

ĐẠI HỌC THĂNG LONG
Khoa Toán - Tin



BÀI TẬP LỚN

Sử dụng Maple biểu diễn bài
toán tìm bao lồi trên mặt phẳng
với thuật toán chuỗi đơn điệu -
Monotone chain.

Sinh viên thực hiện:	Nguyễn Tú Anh
MSV:	A29888
Môn học:	Thực hành tính toán
Giáo viên hướng dẫn:	Nguyễn Hữu Điển

HÀ NỘI - 2019

LỜI MỞ ĐẦU

Nội dung chủ yếu tập trung vào phần giải thuật và cách cài đặt trên ứng dụng Maple, để giải quyết bài toán tìm tập bao lỗi trên mặt phẳng với thuật toán chuỗi đơn điệu - Monotone chain

Phiên bản sử dụng là Maple 17.

Xin chân thành cảm ơn sự giúp đỡ chỉ bảo tận tình của phó giáo sư Nguyễn Hữu Diễn.

Rất mong được sự góp ý của thầy cô và bạn bè.

Mọi đóng góp xin gửi về địa chỉ email : ntanhtm@gmail.com

Sinh viên trình bày: **Nguyễn Tú Anh**

Lớp : **Toán Tin ứng dụng TM29**

Mục lục

I	Bài toán bao lồi và thuật toán Monotone chain	5
1	Giới thiệu bao lồi 2D	5
1.1	Định nghĩa	5
1.2	Giải thích trực quan về bao lồi trên mặt phẳng	5
2	Thuật toán tìm bao lồi - Monotone chain	6
2.1	Giới thiệu	6
2.2	Nguyên lý	6
2.3	Thuật toán	7
2.4	Giải mã (Pseudo-code)	8
II	Biểu diễn bài toán với Maple	9
1	Mã nguồn	9
1.1	Hàm tìm tập bao lồi	9
1.1.1	Hàm "isCCW"	9
1.1.2	Hàm "ConvexHull"	10
1.2	Hàm khởi tạo dữ liệu thực nghiệm	11
1.3	Hàm biểu diễn thuật toán	12
1.3.1	Hàm "myPlot"	12
1.3.2	Hàm "AniConvexHull"	12
2	Thực nghiệm	14
2.1	Sinh ngẫu nhiên danh sách điểm	14
2.2	Tìm bao lồi	15
2.3	Vẽ ảnh động các bước đi của thuật toán	16
3	Giao diện trong ConvexHullSuper.mw	18
3.1	Mã nguồn	18
3.2	Giao diện chính	19
3.2.1	Tạo danh sách điểm	19
3.2.2	Tìm bao lồi và vẽ đa giác	19
3.2.3	Vẽ ảnh động thể hiện các bước của thuật toán	20

MỤC LỤC	4
4 Kết luận	20
Tài liệu tham khảo	21

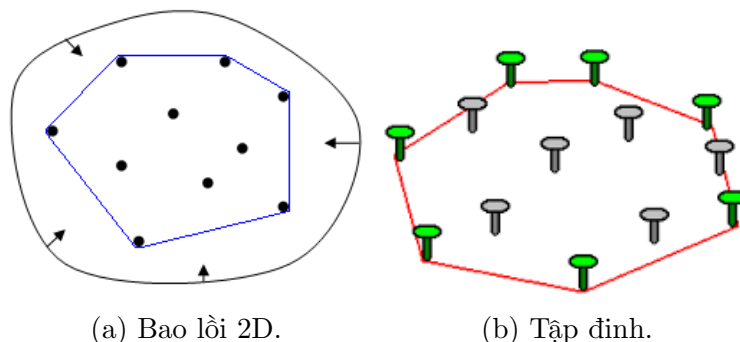
Phần I

Bài toán bao lồi và thuật toán Monotone chain

1 Giới thiệu bao lồi 2D

1.1 Định nghĩa

Trong hình học tính toán (computational geometry), bao lồi (convex hull) của một tập điểm là tập lồi nhỏ nhất (theo diện tích, thể tích, ...) mà tất cả các điểm đều nằm trong tập đó.



Hình 1

1.2 Giải thích trực quan về bao lồi trên mặt phẳng

Sau đây là một số giải thích trực quan về bao lồi trên mặt phẳng.

- Nếu ta coi các điểm trong một tập hợp là các cái đinh đóng trên một tấm gỗ, bao lồi của tập điểm đó có viền ngoài tạo bởi sợi dây chun mắc vào các cái đinh sau khi bị kéo căng về các phía. (Hình 1b)
- Nếu ta coi các điểm trong một tập hợp là các con cừu trên đồng cỏ, bao lồi của tập điểm đó có viền ngoài là hàng rào có độ dài nhỏ nhất bao quanh tất cả các con cừu.
- Nếu ta coi các điểm trong một tập hợp là các đầu mút có thể của các hàng rào, bao lồi của tập điểm đó có viền ngoài là các hàng rào thẳng có điểm đầu và điểm cuối thuộc tập điểm đó và bao quanh diện tích lớn nhất.

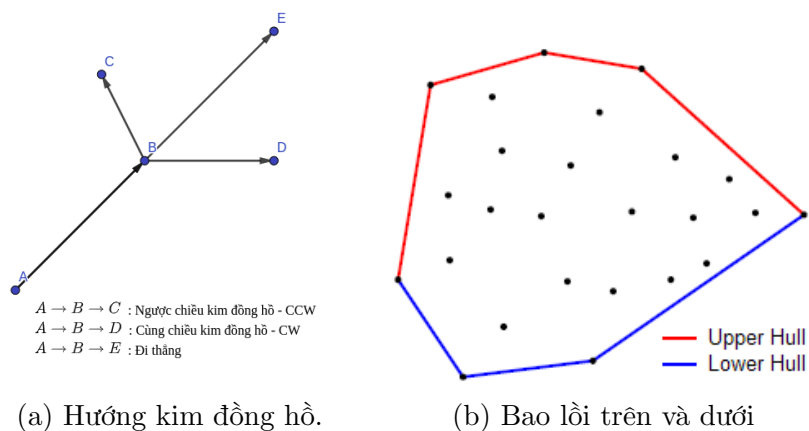
2 Thuật toán tìm bao lồi - Monotone chain

Bài toán tìm bao lồi của một tập điểm trên mặt phẳng là một trong những bài toán được nghiên cứu nhiều nhất trong hình học tính toán và có rất nhiều thuật toán để giải bài toán này. Sau đây là thuật toán tìm bao lồi trên mặt phẳng có tốc độ nhanh và phổ biến nhất - Monotone chain.

2.1 Giới thiệu

Thuật toán Monotone chain của Andrew xây dựng tập điểm bao lồi của một tập điểm trên mặt phẳng (2-dimensional points) với độ phức tạp thuật toán $O(n \log n)$.

2.2 Nguyên lý



Hình 2

Thuật toán dựa trên việc tìm hai chuỗi đơn điệu của bao lồi: chuỗi trên và chuỗi dưới. Ta thấy điểm ở xa về phía bên phải nhất (từ đây gọi là điểm phải nhất) và điểm ở xa về phía bên trái nhất (từ đây gọi là điểm trái nhất) trong dữ liệu vào luôn là hai đỉnh của bao lồi. Phần bao lồi theo chiều kim đồng hồ (Hình 2a) tính từ điểm trái nhất và ngược chiều kim đồng hồ tính từ điểm phải nhất gọi là chuỗi trên, phần còn lại của bao lồi gọi là chuỗi dưới. Ta sẽ tìm chuỗi trên và chuỗi dưới độc lập với nhau.

2.3 Thuật toán

Các bước tiến hành thuật toán Monotone chain như sau:

- Bước đầu tiên là sắp xếp các điểm được cho theo thứ tự tăng dần theo hoành độ. Nếu hai điểm có cùng hoành độ, điểm có tung độ nhỏ hơn sẽ đứng trước. Trong thời gian $O(n \log n)$
- Ta xét việc xây dựng chuỗi trên với độ phức tạp $O(n)$. Gọi H là chuỗi bao lồi trên hiện tại đã tìm được và độ lớn của bao là h . Điểm đầu của chuỗi là H_1 và điểm cuối là H_h . Với mỗi điểm P được xét:

B1. Nếu $h < 2$ xét điểm tiếp theo.

B2. Gọi $\vec{u} = \overrightarrow{H_{h-1}H_h}$ và $\vec{v} = \overrightarrow{H_hP}$. Do ta đang di chuyển theo chiều kim đồng hồ nên để tìm chuỗi trên nên ta kiểm tra xem $\vec{u} \times \vec{v}$ có nhỏ hơn 0 hay không. Nếu có ta xét điểm tiếp theo, nếu không ta loại bỏ H_h quay lại bước B1.

B3. Thêm điểm P vào H

- Việc xây dựng chuỗi bao lồi dưới tương tự. Ta sẽ được kết quả như hình [2b](#)

2.4 Giải mã (Pseudo-code)

Gồm 2 thuật toán chính đó là "Kiểm tra đi ngược chiều kim đồng hồ " (Algorithm 1) và thuật toán tìm tập bao lồi Monotone chain (Algorithm 2). Giải thích rõ hơn sẽ được viết trong phần II.

Algorithm 1 Check Counter-clockwise

```

1: procedure ISCCW( $p_1, p_2, p_3$ )
2:    $d = \overrightarrow{p_1p_2} \times \overrightarrow{p_2p_3}$ 
3:   if  $d > 0$  then return true
4:   else return false

```

Algorithm 2 Convex Hull

```

1: procedure CONVEXHULL(list P of points)
2:   Sort the points of P by x-coordinate (in case of a tie, sort by y-
   coordinate).
3:   Initialize U and L as empty lists.
4:   Constructing upper hulls - U
5:   for  $i = 1..n$  do
6:     while  $\text{sizeof}(U) \geq 2$  do
7:       if  $\text{isCCW}(U[h-1], U[h], P[i]) == \text{true}$  then
8:         Delete the  $h_{th}$  element of U - U[h]
9:       else
10:        Break loop
11:     append P[i] to U
12:   Constructing lower hulls - L
13:   for  $i = n..1$  do
14:     while  $\text{sizeof}(L) \geq 2$  do
15:       if  $\text{isCCW}(L[k-1], L[k], P[i]) == \text{true}$  then
16:         Delete the  $k_{th}$  element of L - L[k]
17:       else
18:        Break loop
19:     append P[i] to L
20:   Remove the first and last point of L.
21:   Concatenate L and U to obtain the convex hull of P.
22:   Points in the result will be listed in counter-clockwise order.

```

Phần II

Biểu diễn bài toán với Maple

1 Mã nguồn

Mã nguồn được viết và thể hiện bằng ứng dụng Maple17. Gồm 2 hàm chính là: "isCCW", "ConvexHull" để tìm tập bao lồi và 3 hàm phụ giúp ta có thực nghiệm rõ ràng hơn.

1.1 Hàm tìm tập bao lồi

1.1.1 Hàm "isCCW"

Hàm kiểm tra hướng đi của vector, được sử dụng trong việc tìm bao lồi trên và dưới.

- Mô tả:
 - Tham số: Gồm 3 điểm - a, b, c với a[1], a[2] lần lượt là tọa độ x, y của điểm a
 - Trả về: boolean (true or false)
 - Độ phức tạp thuật toán: $O(1)$
 - Nhiệm vụ: Kiểm tra hướng đi từ điểm b đến c có đi theo hướng cùng chiều kim đồng hồ hay không ? (Đã đi từ a đến b)
- Code by Maple

```
isCCW := proc(a,b,c)
  #counterclockwise
  local d;
  d:= a[1]*(b[2]-c[2])+b[1]*(c[2]-a[2])+c[1]*(a[2]-b[2]):
  if d > 0 then
    return true;
  else
    return false;
  fi:
end proc;
```

1.1.2 Hàm "ConvexHull"

Hàm tìm tập bao lồi của 1 danh sách điểm đưa vào sử dụng thuật toán Monotone chain với thời gian $O(n \log n)$

- Mô tả:
 - Tham số: Danh sách điểm cần tìm bao lồi (2-D)
 - Trả về: Danh sách điểm nằm trên bao lồi
 - Độ phức tạp thuật toán: $O(n \log n)$
 - Nhiệm vụ: Tìm tập điểm nằm trên bao lồi được liệt kê theo hướng cùng chiều kim đồng hồ
- Code by Maple:

```
ConvexHull := proc(P)
  local res, up, low, nUp, nlow, nPo, i, j, Ps:
  Ps := {op(P)};
  nUp := 0: nlow := 0: nPo := nops(Ps):
  up := vector(nPo): low := vector(nPo):
  for i from 1 to nPo do:
    while nUp >= 2 do:
      if isCCW(up[nUp-1], up[nUp], Ps[i]) then
        nUp := nUp - 1;
      else break; fi;
    od:
    nUp := nUp + 1: up[nUp] := Ps[i]:
  od:
  for i from nPo by -1 to 1 do:
    while nlow >= 2 do:
      if isCCW(low[nlow-1], low[nlow], Ps[i]) then
        nlow := nlow - 1;
      else break: fi:
    od:
    nlow := nlow + 1: low[nlow] := Ps[i]:
  od:
  res := vector(nUp + nlow - 2):
  for i from 1 to nUp do: res[i] := up[i]: od:
  for j from 2 to nlow-1 do: res[j-1+nUp] := low[j]: od:
  return convert(res, list):
end proc:
```

1.2 Hàm khởi tạo dữ liệu thực nghiệm

Hàm "RandPoints"

Hàm tạo tập điểm ngẫu nhiên cho thực nghiệm. Khả dụng sinh 10^6 điểm nhưng với độ phức tạp $O(n \log n)$ của thuật toán chỉ nên tạo 10^5 điểm.

- Mô tả:
 - Tham số: Gồm 3 tham số
 - * n : Số lượng điểm
 - * $range_x$: Khoảng giới hạn của tọa độ x - $(x_{min}..x_{max})$
 - * $range_y$: Khoảng giới hạn của tọa độ y - $(y_{min}..y_{max})$
 - Kiểu trả về: Danh sách n điểm thỏa mãn điều kiện
 - Độ phức tạp thuật toán: $O(n)$
 - Nhiệm vụ: Sinh n điểm ngẫu nhiên
- Code by Maple

```
RandPoints := proc(n,range_x,range_y)
  local P,roll, i, x, y:
  P:= vector(n):
  d:= 1:
  roll_x := rand(range_x);
  roll_y := rand(range_y);
  for i from 1 to n do:
    x:= roll_x():
    y:= roll_y():
    P[d] := [x,y]:
    d:= d+1:
  end do:
  return convert(P,list):
end proc:
```

1.3 Hàm biểu diễn thuật toán

1.3.1 Hàm "myPlot"

Hàm làm nhiệm vụ đơn giản nhưng phải có vì cần sử dụng trong hàm "animate" của Maple giúp vẽ hình ảnh động.

- Mô tả:
 - Tham số: 1 số nguyên - i và danh sách plots - listPlots
 - Kiểu trả về: plot
 - Độ phức tạp thuật toán: $O(1)$
 - Nhiệm vụ: Lấy plot thứ i của danh sách listPlots
- Code by Maple

```
myPlot := (i,listPlots) -> listPlots[trunc(i)];
```

1.3.2 Hàm "AniConvexHull"

Hàm tạo ảnh động biểu diễn thuật toán. Vì đây là ảnh động biểu diễn trực quan nên chỉ khả dụng với tập điểm với số lượng điểm **nhỏ hơn 50** !

Tại mỗi bước của thuật toán, ta tạo 1 plot thể hiện bước đó, xong tập hợp các plot lại và sử dụng hàm animate của maple với hàm "myPlot" vừa tạo ở trên để vẽ được ảnh động.

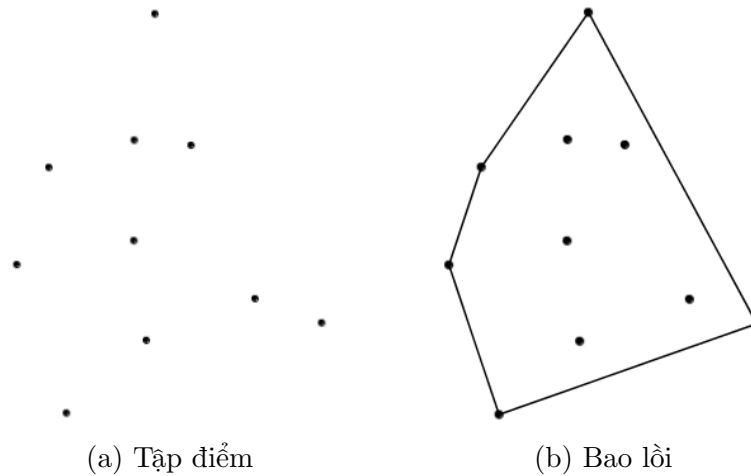
- Mô tả:
 - Tham số: Danh sách điểm - P, Tùy chọn Axes và Scaling
 - Kiểu trả về: 3 plot (2 plot là animate)
 - Nhiệm vụ: Tạo ảnh động thể hiện các bước đi của thuật toán
- Code by Maple

```
AniConvexHull := proc(P,Axes,Scaling)
  local res, up, low, nUp, nlow, nPo, nPl, i, j, Ps, p, l,
    aniUp, aniLow:
  Ps:= {op(P)}: nPo:= nops(Ps):
  up:= vector(nPo): low:= vector(nPo):
  Plots:= vector(5*nPo): nPl:= 0:
  p:= pointplot([Ps[1],Ps[nPo]], color= red,symbol= solidcircle,
    symbolsize = 30, axes = Axes, scaling = Scaling):
  bg:= display(pointplot(P,symbol= solidcircle,
    symbolsize= 15,axes= Axes, scaling= Scaling),p):
  Plots[1]:= p: nPl:= 1:
```

```

#-----Up-----#
up[1] := Ps[1]: nUp:= 1:
for i from 2 to nPo do:
  p := display(p,line(up[nUp],Ps[i],color = red, thickness= 3));
  nPl := nPl + 1: Plots[nPl] := p:
  while nUp >= 2 do:
    if isCCW(up[nUp-1],up[nUp],Ps[i]) then
      l:= line(up[nUp],Ps[i],color= gray, thickness=3);
      p:= display(p,l): nPl:= nPl+ 1: Plots[nPl]:= p:
      l:= line(up[nUp-1],up[nUp],color= gray, thickness=3);
      p:= display(p,l): nPl:= nPl+ 1: Plots[nPl]:= p:
      l:= line(up[nUp-1],Ps[i],color = red, thickness=3);
      p:= display(p,l): nPl:= nPl + 1: Plots[nPl]:= p:
      nUp := nUp - 1:
    else break: fi:
  od:
  nUp:= nUp + 1: up[nUp]:= Ps[i]:
od:
aniUp := animate(myPlot,[t,Plots],t=1..nPl,background=bg,
  frames = nPl, title="Convex Hull - Up");
#-----low-----#
p := pointplot([Ps[1],Ps[nPo]],color= red,symbol= solidcircle,
  symbolsize= 30, axes= Axes, scaling= Scaling):
Plots[1] := p: nPl := 1:
low[1] := Ps[nPo]: nlow := 1:
for i from nPo - 1 by -1 to 1 do:
  p:= display(p, line(low[nlow],Ps[i],color= red,thickness=3));
  nPl := nPl + 1: Plots[nPl] := p:
  while nlow >= 2 do:
    if isCCW(low[nlow-1],low[nlow],Ps[i]) then
      l:= line(low[nlow],Ps[i],color= gray, thickness= 3);
      p:= display(p,l): nPl:= nPl + 1: Plots[nPl]:= p:
      l:= line(low[nlow-1],low[nlow],
        color= gray, thickness= 3);
      p:= display(p,l): nPl:= nPl + 1: Plots[nPl]:= p:
      l:= line(low[nlow-1],Ps[i],color= red, thickness= 3);
      p:= display(p,l): nPl:= nPl + 1: Plots[nPl]:= p:
      nlow := nlow - 1:
    else break: fi:
  od:
  nlow := nlow + 1: low[nlow]:= Ps[i]:
od:
aniLow := animate(myPlot,[t,Plots],t=1..nPl,background=bg,
  frames = nPl,title="Convex Hull - low");
res := vector(nUp + nlow - 2):
for i from 1 to nUp do: res[i] := up[i]: od:
for j from 2 to nlow-1 do: res[j - 1 + nUp]:= low[j]: od:
poly := polygon(convert(res,list),linestyle = solid,
  color = "white",thickness= 3);
return [aniUp,aniLow,display(poly,bg)]:
end:

```



Hình 3: Thực nghiệm

2 Thực nghiệm

Sau đây, ta sẽ sử dụng các hàm khởi tạo ở trên để thực nghiệm giải quyết bài toán tìm tập bao lồi với thuật toán Monotone chain bằng phần mềm Maple 17.

2.1 Sinh ngẫu nhiên danh sách điểm

Lệnh

```
>Points := RandPoints(10,-800 .. 900, -1000 .. 1200): [1]
>display(point(Points,symbol = solidcircle,
symbolsize = 15),axes = Axes, scaling = Scaling)); [2]
```

Giải thích

- 1 : Sinh ngẫu nhiên 10 điểm với tọa độ x nằm trong khoảng -800 đến 900 và tọa độ y trong khoảng -1000 đến 1200 sau đó gán vào biến Points.
- 2 : Vẽ plot các điểm vừa tạo. Với Axes = 'none', Scaling = 'constrained'.

Kết quả

Kết quả ta sẽ có 1 plot thể hiện 10 điểm được sinh ngẫu nhiên như hình [3a](#)

2.2 Tìm bao lồi

Lệnh

```
>p:= ConvexHull(Points): [1]
>poly := polygon(p,linestyle = solid, color = "White" ): [2]
>display(poly,point(Points,symbol = solidcircle,
symbolsize = 15),axes = Axes, scaling = Scaling)); [3]
```

Giải thích

- 1 : Thực hiện thuật toán trả về tập bao lồi của danh sách điểm Points vừa tạo ở trên gắn vào biến p
- 2 : Do danh sách điểm được liệt kê theo chiều kim đồng hồ, nên ta dễ dàng vẽ được 1 đa giác từ tập điểm đó dùng hàm "polygon" trong Maple.
- 3 : Dùng "display" để thể hiện đa giác vừa tìm được và tập điểm ban đầu.

Kết quả

Kết quả giống ta sẽ có 1 đa giác lồi bao 10 điểm đã tạo ở trên như hình [3b](#)

2.3 Vẽ ảnh động các bước đi của thuật toán

Lệnh

```
>listPlots := AniConvexHull(Points,Axes,Scaling): [1]  
>listPlots[1]; listPlots[2]; [2]
```

Giải thích

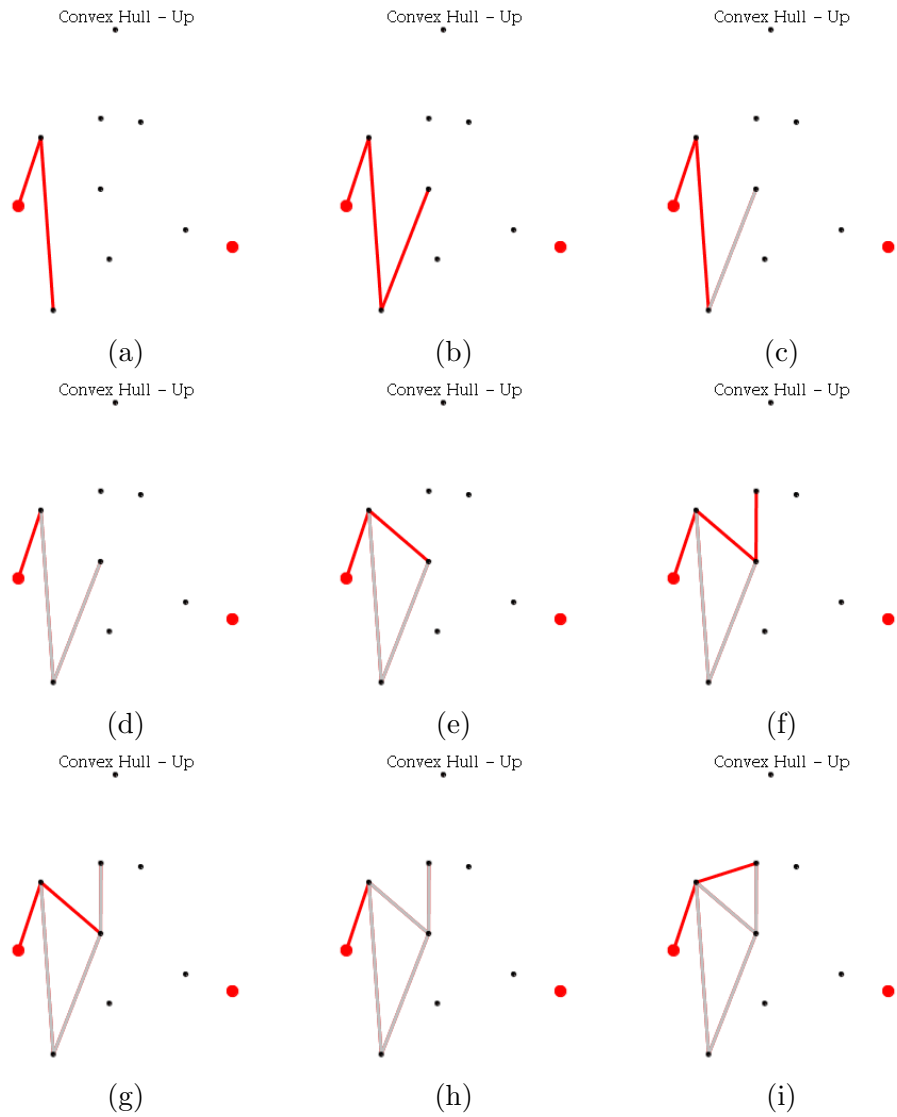
- 1 : Sử dụng hàm "AniConvexHull" để vẽ ảnh động cho bước đi của thuật toán với tập điểm Points và 2 tùy chọn của plot
- 2 : Thể hiện 2 plot bao lồi trên và dưới.
- 3 : Điểm màu đỏ là 2 điểm ở biên được chọn theo thuật toán. Đoạn thẳng màu đỏ là đoạn thẳng được chọn nối các điểm trong tập bao lồi còn đoạn thẳng màu xám thì ngược lại.

Kết quả

Chú ý: Ảnh động khả dụng khi đọc bởi "Adobe Reader" !

Một số hình ảnh

Liệt kê các bước thuật toán khi thực hiện tìm bao lồi bên trên theo thứ tự a, b, c .. được dán nhãn như hình dưới.



Hình 4: Bước đi của thuật toán

3 Giao diện trong ConvexHullSuper.mw

3.1 Mã nguồn

Mã nguồn được viết trong phần "Startup Code" (Ctrl + Shift + E) của file.

```

1 #-----#
2 with(plottools):
3 with(plots):
4 #-----#
5 Points := []:
6 RandPoints := proc(n, range_x, range_y)
7
8     local P, roll, i, x, y:
9     P:= vector(n): d:= 1:
10    roll_x := rand(range_x);
11    roll_y := rand(range_y);
12    for i from 1 to n do:
13        x:= roll_x(): y:= roll_y():
14        P[d] := [x,y]: d:= d+1:
15    end do:
16    return convert(P,list):
17 end proc:
18 #-----#
19 isCCW := proc(a,b,c) #a -> b -> c có đi ngược chiều kim đồng hồ không ?
20     #counterclockwise
21     local d;
22     d := a[1]*(b[2] - c[2]) + b[1]*(c[2] - a[2]) + c[1]*(a[2]-b[2]):
23     if d >= 0 then return true; else return false; fi:
24 end proc:

```

No errors

3.2 Giao diện chính

Giao diện được viết bằng DocumentTools[Components] của Maple

3.2.1 Tạo danh sách điểm

Sau khi điền các tùy chọn thì kích vào nút "Update Plot" để thực hiện khởi tạo tập điểm như mong muốn.

Create list of points

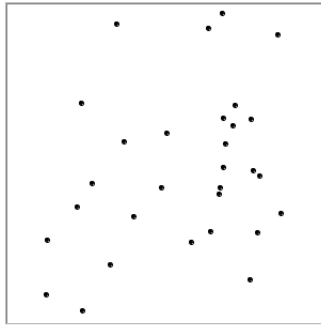
<input checked="" type="radio"/> Random data	Number of Points : <input type="text" value="30"/> Rang of X : From <input type="text" value="-800"/> to <input type="text" value="900"/> Rang of Y : From <input type="text" value="-1000"/> to <input type="text" value="1200"/>
<input type="radio"/> Input data	<input style="width: 100%;" type="text" value="[96, -74], [-5, -84], [-11, 61], [-83, 79], [-45, 70], [0, 0], [1, 90]"/>

Options: ☐ **Axes** ☒ **Scaling constrained**

Update Plot

List of points:

```
[[488, 1200], [511, 292], [704, 105], [872, 1051], [497, 467], [747, 71], [582, 555], [477, -11], [-51, 462], [64, -9], [467, -56], [-249, 1119], [276, -393], [-291, -547], [564, 421], [-482, -862], [-728, -37], [896, -190], [-419, 17], [-193, 303], [679, -647], [-125, -214], [409, -318], [499, 125]]
```



3.2.2 Tìm bao lồi và vẽ đa giác

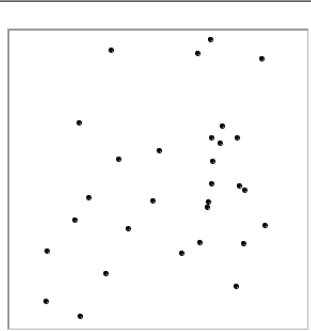
Kích vào nút "Start Convex Hull" để bắt đầu tìm bao lồi từ tập điểm khởi tạo ở trên.

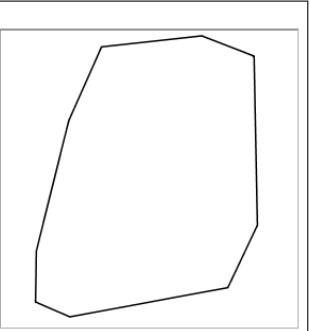
Get polygon

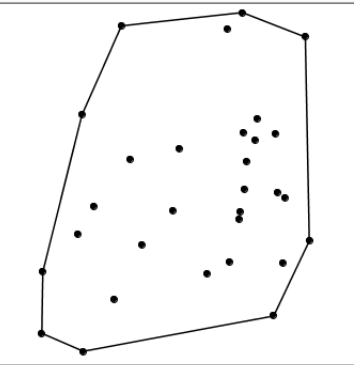
Start Convex Hull

The points belong to convex:

```
[[[-734, -753], [-728, -378], [-490, 576], [-249, 1119], [488, 1200], [872, 1051], [896, -190], [679, -647], [-482, -862]]]
```

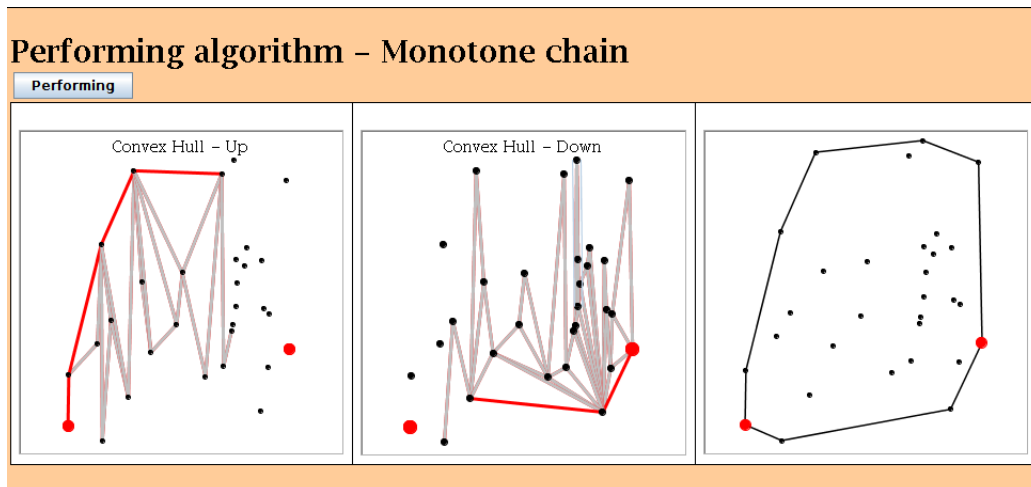






3.2.3 Vẽ ảnh động thể hiện các bước của thuật toán

Kích vào nút "Performing" để vẽ ảnh động.



4 Kết luận

- Thuật toán chuỗi đơn điệu (Monotone chain) vừa dễ cài đặt, vừa là thuật toán nhanh nhất trong các thuật toán tìm bao lồi trên mặt phẳng (Quan điểm cá nhân).
- Maple là phần mềm toán học kết hợp công cụ toán học mạnh nhất thế giới với giao diện giúp dễ dàng phân tích, khám phá, trực quan hóa và giải quyết các vấn đề toán học. Dễ dàng thể hiện các bước thuật toán, và rất dễ lập trình. Mọi thứ trở lên trực quan hơn và rất dễ hiểu.

Tài liệu

- [1] Nguyễn Hữu Điển *Hướng dẫn và sử dụng Maple V*, (1999) NXB Thống Kê.
- [2] Nguồn tài liệu trên mạng:
 - <http://vnoi.info/wiki/translate/wcipeg/Convex-Hull>
 - https://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain
 - <https://www.maplesoft.com/support/help/>