Convex Hull

Degeneracy & Instability

01-XF

Junhui DENG

deng@tsinghua.edu.cn

Degeneracy

- Degenerate geometric data may cause instability or even errors

- For the construction of convex hulls, degenerate cases can occur as:

  a) 3 or more collinear points

  b) 2 or more points lying on a same vertical/horizontal line

  i.e., with the same x/y coordinate

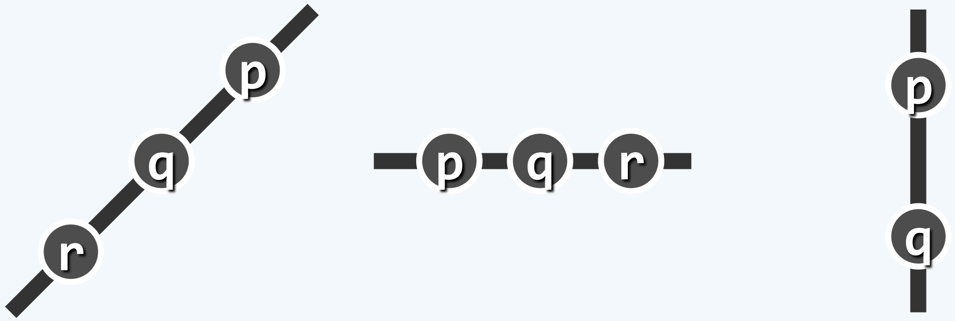
  c) 2 or more coincident points

- (b)\(a) is not difficult to deal with

- In the following pages, let's

  - assume that type (c) cases never occur and

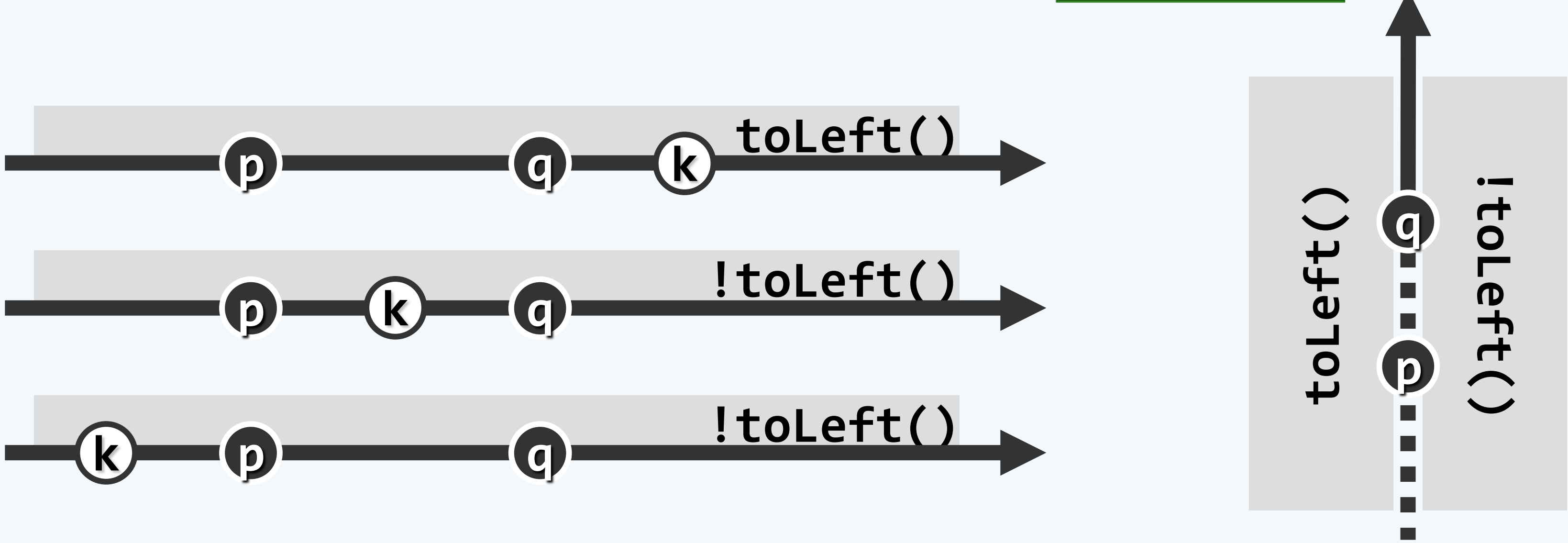
  - consider the solutions for type (a) (including (a)*(b)) cases



(a) (b)*(a) (b)\(a)

Ambiguity of To-left Test

❖ When collinear points exist, how to break ambiguity for toLeft() ?



❖ Here we take the convention that

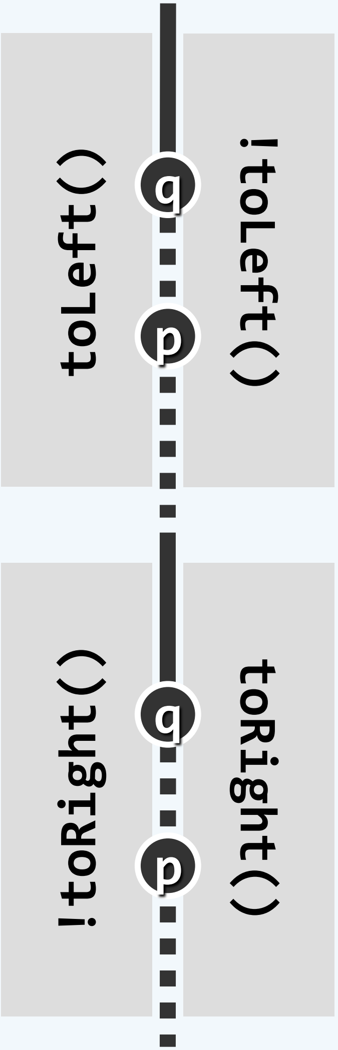
collinear point k lying on the opposite side of p w.r.t. q

belongs to the left region of directed pq

❖ Equivalently, k is classified as a left point if q lies between p and k

Refining To-left Test

```
❖ bool toLeft( Point p, Point q, Point k ) {
      int s = area2( p, q, k );
      if ( s > 0 ) return TRUE;  //left
      if ( s < 0 ) return FALSE; //right
      return between( p, q, k ); //collinear
   }
❖ bool toRight( Point p, Point q, Point k ) {
      int s = area2( p, q, k );
      if ( s > 0 ) return FALSE; //left
      if ( s < 0 ) return TRUE;  //right
      return between( p, q, k ); //collinear
   }
```



Between Test

- For collinear points p, k and q,

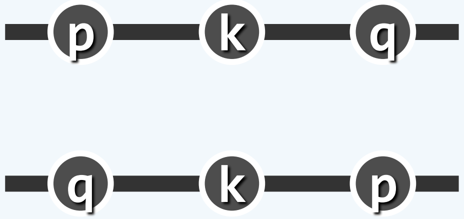
  how to determine if k lies $\boxed{\text{between}}$ p and q?

- Criterion: the $\boxed{\text{dot product}}$ of direct vectors $\overrightarrow{pk}$ and $\overrightarrow{kq}$ is $\boxed{\text{positive}}$

  $$(x_p - x_k, y_p - y_k)(x_k - x_q, y_k - y_q)^T > 0$$

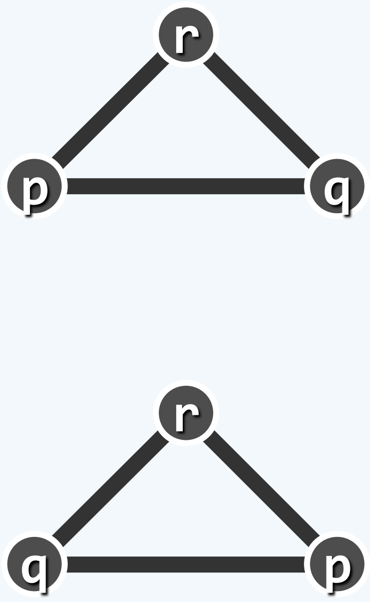
- //determine whether collinear k lies between p and q

```
bool between( Point p, Point k, Point q ) {
   return
      (p.x - k.x) * (k.x - q.x)
    + (p.y - k.y) * (k.y - q.y) > 0;
}
```



Instability in Extreme Point Exclusion

```
❖ void extremePoint( Point * P, int n ) { //n > 2
     /* ...... */
        if ( inTriangle( P[p], P[q], P[r], P[k] ) )
           P[k].extreme = FALSE;
     /* ...... */
  }
```



- EP algorithm should exclude all points covered by a closed triangle pqr

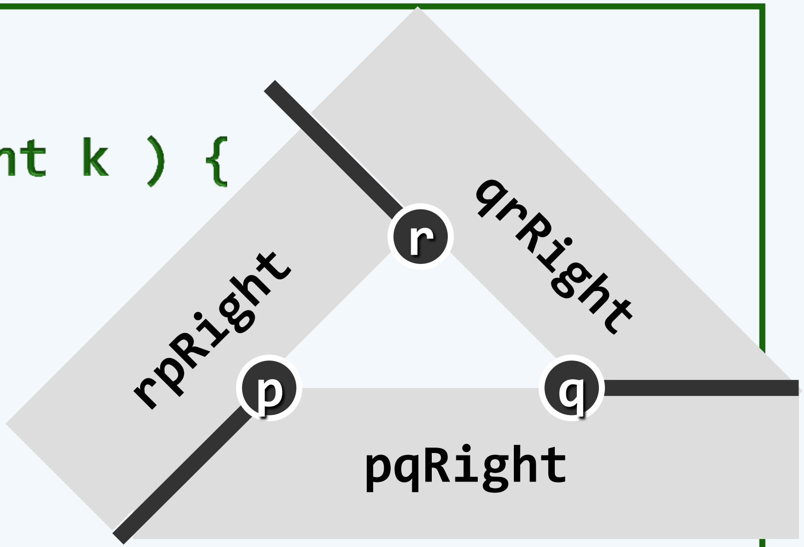
- In other words,

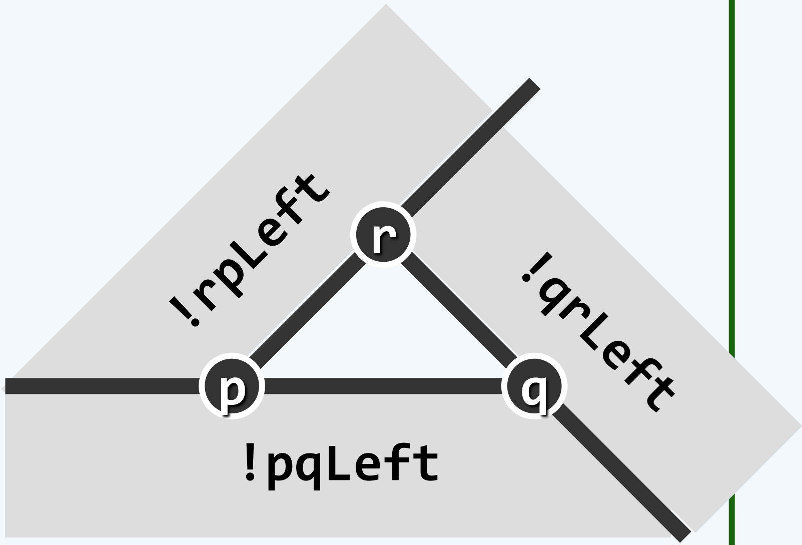
  point k should be classified as non-extreme

  if it lies on segments pq, qr, or rp

Refining In-Triangle Test

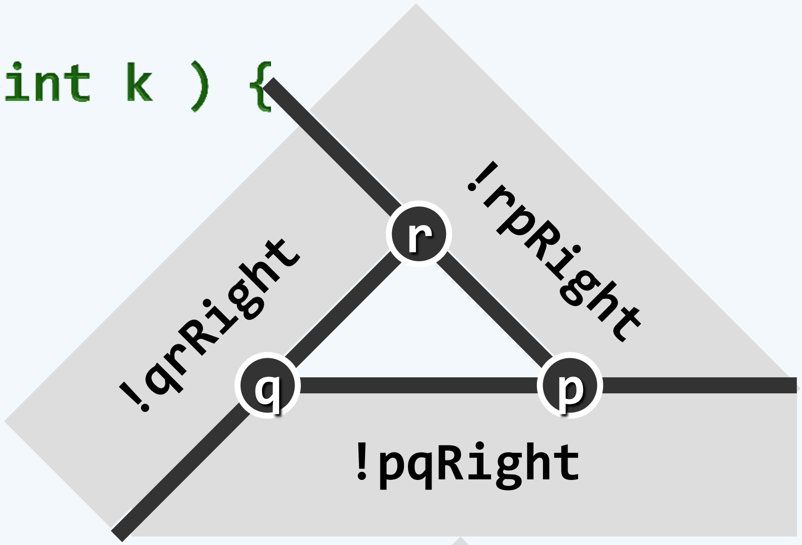
```
❖ bool inTriangle( Point p, Point q, Point r, Point k ) {
     bool pqRight = toRight(p, q, k),
          pqLeft  = toLeft (p, q, k);
     bool qrRight = toRight(q, r, k),
          qrleft  = toLeft (q, r, k);
     bool rpRight = toRight(r, p, k),
          rpLeft  = toLeft (r, p, k);
     return
     ( pqRight == qrRight && qrRight == rpRight
  || ( pqLeft  == qrLeft  && qrLeft  == rpLeft  );
  }
```

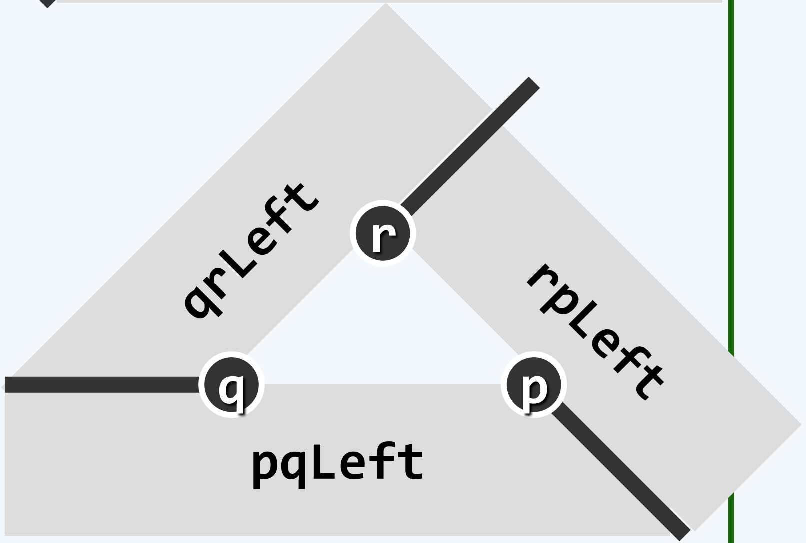




Refining In-Triangle Test

```
❖ bool inTriangle( Point p, Point q, Point r, Point k ) {
    bool pqRight = toRight(p, q, k),
         pqLeft  = toLeft (p, q, k);
    bool qrRight = toRight(q, r, k),
         qrleft  = toLeft (q, r, k);
    bool rpRight = toRight(r, p, k),
         rpLeft  = toLeft (r, p, k);
    return
    ( pqRight == qrRight && qrRight == rpRight
 || ( pqLeft  == qrLeft  && qrLeft  == rpLeft );
  }
```

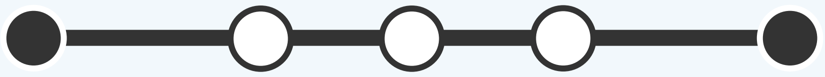




Instability in Extreme Edge Identification

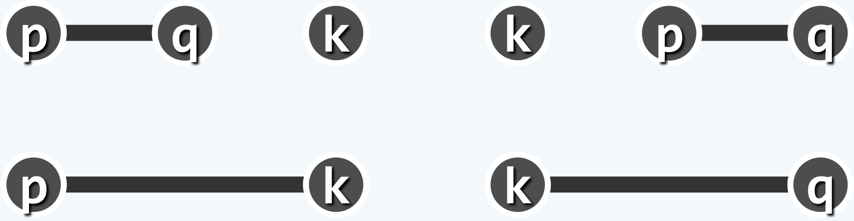
❖ For each group of collinear points, EE algorithm should

- keep only the two endpoints of their convex hull (segment) and
- discard all those lying inside



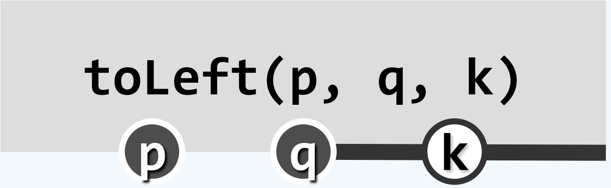
❖ Hence for each EE candidate pq

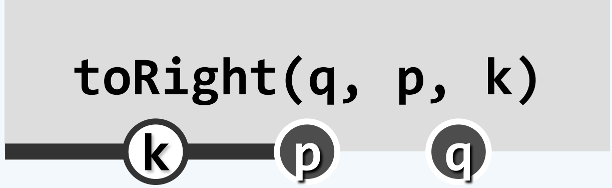
- once a collinear point k is found s.t. q/p lies between p/q and k,
- we should update q/p with k



Refining Extreme Edge Identification

```
❖ void extremeEdge( Point * P, int n ) {
     /* ...... */
     if ( ( k != p ) && ( k != q ) )
        toLeft( P[p], P[q], P[k] ) ? lefFree = FALSE : ritFree = FALSE;
     /* ...... */
  }
```


toLeft(p, q, k)


toRight(q, p, k)

```
❖    if ( ( k != p ) && ( k != q ) ) { //refined version
        if ( toLeft ( P[p], P[q], P[k] ) || toRight( P[q], P[p], P[k] ) )
           lefFree = FALSE;
        if ( toRight( P[p], P[q], P[k] ) || toLeft ( P[q], P[p], P[k] ) )
           ritFree = FALSE;
     }
```

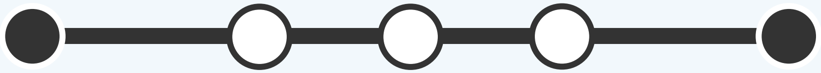
Instability in Gift-Wrapping

❖ Also, for each group of collinear points, GW algorithm should

- keep only the two endpoints

  of their convex hull (segment) and

- discard all those lying inside

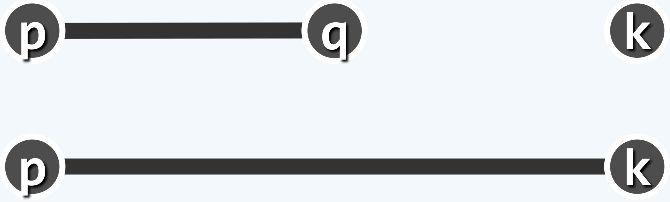


❖ Hence for each EE candidate pq

- once a collinear point k is found

  s.t. q lies between p and k,

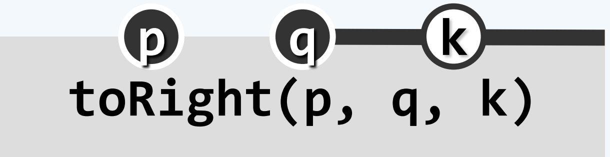
- we should update q with k



❖ Is it possible that p lies between q and k?

Refining Gift-Wrapping

❖
```
void giftWrap( Point * P, int n ) {
   /* ...... */
         if ( ( k != p ) && ( q < 0 || ! toLeft( P[p], P[q], P[k] ) ) )
            q = k; //update q if k lies to right of pq
   /* ...... */
}
```

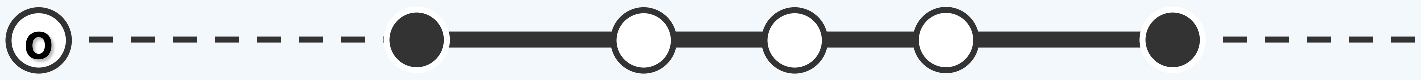


toRight(p, q, k)

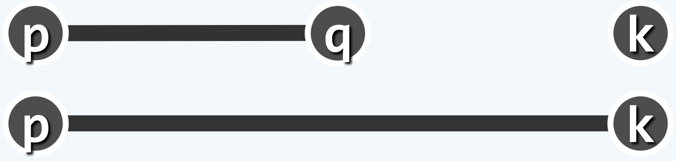
❖
```
         if ( ( k != p ) && ( q < 0 || toRight( P[p], P[q], P[k] ) ) )
            q = k; //update q if k lies to right of pq
```

Instability in Graham Scan

- Again, for each group of collinear points, GS algorithm should keep only the two endpoints of their convex hull (segment) and discard all those lying inside



- Hence for each EE candidate pq once a collinear point k is found such that q lies between p and k, we should update q with k



- Is it possible that p lies between q and k?

Refining Graham Scan

```
❖ while ( ! T.empty() ) do //original version
     ( toLeft( S[1], S[0], T[0] ) ) ?
        S.push( T.pop() ) : S.pop()

❖ while ( ! T.empty() ) do //refined version
     ( toRight( S[1], S[0], T[0] ) ) ?
        S.pop() : S.push( T.pop() )
```