生産情報システム工学

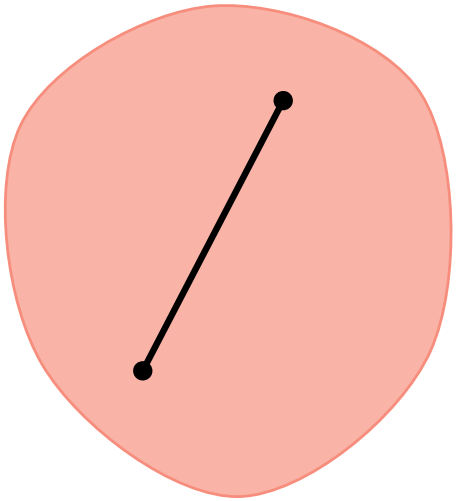
#08 凸包の計算(1)

2015/06/10(水)

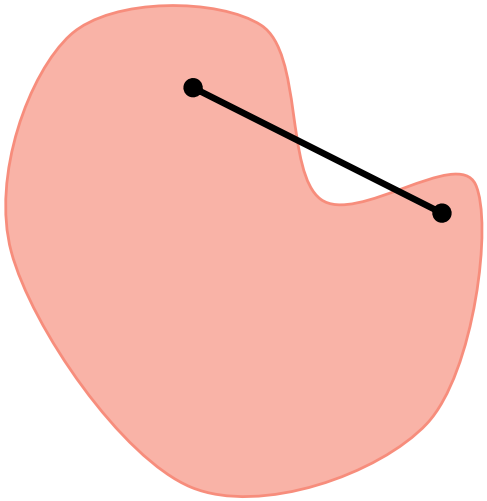
溝口 知広　准教授(居室：61-408室)
mizo@cs.ce.nihon-u.ac.jp

3.0 はじめに

- 凸集合(Convex Set)
  - 平面上に与えられた点集合に対し，任意の2点を結ぶ線分が完全に含まれる

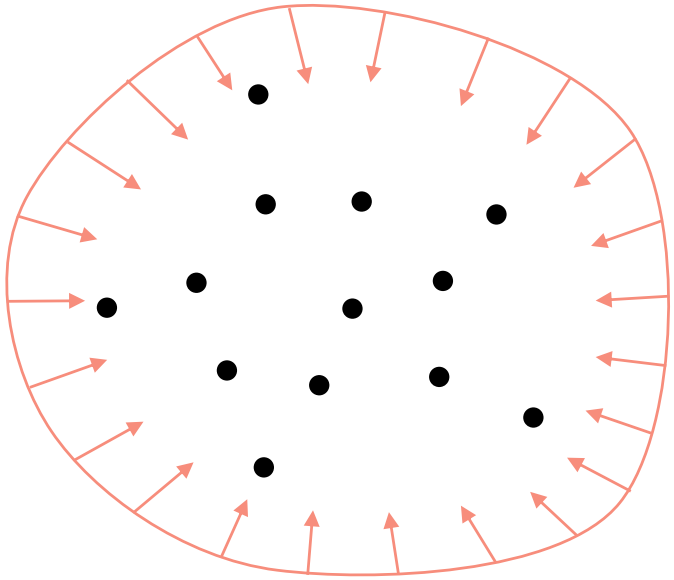


凸



凸ではない

3.0 はじめに

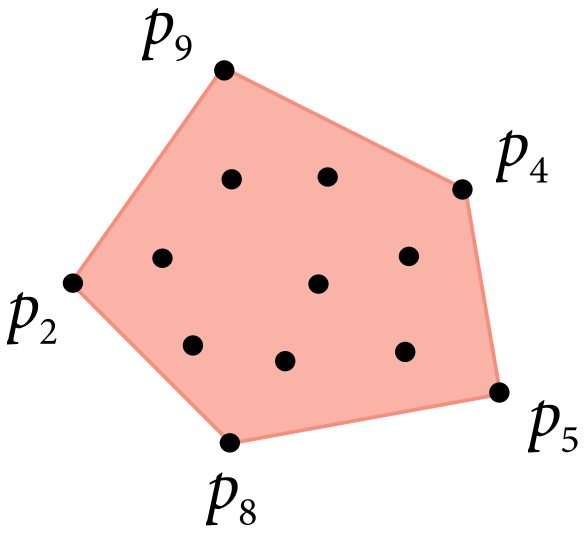


- 凸包(Convex Hull)
  - 平面上に与えれる点集合を含む最小の凸多角形
  - 入力：点集合

$$S = \{p_1, p_2, \cdots, p_n\}$$

  - 出力：凸包を形成する点集合

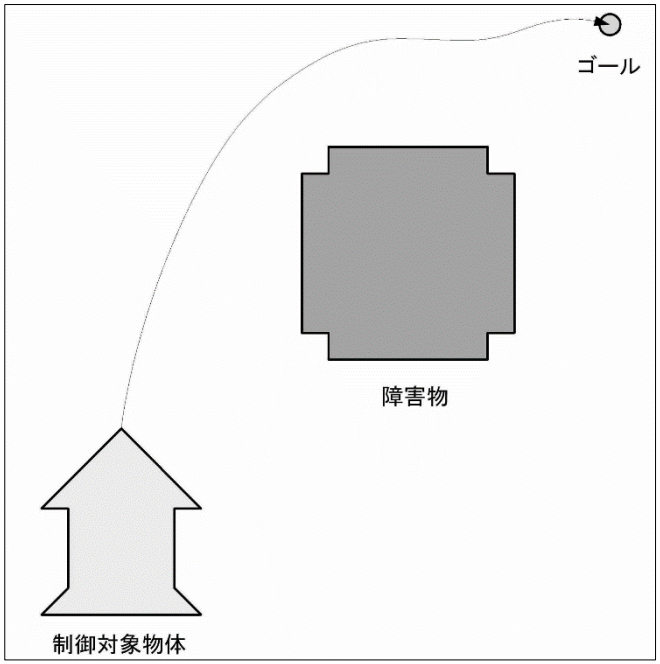
$$CS = \{p_4, p_5, p_8, p_2, p_9\}$$

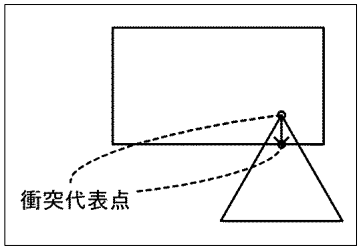


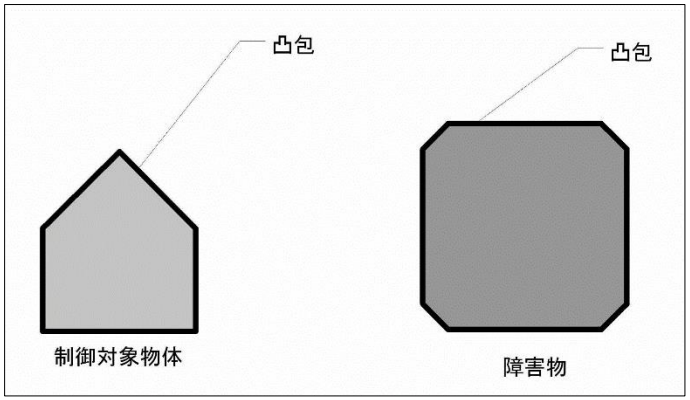
3.0 はじめに

■ 凸包の応用例

- 衝突判定(ロボットなど)
  - 制御対象物と障害物を凸包で近似し計算コストを下げる







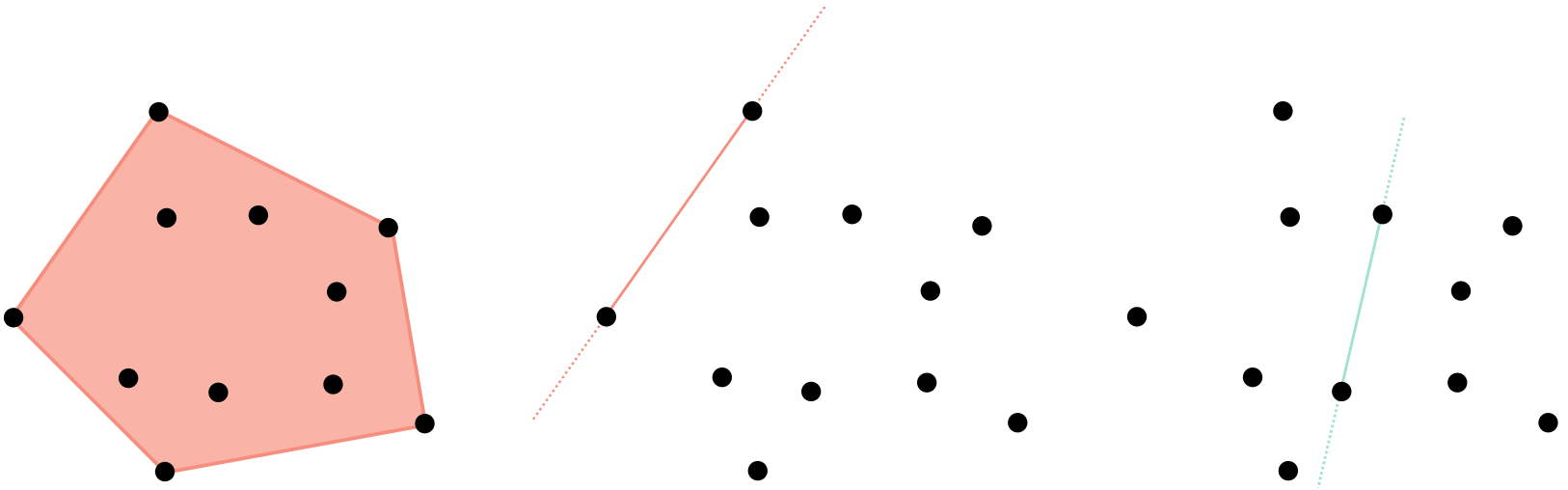
3.0 はじめに

■ 計算方法

- 直接法 (本日の内容)
- 包装法 (本日の内容)
- Graham走査法 (本日の内容)
- 逐次添加法 (来週の内容)
- 分割統治法 (来週の内容)
- …
- など

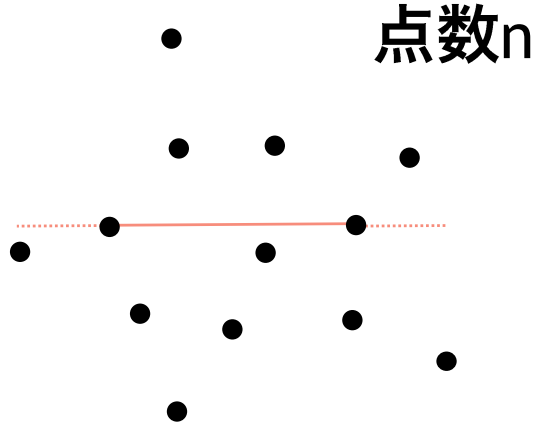
3.0 はじめに

- 直接法(Direct Method)
  - 点集合から2点を取り出し，その線分が凸包を構成する多角形の辺であるかを判定する
  - 残りのすべての点が線分の片側にあれば凸包の線分



実際の凸包　　凸包の線分　　凸包の線分ではない

3.0 はじめに



■ 直接法のアルゴリズムと計算量

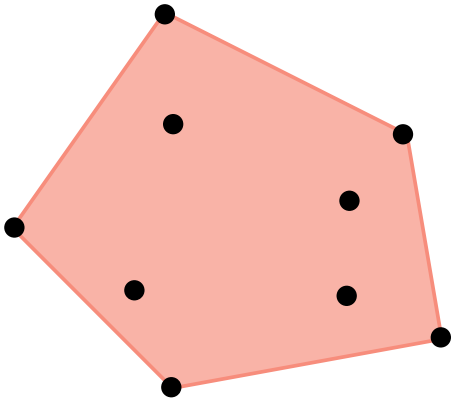
1. 全ての点ペアに対して, ← ペア数：$n(n-1)/2=O(n^2)$
2. 残りの点が線分の片側にあるかを判定する ← 残りの点数：$(n-2)=O(n)$

総計算量：$O(n^3)$

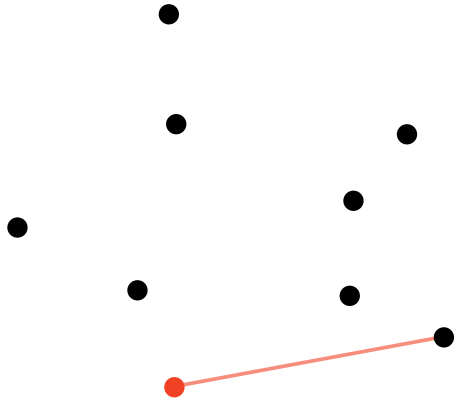
3.1 包装法 (Wrapping method)

■ 基本的な考え方

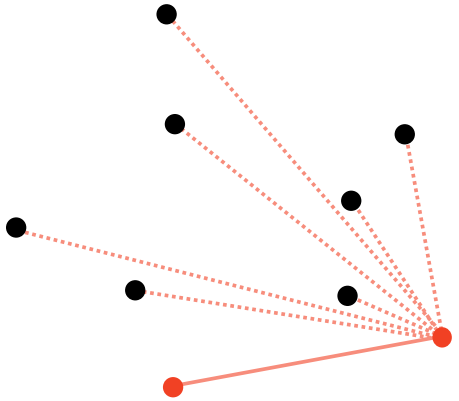
- 見つかった凸包の頂点を，次の頂点を見つける出発点として利用する



実際の凸包



開始点(最小y座標)と1つ目の線分

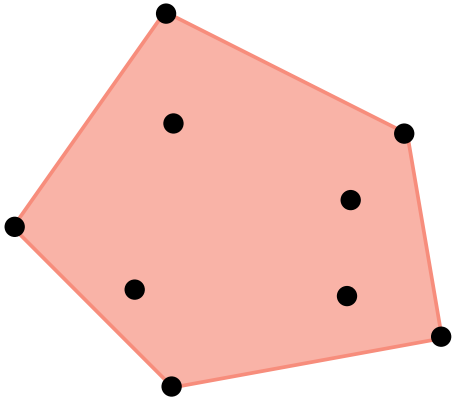


2つ目の線分は1つ目の線分の他方の点と繋がる

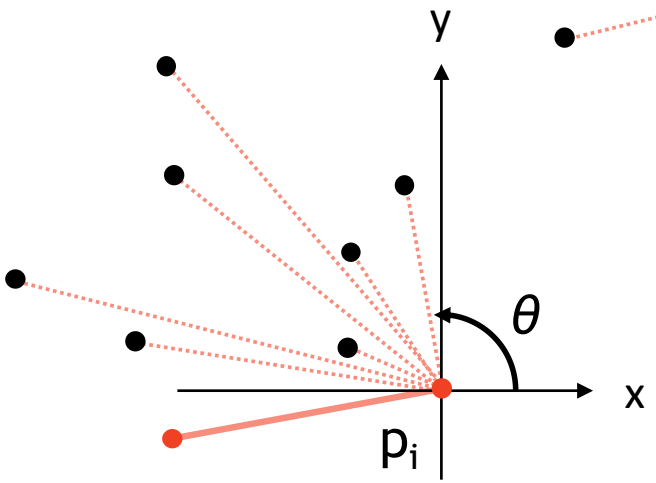
3.1 包装法 (Wrapping method)

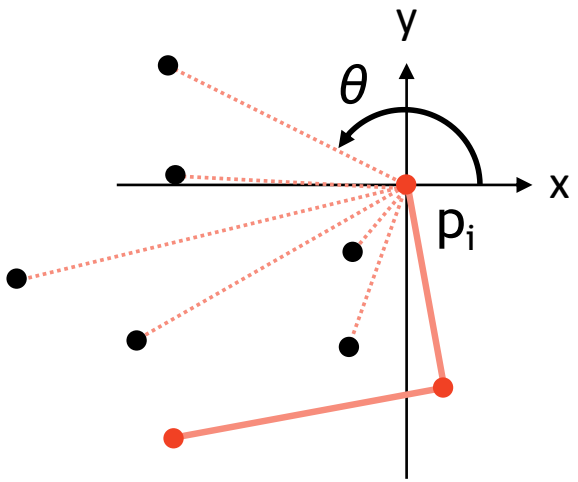
■ 基本的な考え方

- 頂点p_iを原点として，x軸の正の方向からの角度が最小の点を，次の頂点とする



実際の凸包

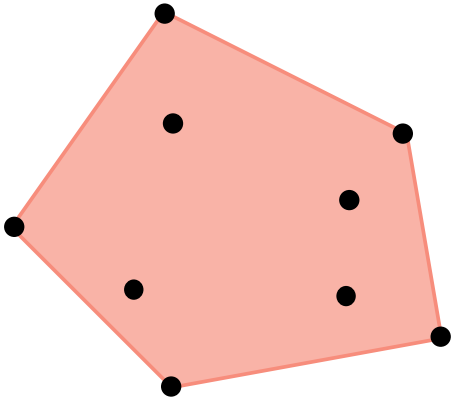




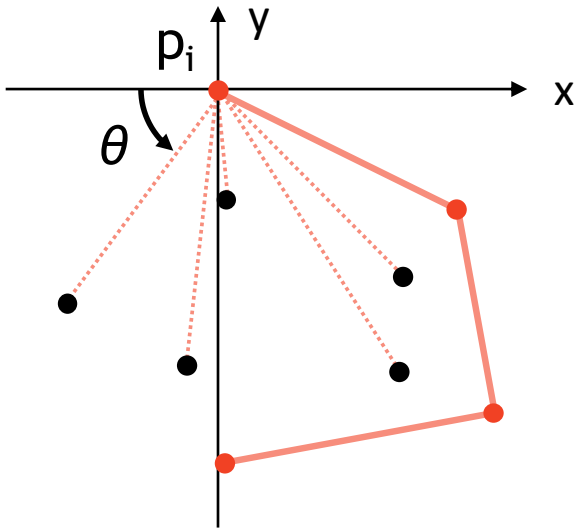
3.1 包装法 (Wrapping method)

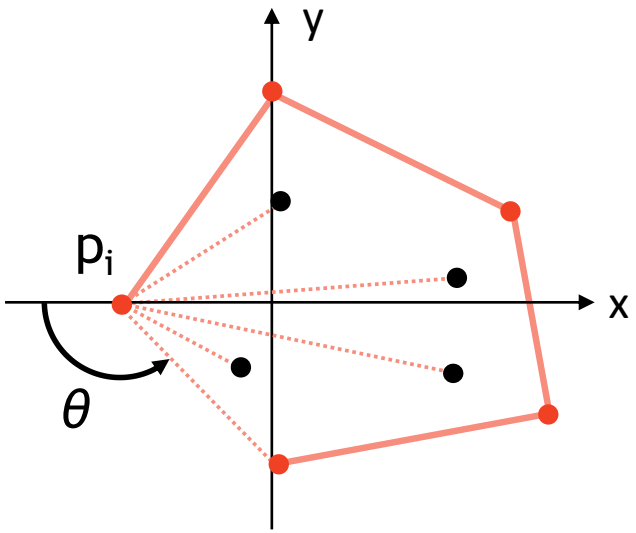
■ 基本的な考え方

- ただし，y座標が最大の頂点が見つかった後は，x軸の負の方向からの角度が最小の点に変更する

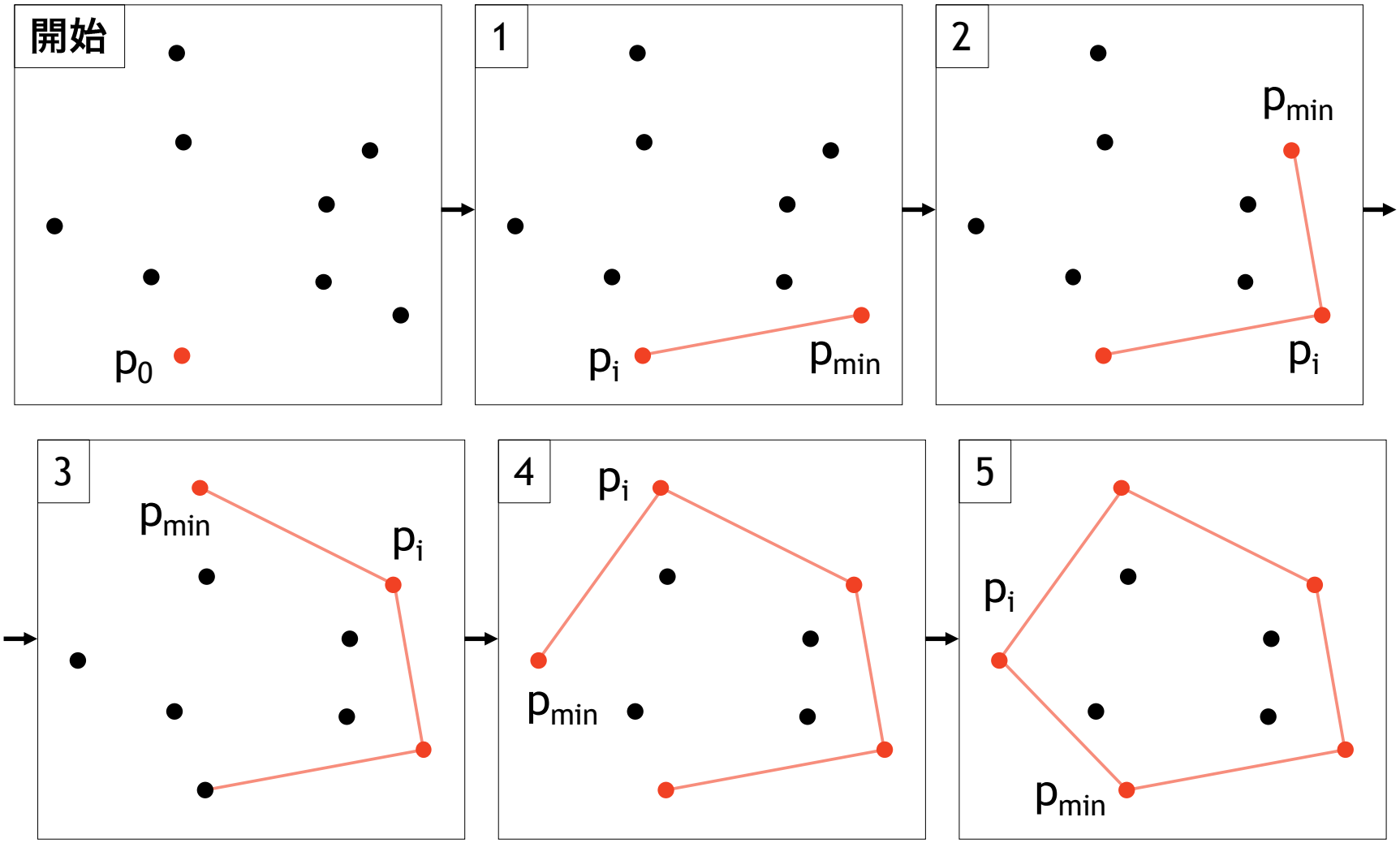


実際の凸包





3.1 包装法 (Wrapping method)

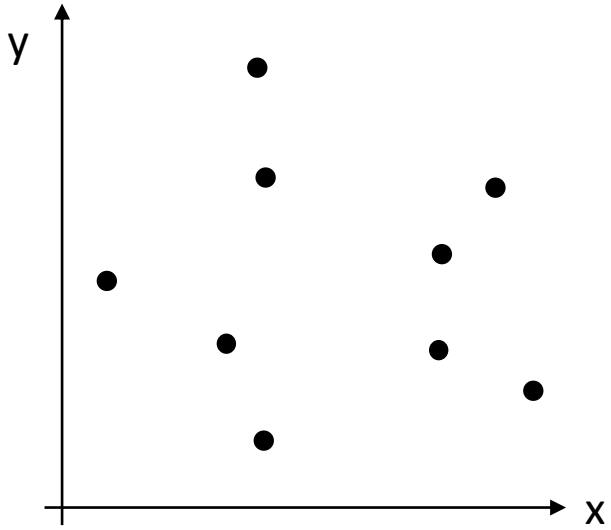


3.1 包装法 (Wrapping method)

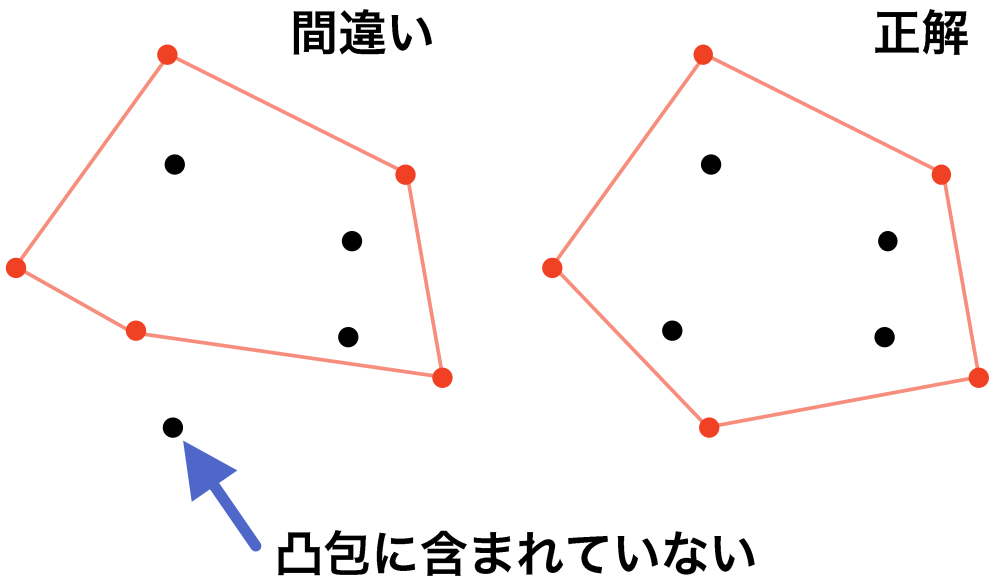
■ アルゴリズム

1. 開始点の探索

- 開始点 = y座標が最小となる点
  (必ず凸包の頂点となるため)

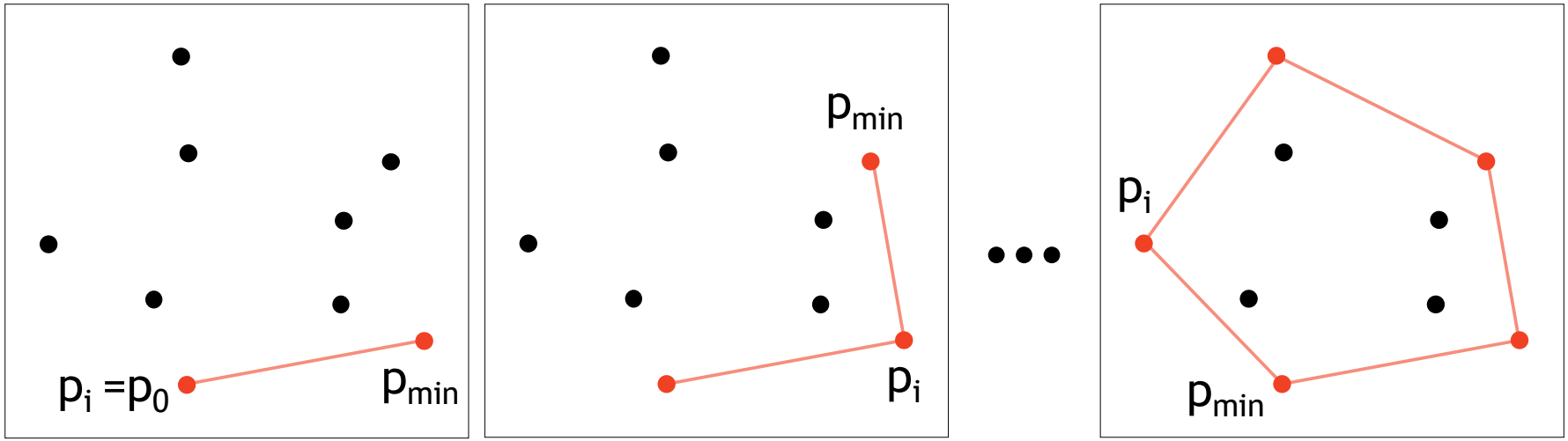


入力点集合



3.1 包装法 (Wrapping method)

2. 開始点p_0をp_iとして，以下を繰り返す
   1. p_iを原点として，偏角が最小の点p_{min}を探す
   2. 点p_{min}を凸包の点，線分(p_i, p_{min})を凸包の線分とする
   3. $p_i \leftarrow p_{min}$として1)に戻る
   4. p_{min}が開始点p_0に一致すれば終了



3.1 包装法 (Wrapping method)

■ 計算量

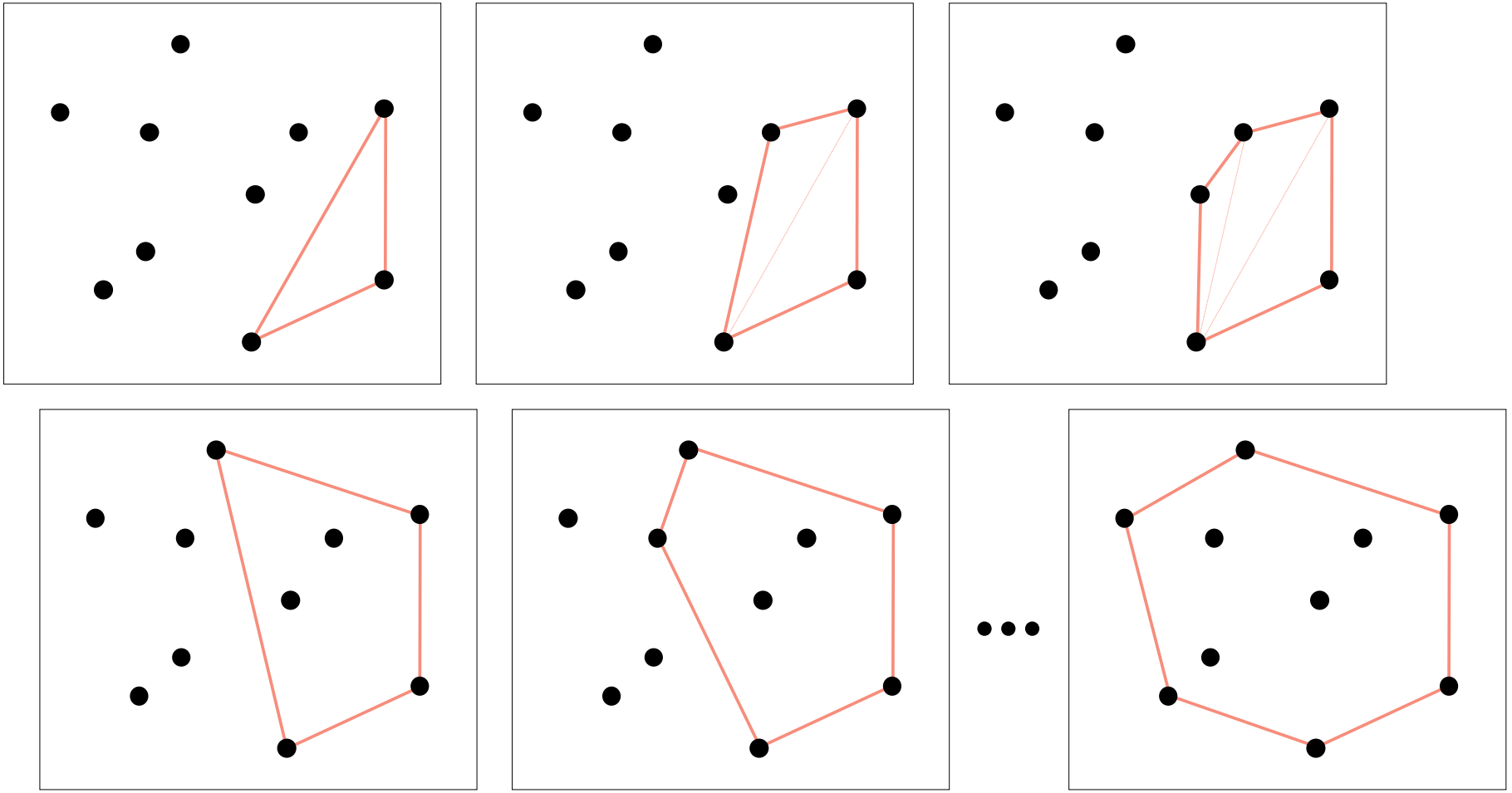
1. 開始点の探索 ← O(n)
   - 開始点 = y座標が最小となる点
2. 開始点p_0をp_iとして，以下を繰り返す ← O(h)
   1. p_iを原点として，偏角が最小の点p_{min}を探す ← O(n)
   2. 点p_{min}を凸包の点，線分(p_i, p_{min})を凸包の線分とする
   3. p_i←p_{min}として1)に戻る
   4. p_{min}が開始点p_0に一致すれば終了

総計算量：O(hn)
(＊hは凸包の点の数)

最悪の場合，総計算量は$O(n^2)$となる．
(h=nとなる場合，つまり，全ての点が凸包の点の場合)

3.2 Graham走査法 (Graham Scan)

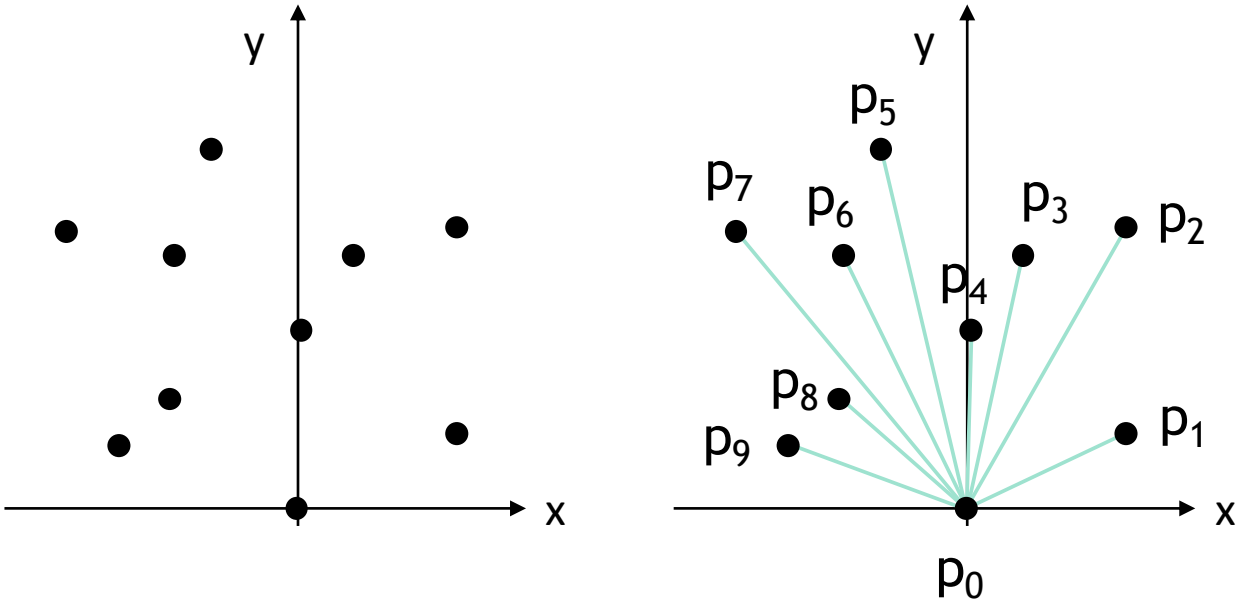
■ 点を1つずつ追加しながら凸包を徐々に作る



3.2 Graham走査法 (Graham Scan)

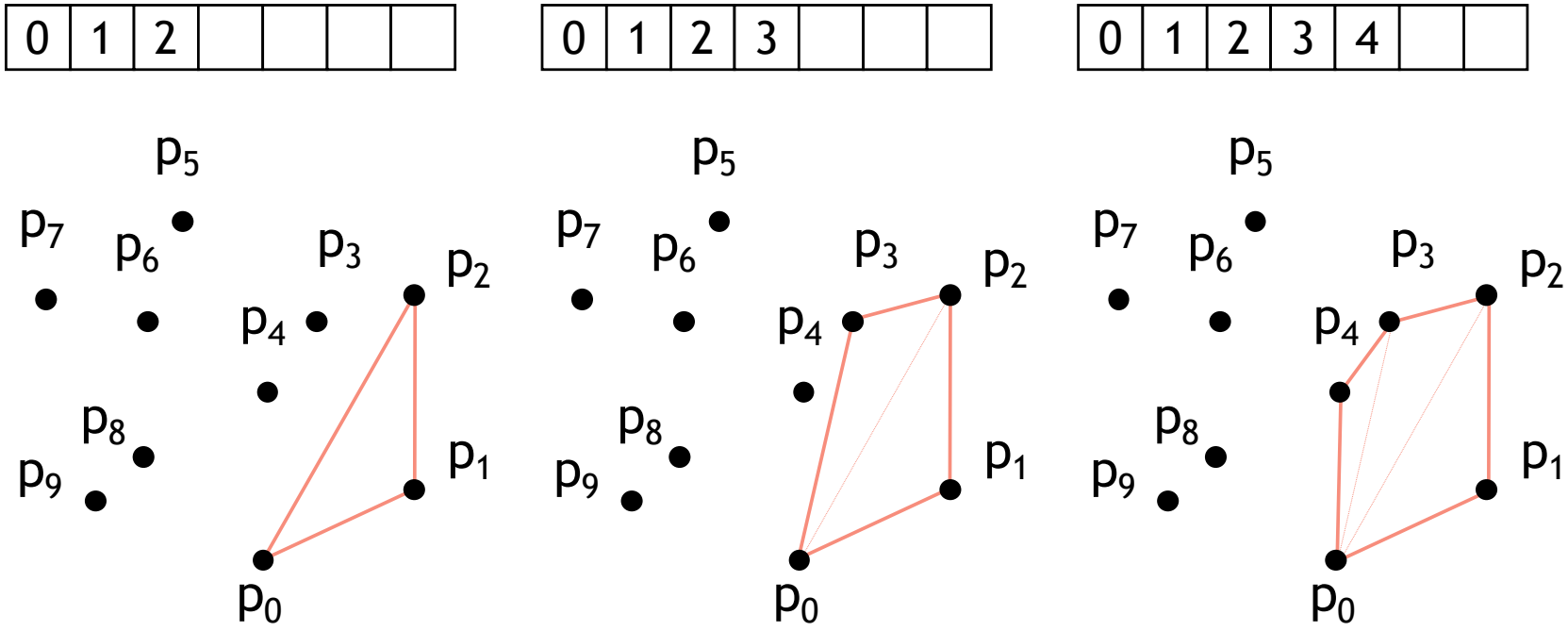
■ 基本的な手順

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)



3.2 Graham走査法 (Graham Scan)

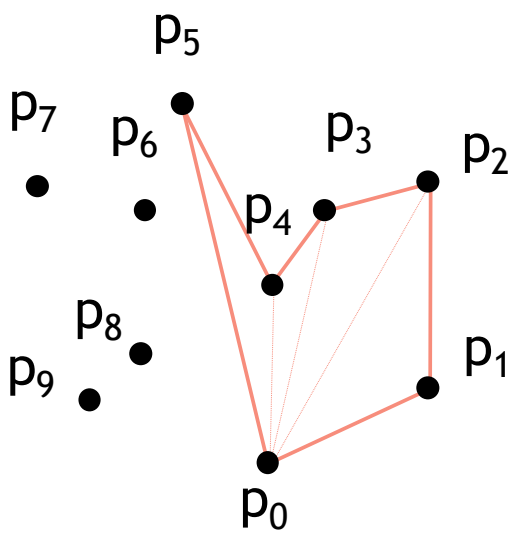
■ 作成途中の凸包はスタックに格納する



3.2 Graham走査法 (Graham Scan)

- 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

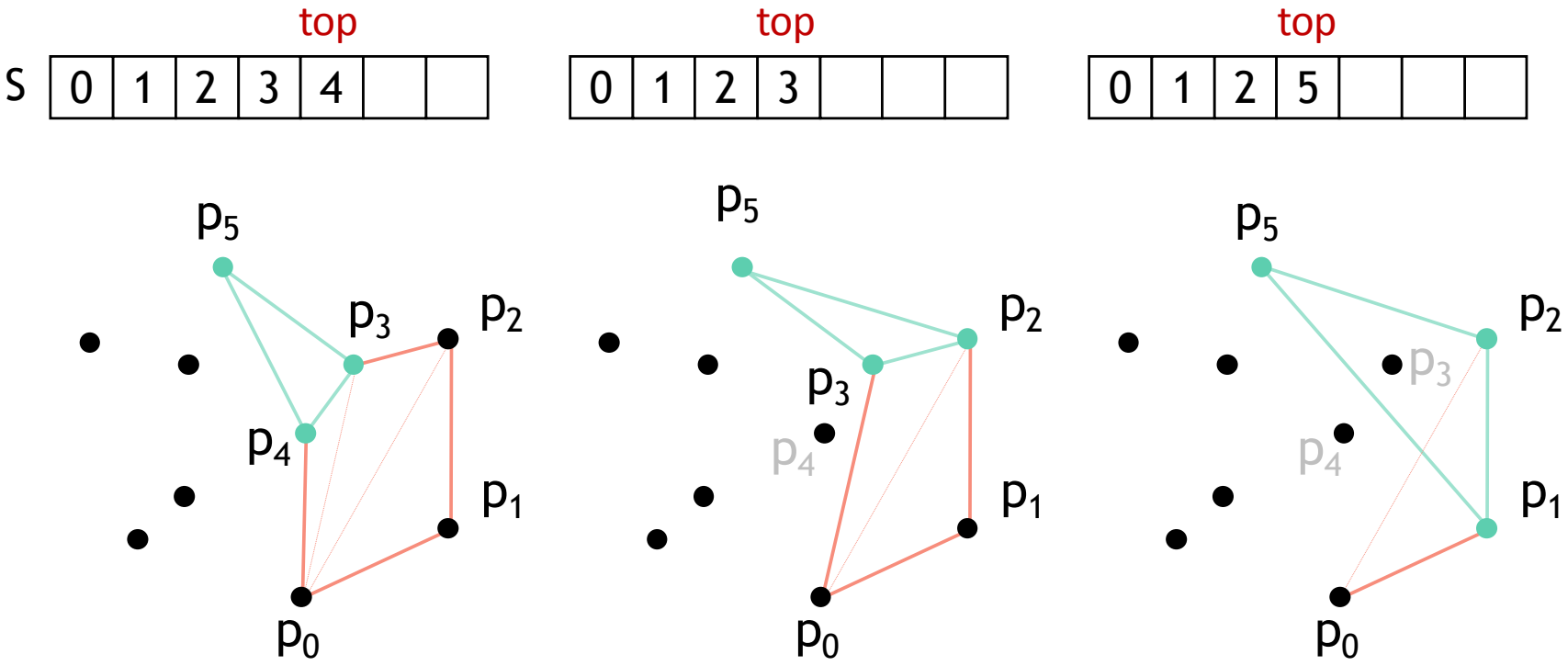
0	1	2	3	4	5	



- **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
  - 時計回りの要因となる頂点を順に削除する

3.2 Graham走査法 (Graham Scan)

■ 削除の方法(新たにp5を追加する場合)



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)

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)

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[top]に
piを追加(図の場合，p5)

3.2 Graham走査法 (Graham Scan)

■ 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