4 凸包問題

平面 $\mathbb{R}^2$ 上にあたえられた多数の点からなる集合 V の凸包 $\mathrm{cov}(V)$ を求める問題が**凸包問題** (convex hull problem) です．凸包 $\mathrm{cov}(V)$ は V のいくつかの点を頂点にもつ有界な凸多面体となりますが，これを求める凸包問題は計算幾何における最も基本的な問題であり，その効率的な解決こそ計算幾何の原点だったともいえます．また，凸包問題を解くアルゴリズムは，点位置決定問題やボロノイ図構成のためのアルゴリズムでサブルーチンとしても使われます．

4.1 分割統治法

分割統治法 (divide-and-conquier method) はアルゴリズムの設計技法としても最も基本的なものであり，凸包問題だけでなく様々な問題に応用することができます．名前のとおり，与えられた問題をいくつかの小さな問題に分割し，それぞれを再帰的に解いたのち，得られた答を統合してもとの問題の答を得ようとするものです．凸包問題を解くまえに，例として与えられたのデータを昇順に並べ替える**ソーティング** (sorting) を使って分割統治法のメカニズムを計算量の観点から説明しましょう．

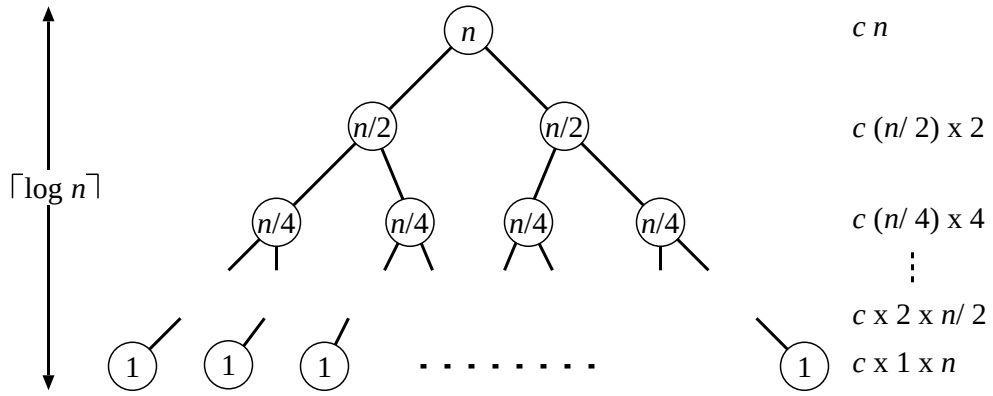
与えられたデータの数が n であれば，それを分割統治法は $\lceil n/2 \rceil$ 個のデータと $\lfloor n/2 \rfloor$ 個のデータに分割し，それぞれを再帰的にソートしたのち，ソートされた両方のデータを統合します．必要となる総計算量を $T(n)$ で表すことにすると，統合操作はデータ数に比例する計算量で行えるので，$c > 0$ を定数として次の漸化式を得ることができます:

$$T(n) = 2T(n/2) + cn.$$

再帰呼び出しはデータの数が 1 つになるまで繰り返されるので，深さ $\lceil \log n \rceil$ の 2 分木で表すことができます．この 2 分木の深さ k には 2^k 個の頂点があり，それぞれで $n/2^k$ 個のデータの統合が行われるので，結局

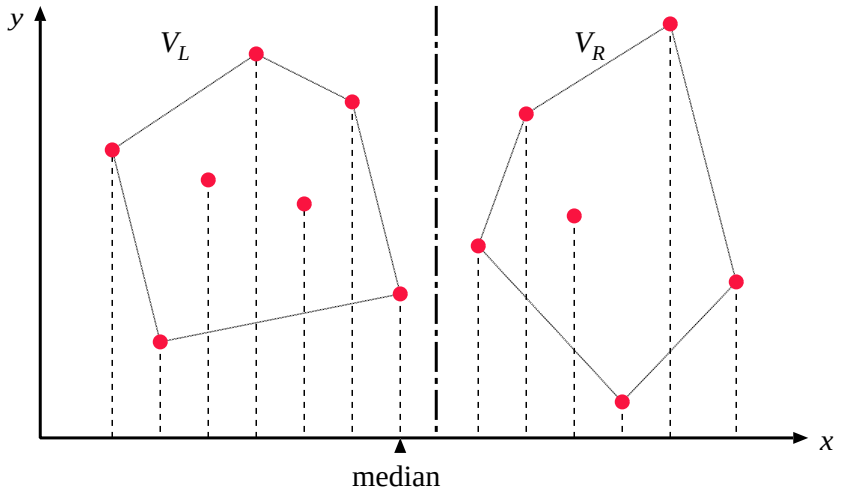
$$T(n) = \sum_{k=1}^{\lceil \log n \rceil} c(n/2^k)2^k = cn\lceil \log n \rceil = O(n \log n)$$

となり，ソーティングの最悪計算量は多項式時間であることがわかります．

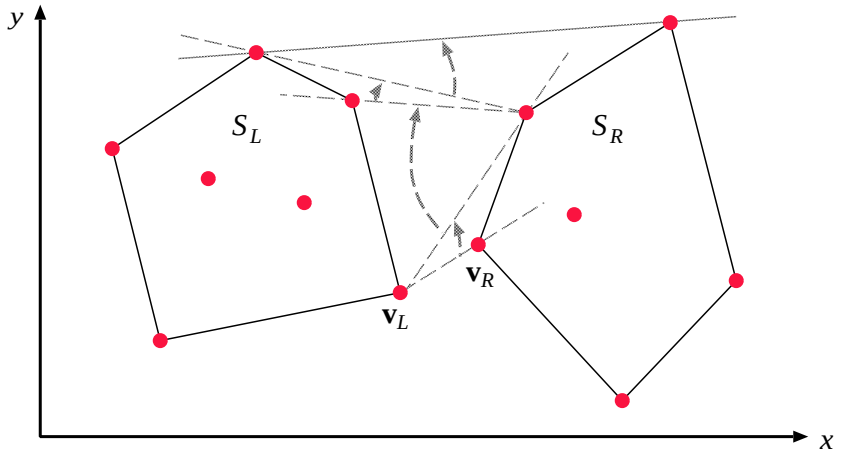


凸包問題への応用

平面上に与えられた n 個の点の集合 V の凸包を求めるためには，まず x 軸に垂直な直線によって V を $\lceil n/2 \rceil$ 個の点集合 V_L と $\lfloor n/2 \rfloor$ 個の点集合 V_R に左右二分割します．そして，V_L，V_R それぞれの凸包を再帰的に求め，最後にこの２つの凸包の共通外接線を求めることで１つの凸多角形に統合します．点集合 V の分割は，点の x 座標を昇順にソートし，その中央値を求めればよいので，先に紹介したソーティングによって $O(n \log n)$ で行うことができます．



２つの凸包をその共通外接線で統合するには，凸包が垂直線によって分離されていることを利用します．左右の凸包を S_L，S_R で表し，それぞれの頂点は時計回りに順序づけられているものとします．



procedure 統合 $(S_L,\ S_R)$

begin

S_L の最も右の頂点 $\mathbf{v}_L$ と S_R の最も左の頂点 $\mathbf{v}_R$ を求める;

while 直線 $\mathbf{v}_L$–$\mathbf{v}_R$ が S_L と S_R の上部外接線でない **do begin**

while 直線 $\mathbf{v}_L$–$\mathbf{v}_R$ が S_R の上部外接線でない **do**

$\mathbf{v}_R$ を S_R の次の頂点に置き換える;

while 直線 $\mathbf{v}_L$–$\mathbf{v}_R$ が S_L の上部外接線でない **do**

$\mathbf{v}_L$ を S_L の前の頂点に置き換える

end;

$\mathbf{v}_L$ の次の頂点を $\mathbf{v}_R$ とする;

S_L の最も右の頂点 $\mathbf{v}_L$ と S_R の最も左の頂点 $\mathbf{v}_R$ を求める;

while 直線 $\mathbf{v}_L$–$\mathbf{v}_R$ が S_L と S_R の下部外接線でない **do begin**

while 直線 $\mathbf{v}_L$–$\mathbf{v}_R$ が S_R の下部外接線でない **do**

$\mathbf{v}_R$ を S_R の前の頂点に置き換える;

while 直線 $\mathbf{v}_L$–$\mathbf{v}_R$ が S_L の下部外接線でない **do**

$\mathbf{v}_L$ を S_L の次の頂点に置き換える

end;

$\mathbf{v}_R$ の次の頂点を $\mathbf{v}_L$ とする

end;

この手続きにしたがえば，凸包 S_L, S_R それぞれの隣接する頂点を $\mathbf{v}_L$, $\mathbf{v}_R$ が一定の方向に移動するので，２つの凸包の頂点数に比例する手間で外接線を求めることができます.

Procedure 統合を用いて点集合 V の凸包を求める分割統治法を記述すれば以下のようになります:

algorithm 分割統治法 I

begin

与えられた n 点を x 座標の昇順にソートして得られる点の配列を V とする;

procedure 分割統治 (V) を呼び出し，V の凸包を求める

end;

procedure 分割統治 (V)

begin

配列 V を中央値で $|V_L| = \lceil |V|/2 \rceil$，$|V_R| = \lfloor |V|/2 \rfloor$ となるように左右２つの部分配列 V_L, V_R に分割する;

if $|V_L| > 1$ **then**

procedure 分割統治 (V_L) を呼び出して V_L の凸包 S_L を求める

else V_L を S_L とする;

if $|V_R| > 1$ **then**

procedure 分割統治 (V_R) を呼び出して V_R の凸包 S_R を求める

else V_R を S_R とする;

procedure 統合 (S_L, S_R) を呼び出し，S_L と S_R を統合して V の凸包 S を作る

end;

計算量の解析

Procedure 統合には，すでに見たとおり２つの凸包の頂点数に比例する手間しかかかりません．凸包の頂点数はそれに含まれる点の数を超えることはないので，最悪計算量は $O(|V|)$ です. Procedure 分割統治は点の数を半分にして自分自身を２回呼び出すことから，その総計算量を

$T(n)$ で表すことにすると次の漸化式が成り立ちます:

$$T(n) = 2T(n/2) + O(n).$$

ソーティングの場合と同様，再帰呼び出しが深さ $\lceil \log n \rceil$ の 2 分木として表されることから，この漸化式の答が

$$T(n) = O(n \log n)$$

であることがわかります．最初に行われる n 点のソーティングには $O(n \log n)$ の手間しかかからないので，結局 algorithm 分割統治 I 全体でも $O(n \log n)$ の手間で済みます．

4.2 逐次添加法

逐次添加法 (incremental method) は，少数の点の凸包を求めておき，それに点を 1 つづつ追加しながら必要に応じて凸包を更新していく方法です．

分割統治法の場合と同様，まず与えられた点集合 V の n 個の点を x 座標の昇順にソートします．その中で i 番めの点を $V(i)$，i 番めまでの点からできる凸包を $S(i)$ で表すことにしましょう．1 点だけからなる点集合の凸包はその点自身であるので，先の procedure 統合 $(S_L,\ S_R)$ を

$$S_L = S(i), \quad S_R = V(i+1)$$

として呼び出せば，$i+1$ 番めまでの点の凸包 $S(i+1)$ を求めることができます．このことから，n 点の凸包 $S(n)$ は次の単純なアルゴリズムで求められることがわかります:

algorithm 逐次添加法 I

begin

与えられた n 点を x 座標の昇順にソートして得られる点の配列を V とする;

$S(1) := V(1)$;

for $i = 1,\ \ldots,\ n$ **do**

procedure 統合 $(S(i),\ V(i))$ を呼び出して凸包 $S(i+i)$ を求める

end;

計算量の解析

すでに調べたように procedure 統合に必要な計算量は，統合すべき 2 つの凸包に含まれる点の数で押さえることができます．このことから直ちに

$$\sum_{i=1}^{n} i = n(n+1)/2 = O(n^2)$$

という algorithm 逐次添加法 I の大ざっぱな最悪計算量が得られます．しかし，「procedure 統合で $\mathbf{v}_L$, $\mathbf{v}_R$ が辿った途中の点は，その後二度と使われることがない」点に注目すれば，計算量の，よりきつい上界を知ることができます．

Procedure 統合の i 回めの呼び出しで，$\mathbf{v}_L$ は $S(i)$ の頂点を m_i 個辿るものとしましょう．この m_i 個の頂点のうち，外接線の接点となる 2 つを除いて $m_i - 2$ 個の頂点は次の凸包 $S(i+1)$ の構成から取り除かれます．いったん取り除かれた点は以後二度と凸包の構成には使われないので，その総数ともとの点の数 n との間には次の関係が成り立つはずです:

$$\sum_{i=1}^{n} (m_i - 2) \leq n. \tag{4.1}$$

一方，procedure 統合が i 回めの呼び出しで m_i 個の頂点を調べるのであれば，すべての呼び出しでは $\sum_{i=1}^{n} m_i$ に比例する手間がかかることになります．このことと (4.1) から，

$$\sum_{i=1}^{n} m_i \leq 3n,$$

つまり計算量は $O(n)$ であることがわかります．残念ながらアルゴリズムの最初に行われるソーティングに $O(n \log n)$ かかりますので，全体の計算量は結局，分割統治法と同じ $O(n \log n)$ です．