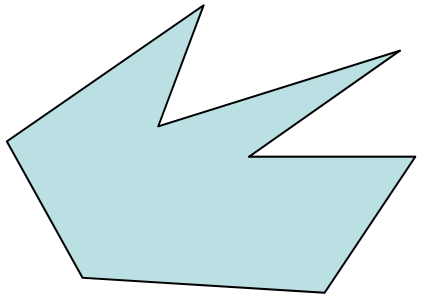
テーマ2：凸多角形に関する問題

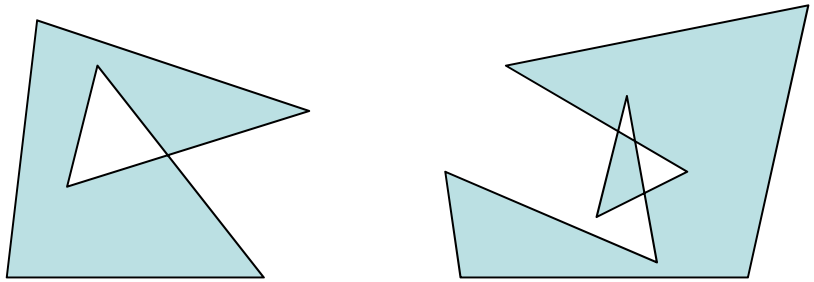
凸性の判定，凸多角形の直径，
内部と外部の区別
オーダ記法

凸多角形の定義と認識方法

定義：平面上の点を順に直線で結んでできる図形を多角形(polygon)といい，連続する2点を結ぶ線分を辺(edge)という．連続しないどの2辺も交差も接触もしないとき，その多角形は単純である(simple)という．



単純な多角形



単純でない多角形

単純な多角形は頂点列として表現するのが一般的．

$$P=(p_0, p_1, p_2, \cdots, p_n = p_0)$$

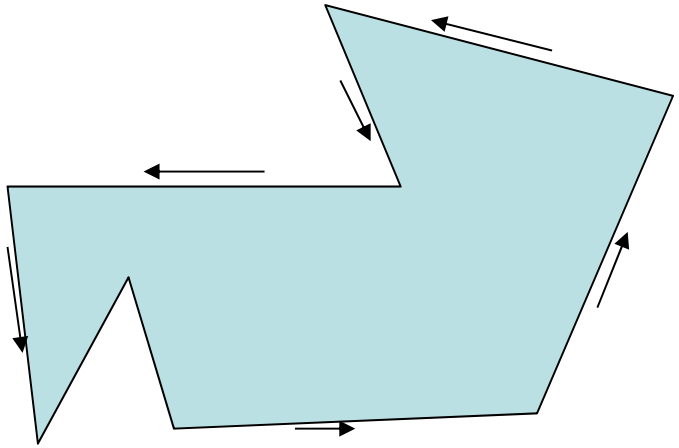
凸多角形の定義と認識方法

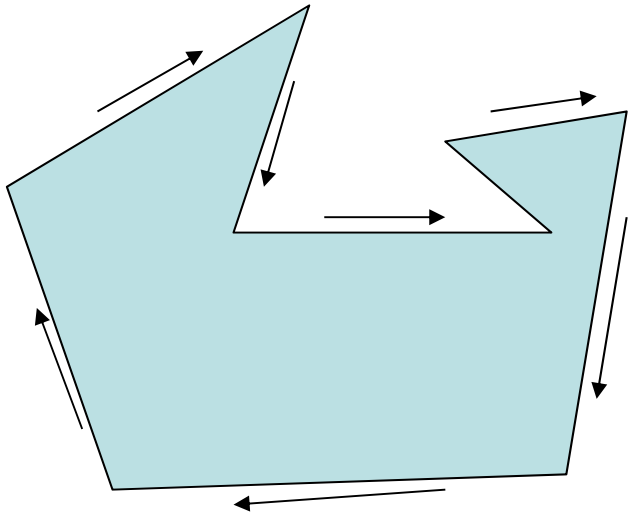
多角形の辺は，内部を左に見るように方向付けられていると仮定．
すなわち，多角形の辺は反時計回りの順を仮定．

$$area(P) = \frac{1}{2}\sum_{i=0}^{n-1} x_i (y_{i+1} - y_{i-1}),$$

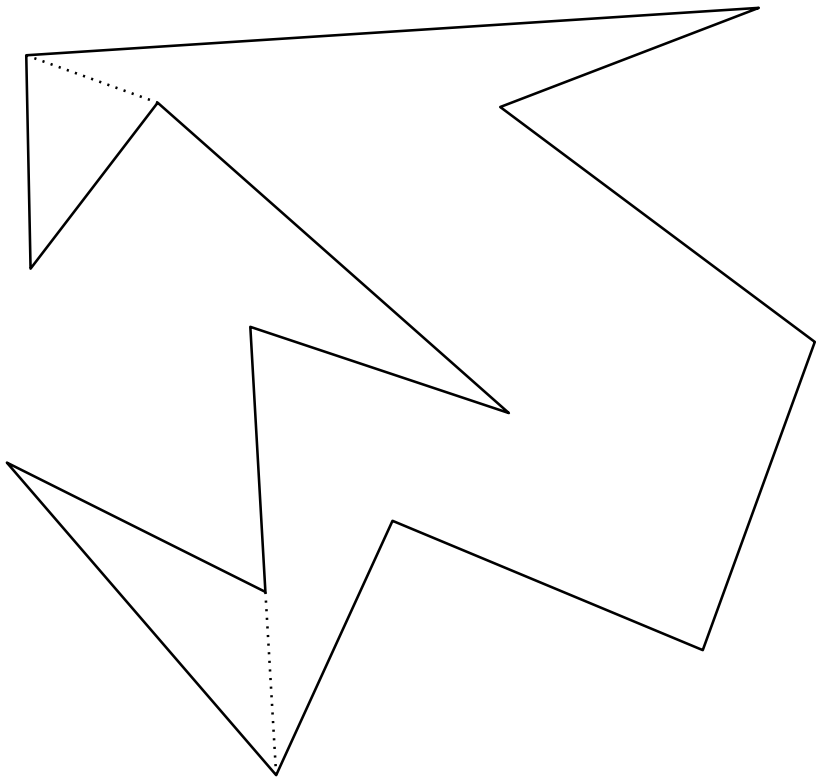
ただし，$y_n = y_0, y_{-1} = y_{n-1}$

上記の符号付面積 > 0 ➔ 多角形は反時計回り
< 0 ➔ 多角形は時計回り





多角形の面積を求める公式を導け.

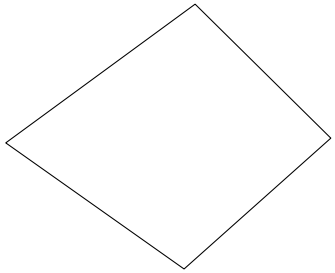


耳（対角線によって切り取ることができる三角形）の存在を仮定して証明せよ.

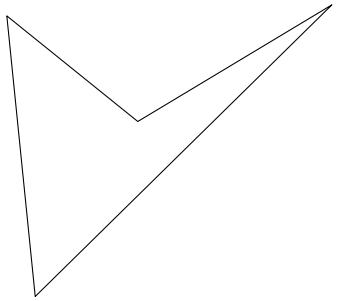
レポート課題1(1)
耳の存在を仮定して,
公式を証明せよ.

命題：4角形以上の多角形は耳をもつ.

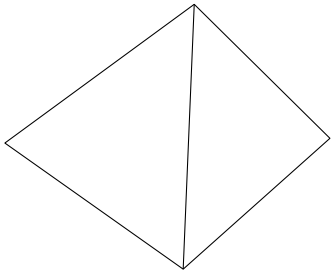
4角形のとき.

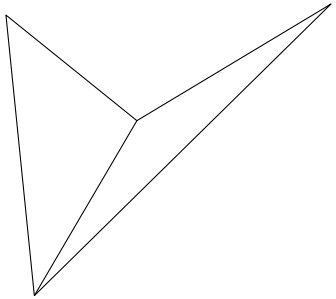


凸多角形



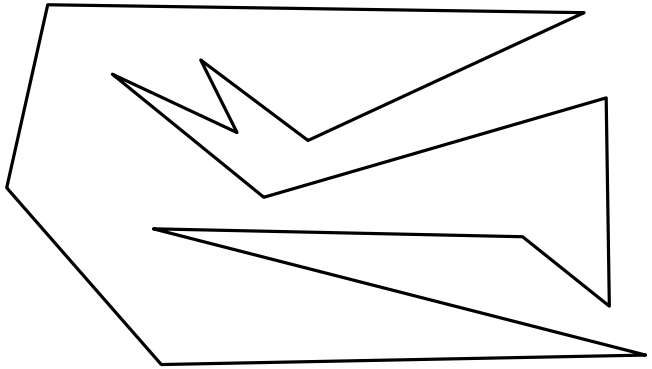
凸でない多角形

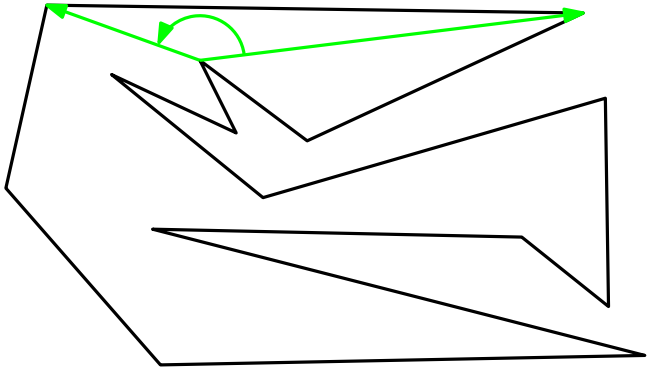




いずれも2個の三角形に分割できる.

n角形のとき（n > 4)





内角が180度以上の頂点があると，
そこから少なくとも1つの頂点は
見える（その頂点と線で結んだ時，
その線分は他の辺と交差しない）．
よって，必ず(n-1)角形以下になる．

内角が180度以上の頂点がなければ凸多角形なので，
どの対角線も多角形の内部だけを通る．その対角線で
多角形を分割すると，(n-1)角形以下のものができる．

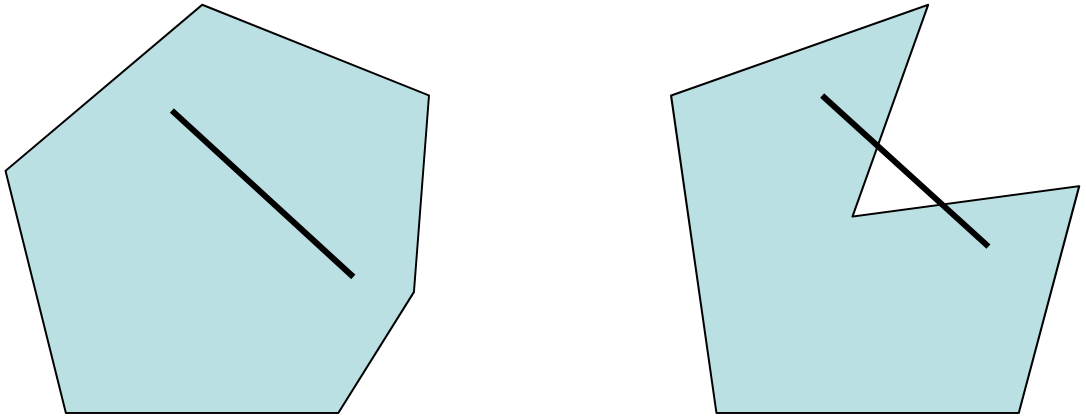
凸多角形の定義と認識方法

凸多角形の定義

多角形の内部の任意の２点を結ぶ線分がその多角形の内部に含まれるような多角形を凸多角形(convex polygon)という.

この定義では対象となる２点対が無限に存在するので，計算不可能.

性質１：多角形Pが凸多角形であるための必要十分条件は，Pのすべての対角線がPの内部にあることである.



凸多角形の定義と認識方法

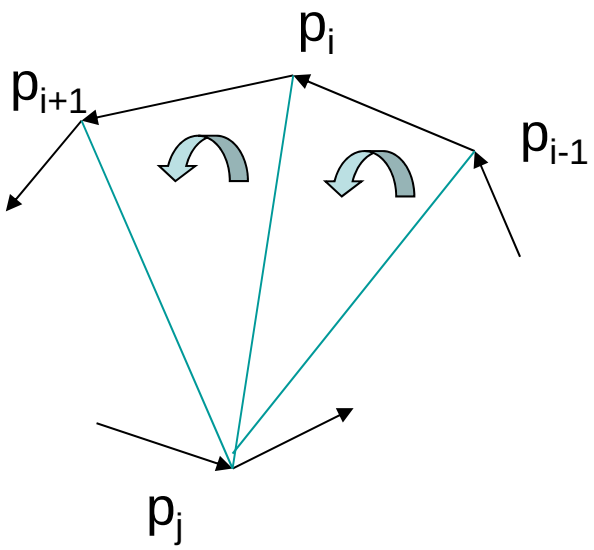
点列$P=(p_0, p_1, p_2, \dots, p_n=p_0)$が凸多角形を成すかどうかの判定：

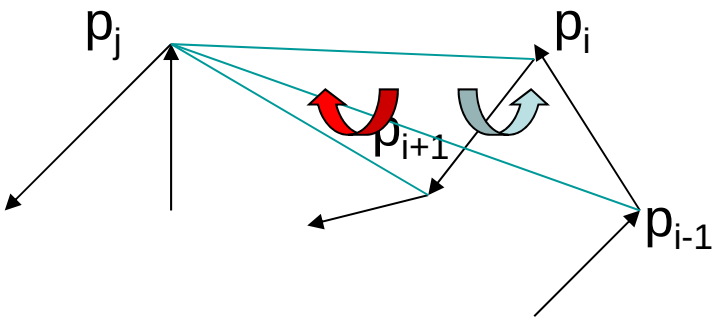
s_{ij}: 頂点p_iとp_jを結ぶ線分（対角線），$|i-j|\geqq 2$

仮定：多角形の辺は反時計回り（内部は辺の左）に順序付けられている．

s_{ij}が多角形の内部を通る

(1) 角（$p_{i-1}p_ip_{i+1}$）の内部を通り，(2) $p_{i-1}p_ip_j$と$p_ip_{i+1}p_j$は共に反時計回り





角の内部を通らない，
時計回りになっている

連続する3点で決まる角の内部を通らない対角線があると，その対角線に関連する2つの三角形の内の一つは必ず時計回りになっている．

よって，(1)の条件は調べなくてもよい．

凸多角形の定義と認識方法

与えられた点列が凸多角形を成すかどうかを判定するアルゴリズム

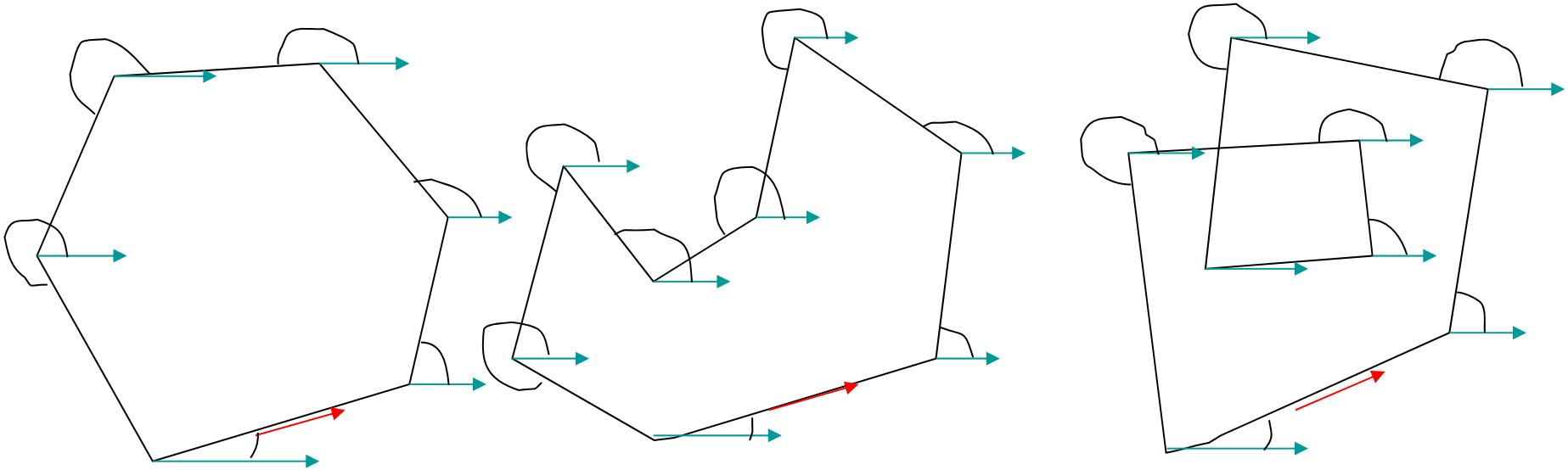
(1) 点列$P=(p_0, p_1, \ldots, p_n=p_0)$を入力する.
(2) for(i=0; i<n-1; i++)
(3) for(j=i+1; j<n; j++)
(4) if(Area(p_{i-1}, p_i, p_j)<0 or Area(p_i, p_{i+1}, p_j)<0)
凸多角形でないと報告して終了;
(5) 凸多角形であると報告して終了.
ただし, インデックスの計算はmod n で行う.

$O(n^2)$時間

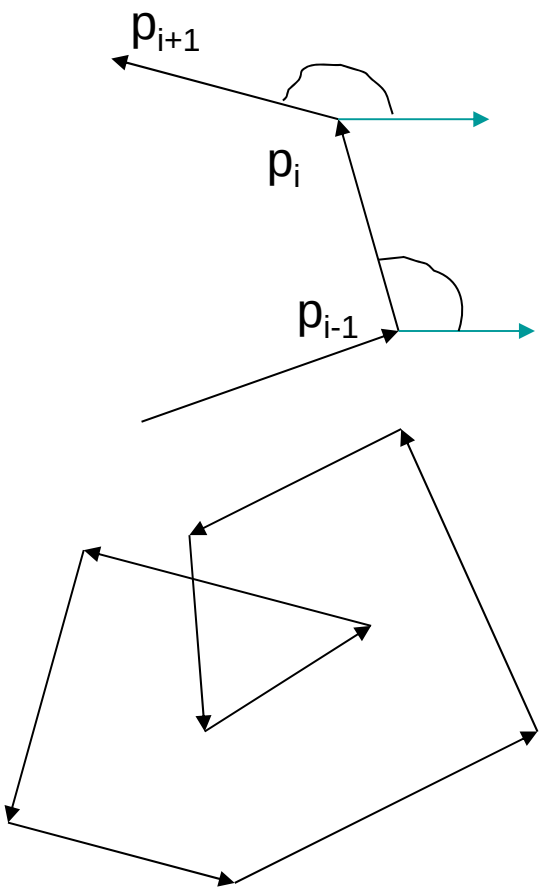
凸多角形の定義と認識方法

性質2：　点列$P=(p_0, p_1, p_2, \cdots, p_n=p_0)$が凸多角形を成す
　　⇔Pを標準形で表したとき，その辺角が単調に増加する．

Pの**標準形**とは，
　y座標最小の点を出発点とし，反時計回りに頂点を並べたもの．
辺角とは，頂点から右に延長した水平線と多角形の辺がなす角のこと．



凸多角形の定義と認識方法



三角形の符号付面積で
順に「反時計回り」を
判定していけば十分か？

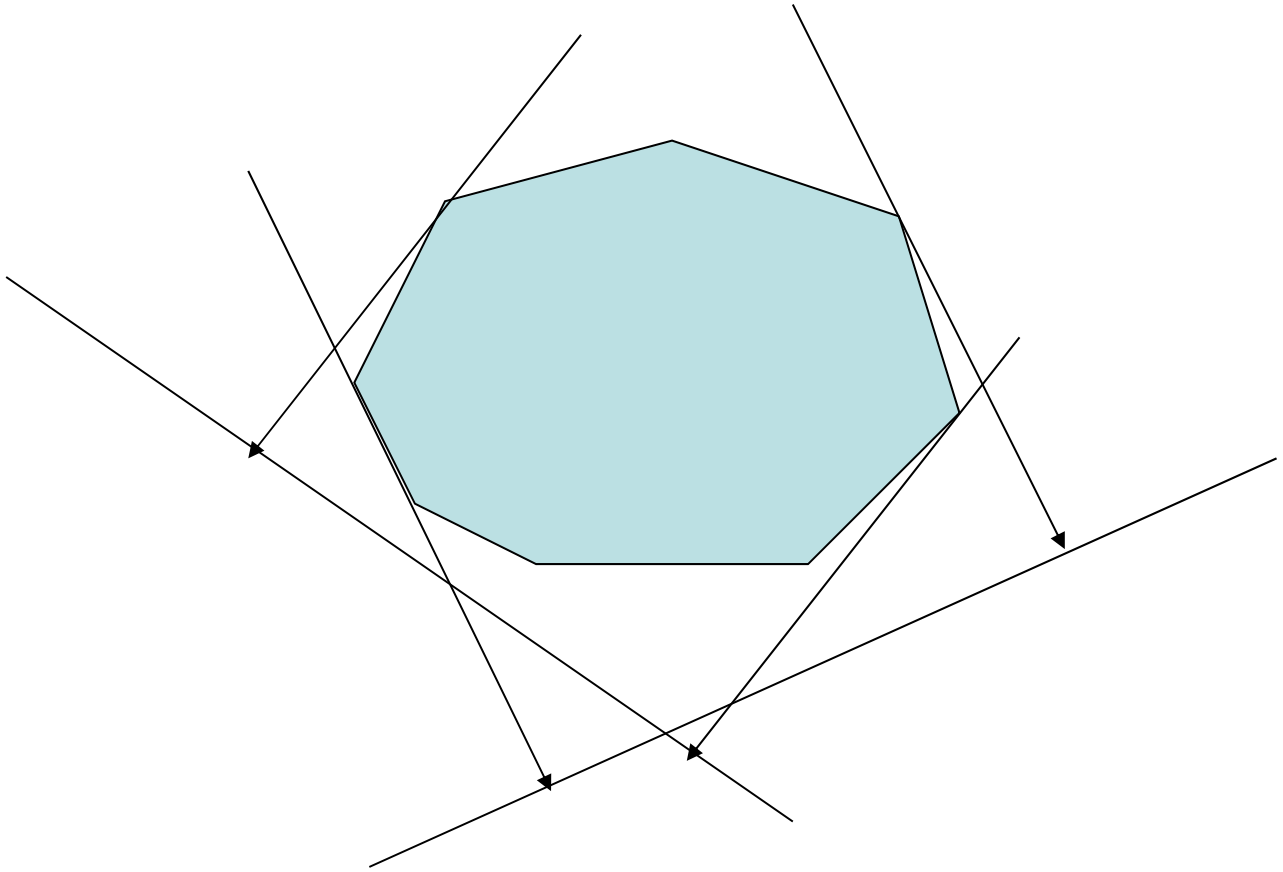
NO！
始点以外のところでy座標が
局所最小値をもつなら，
凸ではない．

まとめ：

1. y座標最小の頂点を見つける
2. 「左回り」であることを確かめる
3. y座標が始点以外の局所最小値を
   持たないことを確かめる

レポート課題1(2)
凸多角形の線形時間判定アルゴリズムを設計せよ

凸多角形の直径の計算



凸多角形の直径

どの方向に投影したとき，影の長さが最大になるか？
影の長さの最大値を直径として定義する.

凸多角形の直径の計算

凸多角形の直径

どの方向に投影したとき，影の長さが最大になるか？
影の長さの最大値を直径として定義する.

傾きを決めると，その傾きをもつ2本の平行な接線の間隔が直径となる.
接線上には頂点が必ず含まれるから，直径を求めるには，頂点対の
距離の最大値を求めればよい.

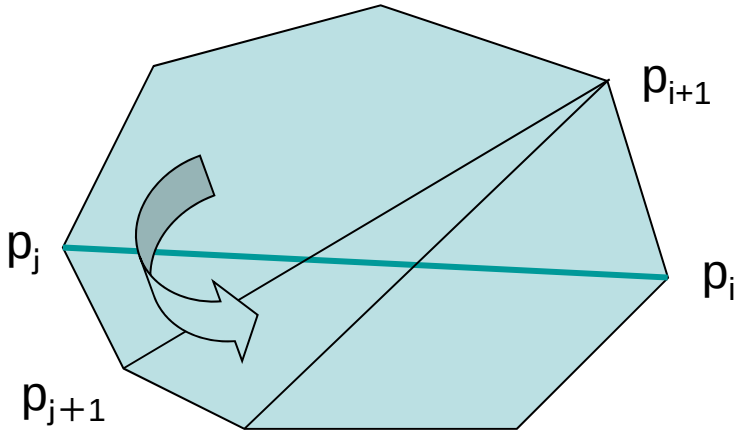
➔$O(n^2)$時間のアルゴリズム
高速化は可能か？

頂点対(p_i, p_j)が対蹠（たいせき）点対である
←p_iから最も遠い頂点がp_j，p_jから最も遠い頂点がp_i

対蹠点対は明らかにO(n)通りしかない.
対蹠点対をうまく列挙できれば，O(n)時間で直径が計算できる.

凸多角形の直径の計算

最初に，頂点0から最も遠い頂点kを求める．
(0, k)を最初の対蹠点対とする．
そこから反時計回りに探索する．

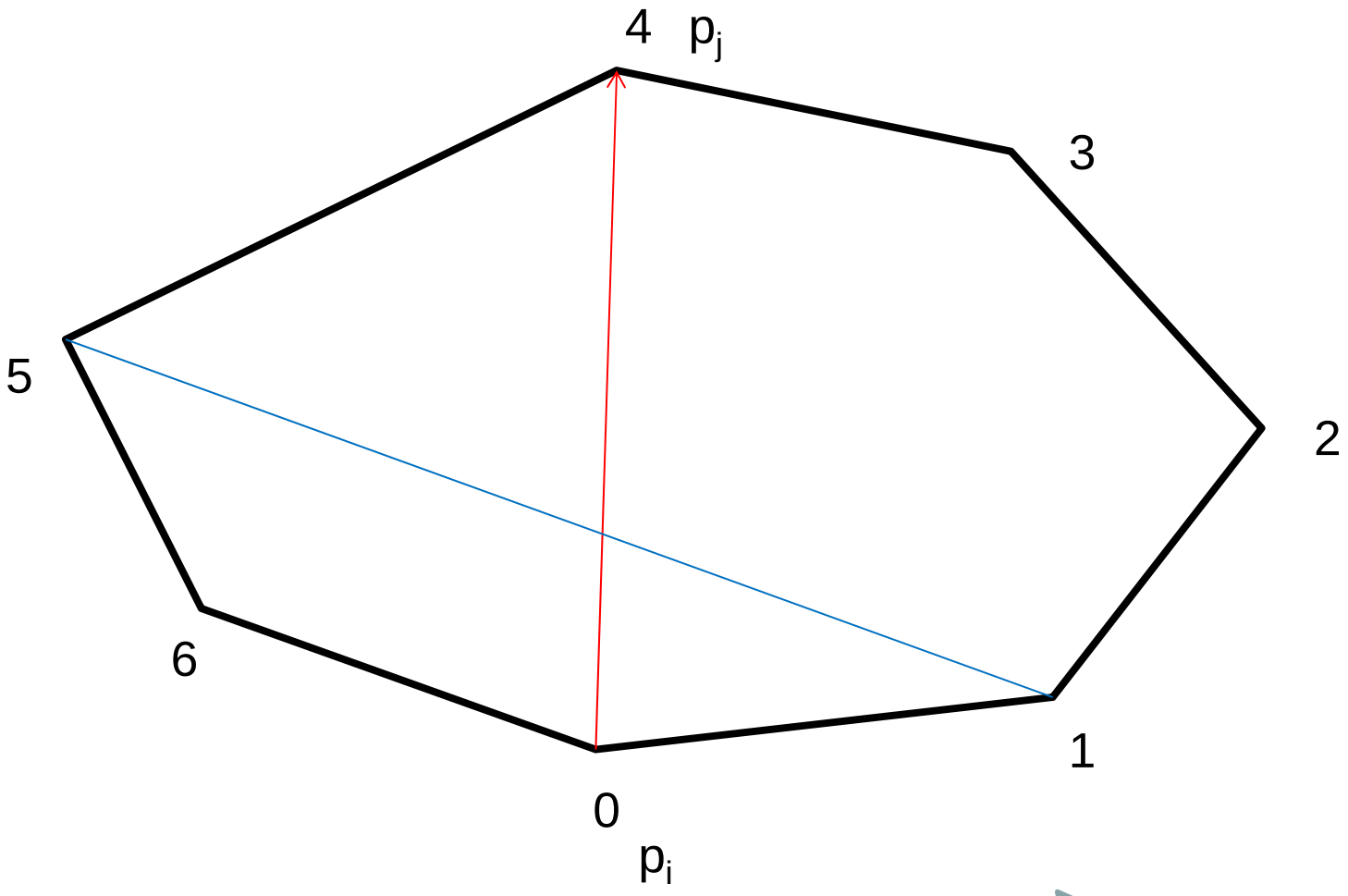


現在の対蹠点対が(p_i, p_j)のとき，
次の点p_{i+1}の対蹠点は
$p_{j+1}, p_{j+2}, \cdots$
と探していけばよい．

- p_i, p_j は対蹠点対なので，p_{i+1}の対蹠点はp_jではありえない．
- 対蹠点をつなぐ二つの線は一般に交わる（Why?）

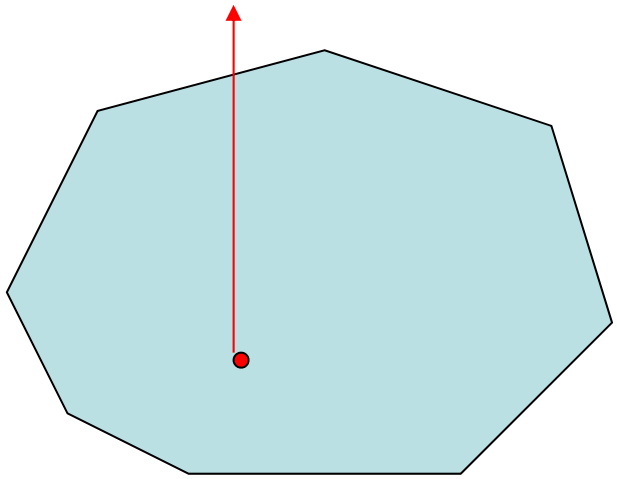
アルゴリズムの実行時間：一つの i に対しては定数ではないが，全体としては線形時間アルゴリズムにできる(Why)

具体的な凸多角形について前ページのアルゴリズムの動作を確かめよ.



ちょっと休憩…

凸多角形の内部と外部の判定



問題:
任意に指定された質問点qが凸多角形Pの内部，外部のどちらにあるかを判定せよ.

鉛直線算法:
質問点qから上方に延長した垂直な半直線が多角形の辺と何回交差するかを求め，偶数回なら外部，奇数回（1回）なら内部と判定する方法.

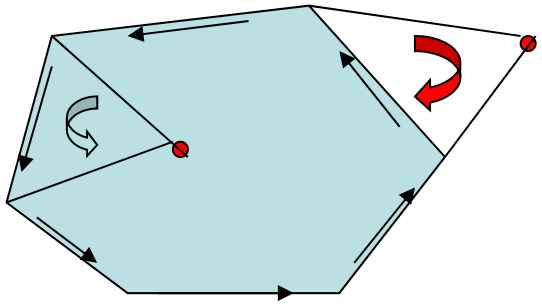
すべての辺と交差判定を行えばよいから，線形時間で判定可能.

しかし，実際には，ちょうど頂点を通る場合，質問点が辺上にある場合など，様々なケースがあり，プログラムは簡単ではない.

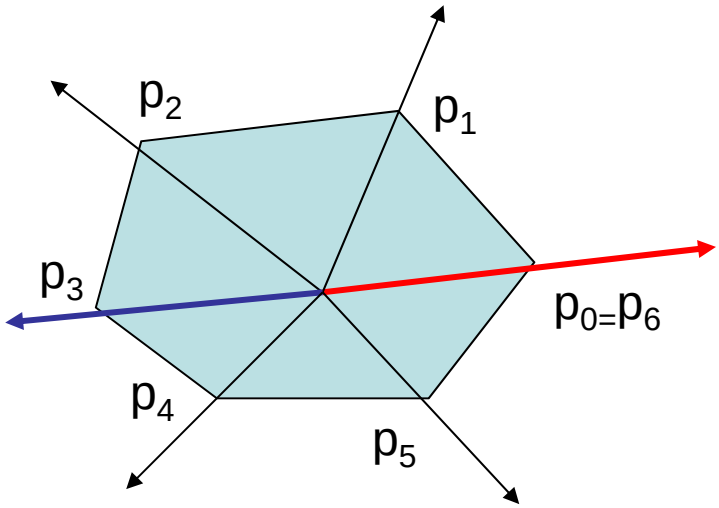
凸多角形の内部と外部の判定

符号付面積を利用する方法:

凸多角形が反時計回りに順序付けられているとき，すべての辺(p_i, p_{i+1})に対して，(p_i, p_{i+1}, q)がすべて反時計回りなら，点qは内部にあり，
そうでなければ外部にある.



凸多角形の内部と外部の判定

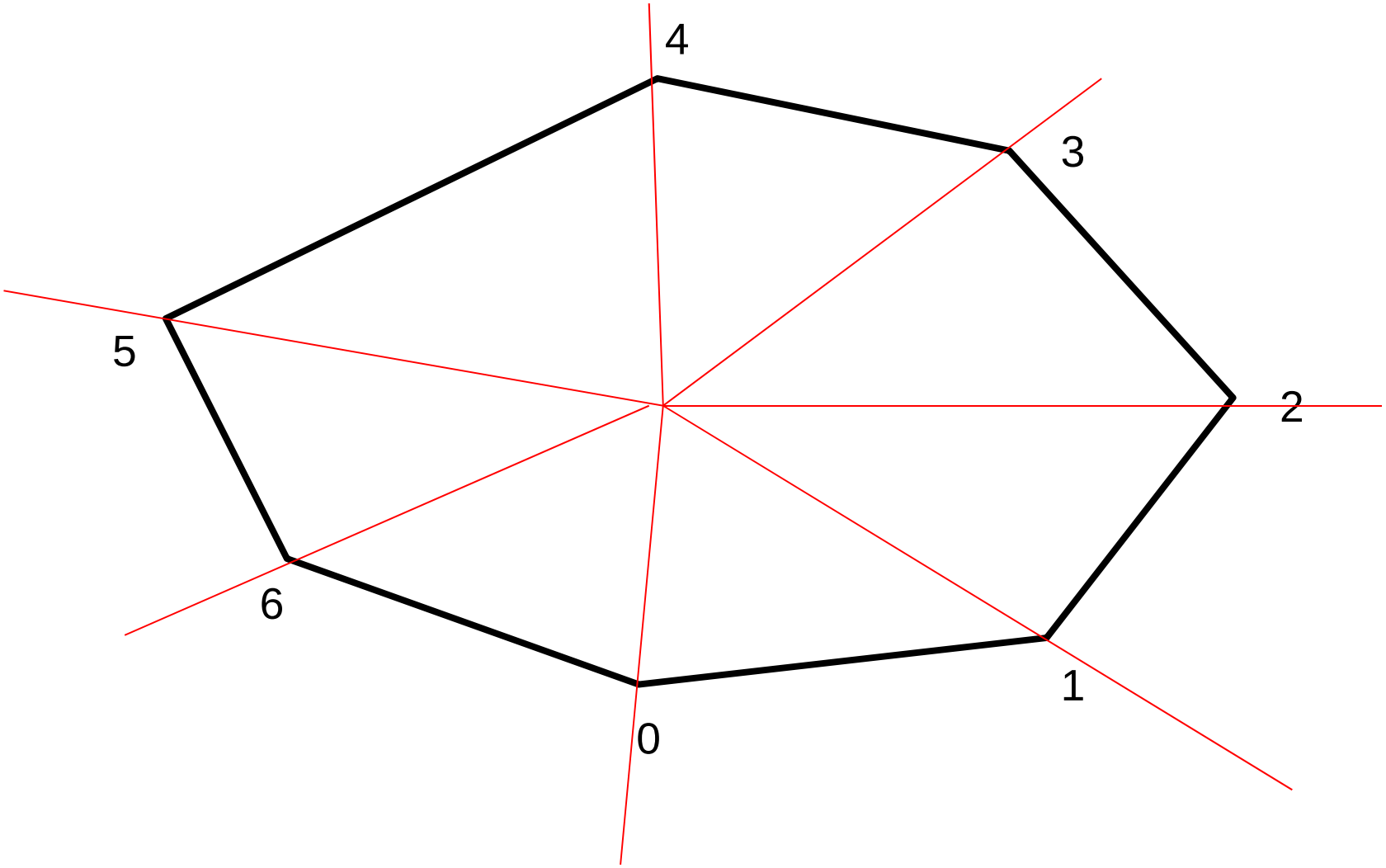


内部に固定する点pの選び方：
p_0と$p_{n/2}$の中点に選ぶとよい．

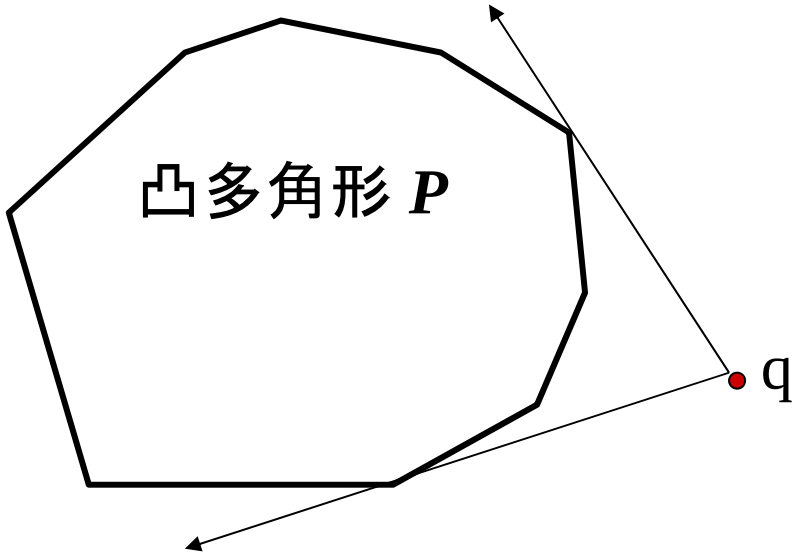
2分探索による方法

(1)多角形の内部に1点pを固定し，
点pから各頂点に半直線を引いて扇形に分割する．
(2)次に質問点がどの扇形に含まれるかを
2分探索によって求める．
(3)求めた扇形において多角形の内部に
含まれるかどうかを判定する．

具体的な凸多角形について前ページのアルゴリズムの動作を確かめよ.



凸多角形への接線

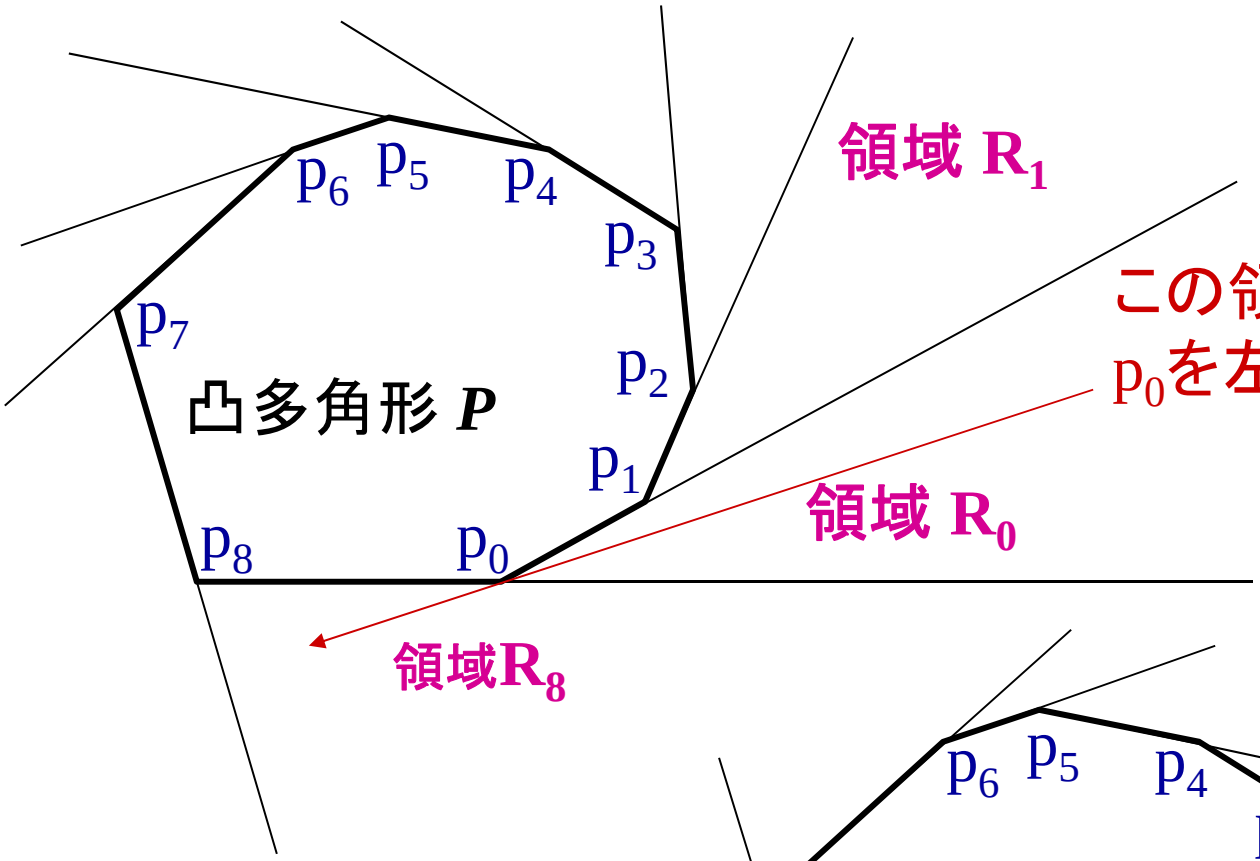


2本の接線が引ける

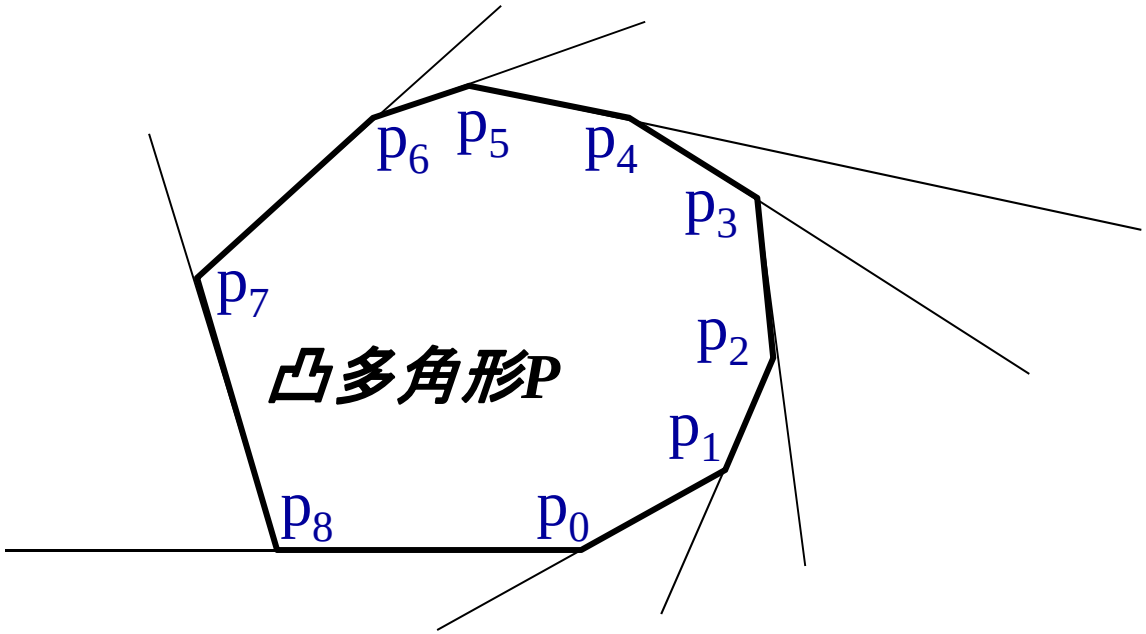
凸多角形の各頂点について，それが接点になっているかどうかを調べるのは容易．しかし，O(n)の時間がかかる．

もっと効率の良いアルゴリズムは？

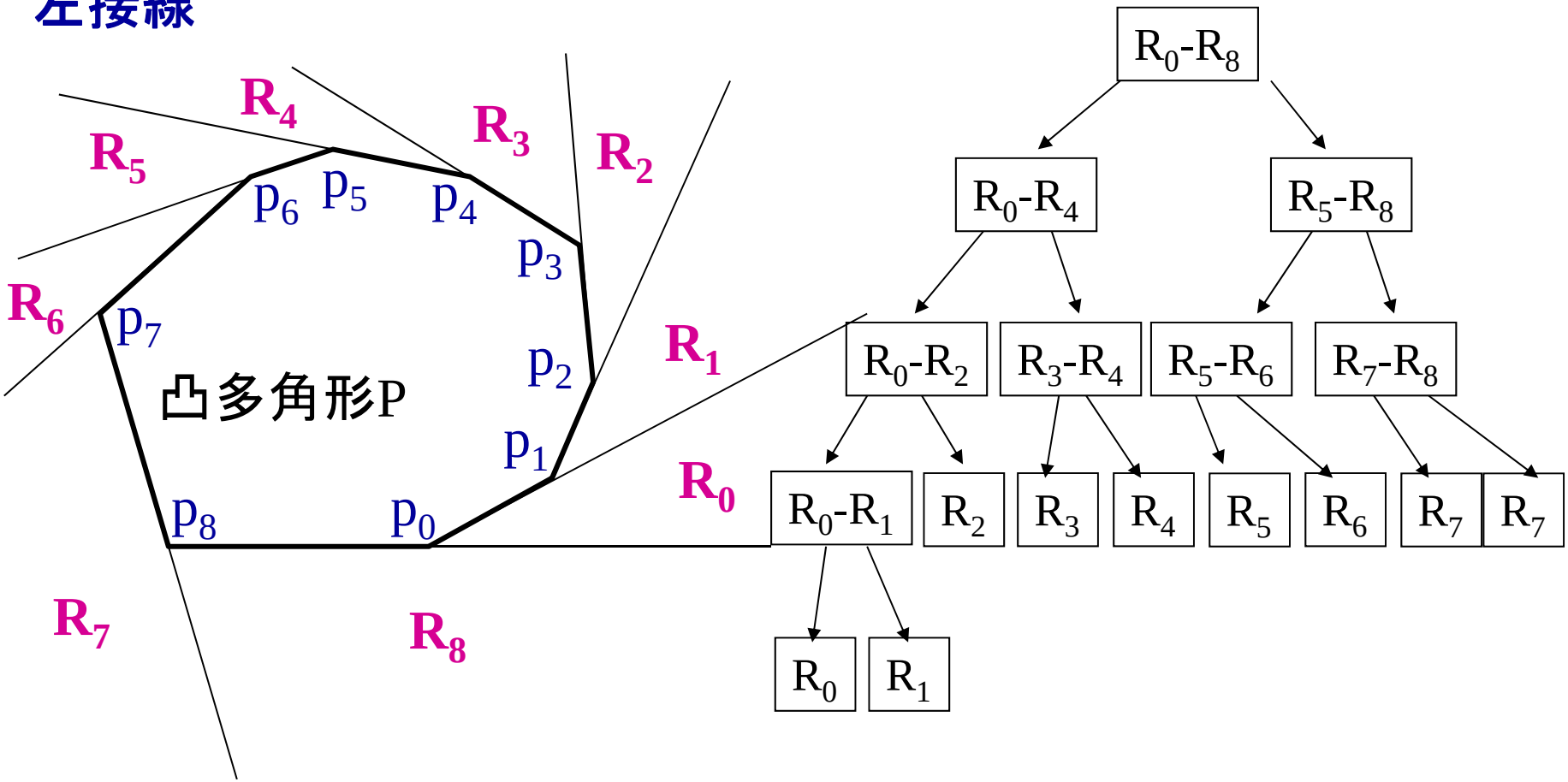
左接線


p_6
p_5
p_4
p_3
p_7
p_2
凸多角形 P
p_1
p_8
p_0
領域 R_1
この領域内の任意の点は
p_0を左接点としてもつ
領域 R_0
領域R_8

右接線


p_6
p_5
p_4
p_3
p_7
p_2
凸多角形P
p_1
p_8
p_0

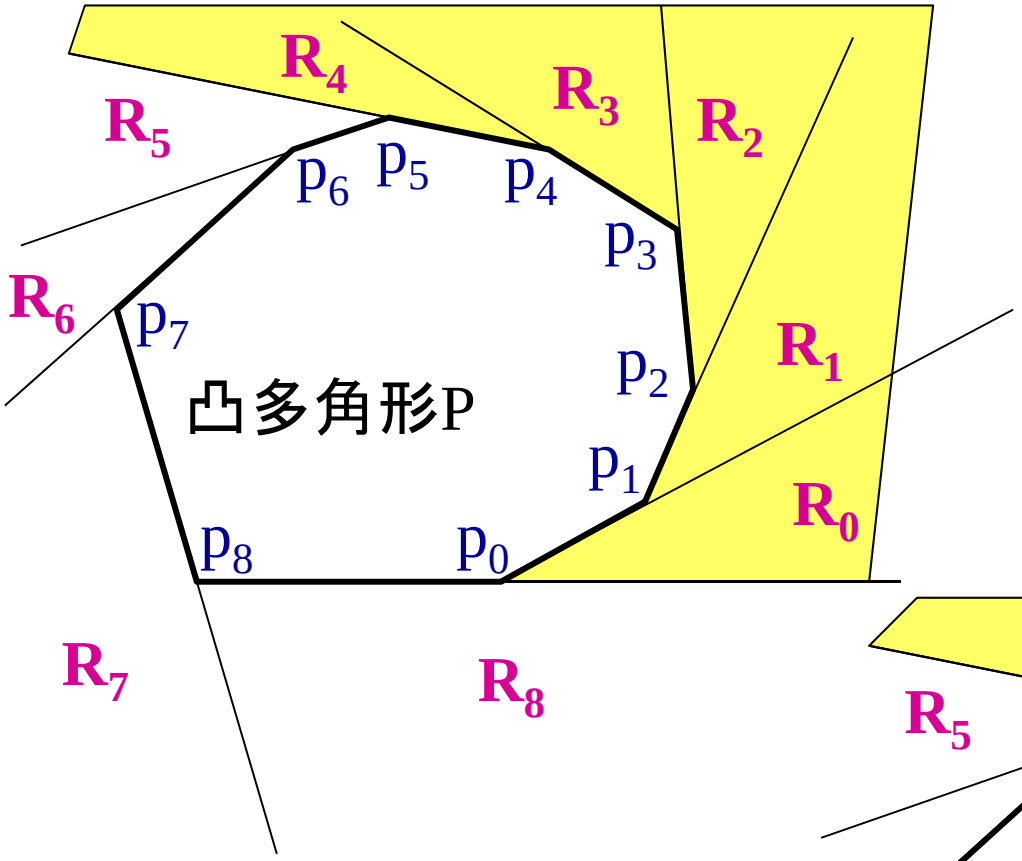
左接線

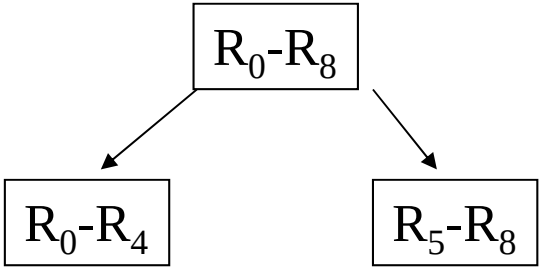


質問点qが与えられたとき，qを含む領域 $\boldsymbol{R_i}$ を求めよ.

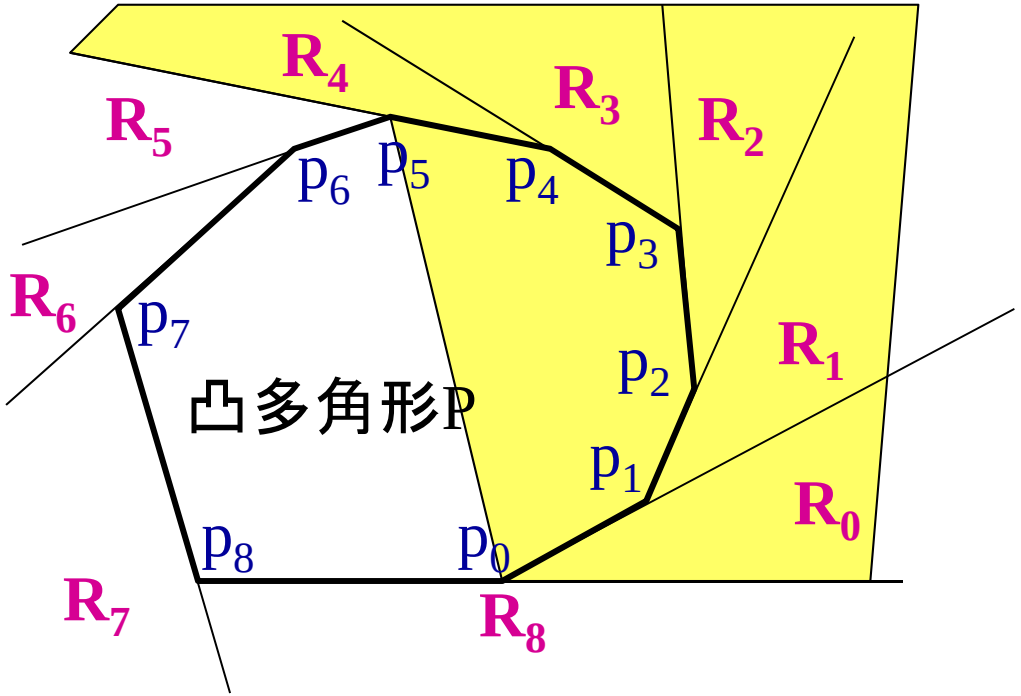
2分探索が可能

左接線



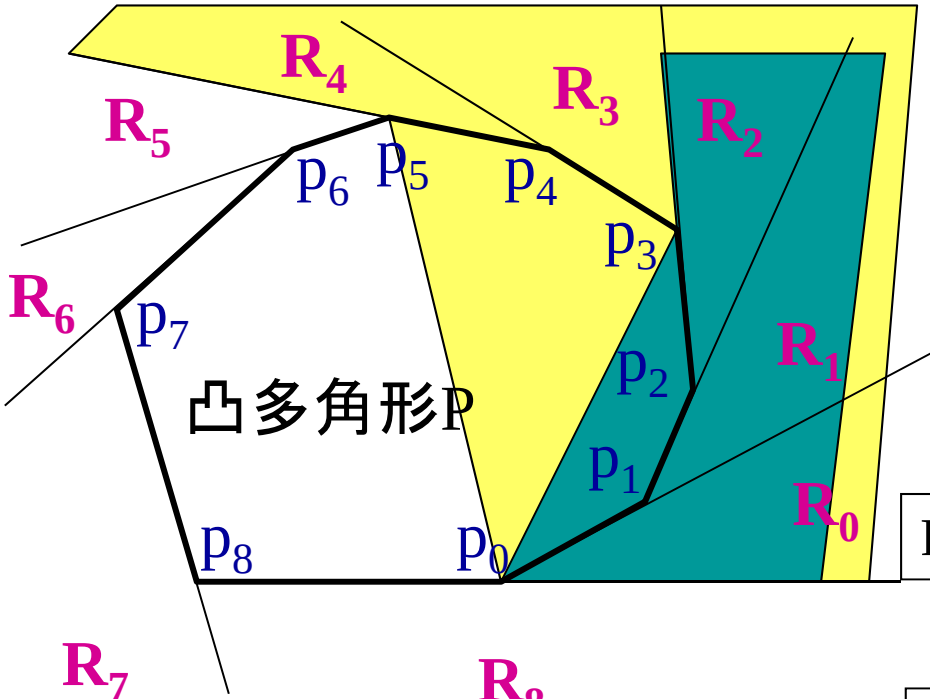


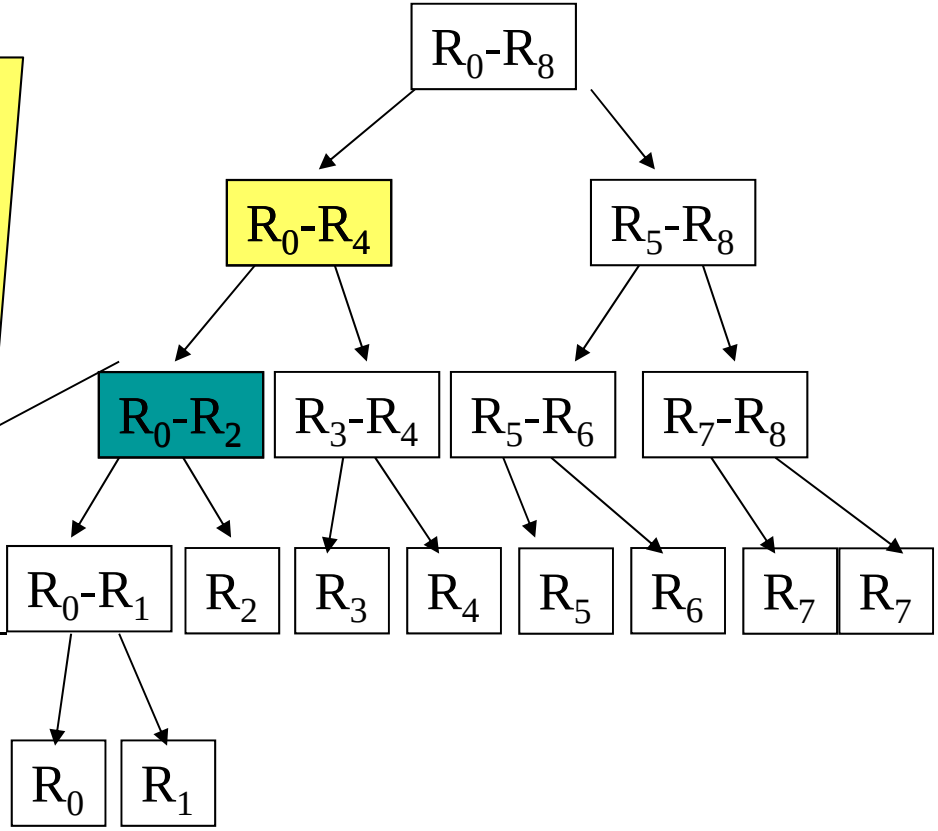
どの領域がqを含むか?

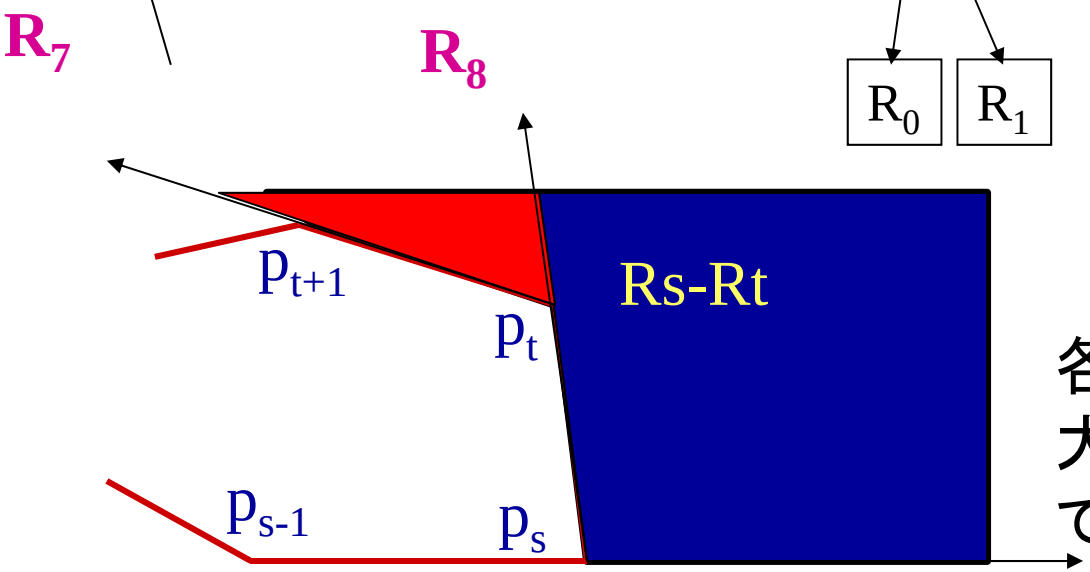


質問点は多角形の外にあることを知っているので,
R_0-R_4 を影をつけた領域で置き換える

左接線







各領域は2つの無限に大きい三角形に分割できる.

オーダ記法

漸近的計算量：入力のサイズnが十分に大きくなったときに
　計算量がどのような割合で増加するかを表したもの．
計算量の増加の割合を示すのが目的なので，
　主要項だけで十分，また係数も重要でない．

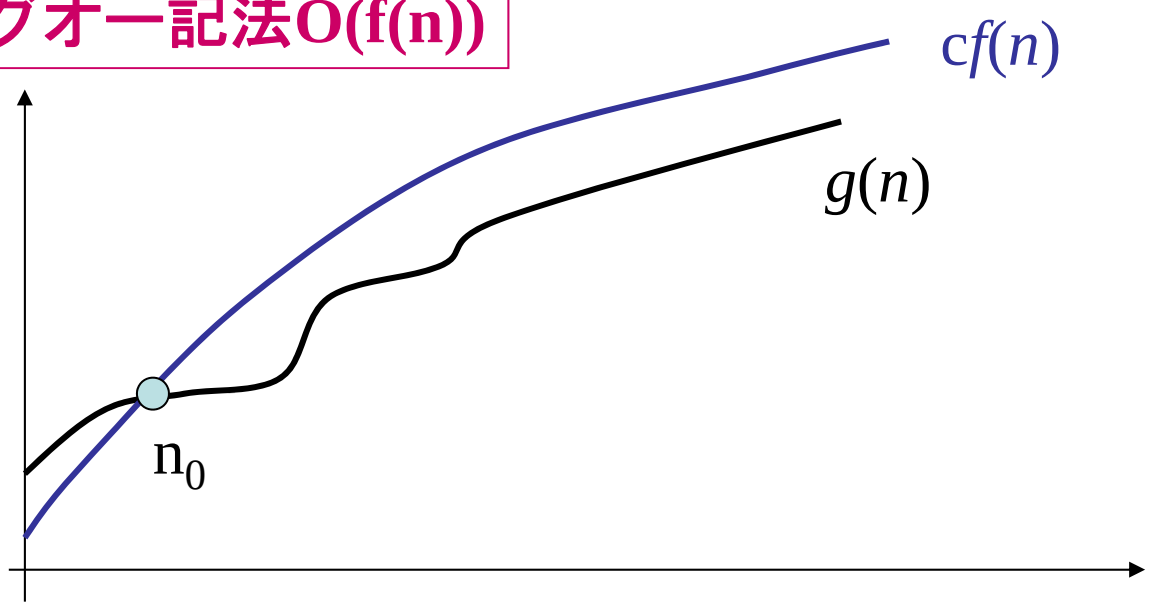
ビッグオー記法O(f(n))

$$O(f(n)) = \{g(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 (g(n) \leq cf(n))\}$$

$O(f(n)) = \{g(n)$：すべての$n \geq n_0$に対して$g(n) \leq cf(n)$
であるような正の定数cとn_0が存在する$\}$

g(n) ∈ O(f(n))と書く代わりに，便宜　上，
g(n)= O(f(n))と書くこともある．

ビッグオー記法O(f(n))



計算量の上界を表すのに用いる.
たとえば，挿入法と呼ばれるデータ整列（ソート）法は，
逆順に並べられたデータが入力されると，データ数nに
対してn^2に比例する時間がかかってしまう.
したがって，挿入法の計算時間は$O(n^2)$と言える.

挿入法はO(n)時間でソートを完了することもある.
すでにソート済みのデータが入力された場合がそうである.

質問:
同じ問題を解く3つのアルゴリズムA, B,Cがあり，それぞれの計算時間が，n, n^2, 2^nであることが分かっているとする．このとき，n=100, 200, 1000, 10000のときの計算時間を求めよ.

答:

n	100	200	1000	10000
n	100	200	1000	10000
n^2	10000	40000	1000000	100000000
2^n	$2^{100} \fallingdotseq 10^{30}$	$\fallingdotseq 10^{60}$	$\fallingdotseq 10^{30\ 1}$	$\fallingdotseq 10^{30\ 10}$

質問:
同じ問題を解く3つのアルゴリズムA, B,Cがあり，それぞれの計算時間が，n, n^2, 2^nであることが分かっているとする．このとき，10^6単位時間に解くことのできる問題のサイズを求めよ.

答: f(n)=nのとき，$n \leqq 10^6$より，$n=10^6$まで解ける.
f(n)= n^2のとき，$n^2 \leqq 10^6$，$n=10^3$まで解ける.
f(n)=2^nのとき，$2^n \leqq 10^6$，$n \leqq 6/\log 2 = 6/0.3 = 20$まで解ける.

質問:
アルゴリズムAがプログラムP(A)として実装されているとする．アルゴリズムAの計算時間複雑度を知るために，入力データのサイズnを何通りも変えて，それぞれ乱数で1000通りの試験データを作成し，その入力に対する計算時間を方眼紙上にプロットすることによって，$an^2 + bn + c$という曲線に最もよくマッチすることを実験的に確かめた．この結果から，このアルゴリズムの時間複雑度は$O(n^2)$だと言えるか．

答:アルゴリズムの計算複雑度が$O(n^2)$であるのは，サイズnのどんな入力に対しても，その計算時間がn^2に比例する関数で抑えられるときに限る．この場合には1000通りしか試していないので，もっと時間のかかる入力の例があるかも知れないので，$O(n^2)$であると言うことはできない．

質問:
アルゴリズムの計算時間を解析したところ，サイズnのどんな入力に対しても $f(n) = an^2 + bn + c$ という式で定まる時間以下であることが分かった．このアルゴリズムの計算時間をO( )の記法で表すとどうなるか．

答:係数のa, b, cはすべて定数であるから，それらの最大値をMとすれば，$an^2 + bn + c \leq Mn^2 + Mn + M \leq 3M\, n^2$ を得る．3Mは定数なので，$an^2 + bn + c = O(n^2)$を得る．

質問:
同じ問題を解くのに2つのアルゴリズムがある．アルゴリズムAの計算時間は$O(n^3)$であることが分かっており，アルゴリズムBは$O(n^2 \log n)$であることが分かっている．同じ入力に対して2つのアルゴリズムを実行したとき，その実行時間に関して何が言えるか？

答: $O(\ )$記法は計算複雑度（計算時間）の上界を与えているだけで，それだけの時間がかかる入力例があるかどうかについては何も言っていない．したがって，この場合，具体的な入力に対してどちらが早く終わるかについては何も言えない．

質問:
$a>0$を定数とするとき，$(n+a)^2 = O(n^2)$であると言えるか？$a<0$のときはどうか？

答: 定義に基づいて考えよう．$a>0$のとき，$(n+a)^2 = n^2+an+a^2 \leqq 3an^2$
であり，$3a$は定数であるから，$(n+a)^2 = O(n^2)$を得る．
$a<0$のときは，$n+a < n + |a|$，$|a|>0$であるから，$(n+a)^2 < (n+|a|)^2 \leqq 3|a|n^2$
であり，$3a$は定数であるから，$(n+|a|)^2 = O(n^2)$を得る．

質問:
任意の正定数eに対して，$\log n = O(n^e)$，$n^e = O(\log n)$は成り立つか？

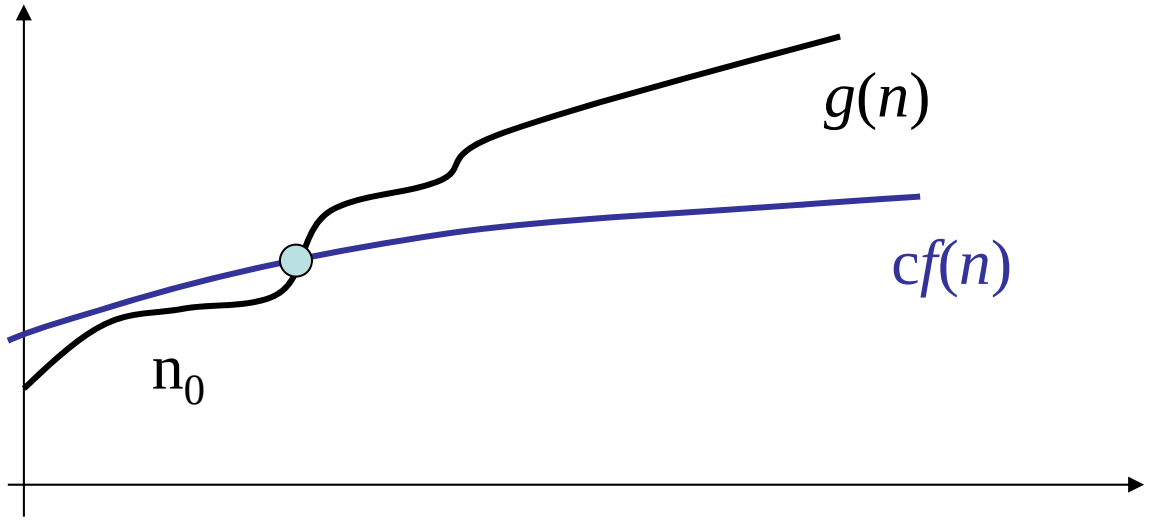
答:
定義に基づいて考えよう．$\log n < cn^e$であるような定数cが存在すれば，
$\log n = O(n^e)$と言える．$n^e > (\log n)/c$なので，両辺の対数を取ると，

$$e \log n > \log((\log n)/c),\ e > \log((\log n)/c)/\log n$$

を得る．右辺はnに関して単調減少なので，いつかはeより小さくなる．そのような最初のnの値n_0を求めれば，$n > n_0$のところでは $\log n < cn^e$ が成り立つ．よって，$\log n = O(n^e)$と言える．

$n^e = O(\log n)$ を言うには，上で不等式の向きを逆にしたものを考える必要があるが，$e < \log((\log n)/c)/\log n$ は十分に大きなnの値に対しては成り立たないので，同じ議論は成り立たない．

ビッグオメガ記法Ω(f(n))



$$\Omega(f(n)) = \{g(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 (cf(n) \leq g(n))\}$$

$\Omega(f(n)) = \{g(n)$: すべての$n \geq n_0$に対して$cf(n) \leq g(n)$

であるような正の定数cとn_0が存在する$\}$

ビッグオメガ記法Ω(f(n))

計算量の下界を表すのに用いる.

n個のデータをデータ間の比較を用いてソートするには,
どんなアルゴリズムでも必ずn log nに比例する時間以上が
かかってしまう最悪の場合が存在する.

よって, **ソーティング問題の下界**は**Ω(n log n)**であると言える.

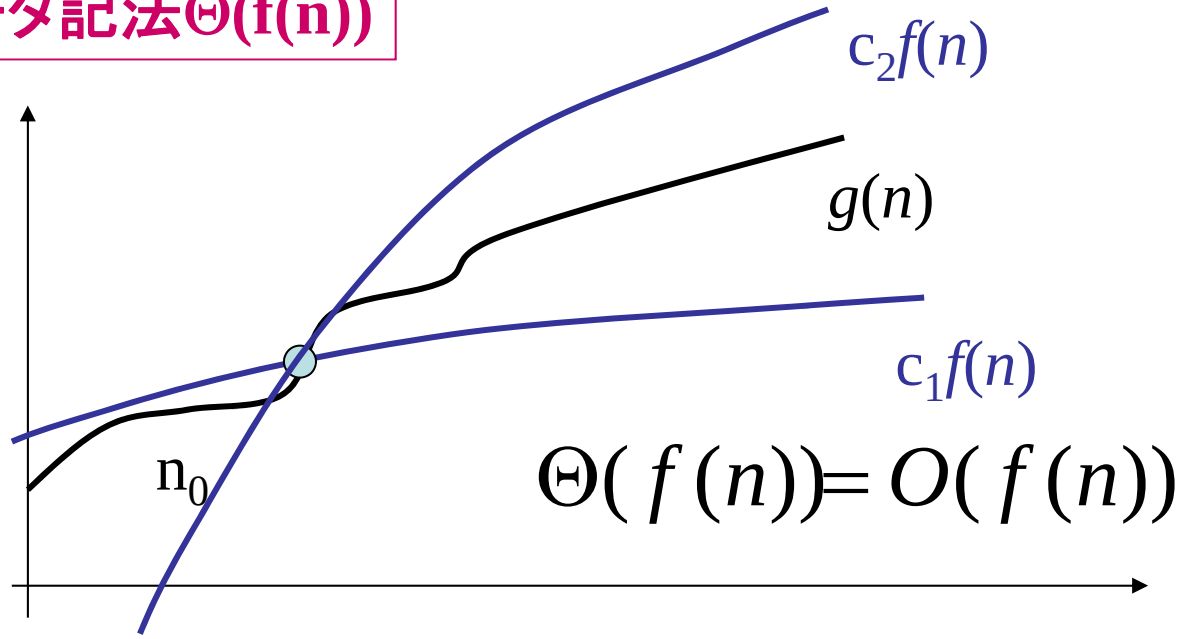
質問:
アルゴリズムの計算量が$\Omega(n^2)$であるというのは，どういう意味ですか？

答: このアルゴリズムに対するサイズnの入力として，n^2に比例する時間がかかってしまうようなものが存在する，すなわち，アルゴリズムの最悪の計算時間は少なくともn^2に比例する程度であるということを意味している．

質問:
ある計算問題の計算量が$\Omega(n^2)$であるというのは，どういう意味ですか？

答: その問題を解くアルゴリズムは何通りも考えられるが，そのようなどんなアルゴリズムに対してもn^2に比例する時間がかかってしまうような入力例が存在する，すなわち，その問題を解くどんなアルゴリズムも最悪時の計算時間は少なくともn^2に比例する程度であるということを意味している．ただし，最悪の入力というのはアルゴリズムごとに違っていてもよい．

シータ記法Θ(f(n))



$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

$$\Theta(f(n)) = \{g(n) \mid \exists c_1, c_2 > 0, \exists n_0, \forall n \geq n_0 \quad (c_1 f(n) \leq g(n) \leq c_2 f(n))\}$$

$\Theta(f(n)) = \{g(n)$: すべての $n \geq n_0$ に対して

$c_1 f(n) \leq g(n) \leq c_2 f(n)$ であるような正の

定数 c_1, c_2 と n_0 が存在する $\}$

質問:
比較だけを用いたソーティングのアルゴリズムとしてヒープソートがあるが，これはどんなn個のデータでもn log nに比例する時間でソートできる．また，比較だけを用いてn個のデータをソートする問題についてはΩ(n log n)という下界も知られている．これらのことから，比較だけを用いてn個のデータをソートする問題の計算複雑度について何が言えるか．

答: 上に述べたように，比較だけを用いてn個のデータをソートする問題についてはO(n log n)の上界とΩ(n log n)の下界が判っているので，その計算複雑度はΘ(n log n)であると言える．