おまけ
幾何的データ構造

符号付三角形の面積

二重連結辺リスト

計算幾何の基礎

点の表現

座標を用いて表現するのが最も一般的
2次元の場合には，p(x, y)のように表現.

2 点間の距離：

ユークリッド距離 $\mathrm{dist}(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

マンハッタン距離 $\mathrm{dist}(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$

L_∞距離 $\mathrm{dist}(p_1, p_2) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$

L_p距離 $\mathrm{dist}(p_1, p_2) = \sqrt[p]{(x_1 - x_2)^p + (y_1 - y_2)^p}$

計算幾何の基礎(2)

直線の表現

(1) 傾きとy切片による表現： $y = ax + b$

y軸に平行な直線($x = c$の形)が表現できない.

(2)3つのパラメータを用いた表現: $ax + by + c = 0$

一つの直線の表現が一意に定まらないという欠点がある.

(3)2点を通る直線として表現する方法:

2点(p, q), (r, s)を通る直線: $(s-q)x - (r-p)y = ps - qr$

2直線の交点

$$l_1 : (y_2 - y_1)x - (x_2 - x_1)y = x_1 y_2 - x_2 y_1$$

$$l_2 : (y_4 - y_3)x - (x_4 - x_3)y = x_3 y_4 - x_4 y_3$$

交点の座標(x, y)は

$$x = \frac{\xi(x_4 - x_3) - \eta(x_2 - x_1)}{\Delta}, \quad y = \frac{\xi(y_4 - y_3) - \eta(y_2 - y_1)}{\Delta}$$

$$\xi = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}, \eta = \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix}, \Delta = \begin{vmatrix} y_2 - y_1 & x_2 - x_1 \\ y_4 - y_3 & x_4 - x_3 \end{vmatrix}$$

計算幾何学の基礎(3)

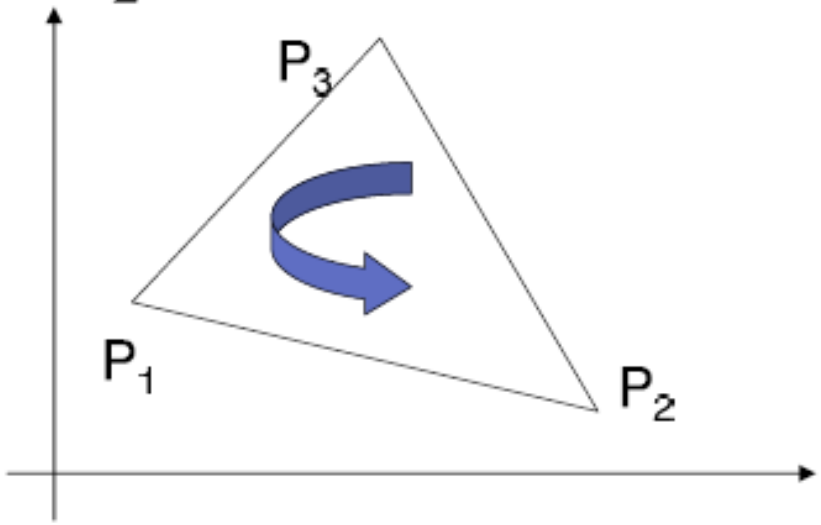
線分のパラメータ表現

$(x_1, y_1), (x_2, y_2)$を結ぶ線分：

$$x = (1-\mu)x_1 + \mu x_2,\ \ y = (1-\mu)y_1 + \mu y_2,\ \ \mu \in [0,1]$$

三角形の面積

$$\text{面積} = \frac{1}{2}((x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1))$$



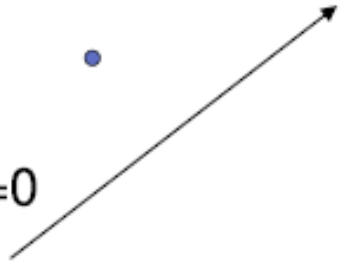
面積＞0：　反時計回り
面積＜0：　時計回り
面積＝0：　一直線上の3点

三角形の符号付面積

計算幾何学の基礎(4)

有向線分と点との位置関係

点pが有向線分ABの左側にある➔面積(A,B,p)>0
点pが有向線分ABの右側にある➔面積(A,B,p)<0
点pが有向線分ABを含む直線上にある➔面積(A,B,p)=0

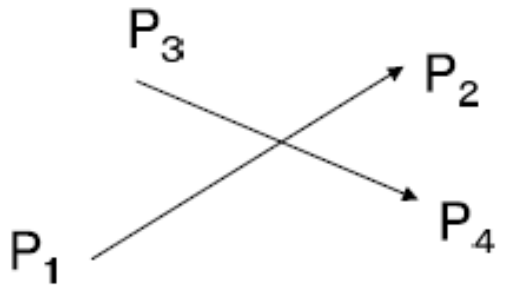


2線分の交差判定

(1)線分を含む直線の式を求めた後，それらの交点を求め，それが両方の線分に含まれるかどうかを判定する.
交点計算に除算を含むので，計算誤差に弱い.

(2)三角形の符号付面積を用いる方法：
2端点P_1P_2を結ぶ線分と，2端点P_3P_4を結ぶ線分が交差するための条件：

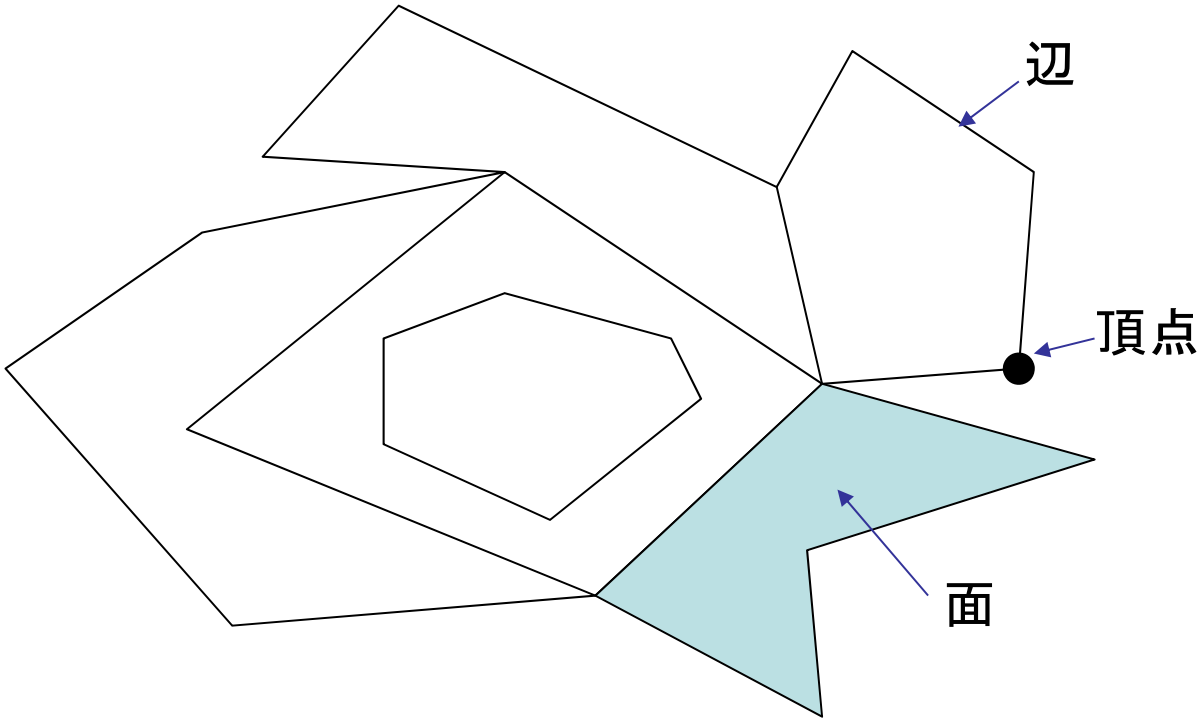
$$\Delta(P_1,P_2,P_3)\times\Delta(P_1,P_2,P_4)<0,\ \text{and}$$
$$\Delta(P_3,P_4,P_1)\times\Delta(P_3,P_4,P_2)<0$$



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]},

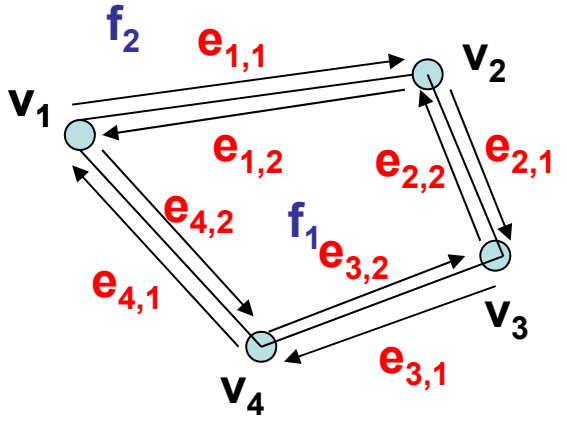
　辺集合　　E = {e[1], e[2], ... , e[m]}

　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

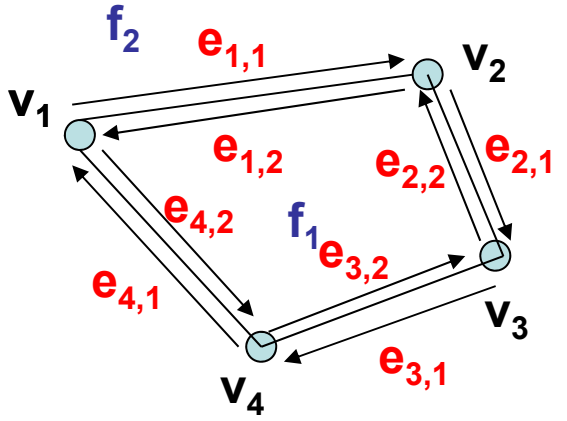
　　inner_component(f)

	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

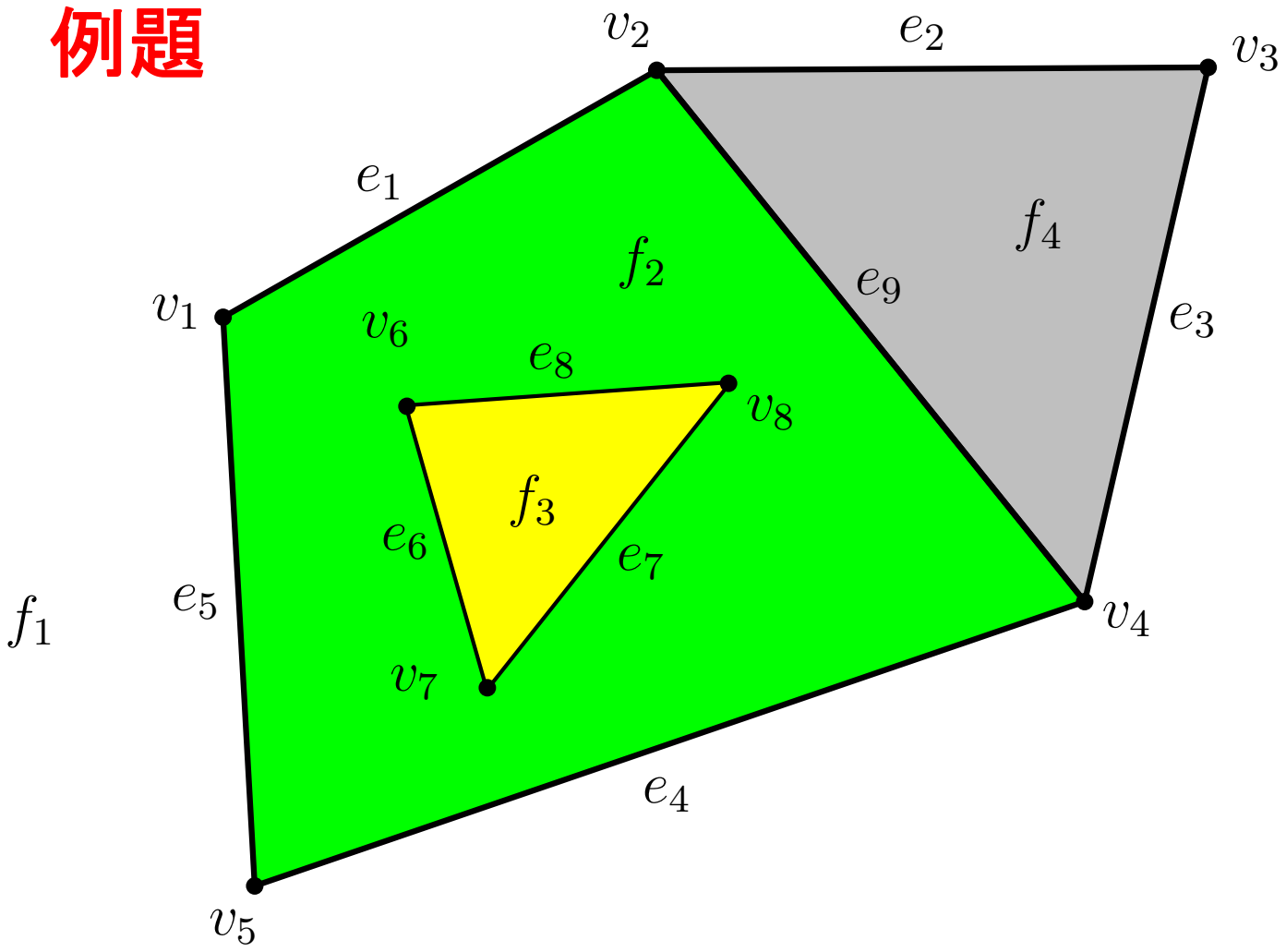
辺に関する情報

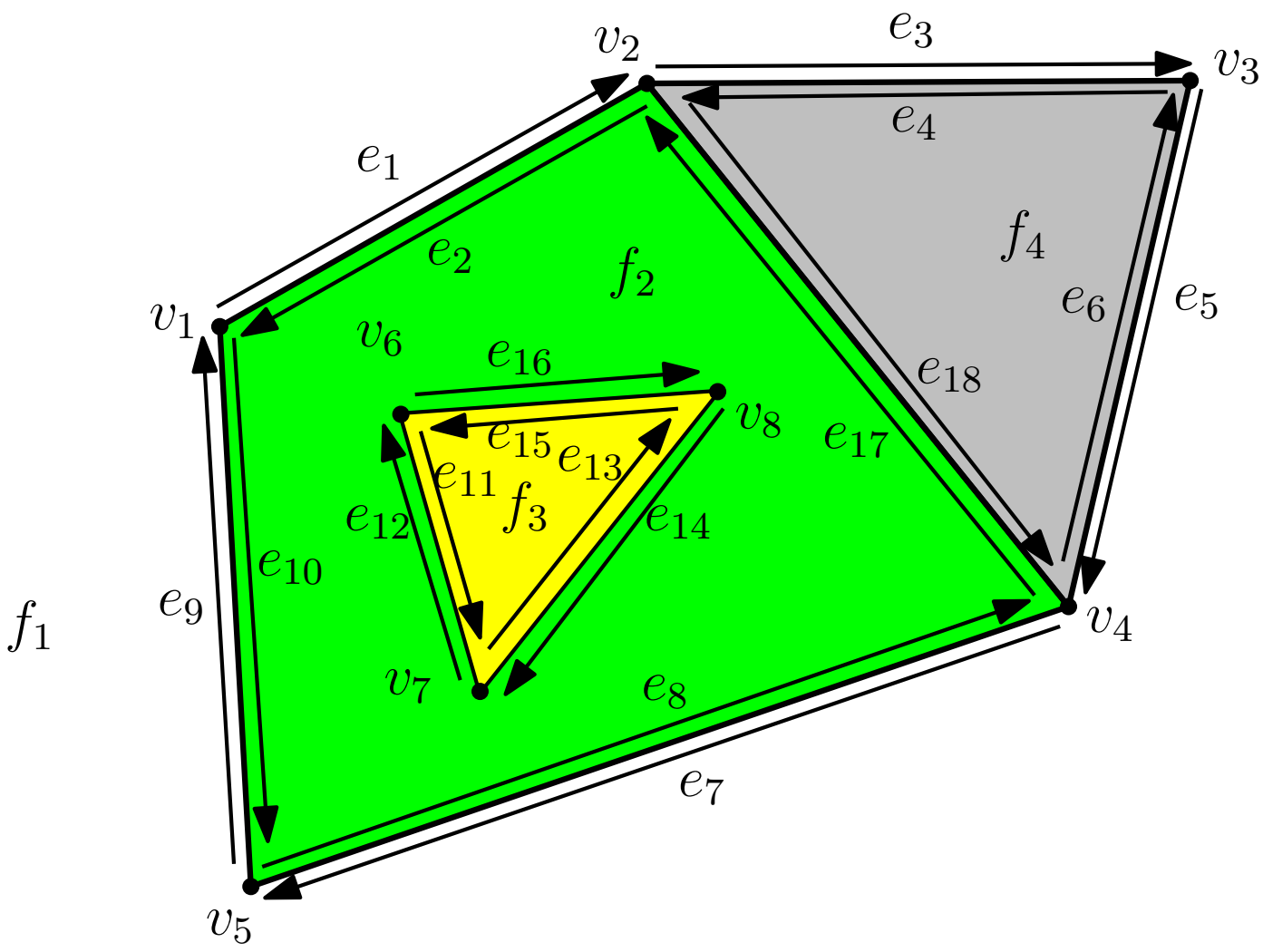
各辺について，
- 異なる方向をもつ2つの有向辺を仮定.
- 各辺の左にある領域の名前: face(e).
- 各辺の始点と終点の頂点名.
- 各辺について，同じ面での次の辺へのポインタをもつ：NextEdge(e).
- 各辺について，反対方向の辺へのポインタをもつ：TwinEdge(e).



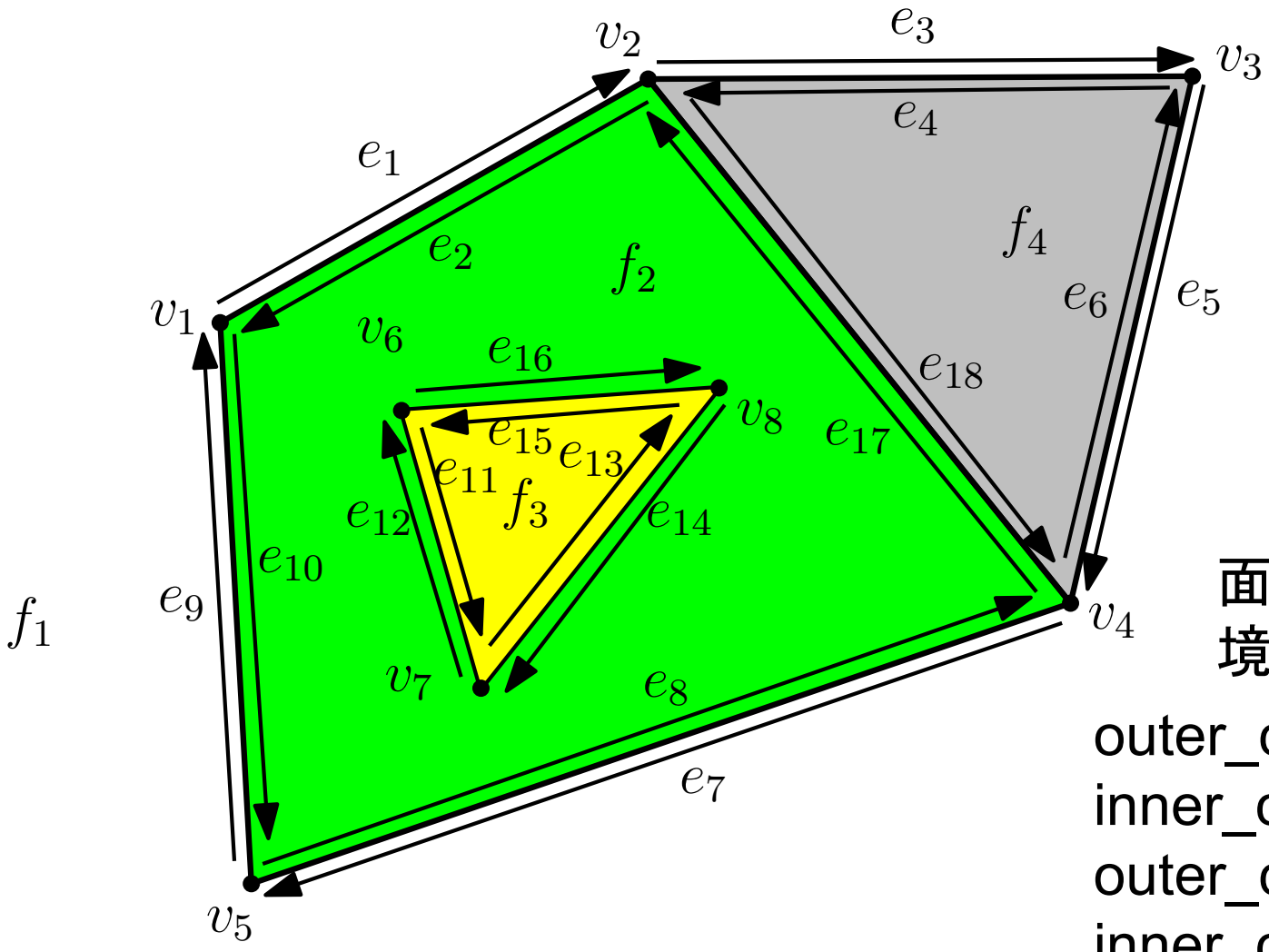
辺	左の領域名	始点	終点	次の辺	反対方向辺
$e_{1,1}$	f_2	v_1	v_2	$e_{2,1}$	$e_{1,2}$
$e_{1,2}$	f_1	v_2	v_1	$e_{4,2}$	$e_{1,1}$
$e_{2,1}$	f_2	v_2	v_3	$e_{3,1}$	$e_{2,2}$
$e_{2,2}$	f_1	v_3	v_2	$e_{1,2}$	$e_{1,2}$
$e_{3,1}$	f_2	v_3	v_4	$e_{4,1}$	$e_{3,2}$
.....	.....	.....	...		

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.

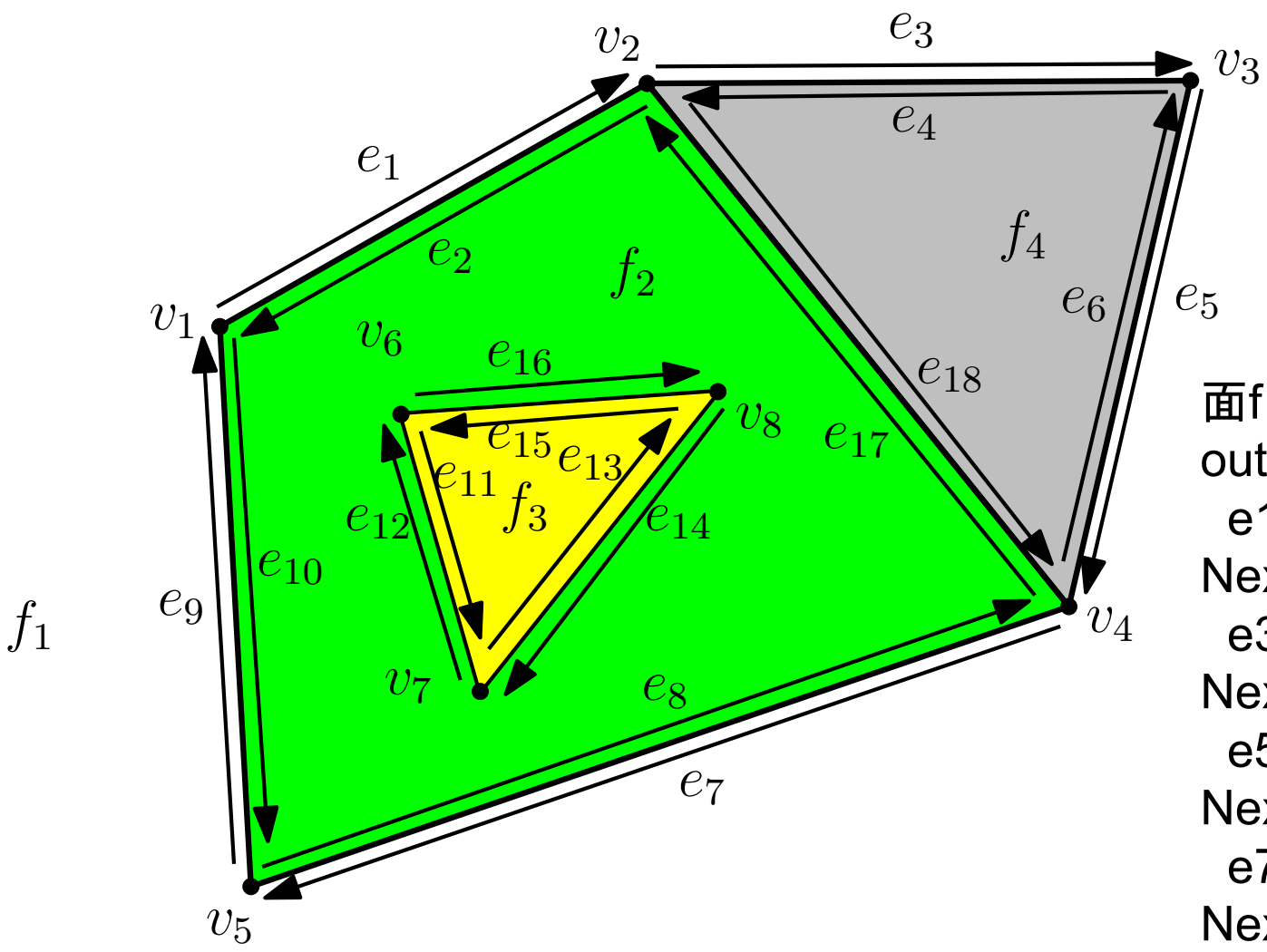
outer_comp(f1)=e1,
inner_comp(f1)=nil
outer_comp(f2)=e2
inner_comp(f2)=e12
outer_comp(f3)=e11
inner_comp(f3)=nil, ...

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

二重連結辺リスト

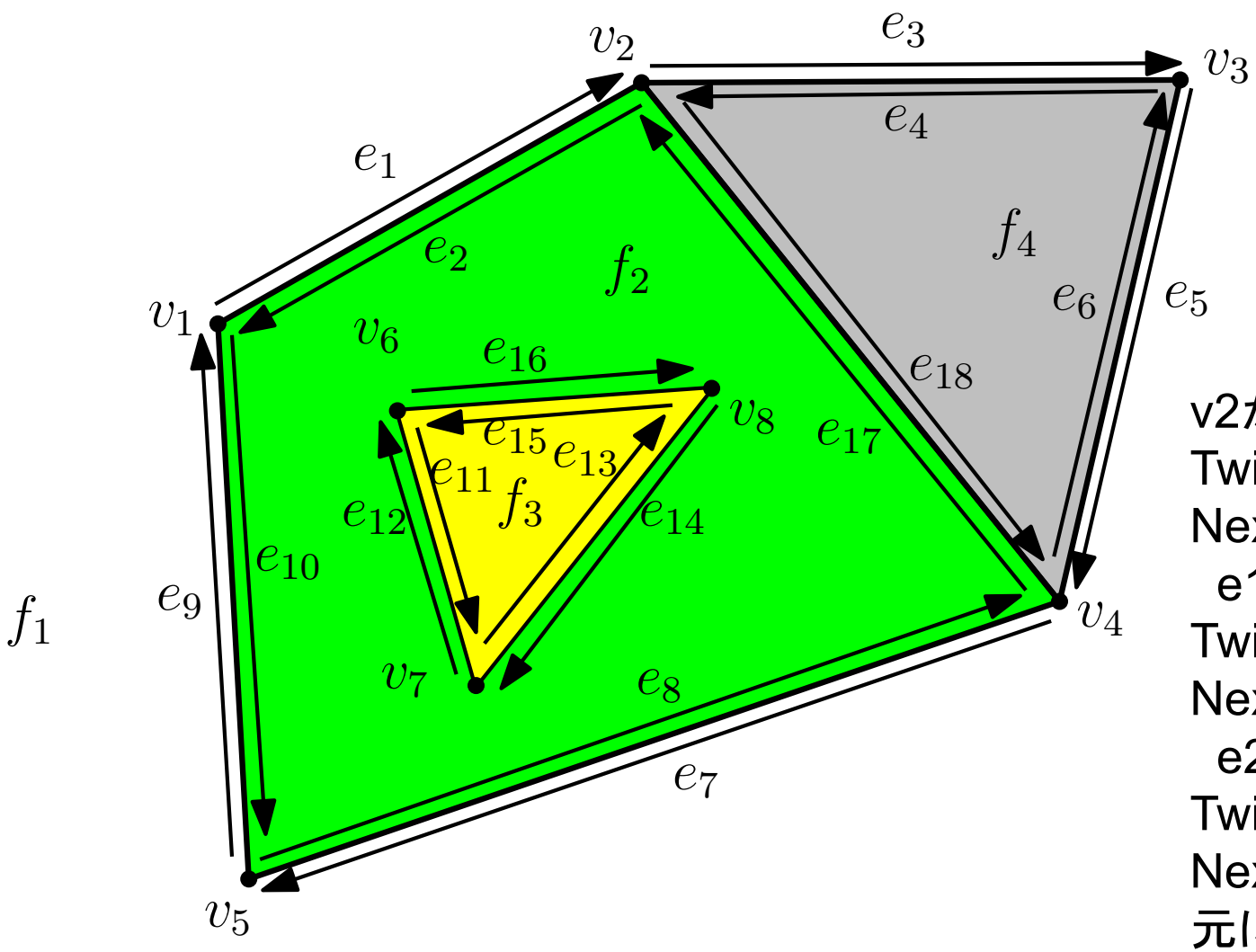
二重連結辺リストを用いてできること:

- 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？