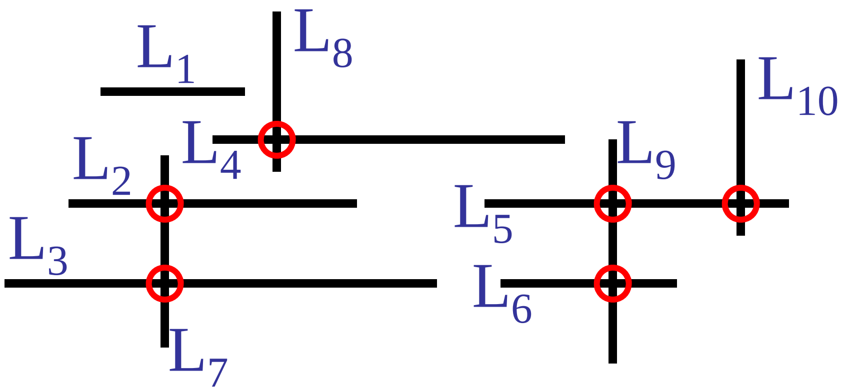
テーマ5：平面走査法

多角形領域の長方形への分割，

線分交差判定問題

平面走査法

問題1: 平面上に合計 n 本の水平線分と垂直線分が与えられたとき，それらの交点をすべて列挙せよ.



腕力法:

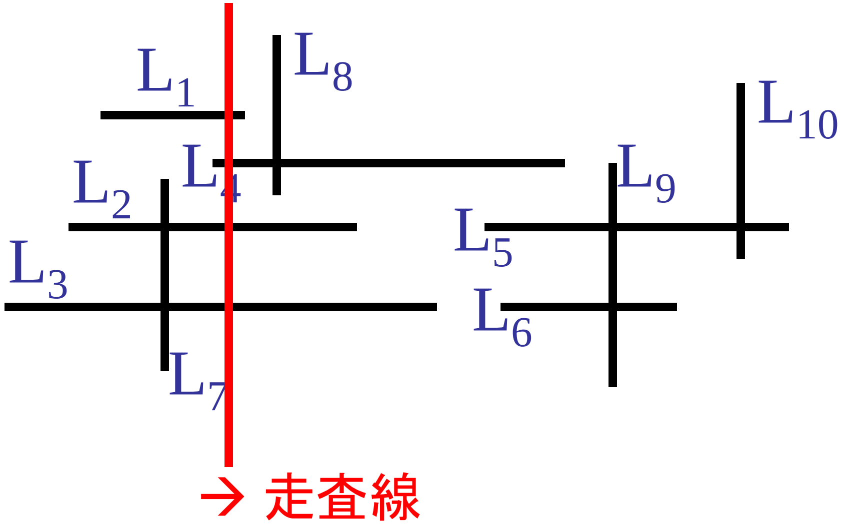
for each 水平線分 L_i do
    for each 垂直線分 L_j do
        if L_i が L_j と交差する
            then 交点をレポート

$O(n^2)$ 時間

平面走査法の考え方

２次元の問題

→ １次元の問題の系列として解決



走査線と交差する水平線分を上下の順を保って管理.

垂直の走査線を左から右に動かしていく.

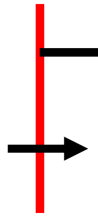
このとき, 走査線と交差する水平線分の集合（走査線状態）は変化するが, その変化を効率よく管理すること.

イベント点：走査線状態の変化

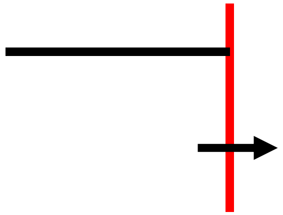
走査線状態とイベント点を管理するデータ構造が必要

イベント点

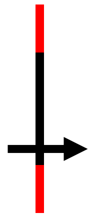
1: 水平線分の左端点 : 走査線は新たな線分に出会う



2: 水平線分の右端点 : 走査線はその線分ともはや交差しない



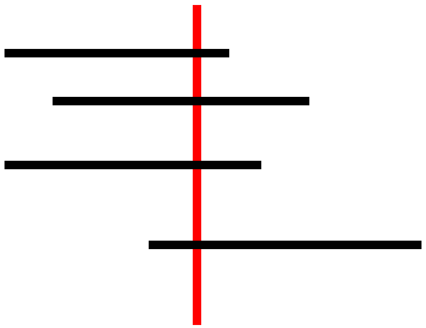
3: 垂直線分と重なるとき :



上記の3種類のイベントはx座標の昇順に生起する:
イベント点スケジュール　または　x-リスト

走査線状態のためのデータ構造: y-list

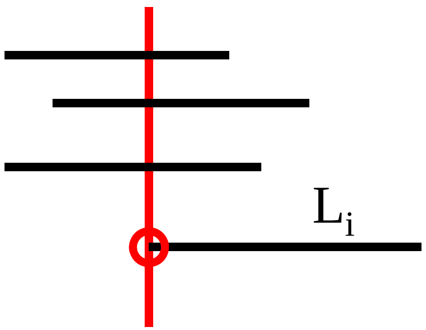
y-list: 走査線と交差する水平線分の集合をy座標の順に管理
➔ 平衡2分探索木として実現 ⇩



O(log n)-時間で可能な操作
1. 新たな水平線分の挿入
2. 水平線分の削除
3. 質問として与えられた垂直線分と交差する水平線分の列挙（＋交点数に比例する時間）

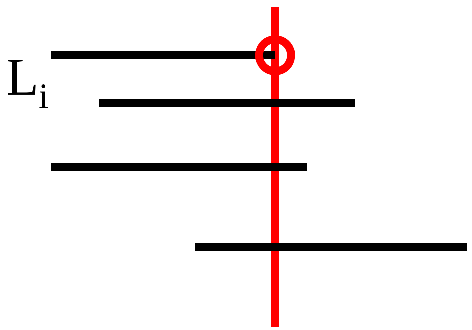
各イベント点での操作

(1)左端点



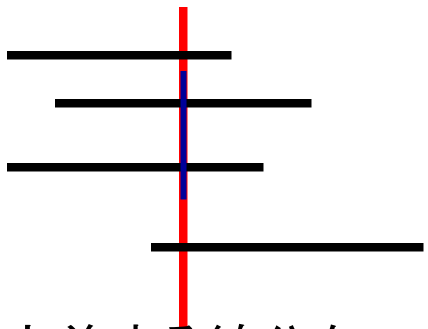
L_i をy-listに挿入

(2)右端点

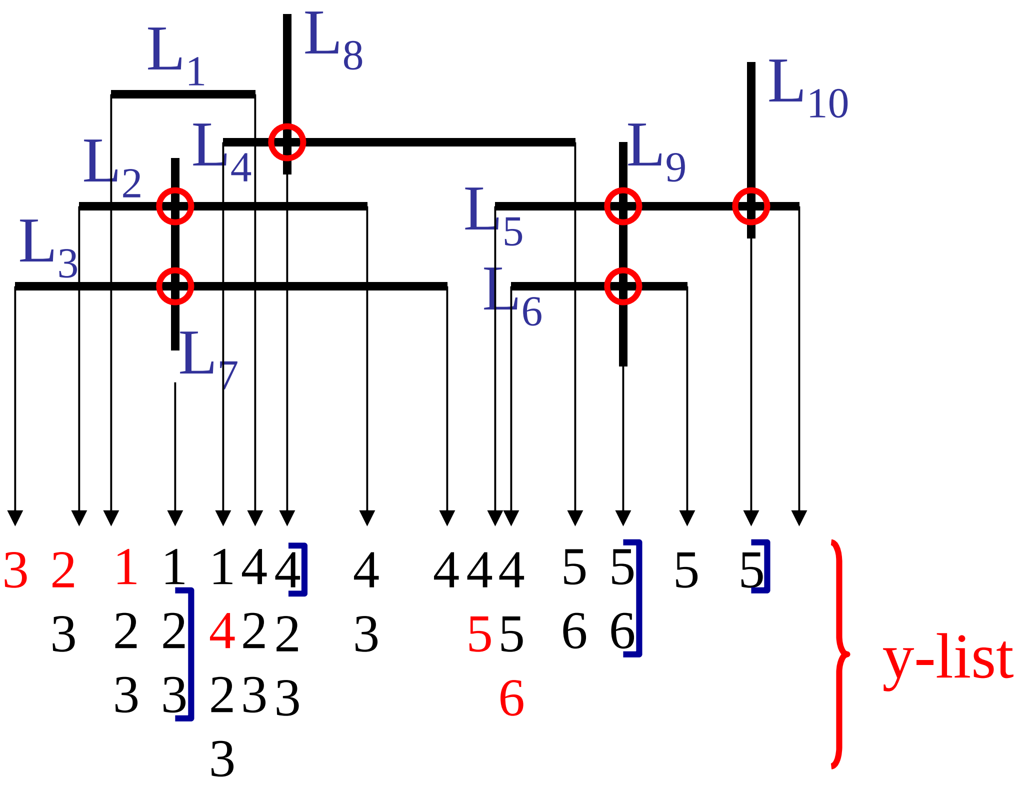


L_i をy-listから削除

(3)垂直線分

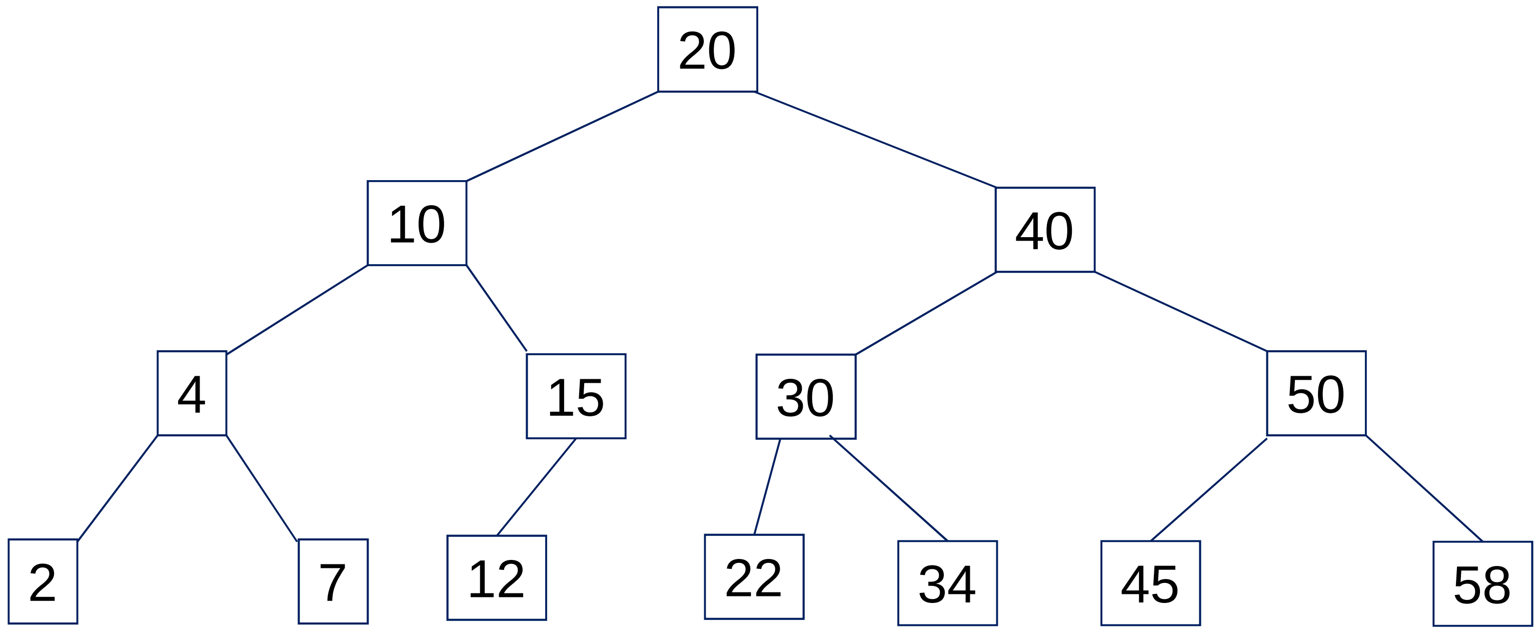


交差する線分を列挙



定理: 平面上にn本の線分が与えられたとき，平面走査法に基づく上記のアルゴリズムにより，O(K + n log n) の時間ですべての交点を列挙することができる．ただし，K は交点数である.

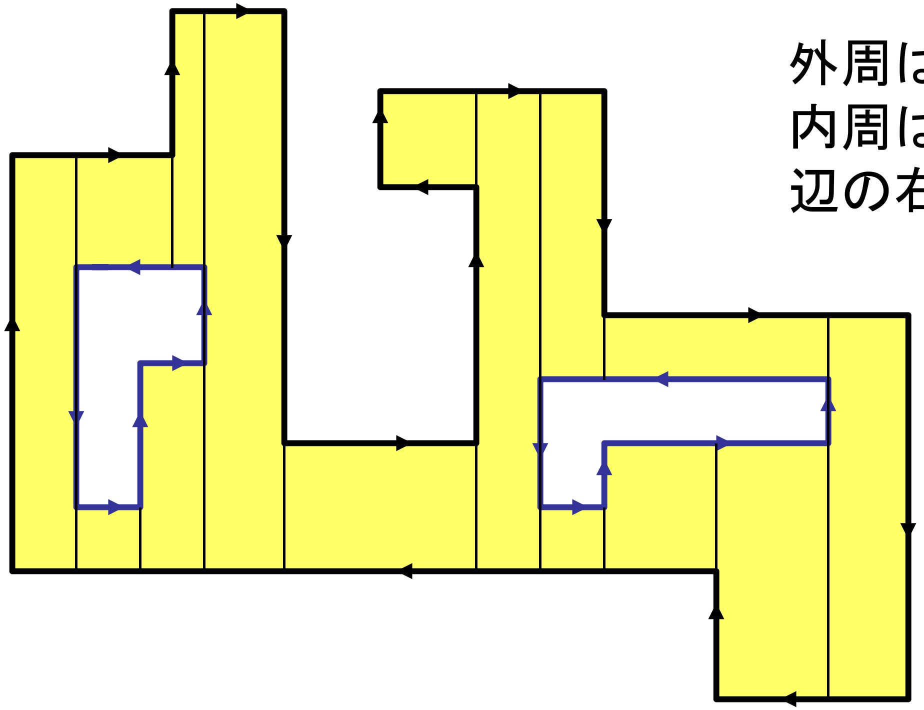
演習：yリストを平衡2分探索木で実現したとき，ある範囲の値を列挙するにはどうすればよいか？たとえば，下の木で[11, 42]の範囲にある数．



レポート課題3：　yリストを単純な配列で実現したとき，１回の操作に必要な時間はどうなるか？また，全体の計算時間はどうなるか？

平面走査法

問題2：　水平線と垂直線からなる多角形領域の内部を長方形に分割せよ．ただし，分割のために引く弦の端点の少なくとも一方は必ず元の多角形領域の頂点でなければならない．



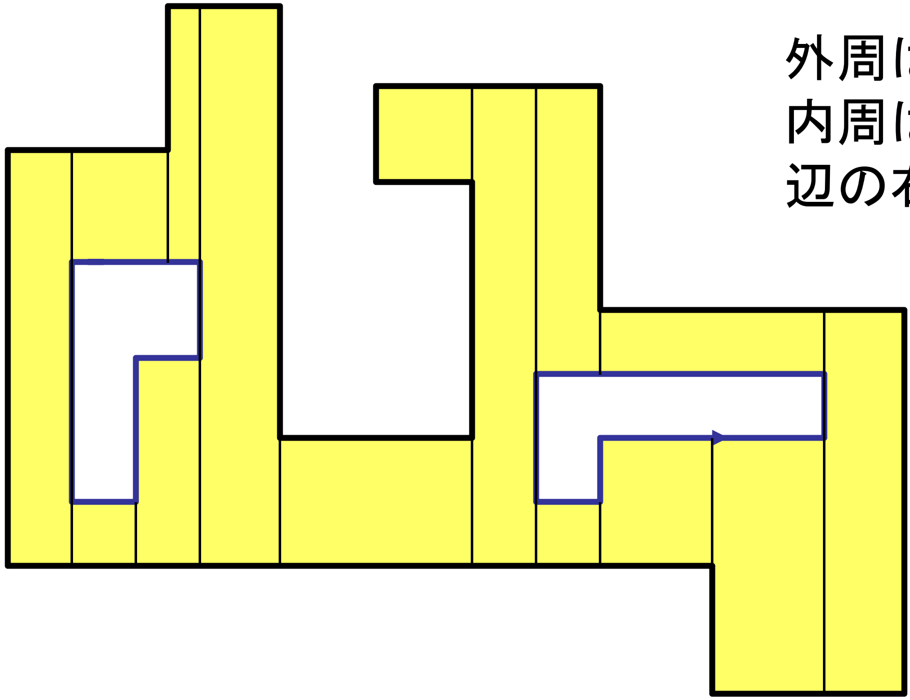
外周は時計回り，
内周は反時計回り
辺の右が領域の内部

各凸頂点で領域の内部に向けて垂直線を引けばよい．

y-リストは水平辺のリスト．

平面走査法

問題2：　水平線と垂直線からなる多角形領域の内部を長方形に分割せよ．ただし，分割のために引く弦の端点の少なくとも一方は必ず元の多角形領域の頂点でなければならない．



外周は時計回り，
内周は反時計回り
辺の右が領域の内部

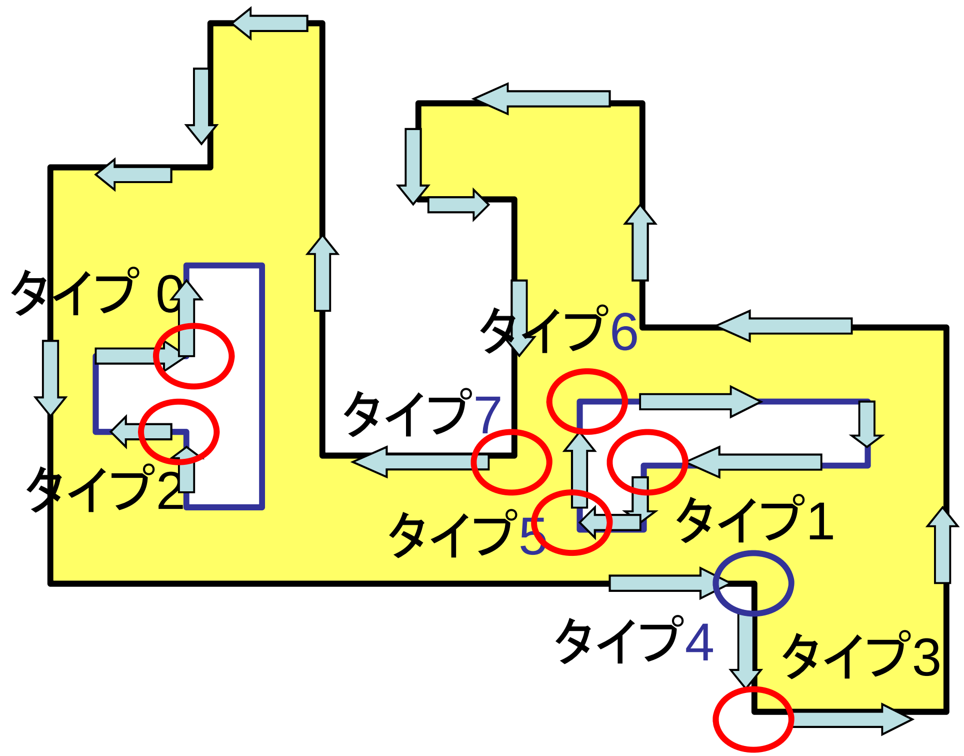
各凸頂点で領域の内部に
向けて垂直線を引けばよい．

y-リストは水平辺のリスト．

イベント点の分類

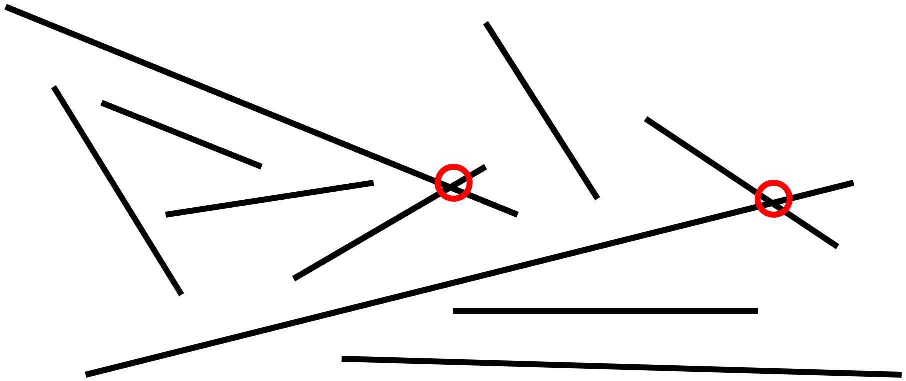
頂点のタイプ分け（辺の左が領域内部）

タイプ 0: 右の次に上:
弦は引かない
タイプ1: 左の次に下:
弦は引かない
タイプ2: 上の次に左:
弦は引かない
タイプ3: 下の次に右:
弦は引かない
タイプ4: 右の次に下:
上方向に弦を引く
タイプ5: 左の次に上:
下方向に弦を引く
タイプ6: 上の次に右:
上方向に弦を引く
タイプ7: 下の次に左:
下方向に弦を引く



線分交差判定問題

問題: 平面上にn本の線分が与えられたとき，それらの中に互いに交差するものがあるかどうかを判定せよ.



仮定:
垂直線は含まれない.

基本的な観察:
2本の線分が互いに交差するなら，必ずある垂直線の上でそれらの線分は隣接している.

➔ 垂直方向に隣接するすべての線分対を調べるアルゴリズム

イベント

1. 線分sの左端点

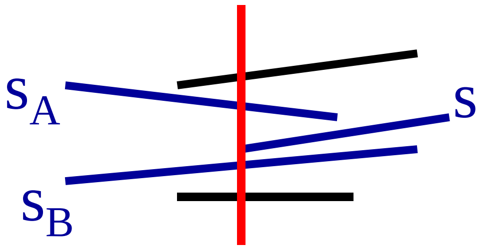
   → s をy-リストに挿入

1. 線分sの右端点

   → s をy-リストから削除

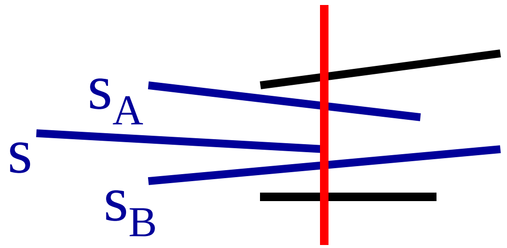
交差判定

1. 左端点



s はs_Aと交差するか？
s はs_Bと交差するか？

2. 右端点



s_Aはs_Bと交差するか？

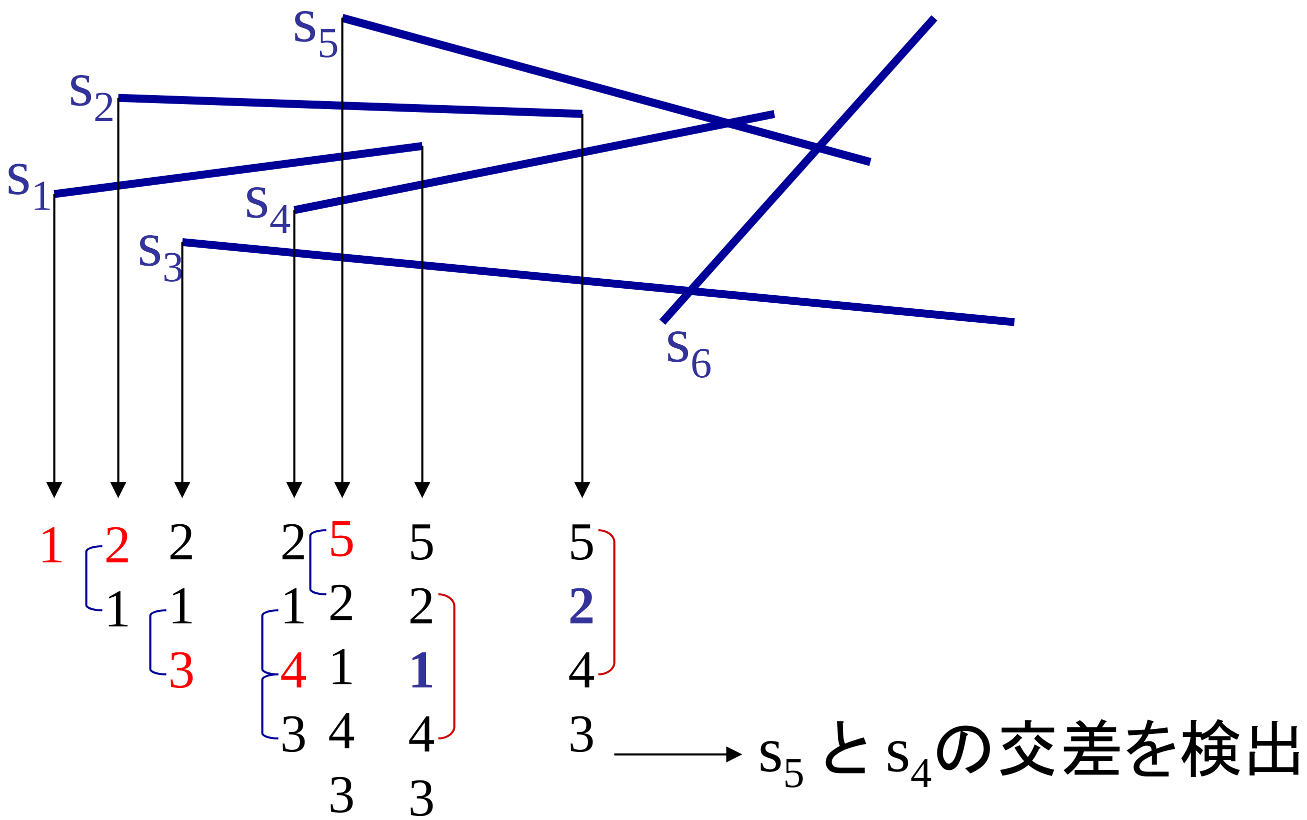
$\mathbf{s_A}$: y-リストでsのすぐ上の線分
$\mathbf{s_B}$: y-リストでsのすぐ下の線分

線分交差判定アルゴリズム (Shamos and Hoey 1976)

線分の端点をx座標の昇順にソートせよ.

→ **x-list**;

- **y-list** = 空;
- for i=1 to 2n do{
- $\boldsymbol{p_i}$ = xリストにおける i 番目の端点;
- $\boldsymbol{s}$ = $\boldsymbol{p_i}$ を端点としてもつ線分;
- if $\boldsymbol{p_i}$ は左端点 then {
- INSERT($\boldsymbol{s}$, y-list);
- $\boldsymbol{s_A}$=above($\boldsymbol{s}$, y-list); $\boldsymbol{s_B}$=below($\boldsymbol{s}$, y-list);
- if $\boldsymbol{s}$ **が** $\boldsymbol{s_A}$ **または** $\boldsymbol{s_B}$ **と交差する** then return YES;
- }else{ //右端点に対する処理
- $\boldsymbol{s_A}$=above($\boldsymbol{s}$, y-list); $\boldsymbol{s_B}$=below($\boldsymbol{s}$, y-list);
- DELETE($\boldsymbol{s}$, y-list);
- if $\boldsymbol{s_A}$ は $\boldsymbol{s_B}$ と交差する then return YES;
- }
- }
- return NO;



定理: 平面上にn本の線分が与えられたとき，もしそれらの間に交差があるなら，平面走査法はO(n log n)時間で交差があると報告する.

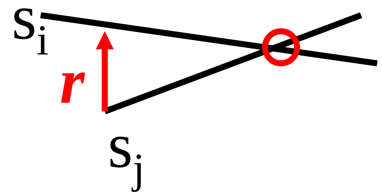
アルゴリズムの正しさ

アルゴリズムが“YES”を出力して停止する場合
→ その答えは正しい.

アルゴリズムが“NO”を出力して停止する場合,
交差がないことを証明せよ.

背理法による証明:
交差があるのに検出できなかったと仮定して矛盾を導く.

c: 最も左の交点
(s_i と s_j が c で交差し, $x_L(s_j) > x_L(s_i)$と仮定)

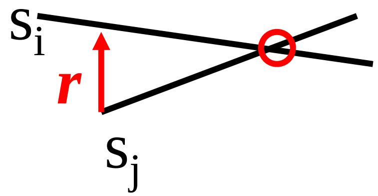


場合 1: 半直線 r はs_iより前にどの線分とも交差しない.

この場合, この交点はs_jの左端点で検出されるはずなので
矛盾である.

c: 最も左の交点

(s_i と s_j が c で交差し, $x_L(s_j) > x_L(s_i)$と仮定)

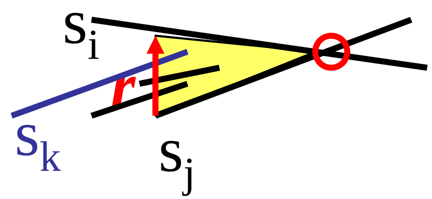


場合 2:半直線 r はs_iより前にある線分と交差する.

s_k : 半直線rと交差する任意の線分.

➔ s_k は s_i とも s_jとも交差しない

(なぜなら, cは最も左の交点だから)



したがって, s_k の右端点は図の黄色の領域になければならない.

$s_{k'}$ = 黄領域の線分の右端点の中で最も右のものをもつ線分

このとき, $s_{k'}$の右端点では

above($s_{k'}$) → s_i

below($s_{k'}$) → s_j

となっているので, s_i と s_j は交差する: 矛盾 !

線分交差列挙アルゴリズム

交差列挙問題: 平面上にn本の線分が与えられたとき，
それらの間の交点をすべて列挙せよ.

Bentley-Ottmanのアルゴリズム(1979)

イベント

1. 左端点

新たな線分をy-リストに挿入.
交差を判定し，あればそれをx-リストに挿入する.

2. 右端点

y-リストから線分を削除する.
交差を判定し，現在の場所より右に交差があれば
x-リストに挿入する.

3. 線分の交点

交点を左右の端点だと見なして，2本の線分のyに
関する順序を交換する.

定理: Bentley-Ottmanのアルゴリズムは，n本の線分の集合に対する合計K個の交点をO((n+K) log n)時間で列挙する.

イベントの数 = $2n + K$

($2n$ 個の端点とK 個の交点)

各イベントはO(log n)時間で処理できる

x-list: 優先順位つきキュー

y-list: 平衡2分探索木

注意!

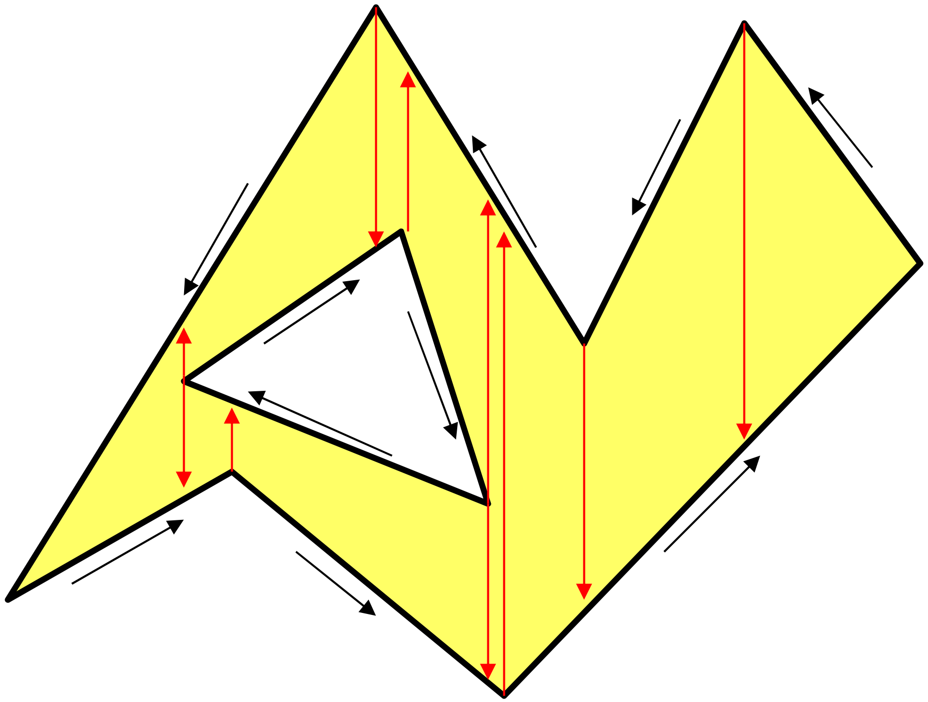
腕力法の計算時間は$O(n^2)$である.

➔もし交点数Kが$O(n^2)$ なら,

最悪の場合，Bentley-Ottmanのアルゴリズムは腕力法より悪い

台形分割

問題: 平面上に多角形領域が与えられたとき，その内部領域を台形に分割せよ.

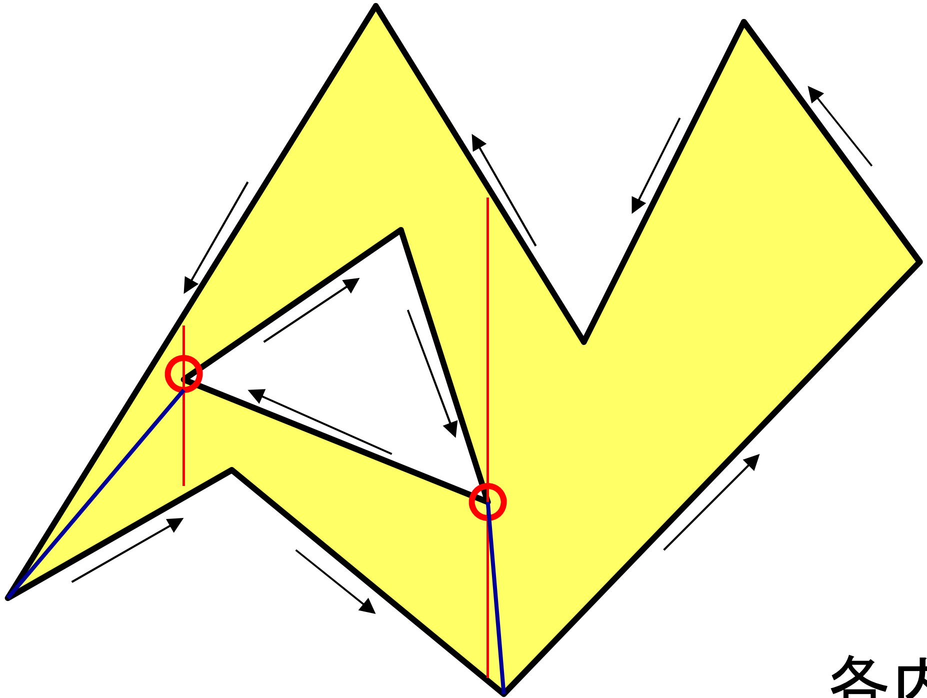


y-リスト: y座標の順に
　並べられた辺のリスト

各頂点において
　領域の内部に向けて
　半直線を延ばす.
　最初の交点で止める

三角形分割

問題: 平面上に多角形領域が与えられたとき，その内部領域を三角形に分割せよ.



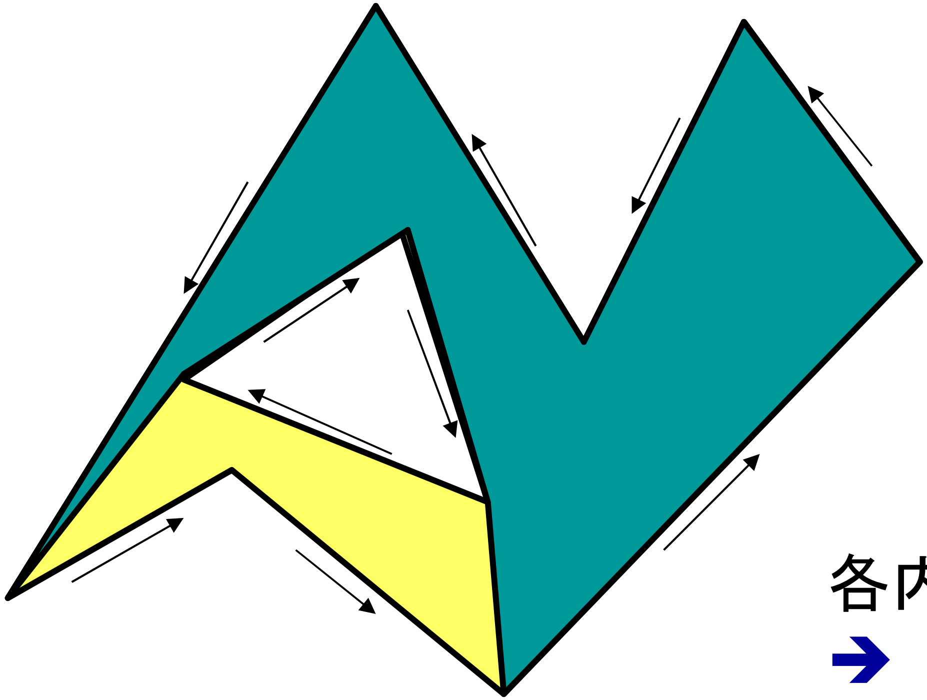
内尖点(ないせんてん)
その内角が垂直方向を含んでいるような頂点

練習問題: 内尖点を正確に定義せよ.

各内尖点から弦を引く
➔ **単調な多角形の集合**

三角形分割

問題: 平面上に多角形領域が与えられたとき，その内部領域を三角形に分割せよ.



内尖点

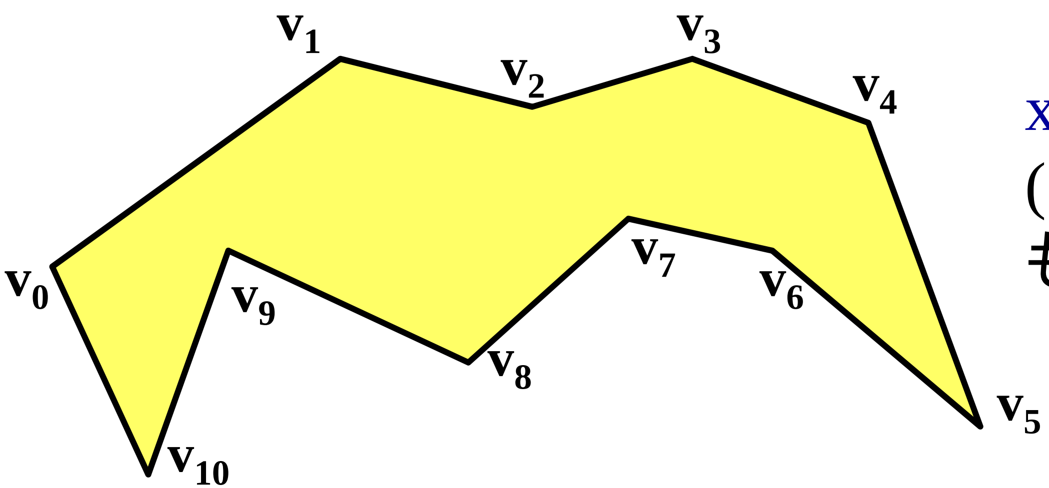
その内角が垂直方向を含んでいるような頂点

各内尖点から弦を引く

➔ **単調な多角形の集合**

レポート課題4: 内尖点を取り除くために引く弦を特徴づけよ.
また，そのような弦の終点をどのように選べばよいか.

単調な多角形の三角形分割



x-単調な多角形
(どこで垂直な直線を引いても高々2回しか交差しない)

1:頂点をx座標の昇順にソート:

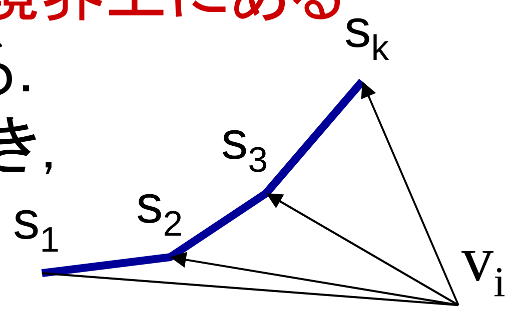
(v_0, v_{10}, v_9, v_1, v_8, v_2, v_7, v_3, v_6, v_4, v_5)

2: 最初の2頂点をスタックに入れる.

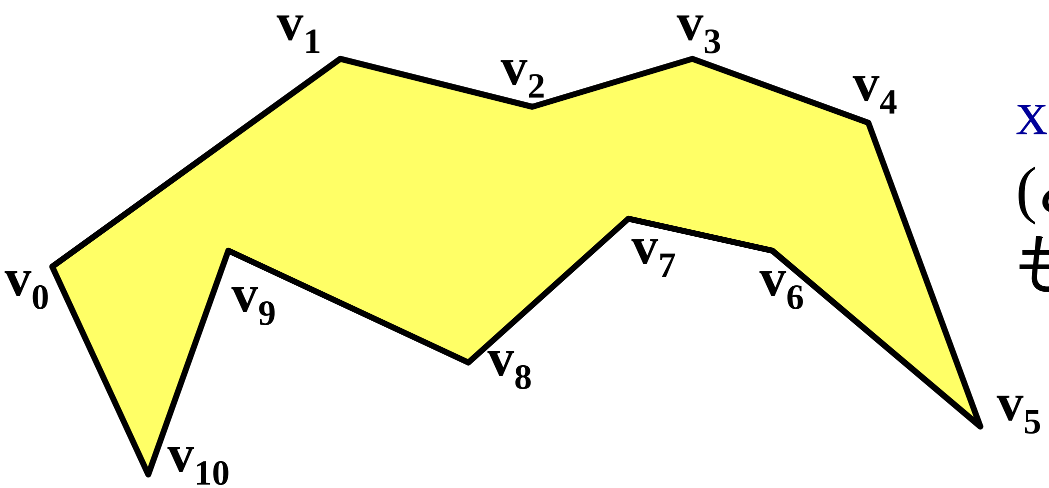
3: for i=2 to n-1 do{ // スタック: (s_1, s_2, s_3, ... , s_k)

ケース1: v_i はスタック内の頂点と反対側の境界上にある

スタック内の頂点は全てv_iから見える.
よって, v_i から s_2, s_3, ... , s_k に弦を引き,
最後にv_i をスタックに入れる.



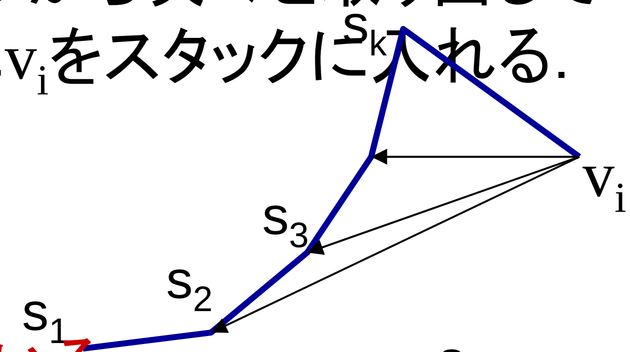
単調な多角形の三角形分割



x-単調な多角形
(どこで垂直な直線を引いても高々2回しか交差しない)

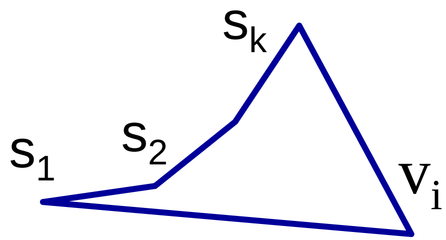
ケース 2: v_i はスタック内の頂点と同じ側の境界上にある

頂点v_iから見える頂点をスタックから次々と取り出してs_{k-1}, s_{k-2}, ... に弦を引き，最後にv_iをスタックに入れる.

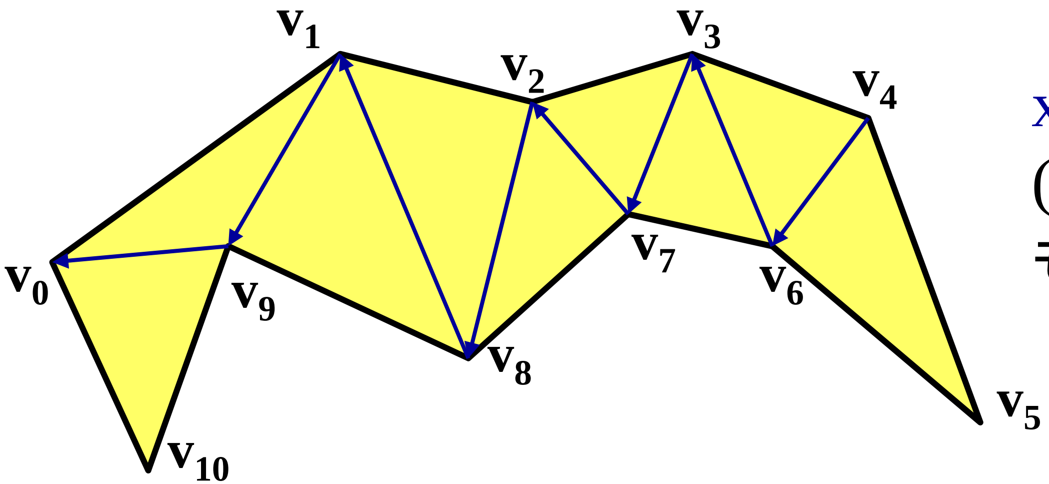


ケース 3: v_i は s_1 と s_k の両方に接続している

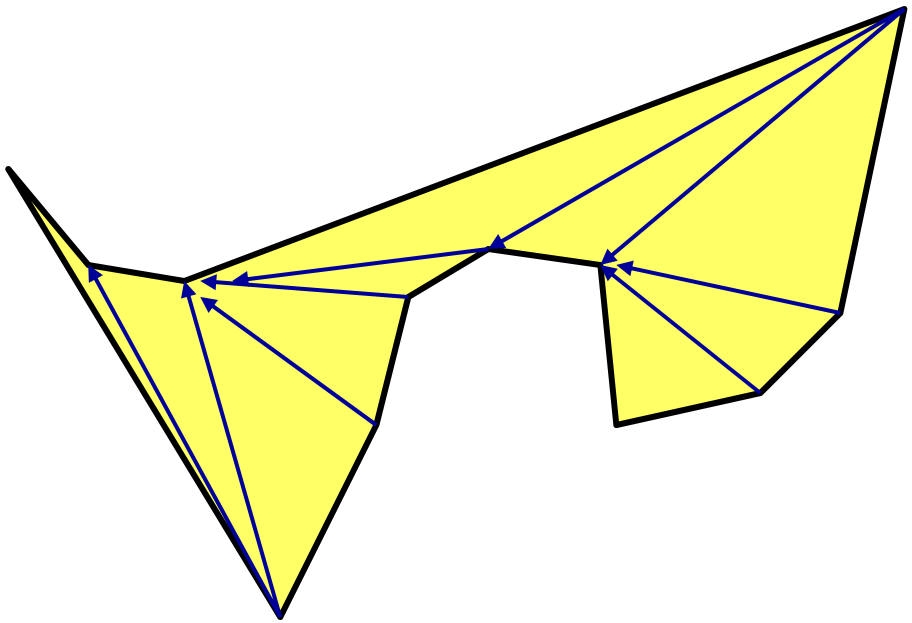
v_iからs_2 , s_3, ... , s_{k-1}に弦を引く.



単調な多角形の三角形分割



x-単調な多角形
(どこで垂直な直線を引いても高々2回しか交差しない)



練習問題: アルゴリズムの実行中にスタックが少なくとも4頂点以上を含むことがあるような多角形の例を示せ.