2.5 凸包の計算

平面上の n 点からなる集合の凸包を $O(n\log n)$ 時間で計算するアルゴリズムがいくつか知られている．このうち最初に発表されたのはRon Grahamによるものであり，これは2つのステップからなる．まず最初に，凸包のある内点の周りで点集合を角度によってソートする．次に，この順に点を次々と読み取り，局所的なテストを行うことで凸包の内部にある点をすべて取り除いていくというものである．以来，このアルゴリズムを変形したものがいくつか出てきた．概念的には非常に単純であるが，点の共線性や他の退化性から，ある種の技術的困難がある．これを単純に避けて通るには，もちろん点集合がそのような不都合な状況を満たさないと仮定し，そういった問題を読者(あるいはプログラマー)に委ねればよい．しかし，多くのプログラマーが，そういった退化性や特殊な状況にこそ最も手間がかかるのだと文句を言っているのが実情である．凸包問題はこの章の後半の中心でもあるので，この節では入力データに対していかなる条件もつけず，できる限り完成度の高い手続きを作り上げるよう努めていく．

ここではGrahamのアルゴリズムを変形して与えよう．この中で点集合は，その第1座標と第2座標により，“辞書式順序”と呼ばれる順序であらかじめソートされているものとする．点集合はこの順に処理され，新たな点が加わるたびに凸包を更新していく．つまり，新たに適当とみなされた点は，順次凸包の頂点となり凸包の頂点リストを更新する．また，もはや頂点ではないという点が出てくれば，それらは凸包の頂点リストから取り除かれなければならない．実際に，このアルゴリズムの核心は，いくつかの局所的なテストを用いて，こういった頂点をすばやく見つけていくことにある．この手続きの主な流れを図2.16に例として示そう．この図において点 p は凸包の頂点リスト $P=\{p_1,\cdots,p_7\}$ に追加される．まず，p から P へ支持線を引くことにより，支持点first, lastが確認される(図2.16(a))．支持線が辺を含む場合には(図において一方の支持線が $[p_2,p_3]$ を含んでいる)，p から離れている方の点を支持点とみなすことにする．そして，リスト P 上でfirstからlastに向かう間の頂点がすべて取り除かれ，1点 p にとって代わられる(図2.16(b))．

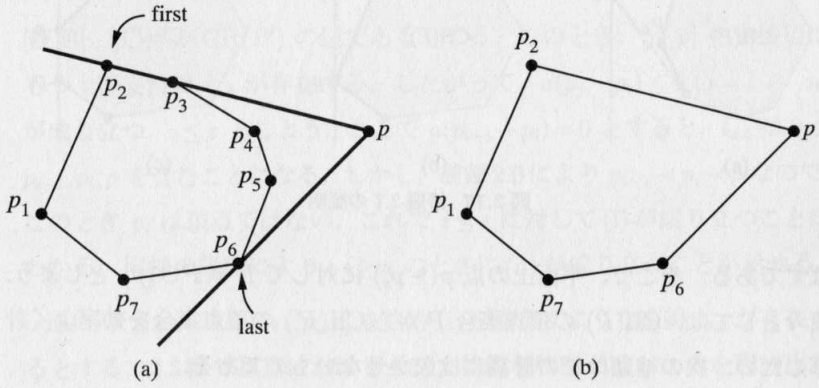


図 **2.16** 凸包の更新

次に，この例で示された手続きの理論的な裏づけをしていくことにしよう．どのようにして実現するかについては後で扱うことにする．いま，$P=\{p_1,\cdots,p_n\}$ $(n\geq 3)$ として，凸包の頂点集合 P が与えられたとしよう．ただし，これらの頂点は，あらかじめ時計まわりの順にソートされているものとする．また，各点 p_i の座標は (p_i^1,p_i^2) であるとする．ここで，P 上での辞書式順序(lexicographical order)「$\prec$」を，

(i) $p_i^1<p_j^1$, あるいは

(ii) $p_i^1=p_j^1$ かつ $p_i^2<p_j^2$

のとき $p_i\prec p_j$ として定義する．この順序によって，最初の点が p_1 であるように P の点が与えられているとしても，一般性を失わない．すなわち，$p_1\prec p_i$ $(i\neq 1)$ としてよい．一方，この順序による最後の点を p_s としよう．このとき，この辞書式順序と先に与えた角度による順序とは，次の意味において両立する(演習問題2.6)．

補題 2.6 $p_1\prec p_2\prec\cdots\prec p_s$ かつ $p_1\prec p_n\prec p_{n-1}\prec\cdots\prec p_s$. ∎

辞書式順序により点を処理していくので，次なる点 p は $p\succ p_s$ を満たす

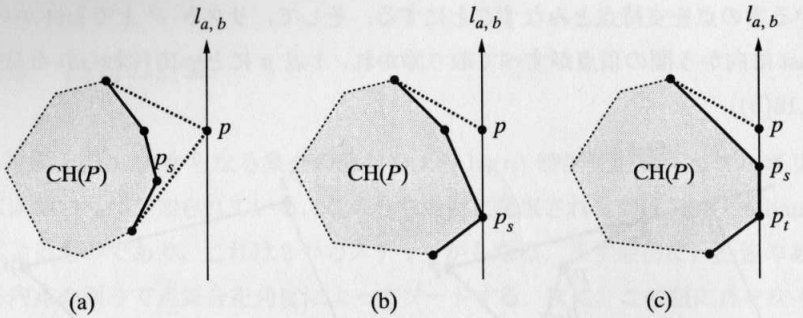


図 **2.17**　補題 2.7 の証明

はずである．そこで，平面上の点 $p(\succ p_s)$ に対して $P'=P\cup\{p\}$ としよう．我々としては，$\mathrm{CH}(P)$ の頂点集合 P から $\mathrm{CH}(P')$ の頂点集合を効率よく計算したい．次の補題はこの計算には欠かせないものである．

補題 2.7　p は $\mathrm{CH}(P')$ の頂点である．

[証明]　$a=(1,0), b=ap$ としよう．$p\succ p_s$ であることから，鉛直線 $l_{a,b}$ が p を通る $\mathrm{CH}(P')$ の上部支持線であることが容易にわかる．支持するものが p であれば自明である (図 2.17(a))．そうでなければ，$[p,p_s]$ または $[p,p_t]$ $(p_s\succ p_t)$ の形の鉛直線分で支持することになる．$[p,p_s]$ の場合には，$p^1=p_s^1$ かつ $p^2>p_s^2$ なので p は線分の端点となっている (図 2.17(b))．また，$[p,p_t]$ の場合には，$p^1=p_t^1$ かつ $p^2>p_s^2>p_t^2$ なので，やはり同じ結論に達する (図 2.17(c))．∎

これにより，$\mathrm{CH}(P')$ を計算するには，後は p を端点にもつ 2 本の辺の他方の端点を計算すればよいことになる．そして，そのような端点は次の定理によって特徴づけられる．

定理 2.8　$P=\{p_1,\cdots,p_n\}$ $(n\geq 3)$ を時計まわりの順にソートされた凸包の頂点集合とする．また，p_s を補題 2.6 により定まる頂点とする．さらに，平面上の点 $p(\succ p_s)$ に対して $P'=P\cup\{p\}$ とおく．このとき，線分 $[p_i,p]$ が $\mathrm{CH}(P')$ の辺であるための必要十分条件は，p_i と p を含むある直線 $l_{a,b}$ が存在して，

(i)　$i\leq s$ のときには $a(p_{i-1}-p_i)<0,\ a(p_i-p_{i+1})\geq 0$,

(ii)　$i\geq s$ のときには $a(p_{i+1}-p_i)<0,\ a(p_i-p_{i-1})\geq 0$

が成り立つことである．

[証明]　$[p_i,p]$ が $\mathrm{CH}(P')$ の辺であるとする．このとき，$[p_i,p]$ を支持辺にもつ上部支持線 $l_{a,b}$ が存在する．したがって，$a(p_j-p_i)\leq 0$ $(j=1,\cdots,n)$ が成り立つ．$i\leq s$ としよう．ここで $a(p_{i-1}-p_i)=0$ とすると，$l_{a,b}$ は 3 点 p_{i-1},p_i,p を含むことになる．しかし，補題 2.6 により $p_{i-1}\prec p_i\prec p$ なので，このとき p_i は頂点ではない．これで $i\leq s$ に対して (i) が成り立つことがわかる．同様の議論により，$i\geq s$ のときに (ii) が成り立つことが示せる．

逆に，$i\leq s$ に対して，$[p_i,p]$ を含み (i) を満たすような直線 $l_{a,b}$ が存在するとする．2.3 節で頂点における上部支持線を特徴づけしたのを思い出そう．条件 (i) により，明らかに a は

$$U_i=\{a: a(p_{i+1}-p_i)\leq 0,\ a(p_i-p_{i-1})\geq 0\}$$

に含まれている．そして，このことと $b=ap_i=ap$ であることから，直線 $l_{a,b}$ が 2 点 p_i, p において上部支持線となっていることがわかる．一方，条件 (i) は，その最初の不等式により，a が

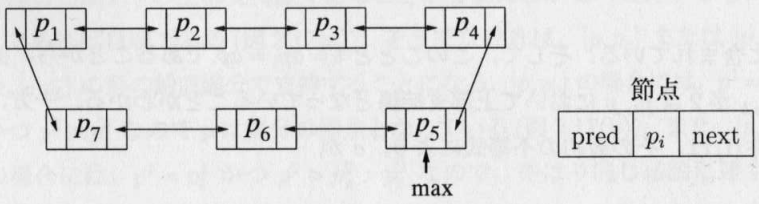
$$U_{i-1}=\{a: a(p_i-p_{i-1})\leq 0,\ a(p_{i-1}-p_{i-2})\geq 0\}$$

に含まれていないことを意味する．したがって，$l_{a,b}$ はちょうど $[p_i,p]$ で $\mathrm{CH}(P')$ を支持する．すなわち，$[p_i,p]$ は $\mathrm{CH}(P')$ の辺である．$i\geq s$ の場合にも同様の議論をすればよい．∎

定理 2.8 は，first が $p_1,\cdots,p_s$ の中の条件 (i) を満たす頂点として，last が $p_s,\cdots,p_n$ の中の条件 (ii) を満たす頂点として一意に定まるということを示唆している．また，2 つの条件は相いれないものなので，first $\neq$ last であることに注意しよう．

2.6 凸包アルゴリズムの実現

この節では，凸包を求めるアルゴリズムを完成させる．その第1ステップとして，まず凸包の頂点集合に適切なデータ構造を選ぼう．前節での議論により，そのようなデータ構造は，頂点が角度の順にソートして確保され，両方向にたどれるようなものである必要がある．この目的のためには双方向巡回リストが理想的であろう．リストの各節点 v には対応する頂点の座標を蓄えるが，これは，凸包の境界上で時計まわりの順に見て，その直前の頂点と次の頂点を指すポインター $\mathrm{pred}(v)$, $\mathrm{next}(v)$ とともに蓄えることにする．簡単のために，ここでは v で節点と頂点の両方を表すことにしよう．また，もう一つ別のポインター "max" が必要である．これは最後にリストに入れられた頂点を指すために用いる．図 2.18 は図 2.16 の例に対するデータ構造を表している．



(a) 双方向巡回リスト　　(b) セル

図 2.18　図 2.16 の例に対するデータ構造

次に，定理 2.8 のテストを左回り・右回りのテストに組直す．新しい点として p を考え，現在 max が指している頂点を v としよう．テストには 4 点 p, v, $\mathrm{pred}(v)$, $\mathrm{next}(v)$ が関係してくるが，テスト (i) は「$\mathrm{pred}(v)$ と $\mathrm{next}(v)$ が p, v を通る直線によって分けられる半平面の同じ側になければならない」ということを意味している．特に，$\mathrm{pred}(v)$ は半平面の内部になければならない．2つの望ましい状況が図 2.19(a), (b) に示されている．したがって，左

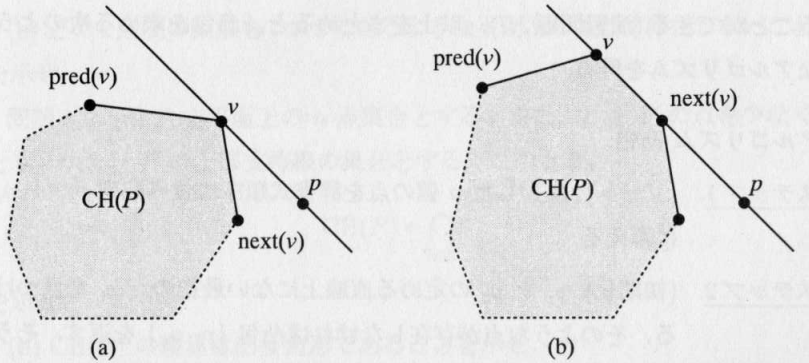


図 2.19　定理 2.8(i) を満たす 4 点 p, v, $\mathrm{pred}(v)$, $\mathrm{next}(v)$ の位置関係

回り・右回りに関して条件 (i) が満たされるのは，

$$\mathrm{left}(p, v, \mathrm{pred}(v)) \textbf{ and not } \mathrm{right}(p, v, \mathrm{next}(v)) = \textbf{true}$$

が成り立つときかつそのときのみである．条件 (ii) についても同様のことがいえる．図 2.20 の手続き $\mathrm{tangent}(p, \mathrm{first}, \mathrm{last}, \mathrm{max})$ は，このことを用いて定理 2.8 のテストを実現している．

```
procedure tangent(p, first, last, max);
    v := max;
    while (not left(p, v, pred(v))) or right(p, v, next(v))
        do v := pred(v);
    first := v;  v := max;
    while (not right(p, v, next(v))) or left(p, v, pred(v))
        do v := next(v);
    last := v;
    return;
```

図 2.20　手続き tangent

後は，first から last に向かう間の頂点をすべて巡回リストから取り除き，代わりに p をその場所に入れる．ところで，この手続きは実は簡略化することができることに注意しよう．"or" 条件の後半の部分を2ヶ所とも省略す

ることができる(演習問題2.7). 以上をまとめると，凸包を求める次のようなアルゴリズムを得る.

アルゴリズム 凸包:

ステップ1. (ソート) 入力した n 個の点を辞書式順序に並べ配列 $p_1,\cdots,p_n$ に蓄える.

ステップ2. (初期化) p_1 と p_2 の定める直線上にない最初の点 p_i を見つける. そのような点が存在しなければ凸包 $\{p_1,p_n\}$ を返す. そうでなければ3点 p_1,p_{i-1},p_i の巡回リストを構成する.

ステップ3. (更新) 各 $j(:=i+1,\cdots,n)$ に対して tangent$(p_j,\text{first},\text{last},p_{j-1})$ を呼び出す. first から last に向かう間の頂点をすべて取り除いて，巡回リストを更新する. first と last の間に p_j を入れる.

このアルゴリズムの正当性は，これまで述べてきたことより保証される. 計算量については，次の定理が成り立ち，この章後半の目的であった平面上の点集合に対する高速凸包アルゴリズムが完成した.

定理2.9 このアルゴリズムは，平面上の n 点の凸包を $O(n\log n)$ 時間で求める. また，n 点が座標に関してソートされて与えられていると，$O(n)$ 時間で凸包を求める.

[証明] ステップ1では，ソートするのに $O(n\log n)$ 時間かかる. ステップ2は，そのような点 p_i を $O(i)$ 時間で求めることができる. ステップ3においては，手続き tangent で走査される点は，first, last 以外すべて取り除かれ，そこで取り除かれると以降2度と扱われない. したがって，ステップ3全体で $O(n)$ 時間かかる. 以上まとめて定理を得る. ∎

演習問題

問題2.1 (a) どの2点間の距離も1であるような平面上の4点は存在しないことを示せ.

(b) どの2点間の距離も1か2であるような平面上の4点は存在しないことを示せ.

問題2.2 (a) P を平面上の n 点集合とする. また，L を P の点を少なくとも2つ含む P の上部支持線の集合とする. このとき，

$$\mathrm{CH}(P)=\bigcap_{l\in L} l^-$$

を示せ.

(b) $\mathrm{CH}(P)$ の境界は凸多角形であることを示せ.

問題2.3 P を平面上の n 点集合とする. このとき，

$$\mathrm{CH}(P)=\{x : x=\sum_{i=1}^{n}\lambda_i p_i,\ \sum_{i=1}^{n}\lambda_i=1,\ \lambda_i\geq 0,\ i=1,\cdots,n\}$$

を示せ.

問題2.4 すべての凸 n 角形は少なくとも $\lceil n/2\rceil$ 組の真の対蹠点対をもつことを示せ. また，偶数 n に対して，この評価が最良であることを示す例を作れ.

問題2.5 (a) 行列式 Δ_{xyz} の符号を計算する手続き sign(x,y,z) を短い pascal 流に書け. ただし，かけ算の回数を最小にするようにせよ.

(b) sign(x,y,z) を高々3回呼び出して disjoint(a,b) を実現する手続きを書け.

(c) sign(x,y,z) を1回呼び出して merge(a,b) の **case** 文の条件判定を実現する手続きを書け.

問題2.6 補題2.6を証明せよ.

問題2.7 手続き tangent$(p,\text{first},\text{last},\text{max})$ において，各 **while** 文における **or** 条件の後半部分が省略できることを示せ.

文献ノート

点集合の直径を実現する点対の数を決定するという問題は極めて古く，H. Hopf, E. Pannwitz [25] によって解決された. 直径の計算のために対蹠点

対を用いるというアイデアは，Shamosの論文で初めて導入され，そこにはすべての対蹠点対を求める線形時間アルゴリズムも与えられている．平面上の点集合に対する最初の $O(n\log n)$ 時間の凸包アルゴリズムはR. Graham [21] による．これはPreparata, Shamos [30] にもある．本書で与えた方法は，上下判定法 (beneath-beyond method) と呼ばれるもので，Edelsbrunner [15] からとってきた．しかし，これらの教科書にあるものには小さな間違いがあり，アルゴリズムが循環することがある．

3　最遠点対と行列の最大値問題

前章での直径問題への取り組み方は，入力した点集合から直径を実現する点対の超集合を定義し，特徴づけることであった．対蹠点対という超集合の大きさは十分に小さかったので，完全に数え上げて，そこから直径となる点対を直ちに選ぶことができた．もう1つ，直径を与える点対の超集合で，もっと直観的ですぐに思い浮かぶものがある．最遠点対の集合である．平面上の点集合 $P=\{p_1,\cdots,p_n\}$ に対して，

$$d(p_i,p_j)\geq d(p_i,p_k)\qquad (k=1,\cdots,n)$$

が成り立つとき，(p_i,p_j) を**最遠点対** (furthest neighbour pair) と呼ぶ．また，このとき p_j を p_i の**最遠点** (furthest neighbour) と呼ぶ．明らかに直径を実現する点対は最遠点対である．そして，以前のときと同様に，2つの自然な疑問が生じてくる．

(a) 最遠点対はどのくらいあるのか？

(b) 効率的に最遠点対を計算できるのだろうか？

全最遠点対問題 (all furthest neighbour pairs problem) とは，点集合 P のすべての最遠点対を求める問題のことである．