生産情報システム工学

#11 中間試験の解説

2015/07/01(水)

溝口 知広　准教授(居室：61-408室)
mizo@cs.ce.nihon-u.ac.jp

問1：ヒープ

下図に示すように配列に格納された初期データに対し，以下の手順でデータの挿入と削除を繰り返した後のヒープの状態を配列で示しなさい．ここで，Ins(x)はヒープにデータxを挿入することを，Del()は最小値を削除することを意味する．

1. I(18) → D() → D() →I(27) → D()
2. D() → I(25) → D() → I(5) → I(3) → I(2) → D()
3. I(22) → I(15) → D() → D()→ I(3) → D() → I(14) → I(9) → D()

0	1	2	3	4	5	6	7
	6	10	23	17	12		

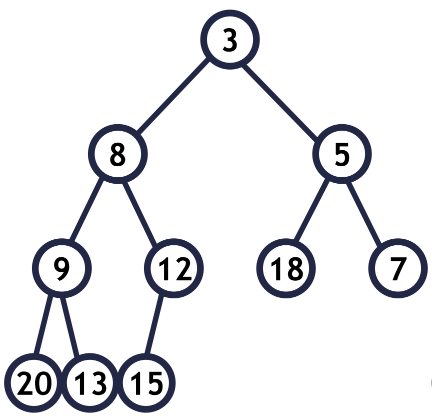
問1：ヒープ

1	2	3	4	5	6	7	8	9	10			
3	8	5	9	12	18	7	20	13	15			

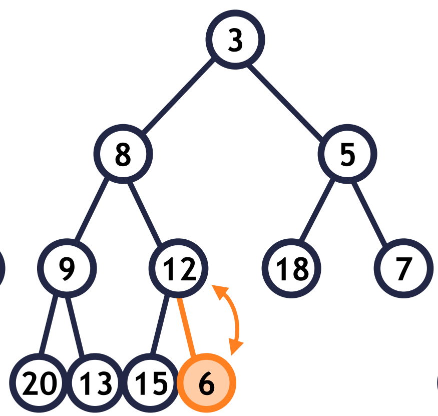
1	2	3	4	5	6	7	8	9	10	11		
3	8	5	9	12	18	7	20	13	15	6		

- **挿入：insert(6)**
  1. 新たに挿入する要素を配列の末尾に入れる
  2. 挿入要素とその親(12)で，ヒープ条件が成立するか調べる
  3. もし成立しなければ，親と子を入れ替える
  4. 終了条件：①ヒープ条件が成立，②挿入要素が根になる

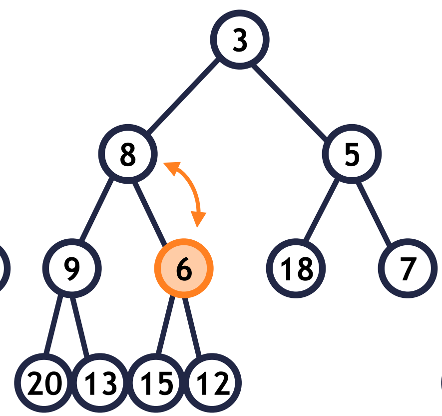
繰返し



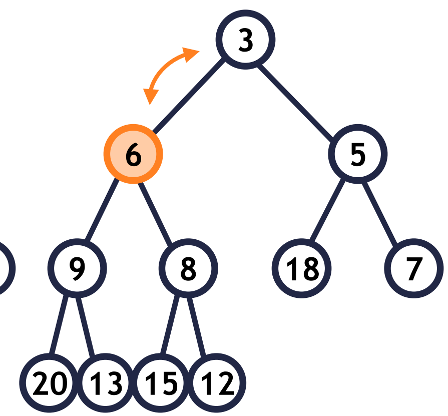
初期状態



6を入れる
12と6で条件を
調べる



12と6を入れ替える
8と6で条件を
調べる



8と6を入れ替える
3と6で条件が成立
終了

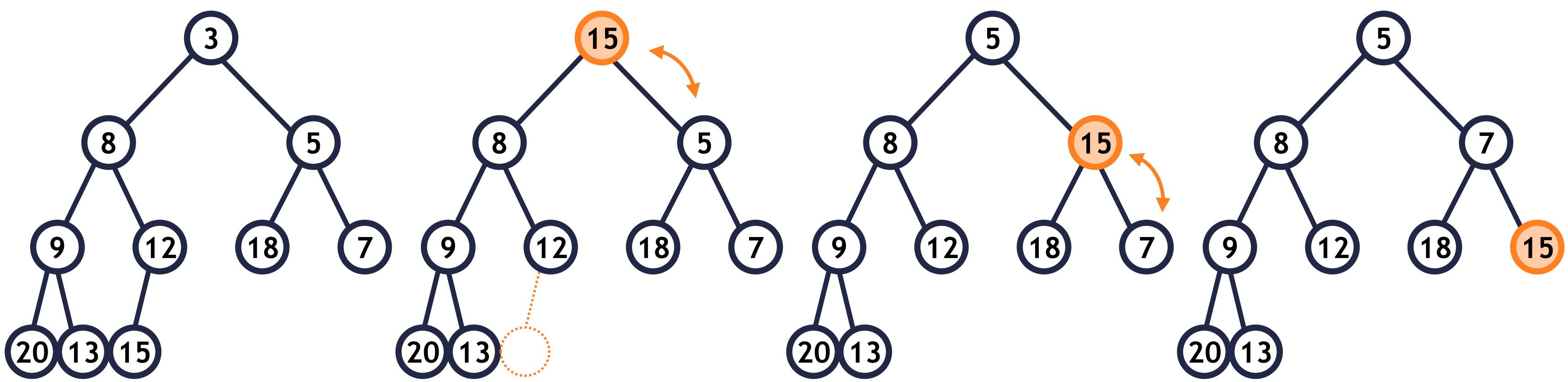
問1：ヒープ

1	2	3	4	5	6	7	8	9	10			
3	8	5	9	12	18	7	20	13	15			

1	2	3	4	5	6	7	8	9				
15	8	5	9	12	18	7	20	13				

- 削除：deletemin()
  1. 末尾の要素を先頭に書き込む(最小要素の削除)
  2. 書き込んだ要素とその子でヒープ条件が成立するか調べる
  3. もし成立しなければ，左右の子の小さい方と入れ替える
  4. 終了条件：①ヒープ条件成立，②書き込んだ要素が葉になる

繰返し



初期状態

15を根に入れる
15と5で条件を
調べる

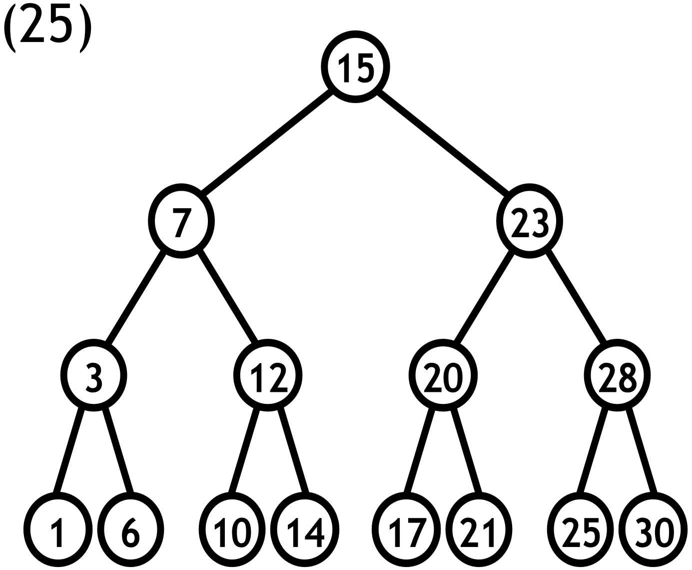
15と5を入れ替える
15と7で条件を
調べる

15と7を入れ替える
15が葉なので条件が
成立，終了

問2：二分探索木

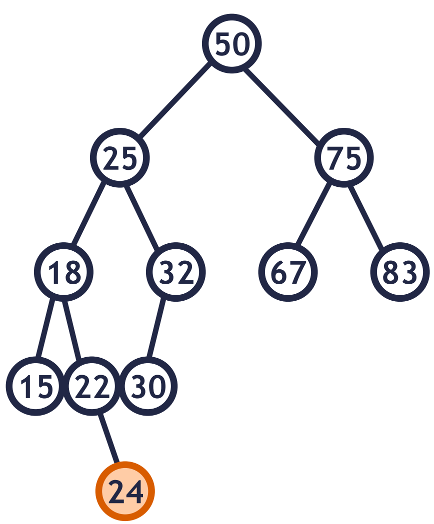
下図に示す初期データに対し，以下の手順でデータの挿入と削除を繰り返した後の2分探索木の状態を図示しなさい．ここで，Ins(x)は木にデータxを挿入することを，Del(x)は木からデータxを削除することを意味する．

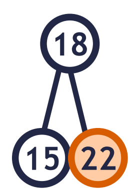
1. D(12) → I(4) → D(23) → I(16) → D(28)
2. I(13) → D(20) → D(25) → I(29) → I(8) → D(7) → I(11)
3. I(32) → D(25) → D(23) → D(15) → I(25) → I(5) → D(20) → I(22) → I(16)



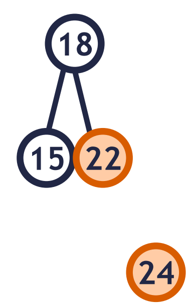
問2：二分探索木

- 挿入の例 (24を新たに追加する場合)
  1. 探索の場合と同様に，根から比較と移動を繰り返す
  2. 最後に訪れたノードにxを新たな子として追加する
     1. 新たな子のためのメモリを割り当てる
     2. データを追加する
     3. 親子関係を更新する

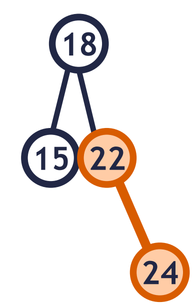




1)葉ノードに到達



2-1)メモリ割り当て
2-2)データの追加



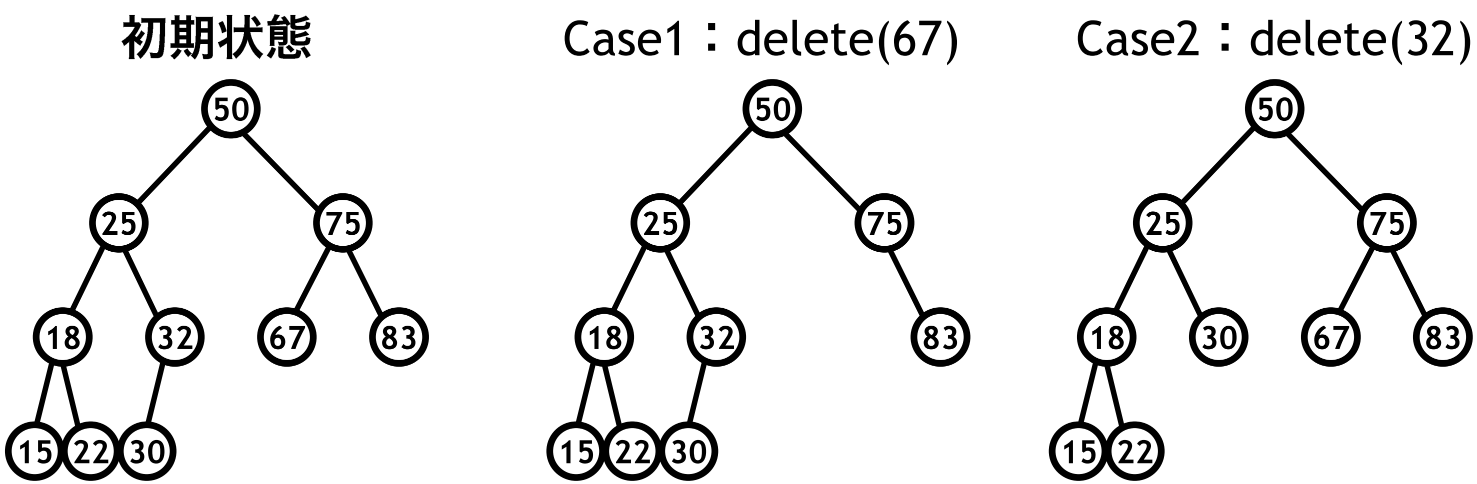
2-3)親子関係の更新

問2：二分探索木

- 削除の例
  1. 探索の場合と同様に，削除するノードへ移動する
  2. ノードを削除する

     Case1：削除するノードが葉の場合 → 葉を削除する

     Case2：葉ではなく，1つの子を持つ場合 → 子で置き換える



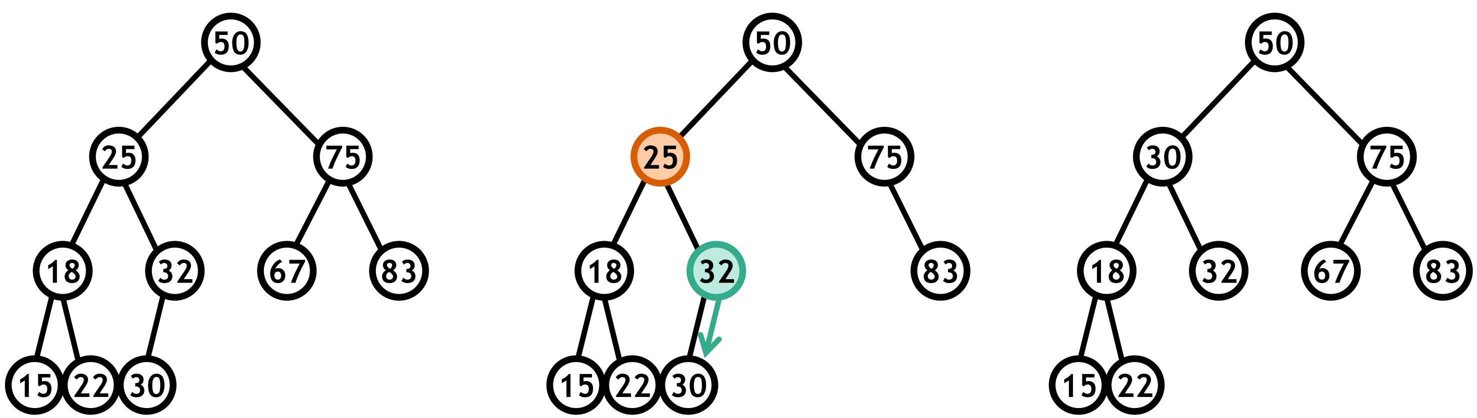
1.4.4 2分探索木

- 削除の例
  1. 探索の場合と同様に，削除するノードへ移動する
  2. ノードを削除する

     Case3：葉ではなく，2つの子を持つ場合

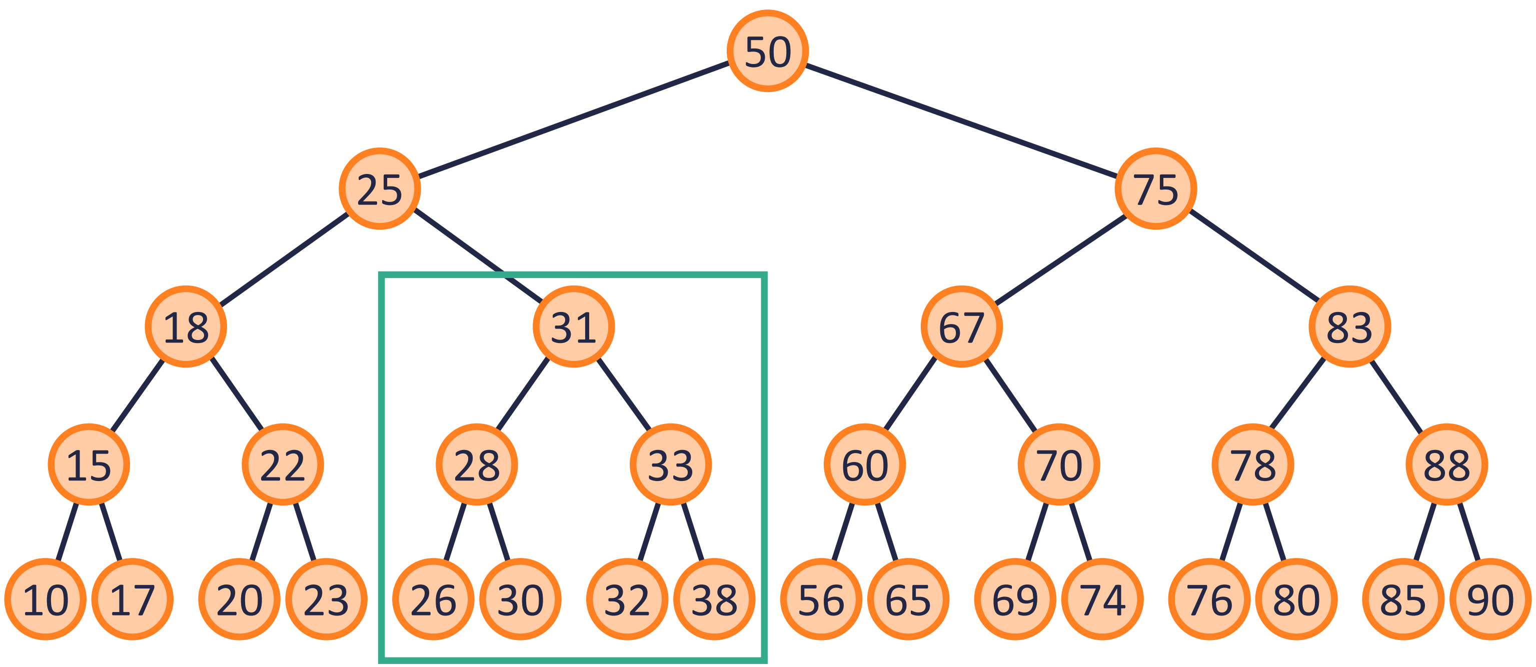
     1. 削除ノードの右の子(32)を出発点とし，左の子を繰り返し辿る
        <u>(削除ノードの次に大きな要素を見つける)</u>
        <u>(＊＊次に小さな要素でも良い)</u>
     2. 到達したノードの要素(30)を削除ノードに上書き

初期状態　　　Case3：delete(25)



問2：二分探索木

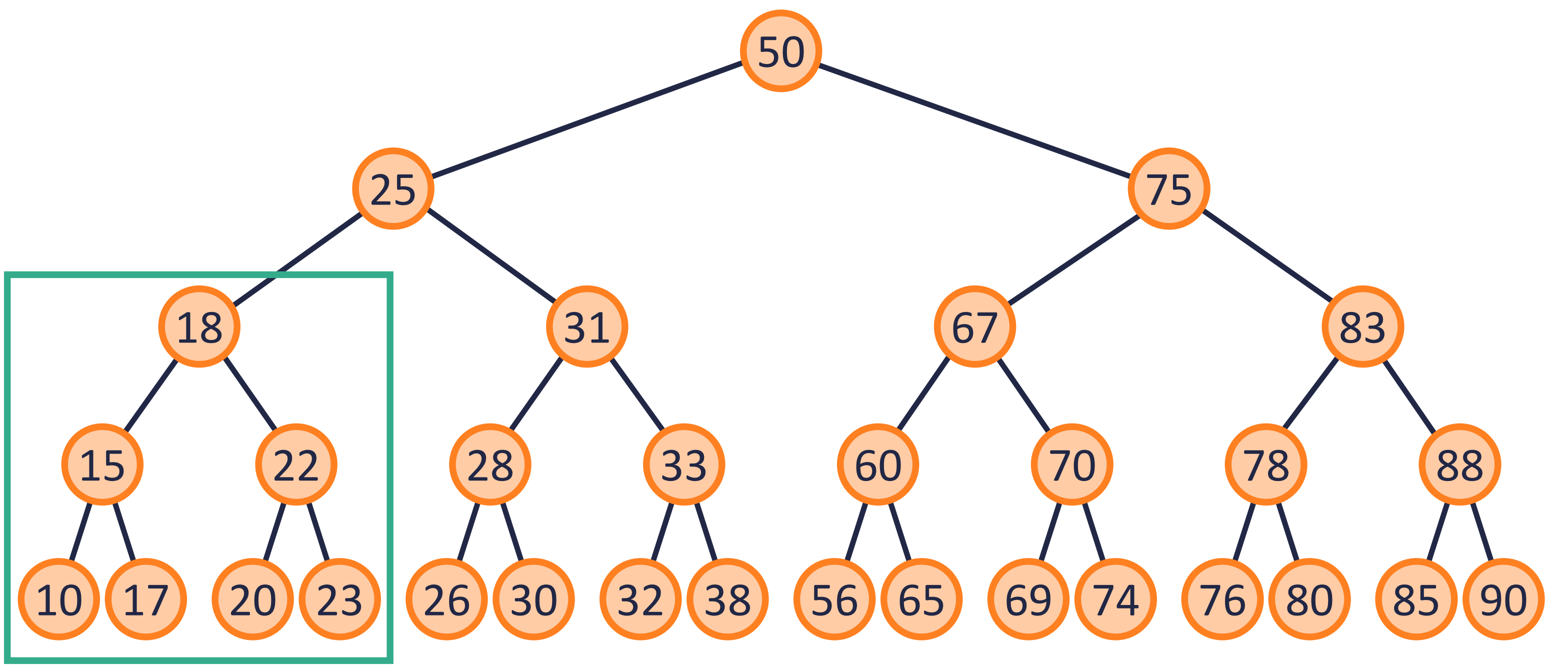
- 削除の例
  - あるノードの要素の次に大きな要素は，右部分木の左を繰り返し辿った先のノードの要素
  - 例：25の次に大きな値は，その右部分木の左端の26



問2：二分探索木

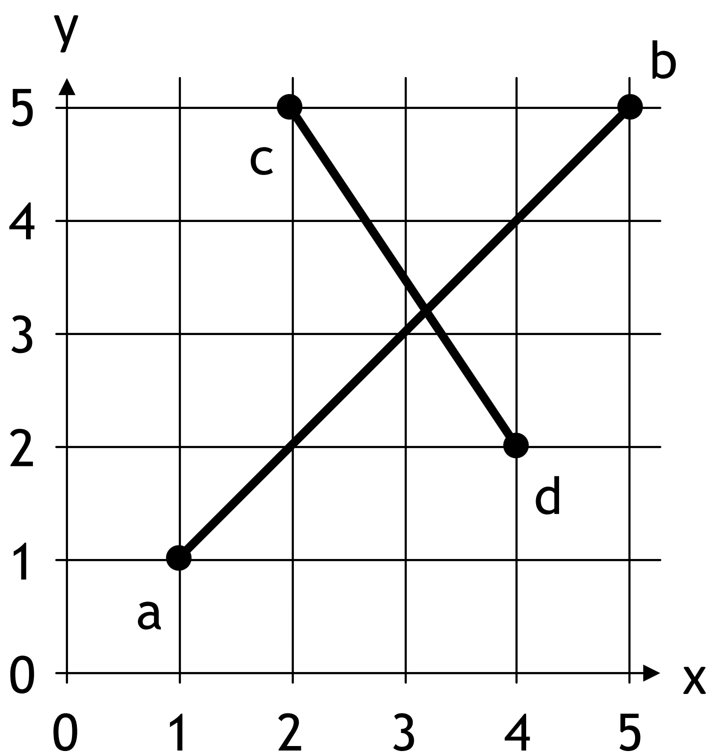
■ 削除の例

- あるノードの要素の次に小さな要素は，左部分木の右を繰り返し辿った先のノードの要素
- 例：25の次に小さな値は，その左部分木の右端の23

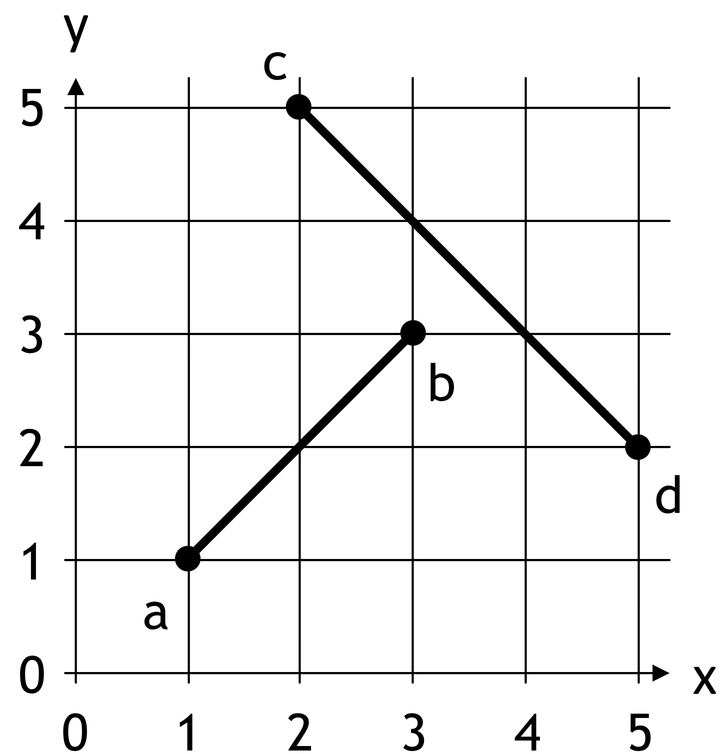


問3：2線分の交差

図a，図bに示す2線分の交差判定を行う過程を，三角形の符号付き面積を実際に計算し記述しなさい．



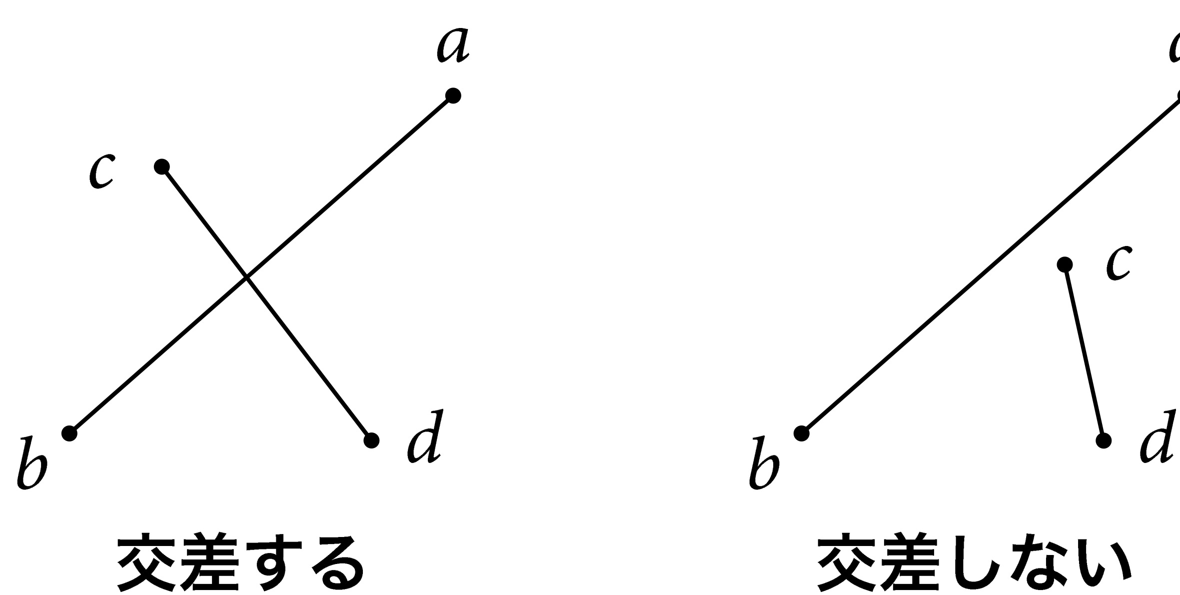
図a



図b

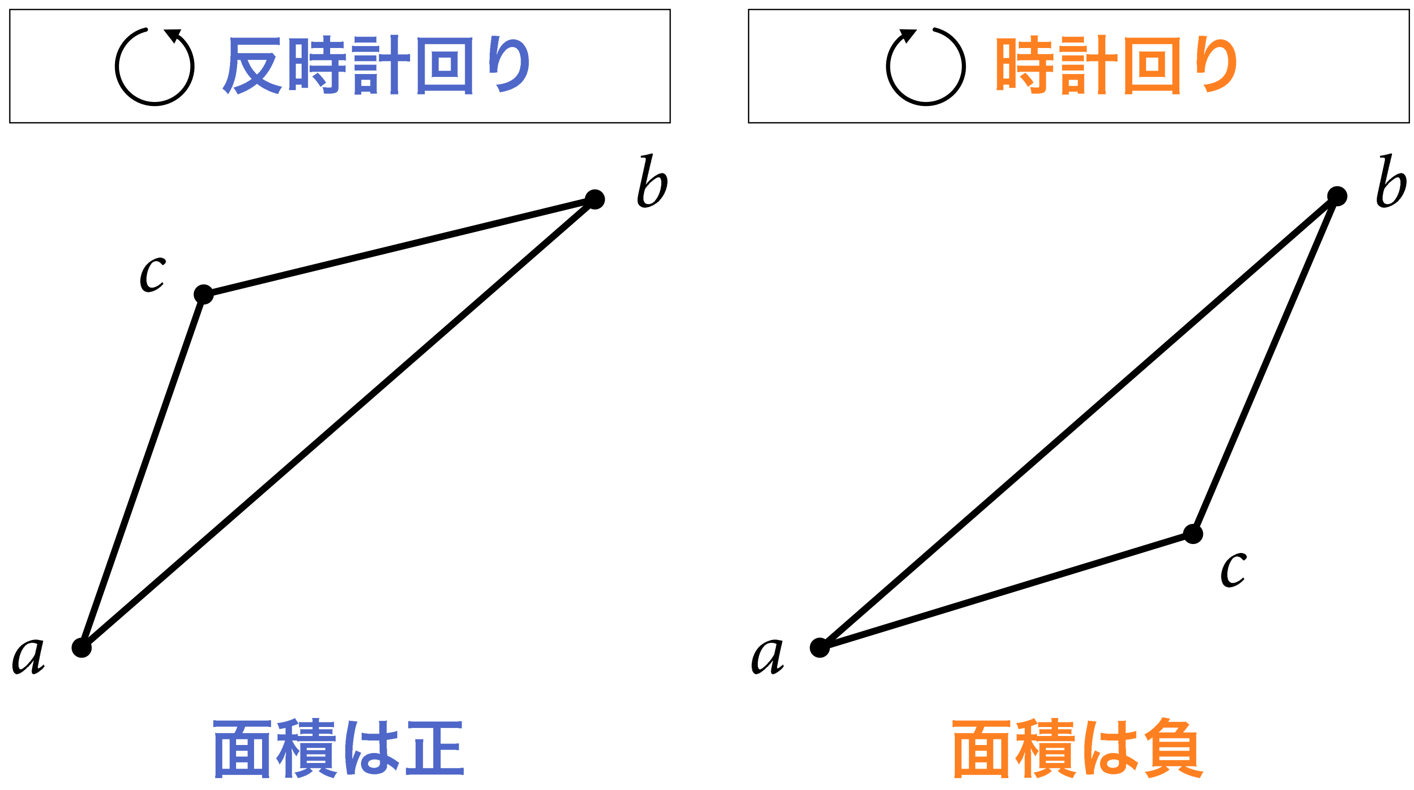
問3：2線分の交差

- 三角形の符号付き面積を利用する方法：
  - 基本的な考え方：
    - 2本の線分abとcdが互いに交わるならば，端点cと端点dが線分abを含む直線によって分離される
    - 同様に，aとbもcdを含む直線によって分離される



問3：2線分の交差

- 三角形の符号付き面積を利用する方法：
  - 面積の符号は3点(a, b, c)の順序で決まる



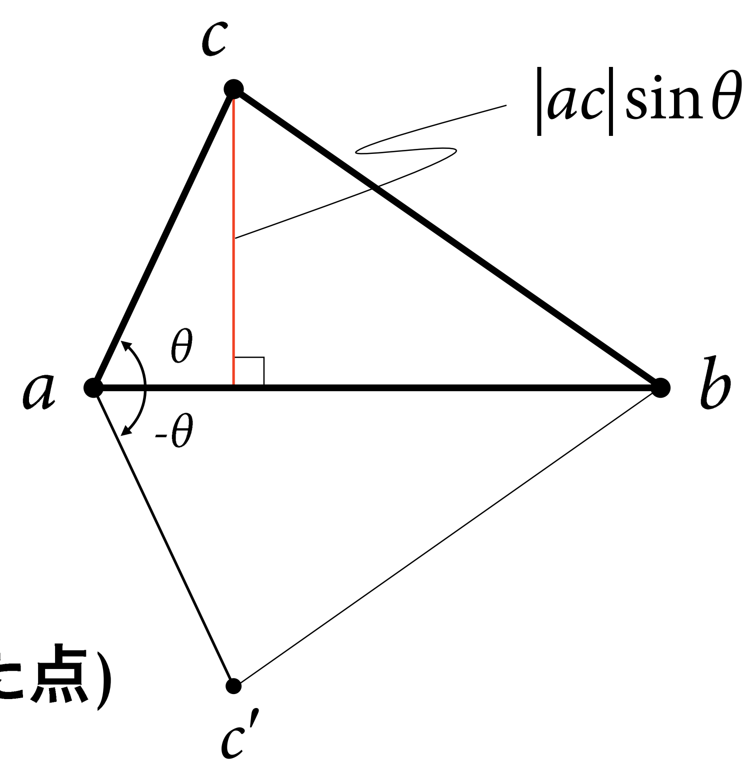
問3：2線分の交差

- 符号付き面積の算出
  - 三角形*abc*の面積

$$S_{abc} = \frac{1}{2}|ab||ac|\sin\theta$$

  - 三角形*abc*’の面積
    (*c*’は*ab*に対して反射させた点)

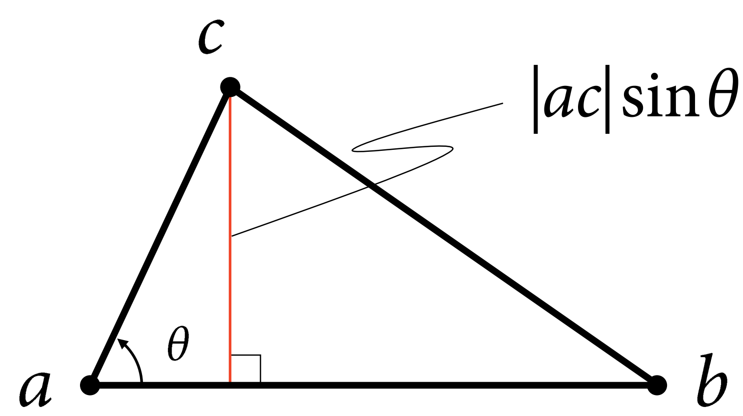
$$S_{abc'} = \frac{1}{2}|ab||ac'|\sin(-\theta)$$
$$= -\frac{1}{2}|ab||ac'|\sin\theta$$



問3：2線分の交差

■ 外積を使った符号付き面積の算出

$$S_{abc} = \frac{1}{2}|ab||ac|\sin\theta$$
$$= \frac{1}{2} ab \times ac$$
$$= \frac{1}{2}\left(\left(x_b - x_a\right)\cdot\left(y_c - y_a\right) - \left(x_c - x_a\right)\cdot\left(y_b - y_a\right)\right)$$

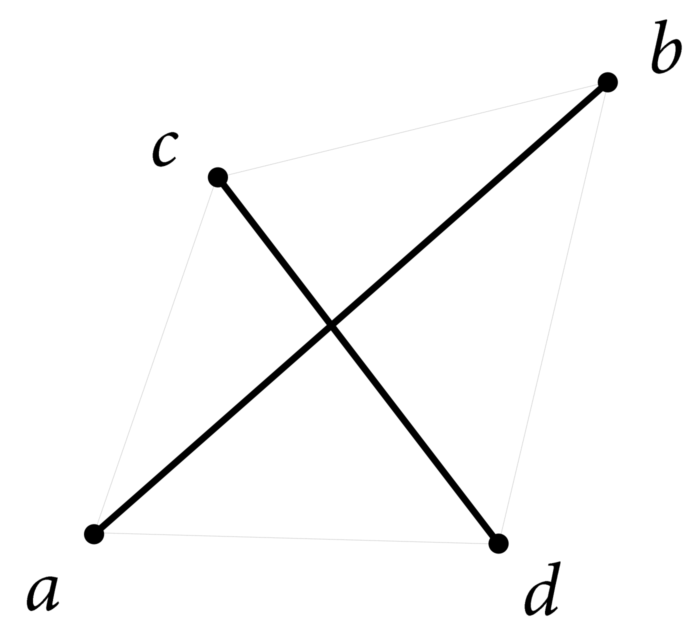


問3：2線分の交差

- 三角形の符号付き面積を利用する方法
  - △abcと△abd，△cdaと△cdbの符号付き面積がいずれも異なる符号を持てば交差する

<u>交差する場合</u>

1. 3点<u>a,b,cの順</u>は反時計回り
   → 符号付き面積は正
2. 3点<u>a,b,dの順</u>は時計回り
   → 符号付き面積は負
3. 3点<u>c,d,aの順</u>は時計回り
   → 符号付き面積は負
4. 3点<u>c,d,bの順</u>は反時計回り
   → 符号付き面積は正



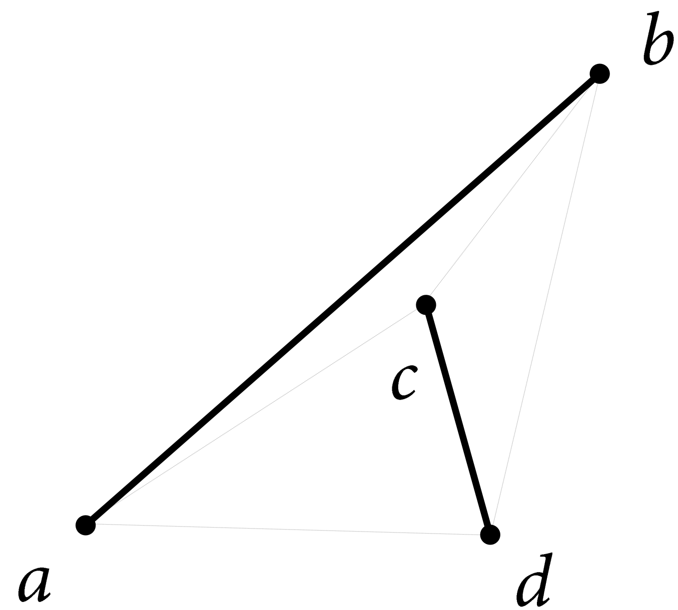
問3：2線分の交差

■ 三角形の符号付き面積を利用する方法

- △abcと△abd，△cdaと△cdbの符号付き面積がいずれも異なる符号を持てば交差する

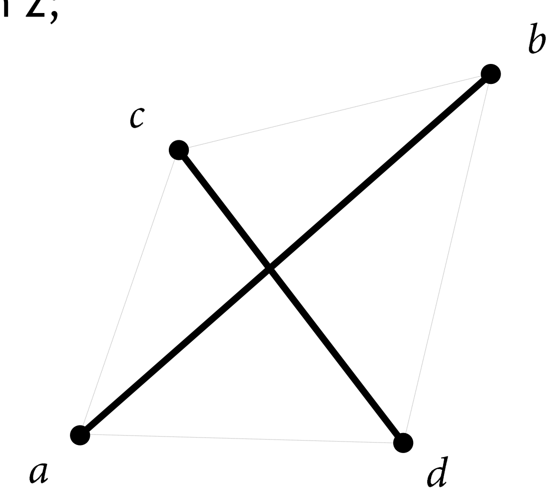
<u>交差しない場合</u>

1. 3点$\underline{a,b,c}$の順は時計回り
   → 符号付き面積は負
2. 3点$\underline{a,b,d}$の順は時計回り
   → 符号付き面積は負
3. 3点$\underline{c,d,a}$の順は時計回り
   → 符号付き面積は負
4. 3点$\underline{c,d,b}$の順は反時計回り
   → 符号付き面積は正



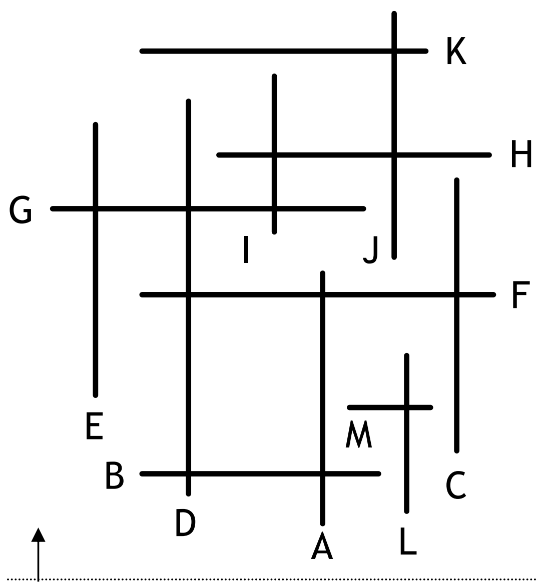
問3：2線分の交差

```
// 線分abとcdが交差するときは1を返し，交差しないときは0を返す
int intersect( struct point a, struct point b, struct point c, struct point d )
{
  // 1) 一方の線分上に，他方の線分のいずれかの端点が乗っている場合
  int  b1 = between( a, b, c );          // 線分ab上に点cがあるかどうか
  int  b2 = between( a, b, d );          // 線分ab上に点dがあるかどうか
  int  b3 = between( c, d, a );          // 線分cd上に点aがあるかどうか
  int  b4 = between( c, d, b );          // 線分cd上に点bがあるかどうか
  if( b1 == 1 || b2 == 1 || b3 == 1 || b4 == 1 )     return 2;

  // 2) 交差する場合
  double  a1 = area( a, b, c );
  double  a2 = area( a, b, d );
  double  a3 = area( c, d, a );
  double  a4 = area( c, d, b );
  if( a1*a2 < 0.0 && a3*a4 < 0.0 )       return 1;

  // 3) 交差しない場合
  else      return 0;
}
```

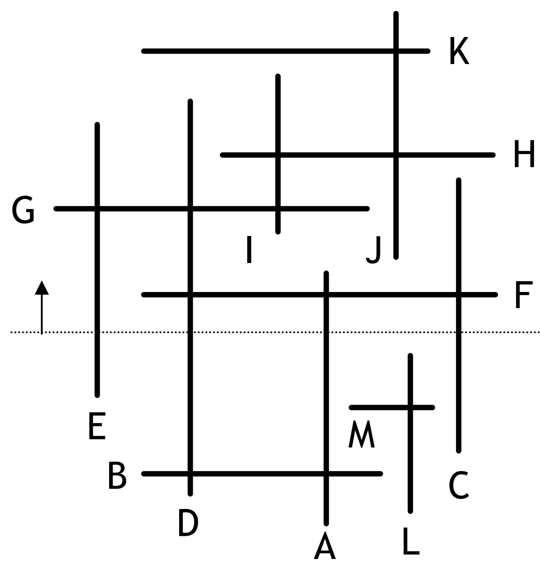


問4：直交する線分の交差(1)

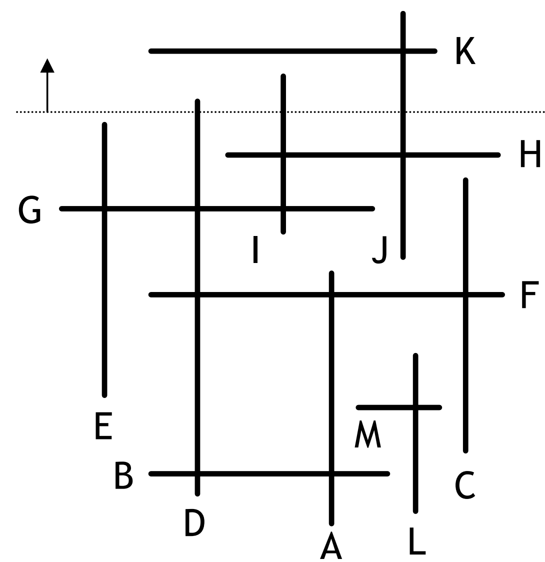
以下の図aに示す複数本の垂直・水平線分の交差を平面走査法で求める．走査線は初期状態では図aに示す位置にあり，その後徐々に+y方向へ移動する．走査線が図b，図cの位置に到達した際の2分探索木の状態を，教科書や講義資料のように記号で示しなさい．



図a



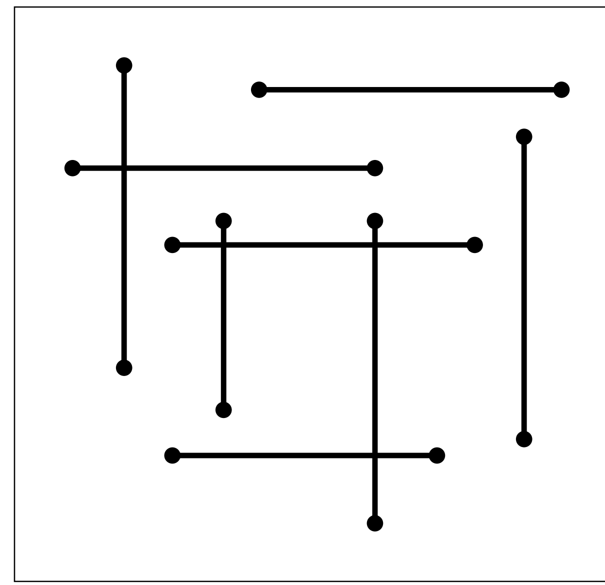
図b



図c

問4：直交する線分の交差(1)

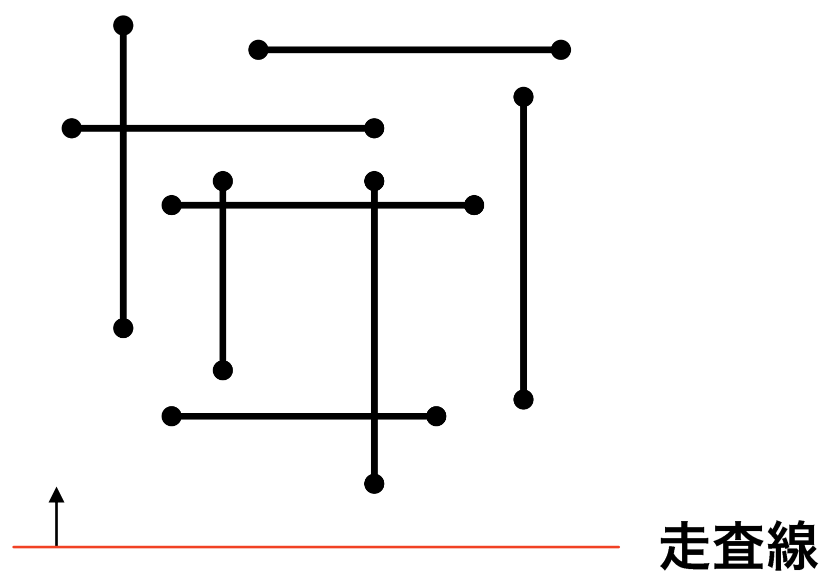
- 問題：水平・垂直なn本の線分の中に，交差はいくつあるか？



- すぐに思いつく簡単な方法
  - n本の線分から2本ずつ選び，それらの交差を調べる
  - 計算量：$O(n^2)$
  - nが大きい場合には使えない
    - 参考：バブルソート $O(n^2)$ vs クイックソート $O(n\log n)$
  - もっと効率的な方法は？

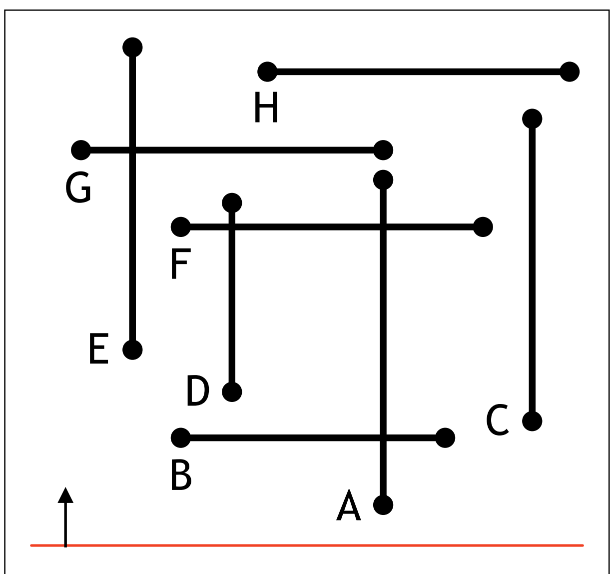
問4：直交する線分の交差(1)

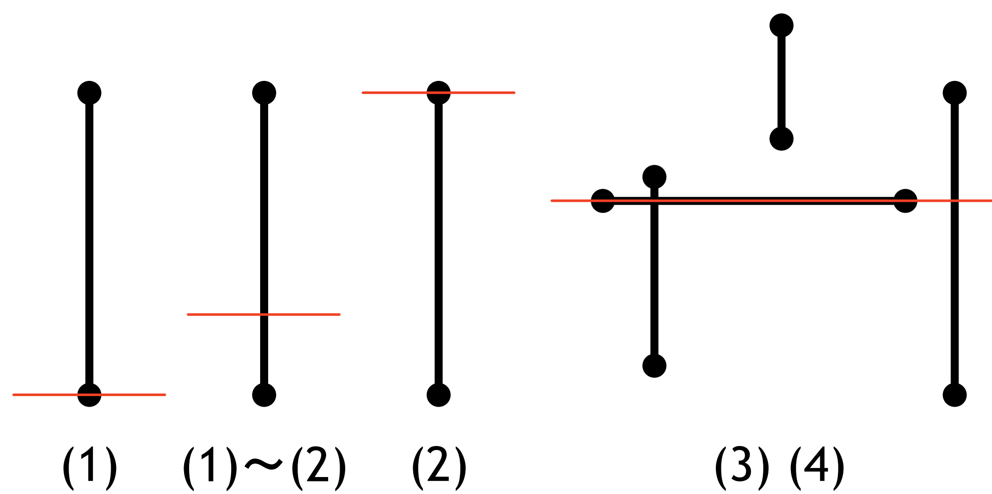
- 平面走査法(plane sweep)
  - 1本の水平，または垂直な直線(=走査線)を平面上を移動させながら，線分の交差を見つける



問4：直交する線分の交差(1)

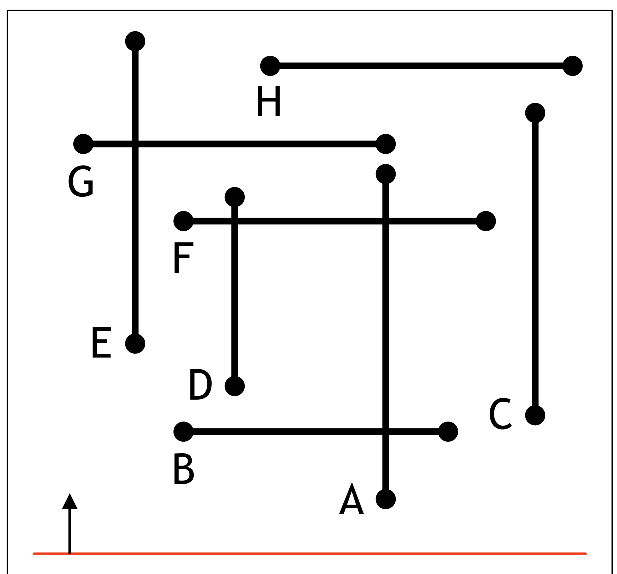
- 平面走査法(plane sweep)の基本的な考え方
  1. 垂直線分と出会う時，垂直線分が走査線上に現れる
  2. 垂直線分と離れる時，上記の点が走査線から消える
  3. 水平線分と出会う時，捜査線と一瞬だけ重なる
  4. 水平線分と出会う時，この水平線分の区間内に垂直線分の点があるか？　あるならばそれが交差である！





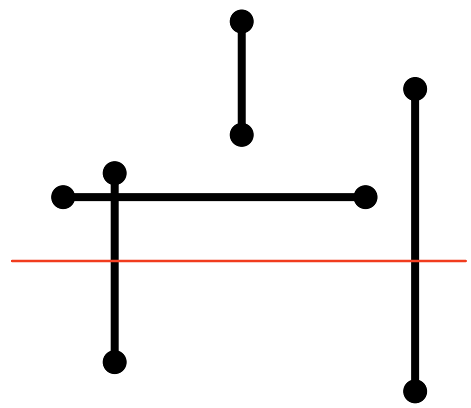
問4：直交する線分の交差(1)

- イベント計画(event schedule)
  - 走査線のイベントポイント(停止位置)を順に教えてくれる
  - y座標順に線分の端点を整列させる
    → A, B, C, D, E, F, D, A, G, C, H, E
  - 垂直線分は2回(上下の端点)，水平線分は1回
  - リストで実現する



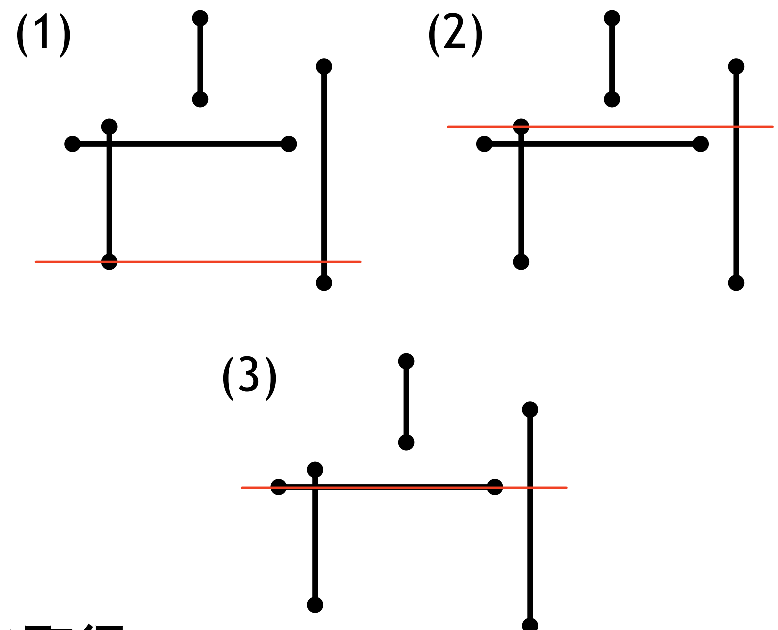
問4：直交する線分の交差(1)

- 走査線計画(sweep-line schedule)
  - 走査線と垂直成分との<u>交差状況を適切に表現するデータ構造</u>
  - 水平線分と出会った時に，それと交差する垂直線分を迅速に見つけ出すことができる構造
  - 走査線計画は垂直線分の各イベントポイントで変更され，水平線分の各イベントポイントで交差を見つけるために使う



問4：直交する線分の交差(1)

- 走査線計画(sweep-line schedule)
  - 2分探索木で実現する



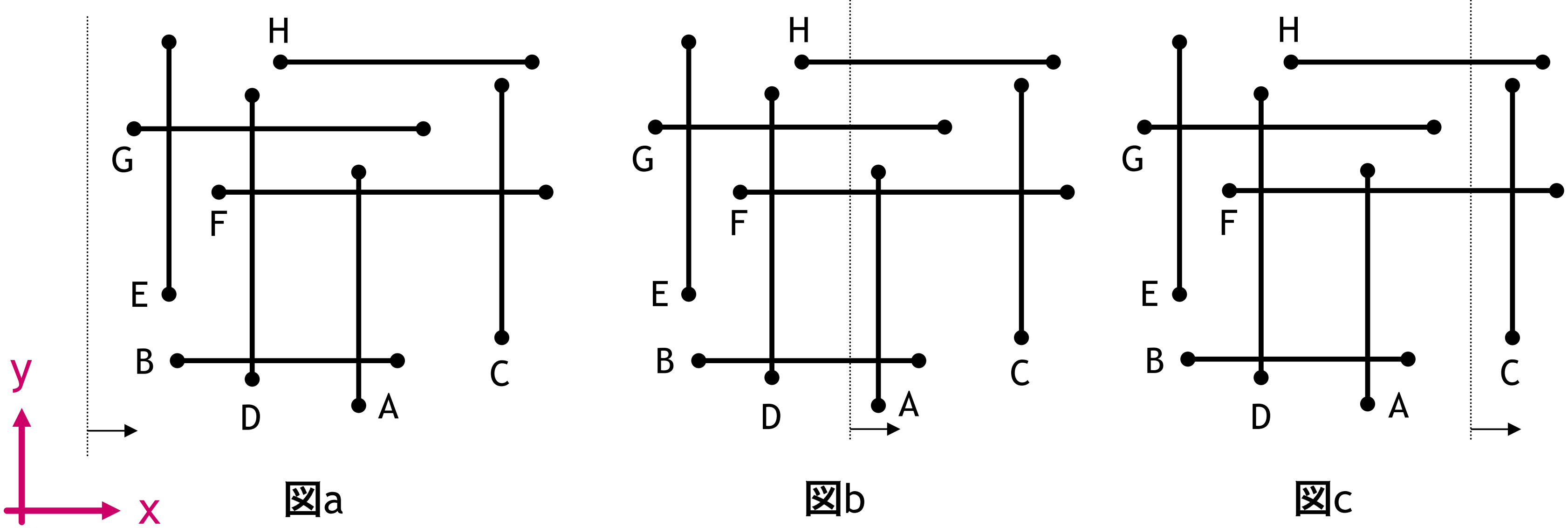
1. 垂直線分の下端点
   → x座標を木に挿入
   (交差の候補とする)

2. 垂直線分の上端点
   → x座標を木から削除

3. 水平線分
   → 2端点のx座標(x1, x2)を取得
   → 区間探索(x1〜x2の範囲)
   (水平線分と交わる垂直線分(=交差)を木から探す)

問5：直交する線分の交差(2)

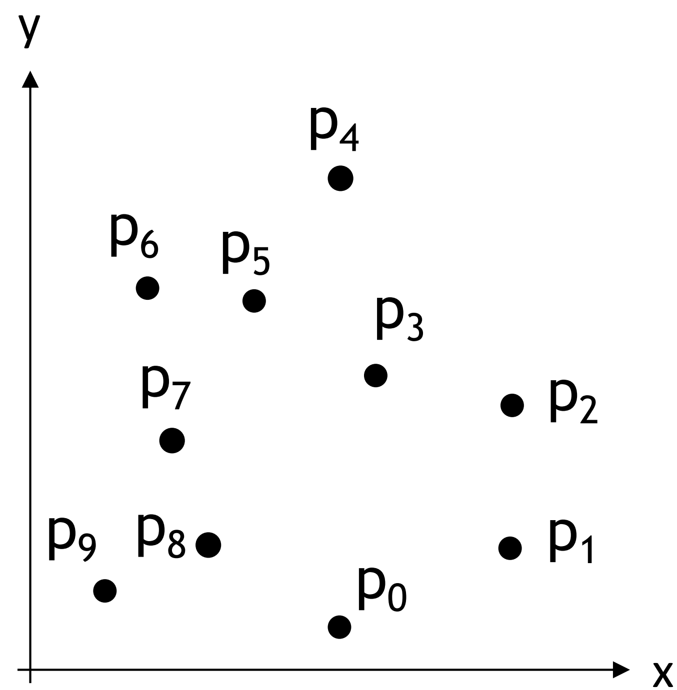
以下の図aに示す複数本の垂直・水平線分の交差を平面走査法で求める．走査線は初期状態では図aに示す位置にあり，その後徐々に+x方向へ移動する．走査線が図b，図cの位置に到達した際の2分探索木の状態を，教科書や講義資料のように記号で示しなさい．問題4の場合とは異なり，走査線が水平線分の左端点に到達すると，その線分のy座標を2分探索木に挿入し，右端点に到達した時に木から削除するものとする．



図a　図b　図c

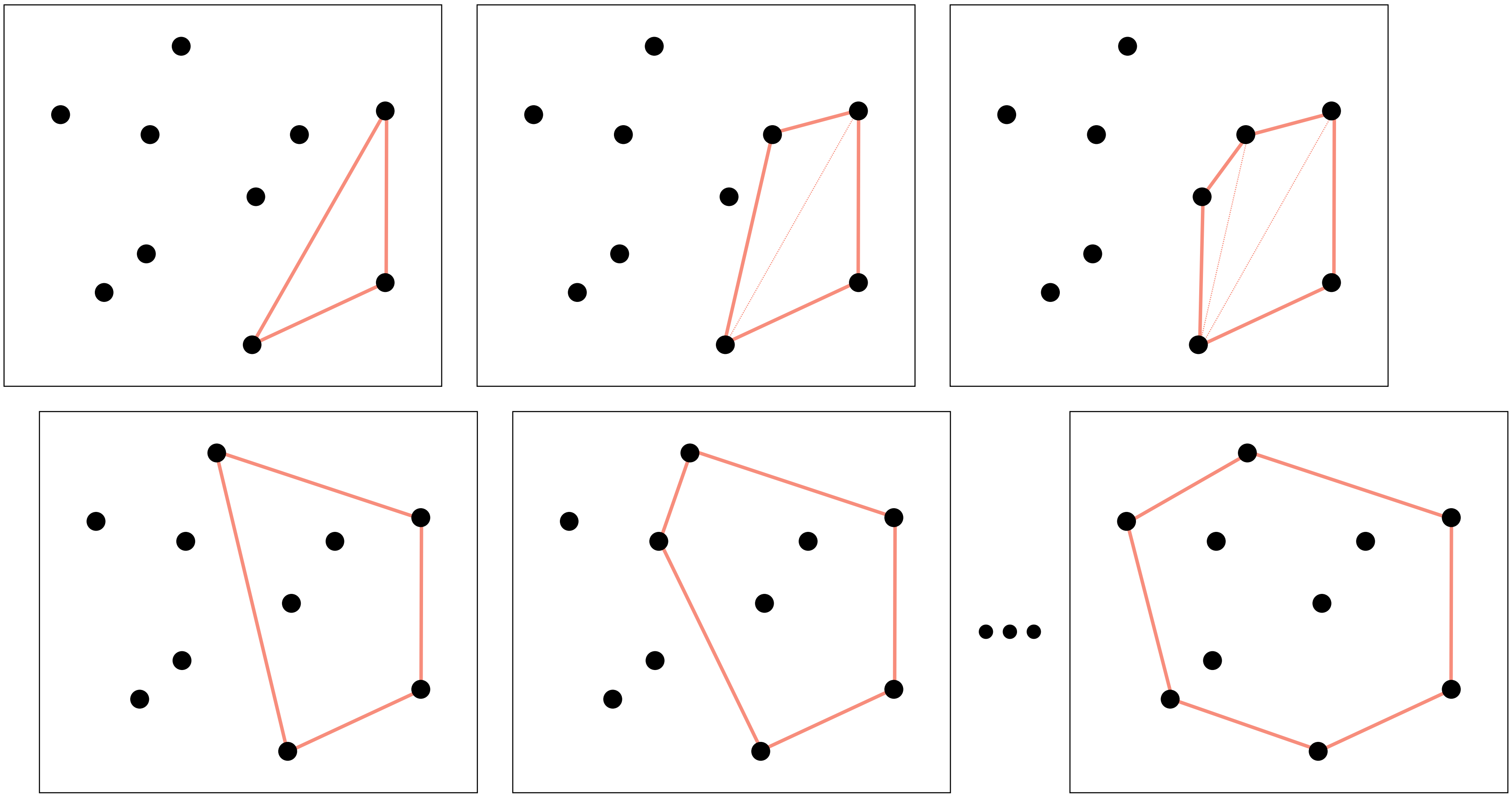
問6：凸包(Graham走査法)

以下に示す10点からGraham走査法で凸包を作ることを考える．この方法ではまず，入力点を偏角順にソートする．ここではp_0を基準点とする．その後p_0とp_1をスタックに追加した後，p_2から順にそれまでに出来上がっている凸包(スタック)に点を加え，その結果，凸包上にありえないことが判明した点をスタックから取り出し，作成中の凸包を作り直す．以下の図の場合，スタックに追加される点(Push)，及びスタックから取り出される点(Pop)をその順に並べなさい．（例：$p_i \rightarrow p_j \rightarrow p_k$）



問6：凸包(Graham走査法)

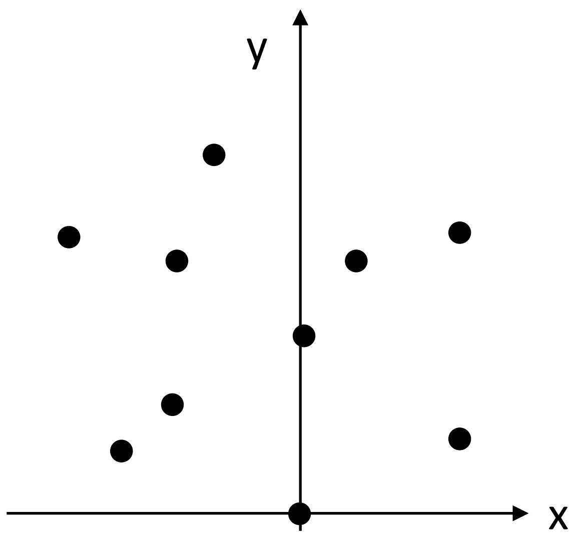
■ 点を1つずつ追加しながら凸包を徐々に作る

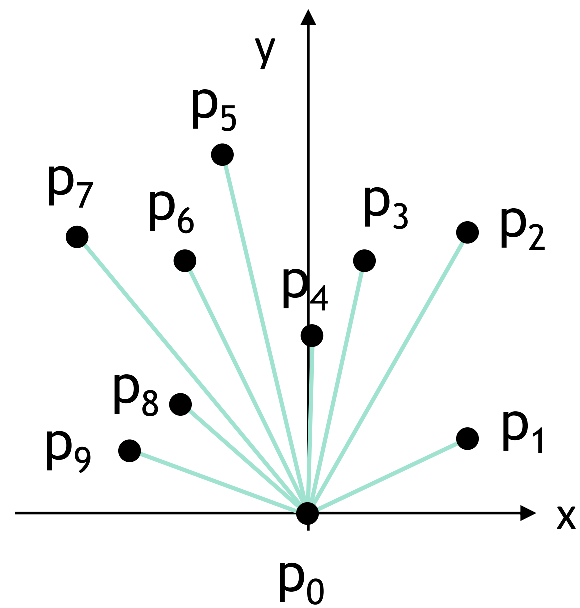


問6：凸包(Graham走査法)

■ 基本的な手順

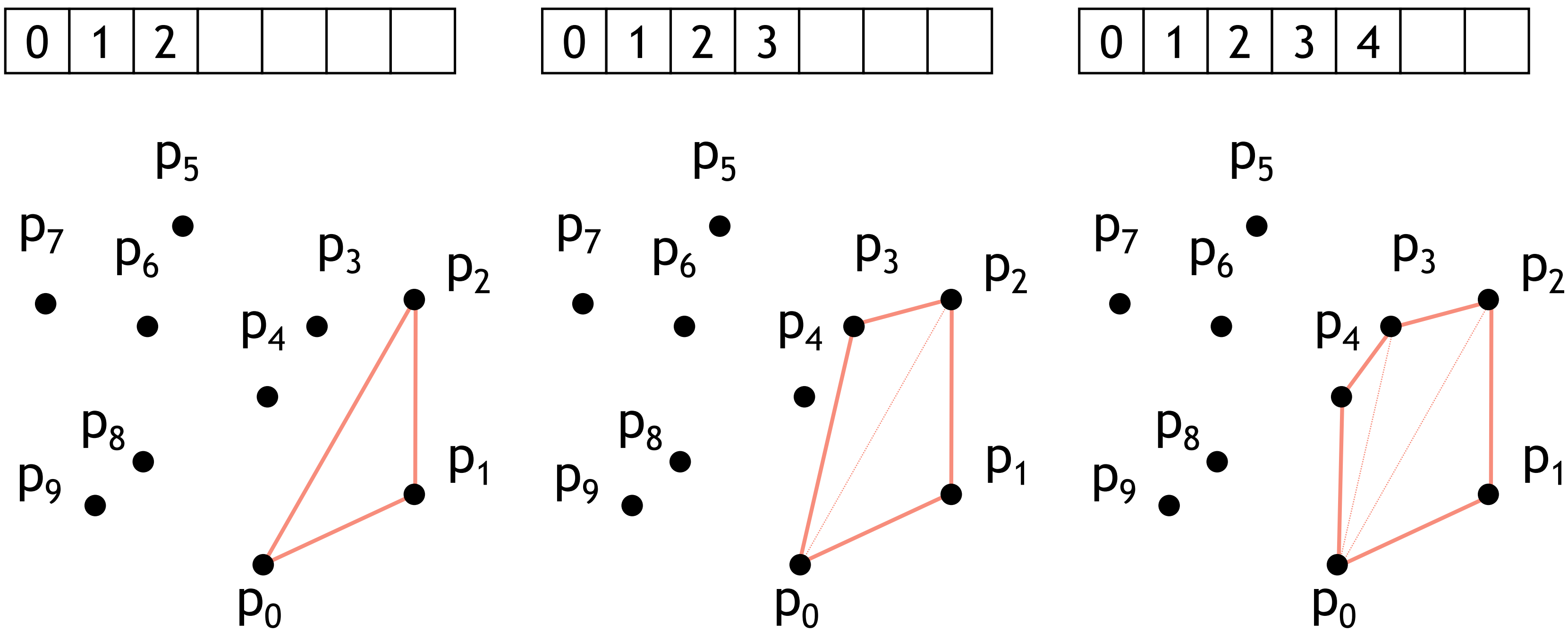
1. 入力点を角度順にソートする
2. ソート順に点を走査しながら凸包を計算する
(p0とp1は先に入れ，その後p2→p3→p4→ … →p9)





問6：凸包(Graham走査法)

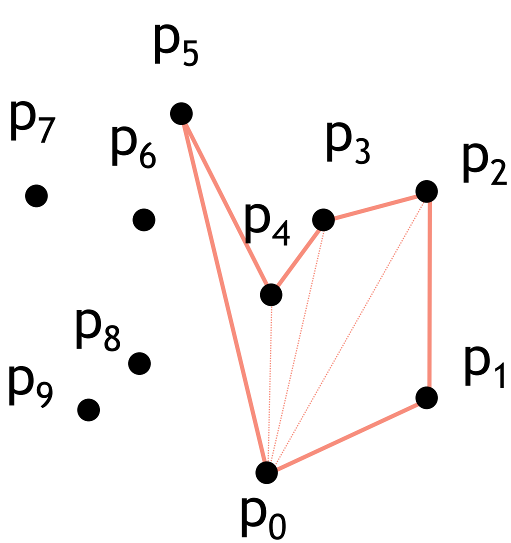
■ 作成途中の凸包はスタックに格納する



問6：凸包(Graham走査法)

- 図の場合，p5を入れると凸包ではなくなる
- 新たに追加したp5は残して，他を削除して凸包にする

0	1	2	3	4	5	



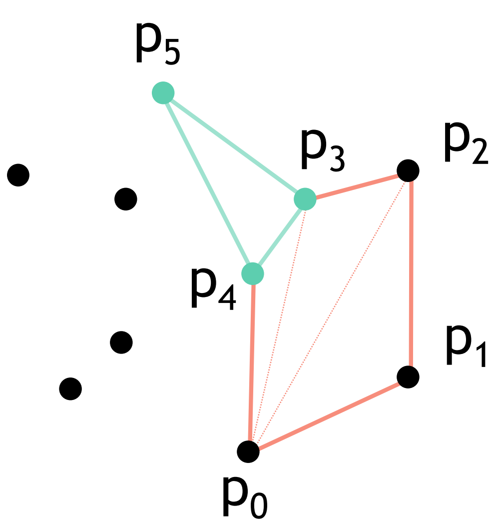
- **頂点削除の考え方**
  - 基準点p0から反時計回りに走査している
  - 凸包の頂点は反時計回りのはず
  - 時計回りに連続する3点を探し，その要因となる頂点を削除する

問6：凸包(Graham走査法)

■ 削除の方法(新たにp5を追加する場合)

top

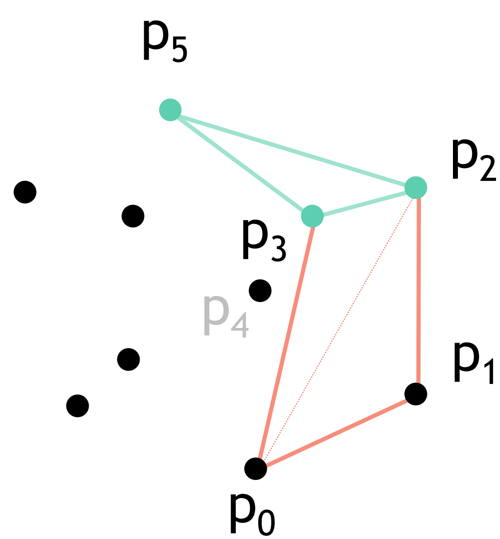
S							
	0	1	2	3	4		



S[top-1], S[top], piの順を確認
(図の場合，p3,p4,p5)
時計回りなのでS[top]を削除
(図の場合，p4)

top

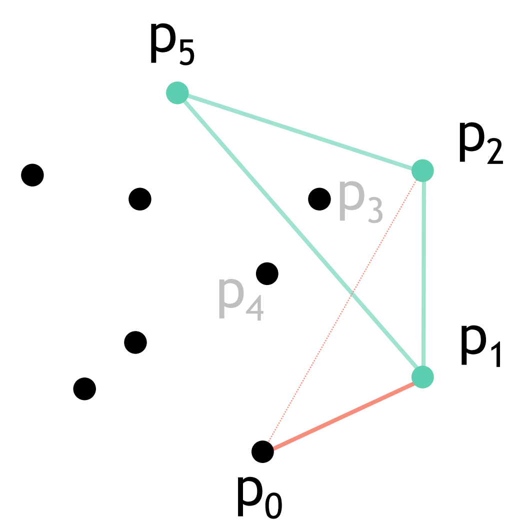
0	1	2	3			



同様に3点の順を確認
(図の場合，p2,p3,p5)
時計回りなのでS[top]を削除
(図の場合，p3)

top

0	1	2	5			



同様に3点の順を確認
(図の場合，p1,p2,p5)
反時計回りなのでSに
piを追加(図の場合，p5)

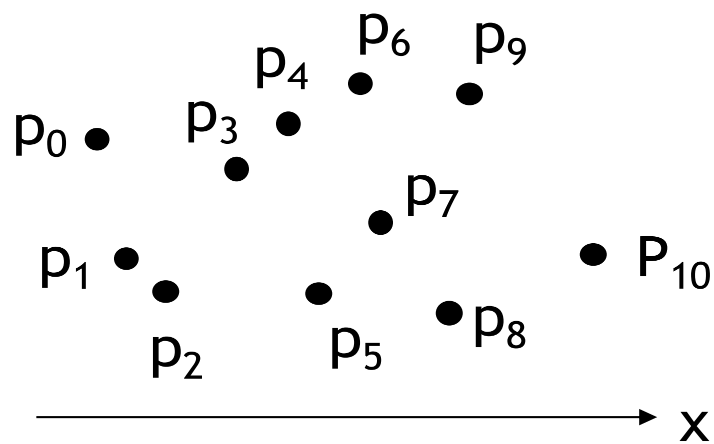
問6：凸包(Graham走査法)

■ アルゴリズム

1. y座標が最小となる頂点を探索し，基準点p_0とする
2. 他の点をp0に対する角度の昇順に並び替える
   (p_1, p_2, p_3, ..., p_{n-1})
3. スタックSにp_0, p_1を追加する
4. p_i(i=2, ..., n-1) に対して順に以下の処理を行う
   1. while(S[top-1], S[top], p_iが時計回りならば)
      S[top]をスタックから取り出す
   2. p_iをスタックSに入れる

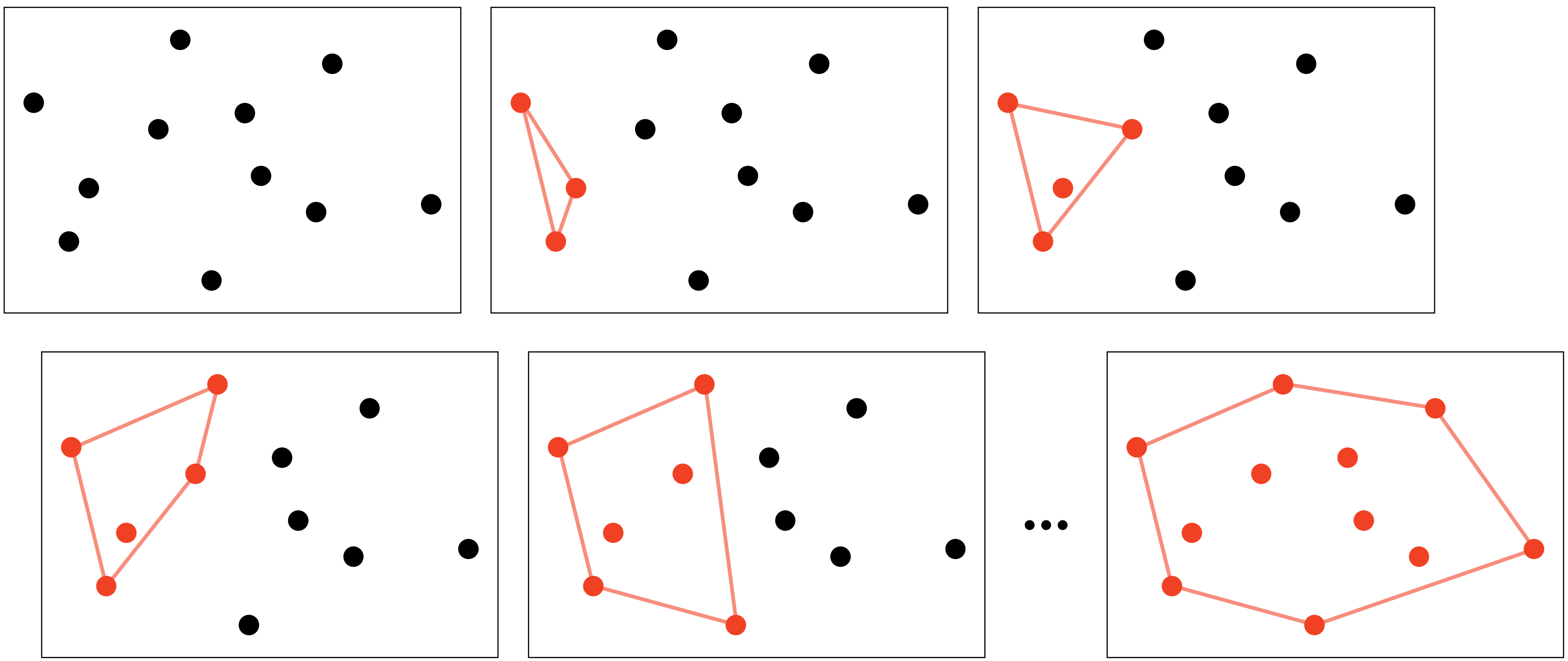
問7：凸包(逐次構成法)

以下に示す10点から逐次構成法で凸包を作ることを考える．この方法ではまず，入力点をx座標順にソートする．その後，上部凸包と下部凸包を別々に計算し，最後に1つに統合する．それぞれの凸包を計算する際には，p_0とp_1をスタックに追加した後，p_2から順にそれまでに出来上がっている凸包(スタック)に点を加え，その結果，凸包上にありえないことが判明した点をスタックから取り出し，作成中の凸包を作り直す．以下の図の場合，上部/下部凸包の計算過程で，スタックに追加される点(Push)，及びスタックから取り出される点(Pop)をその順に並べなさい．ただし，上部凸包は反時計回りに，スタックのトップからp_{10}，p_9，p_6，p_0の順に点を格納するとし，下部凸包は時計回り順に並べるものとする．



3.3 逐次構成法 (Incremental Method)

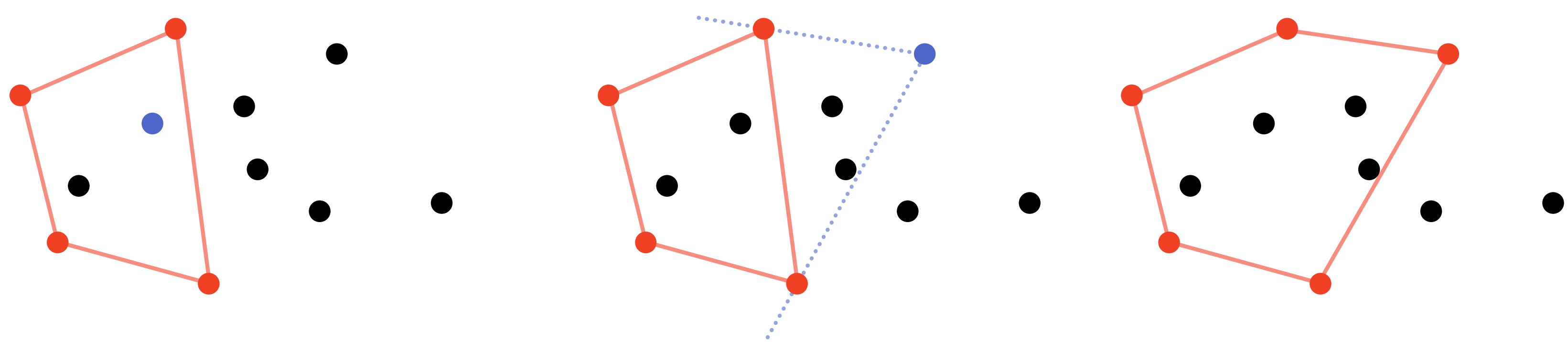
- 最初に少数のデータに対して問題の解を求め，その後データを1つずつ追加しながら解を更新する方法



3.3 逐次構成法 (Incremental Method)

1) 点の内外判定を利用する方法

1. 3点p_0,p_1,p_2による凸包(三角形)を構成する
2. 1点ずつ追加しながら
   a. 点が現在の凸包に含まれるなら，何もしない
   b. 含まれなければ，2本の接線を求め，凸包を作り直す

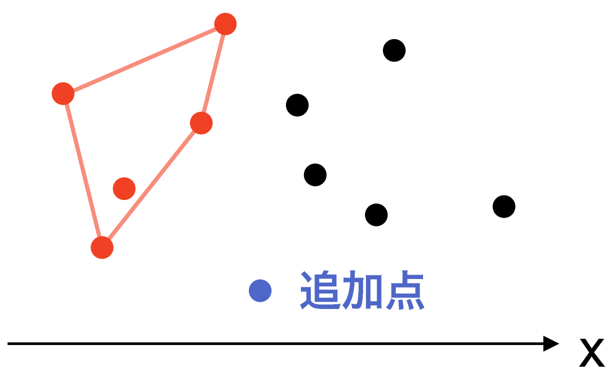


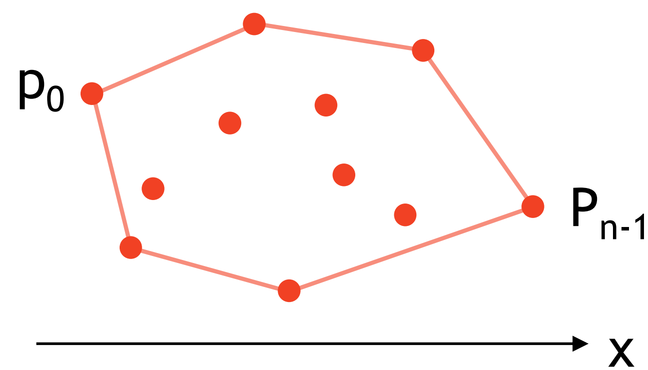
点が凸包に含まれる　　　点が凸包に含まれない

3.3 逐次構成法 (Incremental Method)

2) ソートを利用する方法

- ソート順に点を追加しながら，凸包を作り直す
- 追加点は凸包の外側にあるので，内外判定は不要

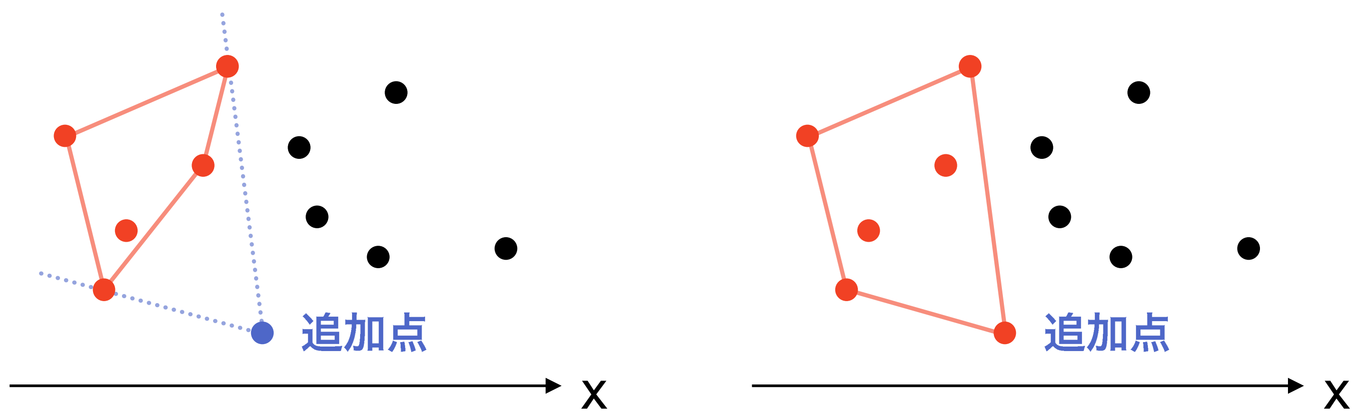




3.3 逐次構成法 (Incremental Method)

2) ソートを利用する方法

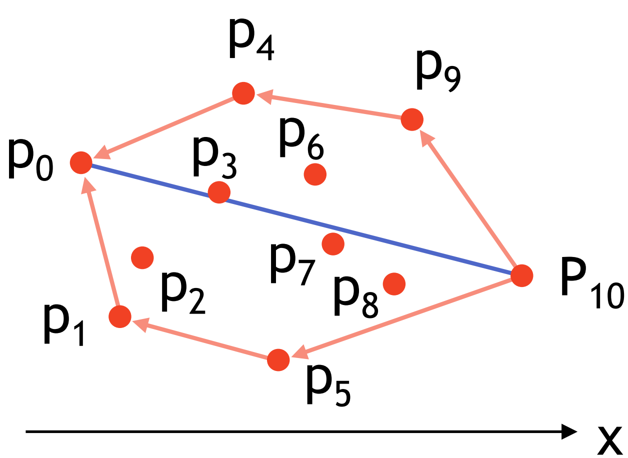
- 点を追加する時に，凸包への接線を2本求め，新たな線分とする
- 2本の接線に挟まれた線分を除外する



3.3 逐次構成法 (Incremental Method)

2) ソートを利用する方法

- 上部凸包，下部凸包を別々に作り，後で統合する
- p_0とp_{n-1}を結ぶ線分で上下に分ける(図の場合n=10)



上部凸包 = $\{p_{10}, p_9, p_4, p_0\}$
→ 反時計回り

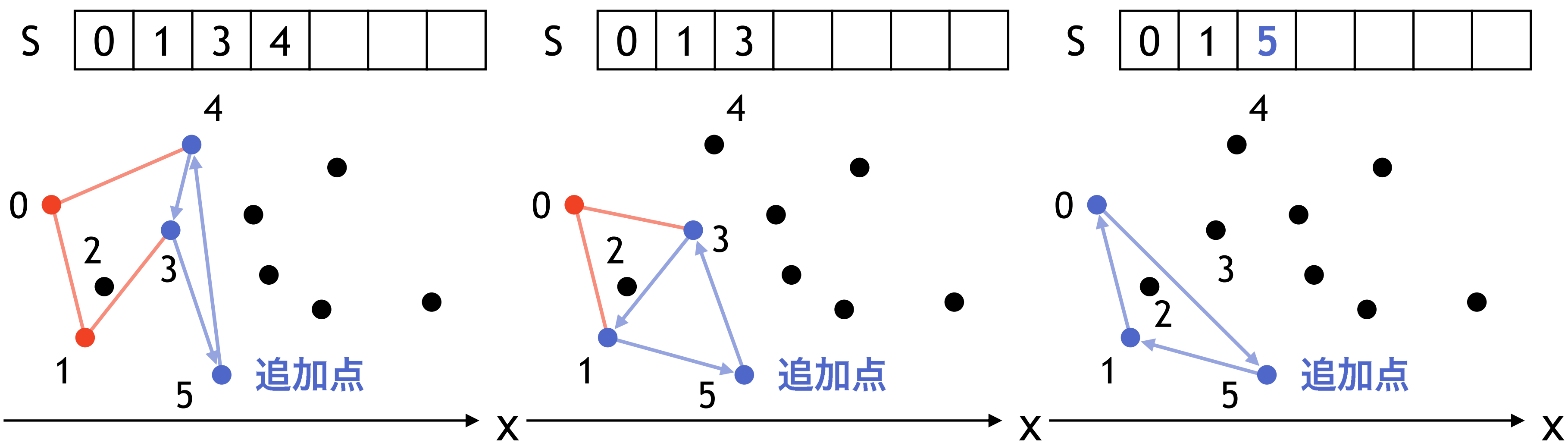
下部凸包 = $\{p_{10}, p_5, p_1, p_0\}$
→ 時計回り

3.3 逐次構成法 (Incremental Method)

2) ソートを利用する方法

– 接線の求め方 = 新たに追加する点と，最後に追加した2つの点の並び順を評価する

<u>下部凸包(時計回り)，点p_5を追加する場合</u>



(5,4,3)は反時計回り
→ 4を削除

(5,3,1)は反時計回り
→ 3を削除

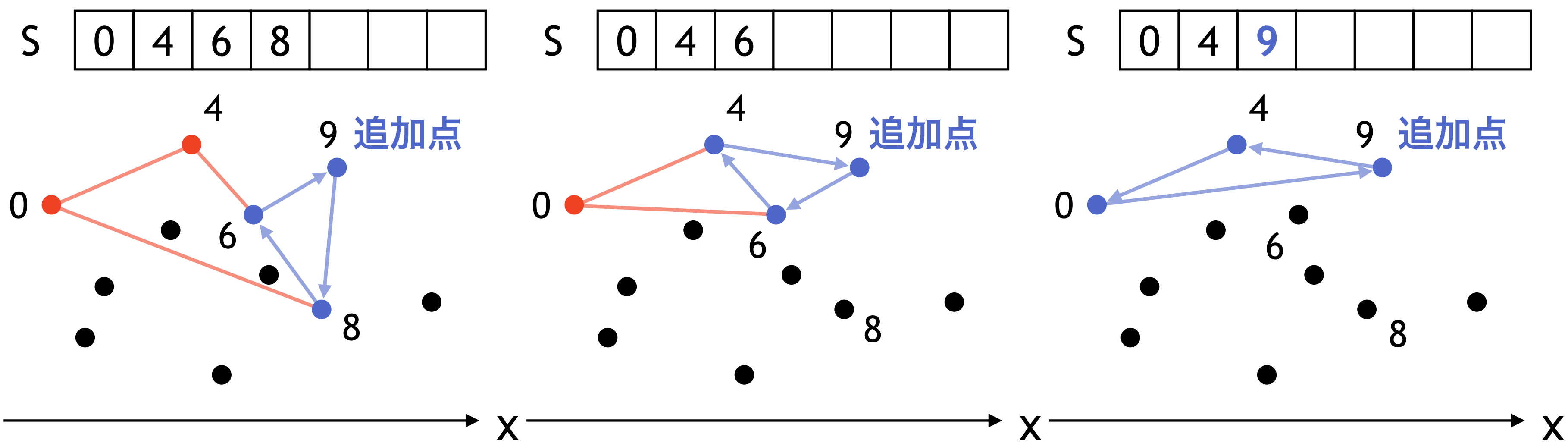
(5,1,0)は時計回り
→ 5を追加

3.3 逐次構成法 (Incremental Method)

2) ソートを利用する方法

- 接線の求め方 = 新たに追加する点と，最後に追加した2つの点の並び順を評価する

上部凸包(反時計回り)，点p_9を追加する場合



(9,8,6)は時計回り
→ 8を削除

(9,6,4)は時計回り
→ 6を削除

(9,4,0)は反時計回り
→ 9を追加

3.3 逐次構成法 (Incremental Method)

2) ソートを利用する方法

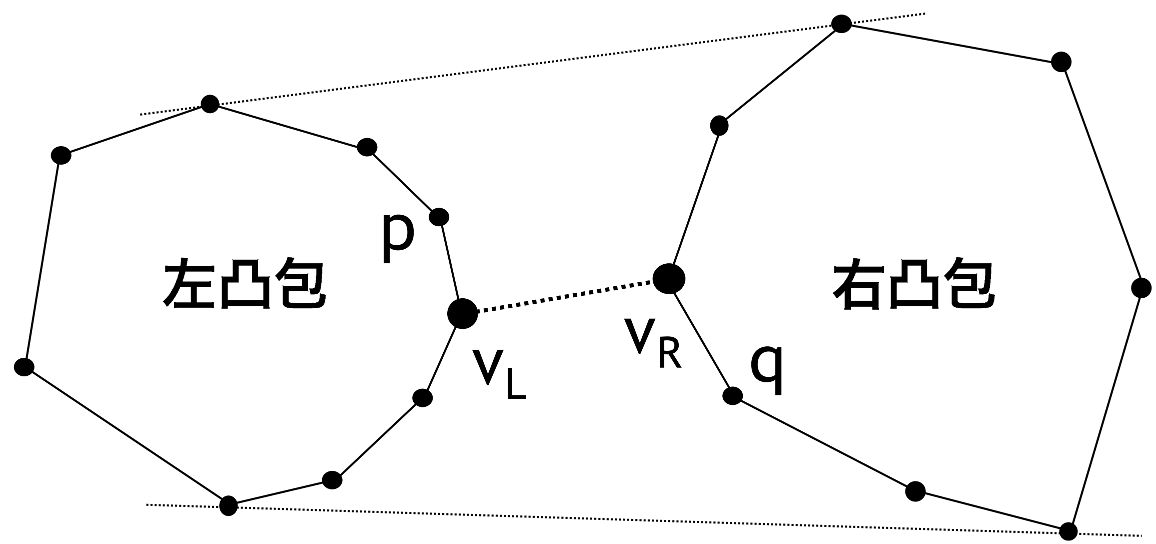
上部凸包の構成アルゴリズム

1. 全ての点をx座標の昇順にソートする　　$O(n\log n)$
2. p_0,p_1をスタックSに積む
3. p_i(i=2,3,…,n-1)に対し以下を行う　　$O(n)$
   a. p_i, S[top], S[top-1]が時計回りであれば，S[top]を取り出す
   b. これを3点の順が反時計回りになるまで繰り返す
   c. p_iをスタックSに積む

総計算量　$O(n\log n)$

問8：凸包(分割統治法)

分割統治法で凸包を作ることを考える．この方法では，小さな凸包ペアの上部/下部接線を求めることで統合を行う．以下のような2つの凸包から接線を求める際には，左凸包の右端点(v_L)と，右凸包の左端点(v_R)から処理を開始し，3点の並び順を評価すること(三角形の符号付き面積の計算)を繰り返す．以下の例の場合，上部接線，下部接線の計算に，それぞれ何回の評価が必要か．なお最初の評価は，上部接線を求める際には三角形$v_R v_L p$から，下部接線を求める際には三角形$v_L v_R q$から，図に示す三角形より開始するものとする．



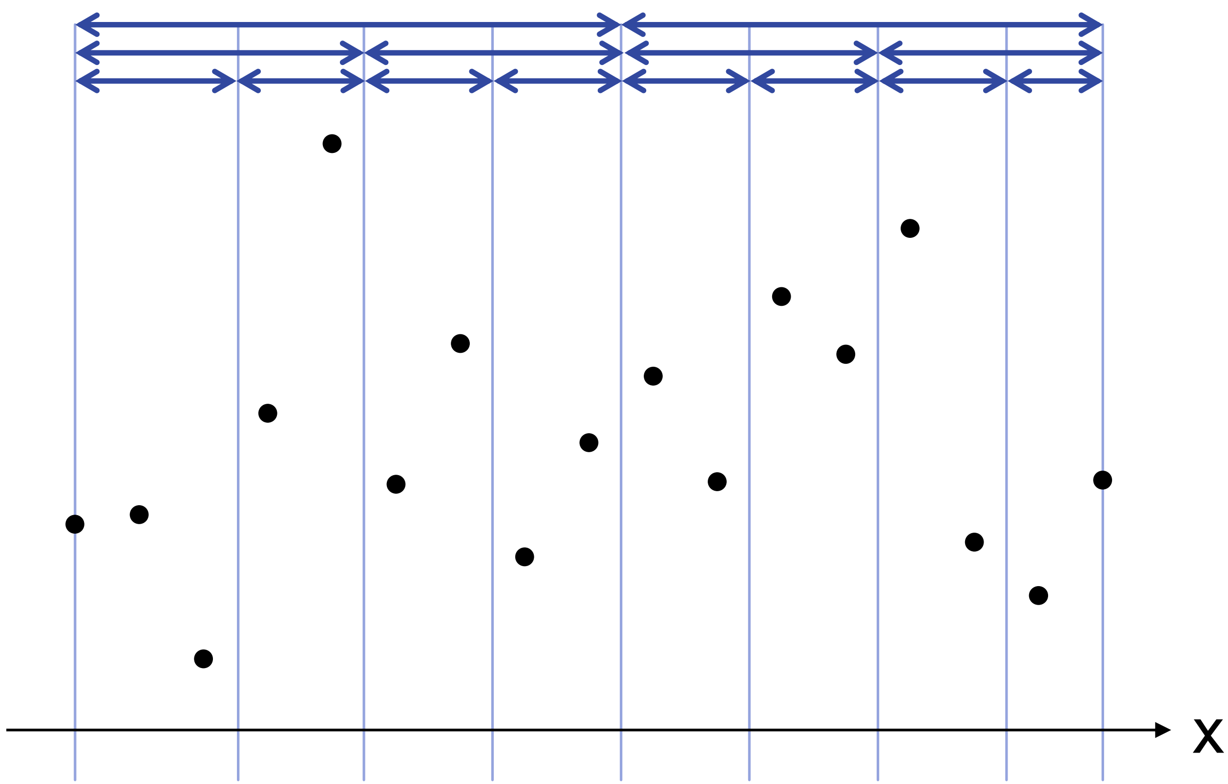
問8：凸包(分割統治法)

■ 分割統治法を利用した凸包の計算

1. ソート：点をx座標順にソートする
2. 分割：各グループをx座標の中央値で2グループに分割する(点数が3以下になるまで繰り返す)
3. 凸包の計算：各グループの凸包を計算する
4. 統合：2つの凸包の上下の2接線を求めて統合する(1つになるまで繰り返す)

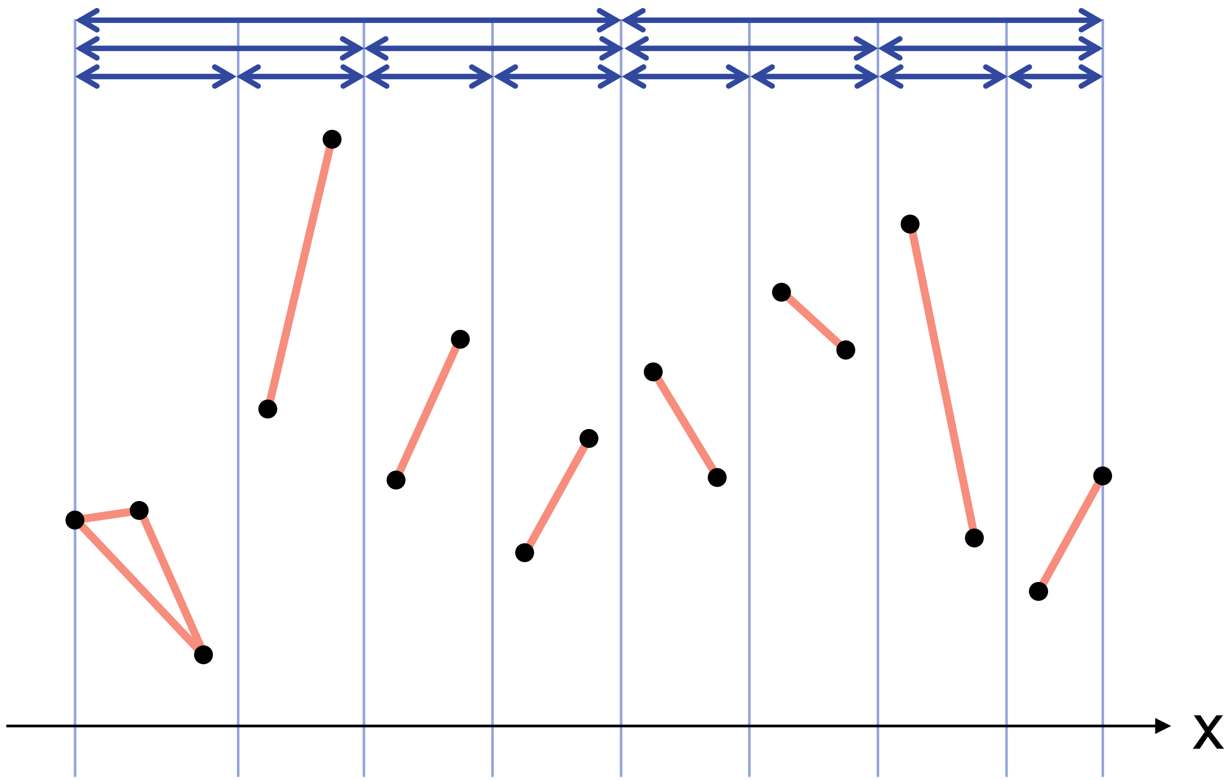
問8：凸包(分割統治法)

2. 分割：各グループをx座標の中央値で2グループに分割する(点数が3以下になるまで繰り返す)



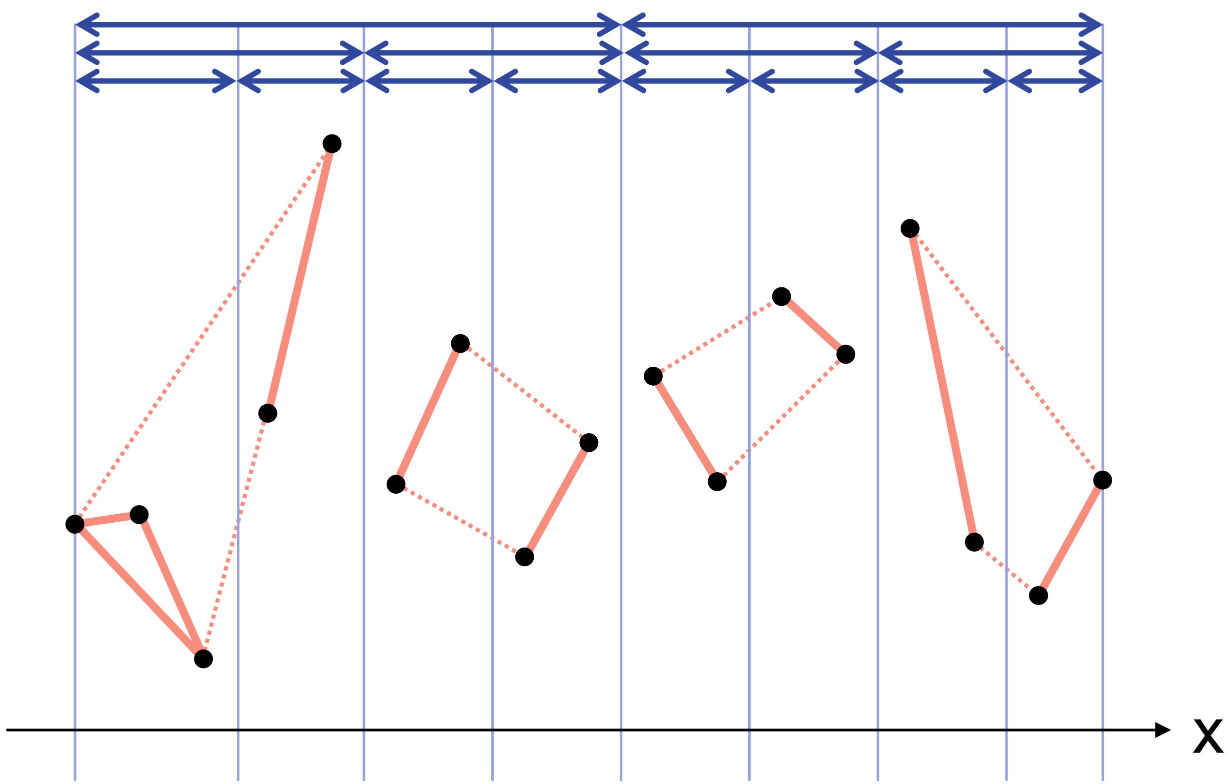
問8：凸包(分割統治法)

3. 凸包の計算：各グループの凸包を計算する



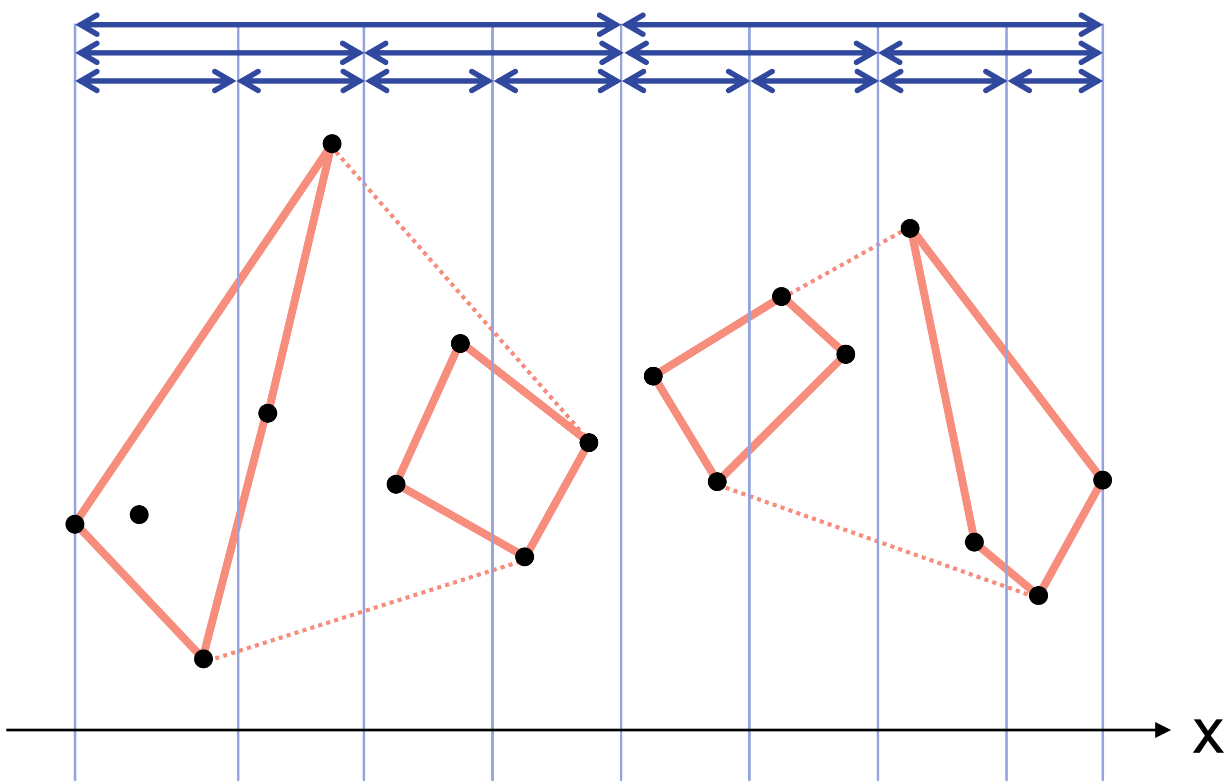
問8：凸包(分割統治法)

4. 統合：2つの凸包の上下の2接線を求めて統合する(1つになるまで繰り返す)



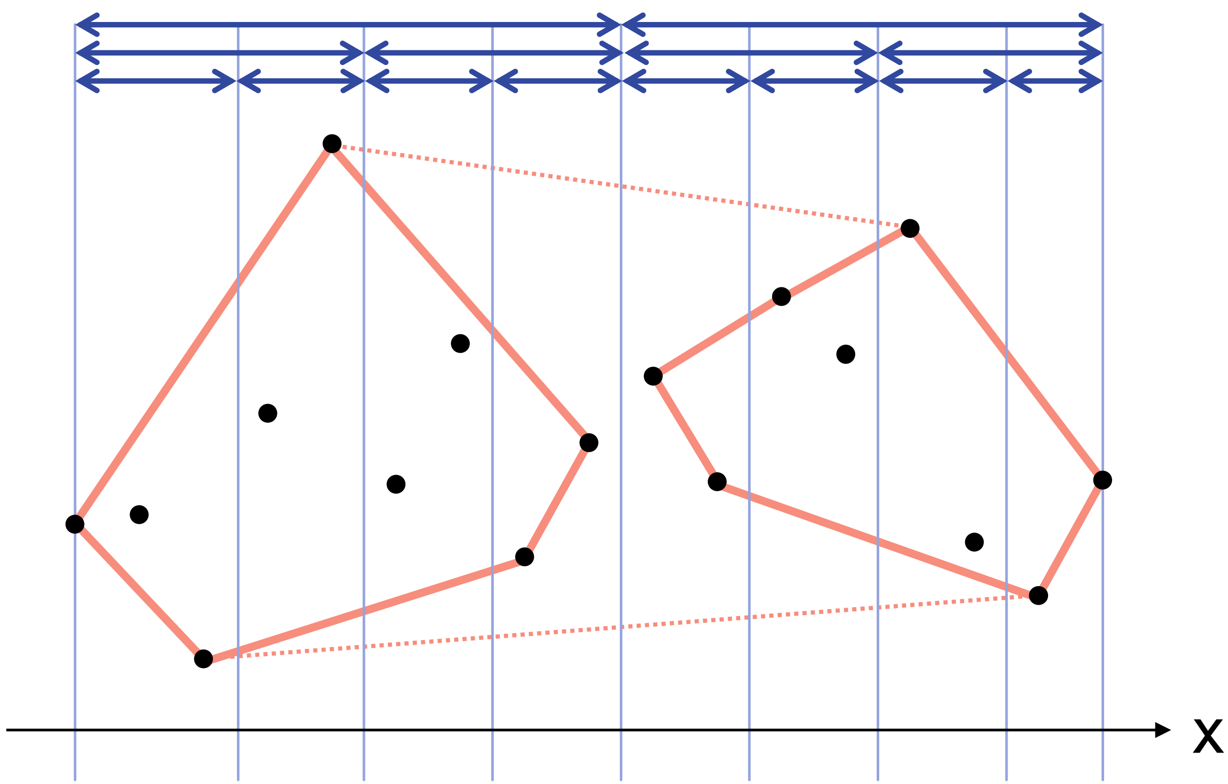
問8：凸包(分割統治法)

4. 統合：2つの凸包の上下の2接線を求めて統合する(1つになるまで繰り返す)



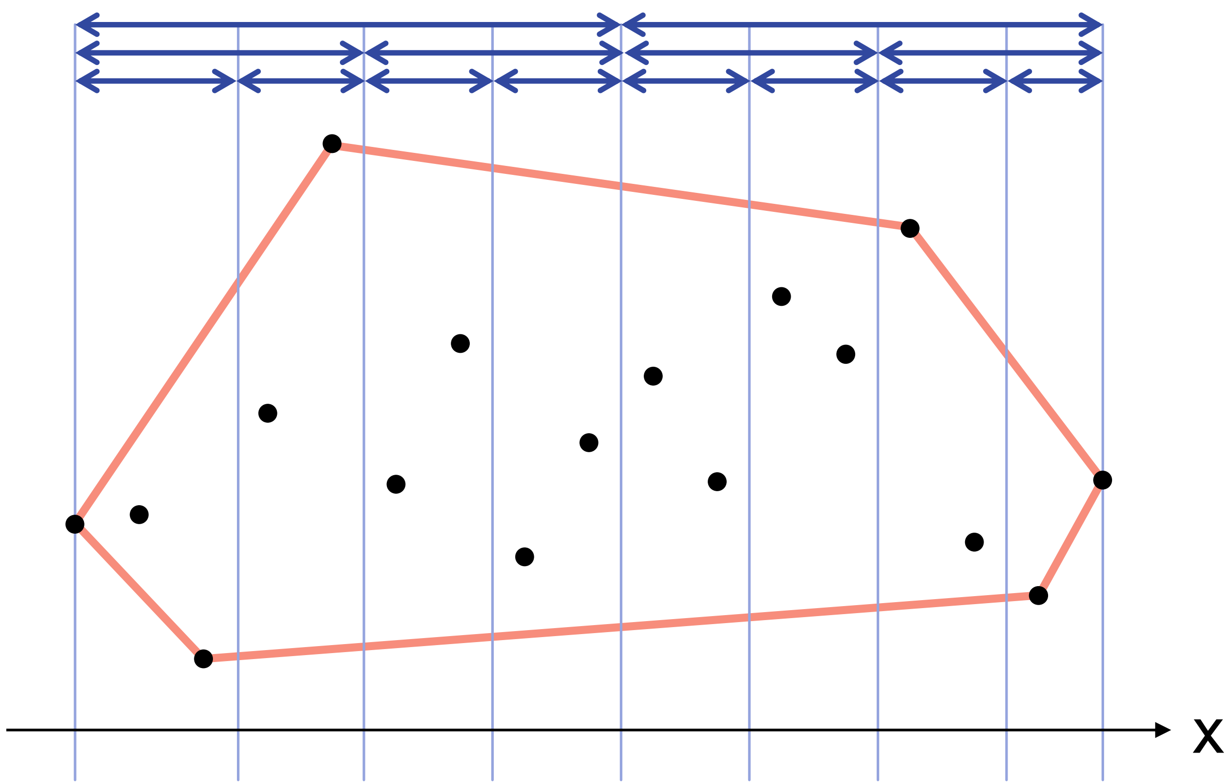
問8：凸包(分割統治法)

4. 統合：2つの凸包の上下の2接線を求めて統合する(1つになるまで繰り返す)



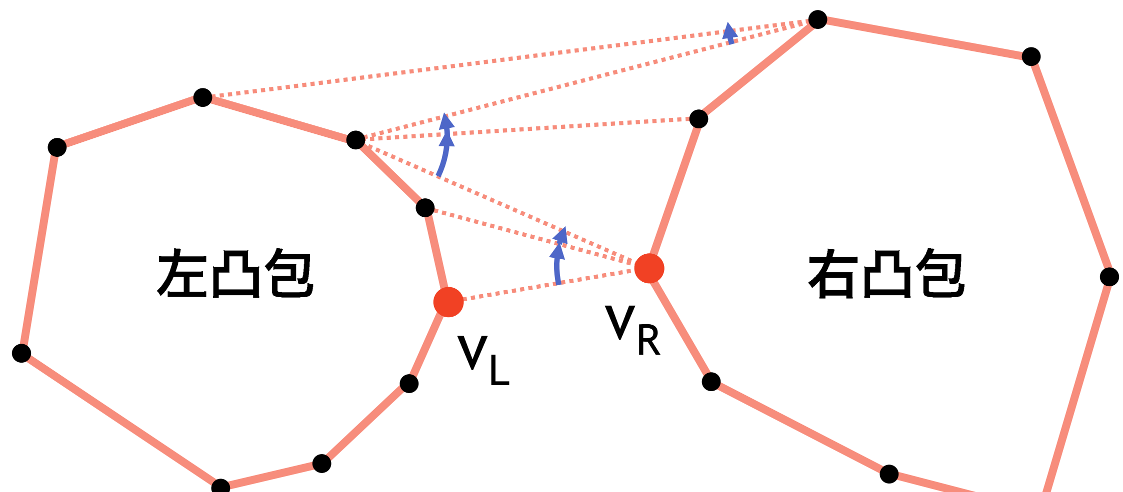
問8：凸包(分割統治法)

4. 統合：2つの凸包の上下の2接線を求めて統合する(1つになるまで繰り返す)

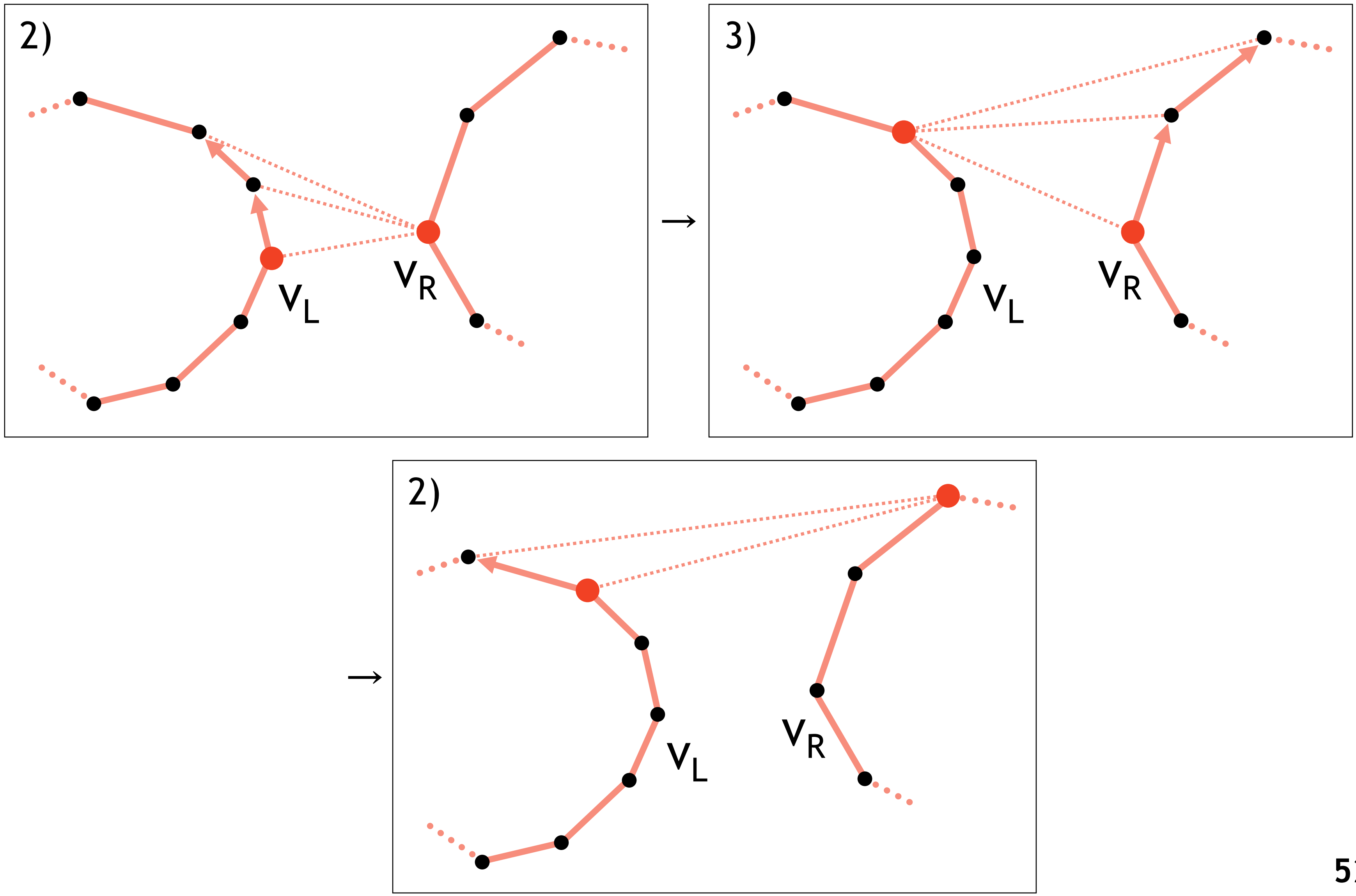


問8：凸包(分割統治法)

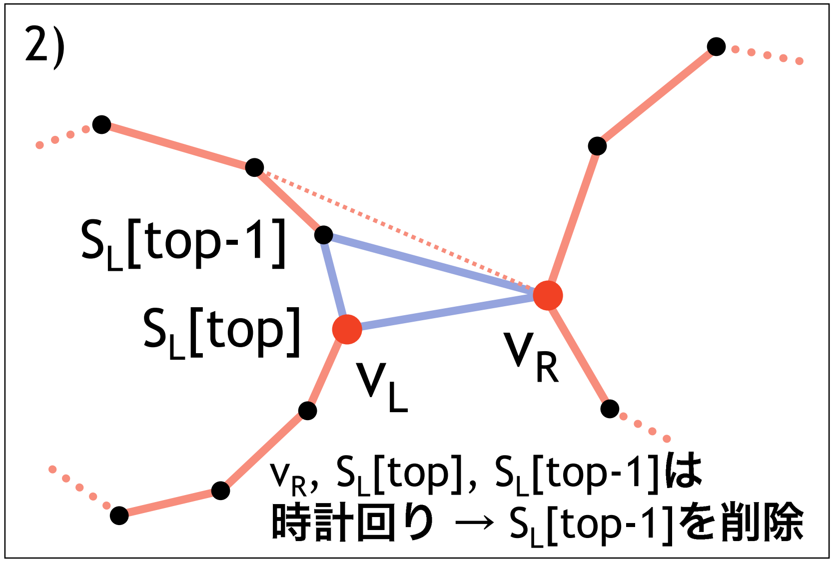
- 上部接線の求め方
  1. 左凸包の右端点(v_L)，右凸包の左端点(v_R)を探す
  2. v_Rを固定して，v_Lを移動させながら接線を見つける
  3. v_Lを固定して，v_Rを移動させながら接線を見つける
  4. 手順2, 3を凸包の接線が見つかるまで繰り返す



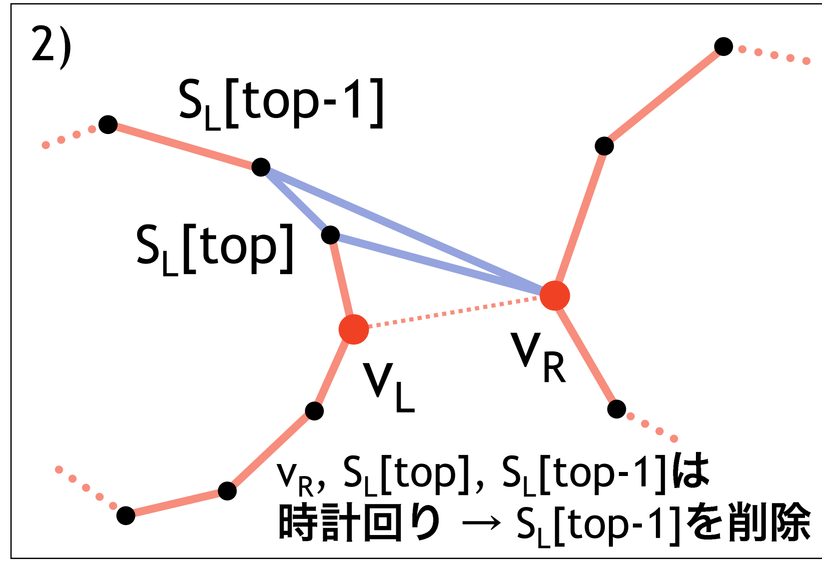
問8：凸包(分割統治法)



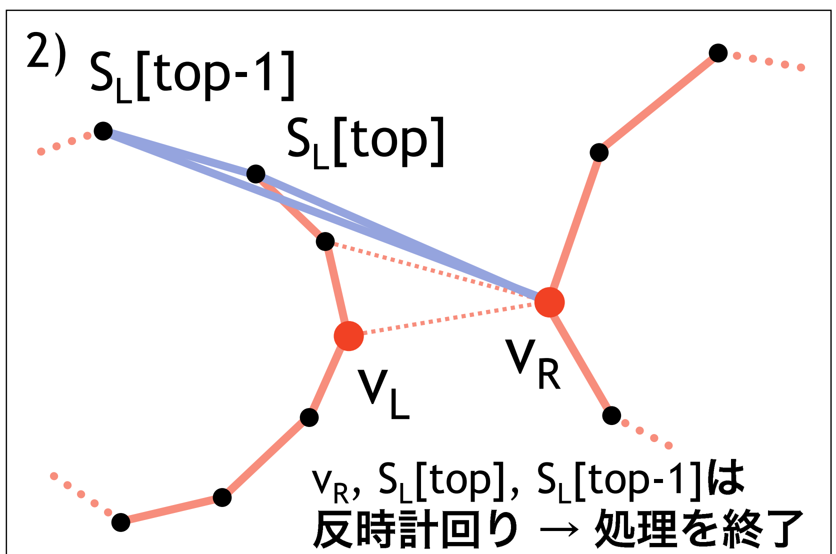
問8：凸包(分割統治法)



2)
S_L[top-1]
S_L[top]
v_L
v_R
v_R, S_L[top], S_L[top-1]は
時計回り → S_L[top-1]を削除



2)
S_L[top-1]
S_L[top]
v_L
v_R
v_R, S_L[top], S_L[top-1]は
時計回り → S_L[top-1]を削除



2)
S_L[top-1]
S_L[top]
v_L
v_R
v_R, S_L[top], S_L[top-1]は
反時計回り → 処理を終了