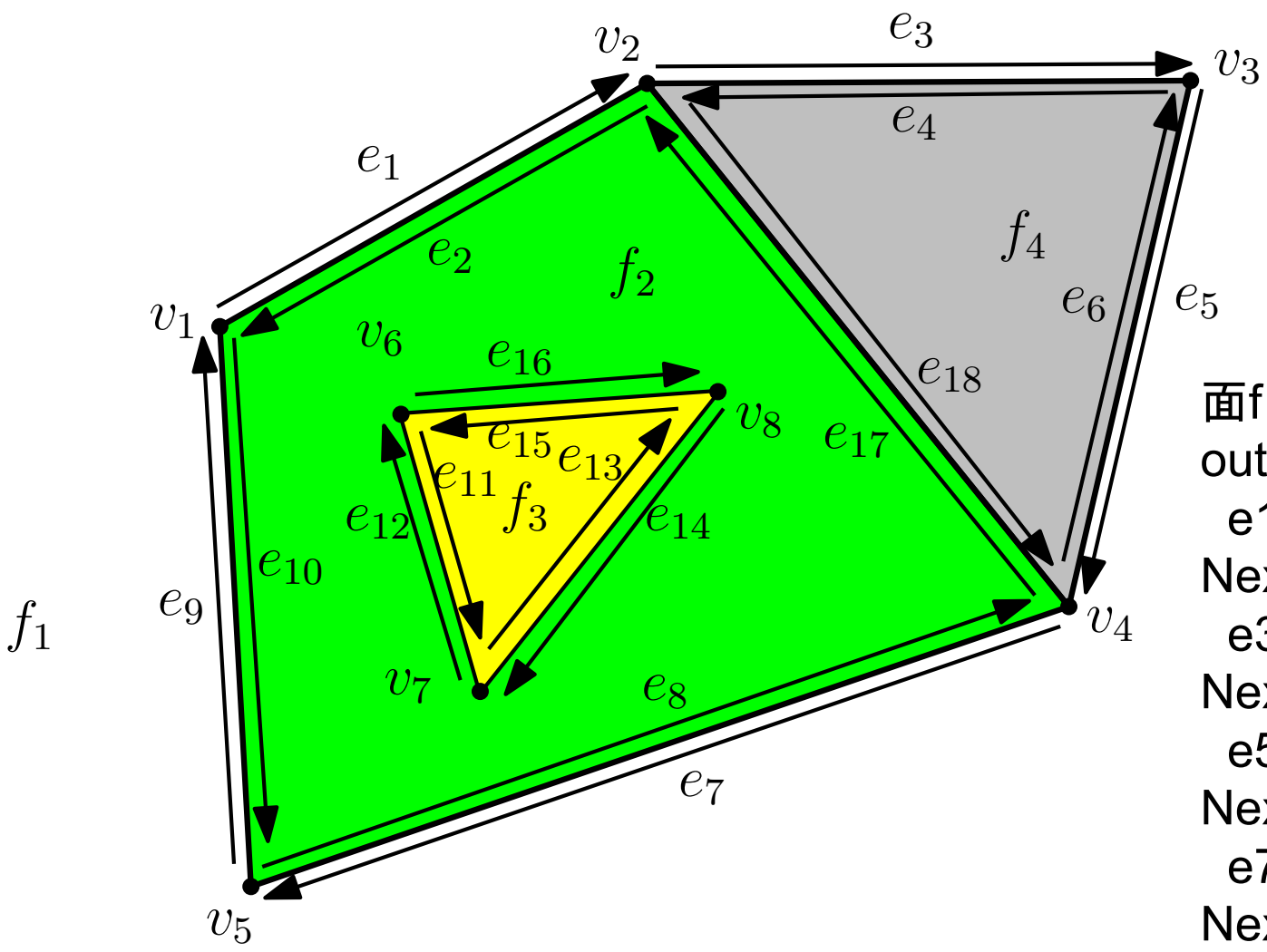
outer_comp(f1)=e1,
inner_comp(f1)=nil
outer_comp(f2)=e2
inner_comp(f2)=e12
outer_comp(f3)=e11
inner_comp(f3)=nil, ...

NextEdge(e1)=e3, NextEdge(e3)=e5, ...
TwinEdge(e1)=e2, TwinEdge(e2)=e1, ...

二重連結辺リスト

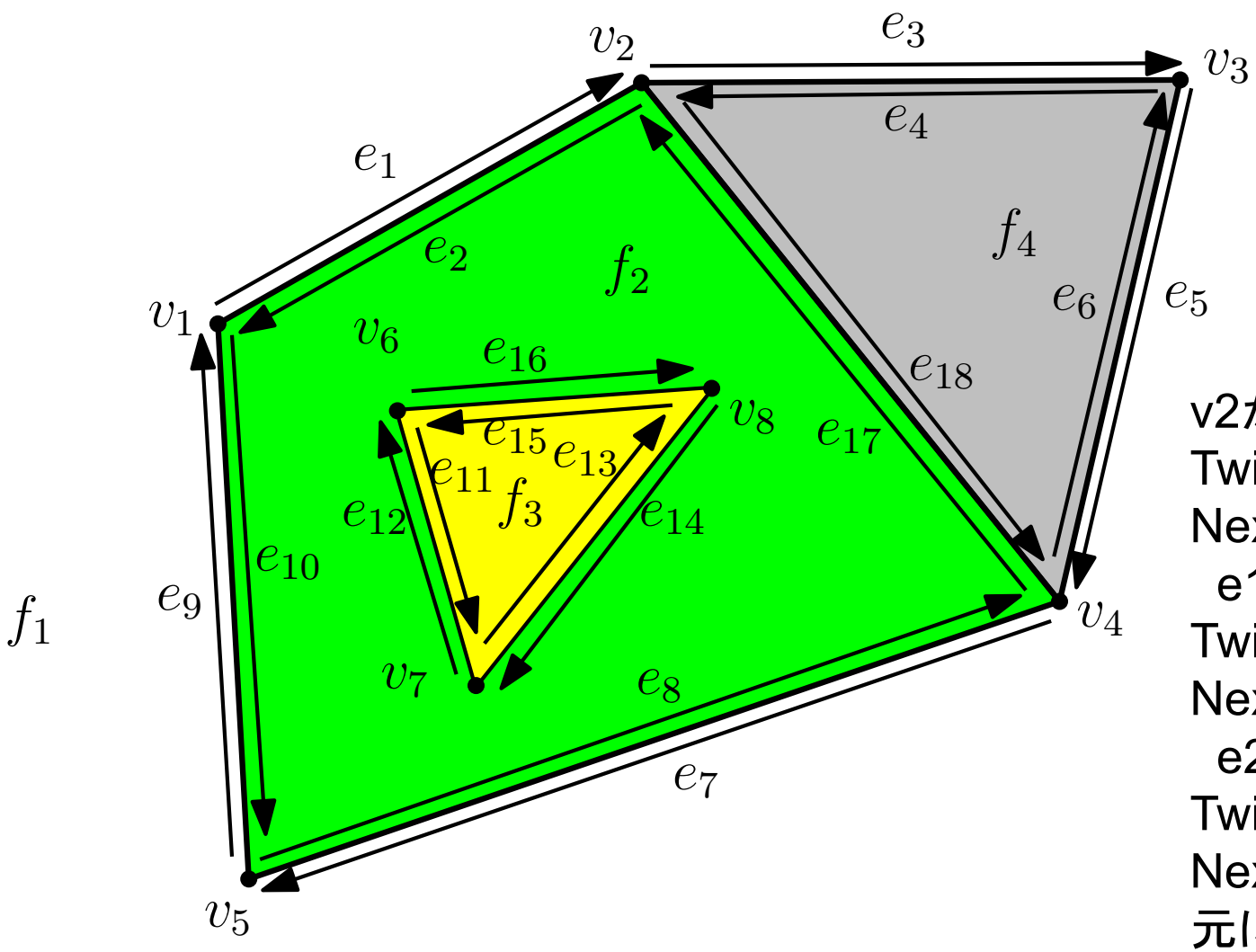
二重連結辺リストを用いてできること:

- 面fを与えると，一つの辺から始めて次の辺へのポインタをたどれば，面の境界上の辺を列挙できる.
- 頂点を与えると，その頂点に入るすべての辺を列挙できる.
- 頂点を与えると，その頂点から出るすべての辺を列挙できる.
- 辺を与えると，その左にある面の名前を出力できる.



一つの面の境界を辿る.

面f1を辿る:
outer_comp(f1)=e1
e1を出力
NextEdge(e1)=e3
e3を出力
NextEdge(e3)=e5
e5を出力
NextEdge(e5)=e7
e7を出力
NextEdge(e7)=e9
e9を出力
NextEdge(e9)=e1
元に戻ったので終了



v2から出る辺e3を出力
TwinEdge(e3)=e4
NextEdge(e4)=e18
　e18を出力
TwinEdge(e18)=e17
NextEdge(e17)=e2
　e2を出力
TwinEdge(e2)=e1
NextEdge(e1)=e3
元に戻ったので終了

たとえば，頂点v2から出る辺をすべて列挙するには？

演習：前ページでは二重連結辺リストを用いてできる操作を列挙したが，
単に頂点に接続する辺を番号順に蓄えるだけのデータ構造でそれらの
操作を実現しようとすると，どれだけの時間が必要か？