* **[Trang chủ](http://www.chuyennguyentrai.edu.vn/)**
* [**Giới thiệu**](http://www.chuyennguyentrai.edu.vn/category/phong-truyen-thong.html)
* [**Tin tức**](http://www.chuyennguyentrai.edu.vn/category/tin-tuc-su-kien.html)
* [**Giáo dục**](http://www.chuyennguyentrai.edu.vn/category/giao-duc-du-hoc.html)
* [**25 Năm**](http://www.chuyennguyentrai.edu.vn/category/25_nam_truong_nguyen_trai.html)
* [**Trang nội bộ**](http://www.chuyennguyentrai.edu.vn/category/tin-noi-bo.html)
* [**Thư điện tử**](http://mail.chuyennguyentrai.edu.vn/)
* [**Diễn đàn**](http://www.nguyentraionline.org/)
* [**Liên hệ**](http://www.chuyennguyentrai.edu.vn/contact.html)

**Thứ hai, ngày 11/03/2013**



|  |
| --- |
| **THƯ VIỆN BÀI GIẢNG »**[**TIN HỌC**](http://www.chuyennguyentrai.edu.vn/category/tvbt-tin-hoc.html)**»**  Số lượt xem: 4810  Gửi lúc 23:41' 12/07/2009    **BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHẤT**  **BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHẤT**  **A. Trích "Cẩm nang thuật toán"-Tập 2**  Bài toán cặp điểm gần nhất tìm hai điểm gần nhau nhất trong một tập hợp điểm cho trước. Bài toán này có quan hệ với bài toán tìm láng giềng gần nhất; mặc dù không được ứng dụng rộng rãi bằng.  Có vẻ như ta phải tính khoảng cách giữa mọi cặp điểm để tìm khoảng cách nhỏ nhất: Cho N điểm sẽ có thời gian thực hiện tỷ lệ với N2. Tuy nhiên ta có thể dùng một thuật toán sắp xếp để chỉ phải tính chừng NlogN khoảng cách trong trường hợp xấu nhất (về mặt trung bình sẽ ít hơn nhiều) và thời gian thực hiện trong trường hợp xấu nhất là NlogN (về mặt trung bình sẽ tốt hơn nhiều). Dưới đây, ta sẽ xem xét chi tiết một thuật toán như vậy.  Ta sẽ dùng một thuật toán dựa trên chiến lược "chia để trị". Ý tưởng ở đây là sắp xếp các điểm theo một trục toạ độ, như trục x chẳng hạn, rồi dùng thứ tự này chia tập điểm thành hai phần. Trong toàn bộ tập điểm đã cho, cặp điểm gần nhất hoặc là cặp gần nhất trong cùng một bên nào đó, hoặc là một cặp điểm cắt ngang đường thẳng phân giới giữa hai tập điểm thành phần. Dĩ nhiên trường hợp đáng chú ý là khi cặp điểm gần nhất cắt ngang đường phân giới. Cặp điểm gần nhất trong mỗi nửa bên rõ ràng là tìm được bằng các lời gọi đệ qui, nhưng còn những cặp có mỗi điểm ở một bên đường phân giới sẽ được kiểm tra như thế nào?  Vì đang tìm cặp điểm gần nhất trong một tập hợp điểm, ta chỉ cần xét những điểm nằm trong khoảng cách min ở hai bên đường phân giới, với min là khoảng cách nhỏ hơn giữa các cặp điểm gần nhất ở mỗi nửa bên. Tuy nhiên, sự nhận xét này là không đủ trong trường hợp xấu nhất, vì có thể có nhiều cặp điểm gần với đường thẳng phân giới; ví dụ tất cả các điểm ở mỗi nửa bên có thể sắp thành một hàng ngay cạnh đường thẳng phân giới.  Để xử lý những tình huống như vậy, có vẻ như cần phải sắp xếp các điểm theo y. Vậy thì, ta có thể giới hạn số các khoảng cách phải tính như sau: xứ lý các điểm theo chiều tăng của y, kiểm tra xem mỗi điểm có nằm trong dải đứng chứa các điểm trong phạm vi min kể từ đường phân giới. Với mỗi điểm trong dải trên, tính khoảng cách giữa điểm này với các điểm trong dải và có tung độ y nhỏ hơn tung độ của đểm đang xét nhưng không nhỏ quá min. Do khoảng cách giữa các điểm ở mỗi nửa bên tối thiểu là min nên số điểm phải kiểm tra sẽ ít hơn.  Mặc dù thuật toán này đơn giản, có một vài chú ý để cài đặt có hiệu quả. Ví dụ, có thể sẽ trả giá đặt để sắp xếp các điểm theo y bằng một thủ tục đệ qui. Ta đã xét nhiều thuật toán có thời gian thi hành được mô tả bằng biểu thức qui nạp CN=2CN/2+N, dẫn đến CN tỷ lệ với NlogN; nếu ta sắp xếp theo y thì biểu thức qui nạp sẽ là CN=2CN/2+NlogN, dẫn đến CNtỷ lệ với Nlog2N. Để loại bỏ điều này, ta tránh sắp xếp theo y.  Lời giải vấn đề này thì đơn giản nhưng tinh vi. Phương pháp *mergesort* dựa trên việc chia những phần tử được sắp, cũng giống như việc chia các điểm nêu trên. Ta có hai vấn đề và có chung một lời giải cho chúng, như vậy ta có thể giải chúng đồng thời !. Đặc biệt là ta sẽ viết một thủ tục đệ qui để vừa sắp xếp theo y và vừa tìm cặp điểm gần nhất. Thủ tục này sẽ chia đôi tập điểm, rồi gọi lại chính nó để sắp xếp hai nửa bên theo y và tìm cặp điểm gần nhất trong mỗi nửa, sau đó trộn hai nửa bên để hoàn tất việc tính cặp điểm gần nhất. Trong cách này, ta tránh được chi phí cho sự thêm việc sắp xếp theo y bằng cách trộn dữ liệu được yêu cầu trong việc sắp xếp với dữ liệu được yêu cầu khi tính cặp điểm gần nhất:  Khi sắp xếp theo y, việc chia đôi có thể được làm bằng bất kỳ cách nào, nhưng với phép tính cặp điểm gần nhất, việc chia đôi yêu cầu có một nửa bên có hoành độ x nhỏ hơn nửa bên còn lại. Điều này được cài đặt dễ dàng bằng cách sắp xếp theo x trước khi chia đôi tập điểm. Thực tế, ta có thể dùng cũng thủ tục này để sắp xếp theo x! Khi đã hiểu dự định tổng thể, việc cài đặt không còn khó khăn nữa.  Như đã lưu ý ở trên, việc cài đặt sẽ dùng các thủ tục sort và merge trong thuật toán sắp xếp trộn. Bước đầu tiên là thay đổi các cấu trúc danh sách để lưu các điểm thay cho các khoá, và thay đổi thủ tục merge để kiểm tra biến toàn cục pass dùng cho việc chọn cách so sánh. Nếu pass=1, ta sẽ so sánh hoành độ x của hai điểm; nếu pass=2 ta sẽ so sánh tung độ y giữa hai điểm. Trình merge được cài đặt như sau:  function merge(a, b: link): link;  var c: link; comp: boolean;  begin     c:=z;     repeat        if pass=1 then comp:=a^.p.x<b^.p.x        else comp:=a^.p.y<b^.p.y;        if comp then           begin              c^.next:=a;              c:=a;              a:=a^.next;           end        else           begin              c^.next:=b;              c:=b;              b:=b^.next;           end;     until c=z;     merge:=z^.next;     z^.next:=z;  end;  Nút rỗng z ở cuối danh sách làm điểm cầm canh với toạ độ x, y giả. Ta dùng một thủ tục đơn giản khác để kiểm tra khoảng cách giữa hai điểm đã cho có nhỏ hơn biến toàn cục min hay không. Nếu nhỏ hơn, biến toàn cục min sẽ lưu lại khoảng cách này và các điểm nút được lưu vào hai biến toàn cục cp1 và cp2.  procedure check(p1,p2: point);  var dis: real;  begin     if (p1.y<>z^.p.y) and (p2.y<>z^.p.y) then     begin        dis:=sqrt(sqr(p1.x-p2.x)+sqr(p1.y-p2.y));        if dis<min then        begin           min:=dis;           cp1:=p1;           cp2:=p2;        end;     end;  end;   Như vậy, min luôn chứa khoảng cách giữa cp1 và cp2, là hai điểm gần nhất được lấy tới lúc này.  Bước thứ hai là thay đổi thủ tục sort để qui trong sắp xếp trộn để tìm điểm gần nhất khi pass=2:  function sort(c: link; N: integer): link;  var a, b: link; i: integer; middle: real;      p1,p2,p3,p4: point;  begin     if c^.next=z then sort:=c else     begin        a:=c;        for i:=2 to N div 2 do c:=c^.next;        b:=c^.next; c^.next:=z;        if pass=2 then middle:=b^.p.x;        c:=merge(sort(a,N div 2),sort(b,N-(N div 2)));        sort:=c;        if pass=2 then           begin              a:=c;              p1:=z^.p; p2:=z^.p; p3:=z^.p; p4:=z^.p;              repeat                 if abs(a^.p.z-middle)<min then                 begin                    check(a^.p,p1);                    check(a^.p,p2);                    check(a^.p,p3);                    check(a^.p,p4);                    p1:=p2; p2:=p3; p3:=p4;                    p4:=a^.p;                 end;                 a:=a^.next;             until a=z;          end;     end;  end;  Nếu pass=1, đây chính là thủ tục mergesort đệ qui trong sắp xếp trộn; Nó trả về danh sách kết nối các điểm được sắp xếp theo hoành độ x (vì merge được thay đổi như đã nêu trên để so sánh x trong pass=1). Điểm hay của cài đặt này là khi pass=2, chương trình không chỉ sắp xếp theo y (vì merge được thay đổi như đã nêu trên để so sánh theo y khi pass=2) mà còn hoàn thành việc tính điểm gần nhất sẽ được mô tả chi tiết sau.  Trước tiên, ta sắp xếp theo x, rồi sắp xếp theo y và tìm cặp điểm gần nhất bằng lời gọi sort như đoạn sau:  new(z); z^.next:=z;  z^.p.x:=maxint;  z^.p.y:=maxint;  new(h);  h^.next:=readlist; // tao danh sach  min:=maxint;  pass:=1; h^.next:=sort(h^.next,N); // sap xep theo x  pass:=2; h^.next:=sort(h^.next,N);  Sau đó hai biến toàn cục cp1, cp2 (trong thủ tục check "tìm cặp nhỏ nhất") chứa cặp điểm gần nhất.  Vấn đề then chốt trong cài đặt này là thao tác sort khi pass=2. Trước khi vào các lời gọi đệ qui, các điểm đã được sắp theo x; trật tự này dùng để chia đôi các điểm và tìm hoành độ x của đường phân giới. Sau các lời gọi đệ qui, các điểm được sắp theo y và khoảng cách giữa các cặp điểm trong mỗi bên nhỏ hơn min. Trật tự theo y được dùng để quét các điểm ở gần đường phân giới; giá trị min được dùng để giới hạn số điểm kiểm tra. Mỗi điểm trong phạm vi min, kể từ đường phân giới được kiểm tra với từng điểm trong 4 điểm trước đó và 4 điểm này cũng trong phạm vi min kể từ đường phân giới (bạn đọc có thể kiểm tra điều này). Sự kiểm tra này đảm bảo tìm cặp điểm gần nhau hơn min và mỗi điểm nằm ở một bên của đường phân giới. Ta biết rằng mỗi điểm rơi vào cùng một bên của đường phân giới cách nhau một khoảng tối thiểu là min, như vậy các điểm rơi vào vòng tròn có bán kính min sẽ được giới hạn lại.  Bạn đọc chăm chú có thể thấy rằng ta không cài đặt thuần tuý thuật toán chia để trị được mô ta ở trên - ta không thực sự tính cặp điểm gần nhất trong hai nửa bên, rồi lấy cặp gần hơn trong hai cặp có được. Thay vào đó, ta đơn giản chỉ tính cặp điểm gần hơn giữa hai cặp điểm gần nhất bằng cách dùng một biến toàn cục min trong suốt việc tính toán đệ qui. Mỗi lần ta tìm được một cặp điểm gân hơn, ta có thể xét một dải đứng hẹp hơn, quanh đường phân giới đang xét.  **Tính chất:** *Cặp điểm gần nhất trong tập N điểm có thể được tìm trong NlongN bước*  Về bản chất, sự tính toán trên được làm trong thời gian cho hai lần mergesort (một theo trục x và một theo trục y) cộng với phí tổn cho việc tìm đường phân giới. Phí tổn này được cho bởi biểu thức đệ qui TN=TN/2+N do đó chi phí NlogN là hệ quả của công thức trên  Cách tiếp cận tổng quát - dùng ở đây cho bài toán tìm cặp điểm gần nhất - có thể được dùng cho các bài toán hình học khác. Ví dụ một vấn đề là tìm *tất cả láng giềng gần nhất* với mỗi điểm ta tìm điểm gần nhất của nó. Bài toán này có thể giải bằng cách dùng một chương trình tương tự như trên với việc mở rộng xử lý theo đường phân giới: với mỗi điểm xét có hay không một điểm ở bên kia đường phân giới gần với nó hơn là điểm gần nhất cùng bên. Lần nữa, việc sắp xếp theo y giúp ích cho quá trình tính toán này.  (Theo Robert Sedgewick)  B. Cài đặt  Dưới đây là chương trình cài đặt bằng pascal theo tư tưởng trên của R. Sedgewick. Chú ý rằng ở đây, thay vì dùng danh sách liên kết chúng ta sử dụng mảng:    const     tfi='gannhat.inp';     tfo='gannhat.out';     maxn=10001;     vc=1.0e+15;     e=1.0e-4;    type     arrN=array[1..maxn] of double;    var     n: longint;     x, y,xt,yt: arrN;     res: double;    function dis(i,j: longint): double;  begin     dis:=sqrt(sqr(x[i]-x[j])+sqr(y[i]-y[j]));  end;    procedure doc;  var i: longint;  begin     read(n);     for i:=1 to n do read(x[i],y[i]);  end;    procedure doi(var u,v: double);  var w: double;  begin     w:=u;     u:=v;     v:=w;  end;    procedure sortx(k,l: longint);  var r: double;      i,j: longint;  begin     r:=x[(k+l) div 2];     i:=k; j:=l;     repeat        while x[i]<r do inc(i);        while x[j]>r do dec(j);        if i<=j then           begin              doi(x[i],x[j]);              doi(y[i],y[j]);              inc(i);              dec(j);           end;     until i>j;     if k<j then sortx(k,j);     if i<l then sortx(i,l);  end;    procedure merge(k,l: longint);  var i,j,u: longint;  begin     for i:=k to l do        begin           xt[i]:=x[i];           yt[i]:=y[i];        end;     j:=(k+l) div 2+1;     i:=k;     u:=k-1;     repeat        inc(u);        if i>(k+l) div 2 then           begin              x[u]:=xt[j];              y[u]:=yt[j];              inc(j);           end        else if j>l then           begin              x[u]:=xt[i];              y[u]:=yt[i];              inc(i);           end        else           begin              if yt[i]<=yt[j] then                 begin                    x[u]:=xt[i];                    y[u]:=yt[i];                    inc(i);                 end              else                 begin                    x[u]:=xt[j];                    y[u]:=yt[j];                    inc(j);                 end;           end;     until u=l;  end;    procedure check(u,v: longint);  var r: double;  begin     if u<>v then        begin           r:=dis(u,v);           if r<res-e then res:=r;        end;  end;    procedure sorty(k,l: longint);  var mid: longint;      r: double;      i, i1, i2, i3, i4: longint;  begin     if k=l then exit;     mid:=(k+l) div 2;     r:=x[mid];     sorty(k,mid);     sorty(mid+1,l);     merge(k,l);     i1:=k; i2:=k; i3:=k; i4:=k;     for i:=k+1 to l do        if abs(x[i]-r)<res-e then           begin              check(i,i1); check(i,i2);              check(i,i3); check(i,i4);              i1:=i2; i2:=i3; i3:=i4;              i4:=i;           end;  end;      procedure main;  begin     assign(input,tfi); reset(input);     assign(output,tfo); rewrite(output);     doc;     sortx(1,n);     // Tinh khoang cach min     res:=vc;     sorty(1,n);     writeln(res:0:4);     close(input); close(output);  end;    BEGIN     main;  END.      Tác giả: Lê Thanh Bình  **[**[**Gửi bạn bè**](mailto:?subject=Chao%20ban)**]    [**[**Bản in**](http://www.chuyennguyentrai.edu.vn/?mod=news&act=print&news_id=45)**]** |

**Thăm dò dư luận**

Top of Form

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

Khoảng cách giữa các điểm thường được xét trong các bài toán hình học. Đã có rất nhiều ứng dụng của một bài toán quen thuộc “tìm láng giềng gần nhất: Tìm điểm gần nhất của một điểm cho trước trong một tập cho trước”. Thông thường, để giải quyết bài toán trên có lẽ hướng thông dụng nhất là kiểm tra khoảng cách giữa điểm được cho với mỗi điểm trong tập điểm đã cho, nhưng có những cách giải quyết tốt hơn. Ở đây, tôi xin đưa ra và cùng các bạn xem xét một bài toán khác về khoảng cách, một thuật toán mẫu được dùng có hiệu quả trong các biến thể của các bài toán như vậy. Cách tiếp cận của chúng ta là sẽ thảo luận một phương pháp tổng quát để giải các bài toán điểm gần nhất thông qua việc xem xét kỹ lưỡng một cài đặt mẫu giải quyết một bài toán đơn giản.

**Bài toán tìm cặp điểm gần nhất**

Bài toán cặp điểm gần nhất tìm hai điểm gần nhau nhất trong một tập điểm cho trước?

Với một yêu cầu đề bài như trên, dường như công việc của chúng ta là phải tính khoảng cách giữa mọi cặp điểm để tìm khoảng cách nhỏ nhất: Giả sử cho trước N điểm, vậy thời gian thực hiện sẽ tỉ lệ với N2. Tuy nhiên, ta có thể dùng một thuật toán sắp xếp để chỉ phải tính chừng NlogN khoảng cách trong trường hợp xấu nhất (về mặt trung bình sẽ ít hơn nhiều). Và bây giờ chúng ta hãy cùng xem xét một thuật toán như vậy nhé.

Chúng ta sẽ dùng một thuật toán dựa trên chiến lược “*chia để trị*”. Ý tưởng ở đây là sắp xếp các điểm theo một trục toạ độ, như trục x chẳng hạn, rồi dùng thứ tự này để chia tập điểm thành hai phần. Trong toàn bộ tập điểm đã cho, cặp điểm gần nhất hoặc là cặp gần nhất trong cùng một bên nào đó, hoặc là một cặp điểm cắt ngang đường thẳng phân giới giữa hai tập điểm thành phần. Dĩ nhiên, trường hợp đáng chú ý là khi cặp điểm gần nhất cắt ngang đường phân giới. Cặp điểm gần nhất trong mỗi nửa bên rõ ràng là tìm được bằng các lời gọi đệ quy, nhưng còn các cặp có mỗi điểm ở một bên đường phân giới sẽ được kiểm tra như thế nào?

Điều tất nhiên là chúng ta sẽ chỉ cần xét những điểm trong khoảng cách min ở hai bên đường phân giới (vì đang tìm cặp điểm gần nhất), với min là là khoảng cách nhỏ hơn giữa các cặp điểm gần nhất ở mỗi nửa bên. Tuy nhiên, trong trường hợp xấu nhất thì nhận xét này là không đủ, vì có thể có nhiều cặp gần với đường phân giới; ví dụ như tất cả các điểm ở mỗi nửa bên có thể sắp thành một hàng ngay cạnh đường thẳng phân giới.

Để xử lý tình huống trên, có vẻ như chúng ta cần phải sắp các điểm theo y. Như vậy, chúng ra có thể giới hạn các khoảng cách phải tính như sau:

-Xử lý các điểm theo chiều tăng của y

-Kiểm tra xem mỗi điểm có nằm trong dải đứng chứa các điểm trong phạm vi min kể từ điểm phân giới

-Với mỗi điểm trong dải trên, tính khoảng cách giữa điểm này với các điểm cũng trong dải và có tung độ y nhỏ hơn tung độ của điểm đang xét nhưng không nhỏ quá min.

Khoảng cách giữa các điểm ở mỗi nửa bên tối thiểu là min nên số điểm phải kiểm tra sẽ ít hơn.

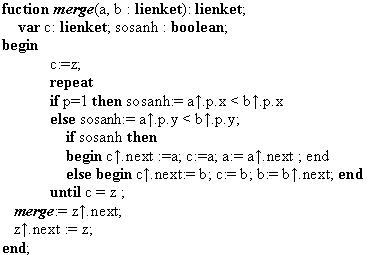
Dù rằng thuật toán này đơn giản, nhưng chúng ta cũng nên chú ý một vài điều để cài đặt có hiệu quả. Chúng ta có thể trả giá để sắp các điểm theo y bằng một thủ tục đệ quy. Nhiều thuật toán có thời gian thi hành được mô tả bằng biểu thức quy nạp CN= 2CN/2+ N, dẫn đến CNlà tỉ lệ với NlogN; Nếu chúng ta xấp xếp theo y thì biểu thức quy nạp sẽ là:

CN=2\*CN/2 + NlogN, dẫn đến CN tỉ lệ với Nlog2N( NlogN\*logN). Bởi vậy, để loại bỏ điều này, chúng ta sẽ tránh sắp theo y

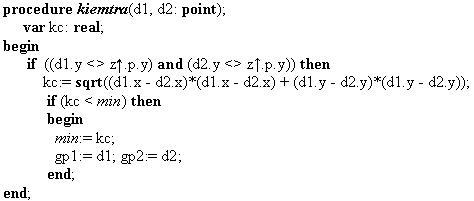
Lời giải vấn đề này tuy đơn giản nhưng cũng thật tinh vi. Ta có hai vấn đề và có chung một lời giải cho chúng, như vậy chúng ta có thể giải chúng đồng thời. Đặc biệt chúng ta sẽ viết một thủ tục đệ quy vừa sắp theo y lại vừa tìm cặp điểm gần nhất. Thủ tục này sẽ chia đôi tập điểm, rồi gọi lại chính nó để sắp hai nửa bên theo y và tìm cặp điểm gần nhất trong mỗi nửa, sau đó trộn hai nửa bên để hoàn tất việc sắp theo y và áp dụng lại thủ tục trên để hoàn tất việc tính cặp điểm gần nhất. Cách này chúng ta đã tránh được chi phí cho sự thêm việc sắp theo y bằng cách trộn dữ liệu được yêu cầu trong việc sắp xếp với dữ liệu được yêu cầu khi tính cặp điểm gần nhất.

Khi sắp theo y, việc chia đôi có thể làm bằng bất kỳ cách nào, nhưng với phép tính cặp điểm gần nhất, việc chia đôi yêu cầu có một nửa bên có hoành độ x nhỏ hơn nửa bên còn lại. Điều này được cài đặt dễ dàng bằng cách sắp theo x trước khi chia đôi tập điểm. Thực tế, ta có thể dùng ngay thủ tục này để sắp theo x! Khi đã hiểu dự định tổng thể, việc cài đặt không còn khó khăn nữa

Chúng ta sẽ cần sử dụng đến thủ tục *sort*và ***merge*** đệ quy. Đầutiên là thay đổi các cấu trúc danh sách để lưu các điểm thay cho các khóa, và sử dụng thủ tục ***merge*** để kiểm tra biếnt toàn cục p dùng cho việc chọn cách so sánh. Nếup= 1, ta sẽ so sánh hoành độ x của hai điểm; nếu p=2 ta so sánh tung độ y giữa hai điểm. Chúng ta cùng xét hàm ***merge****:*

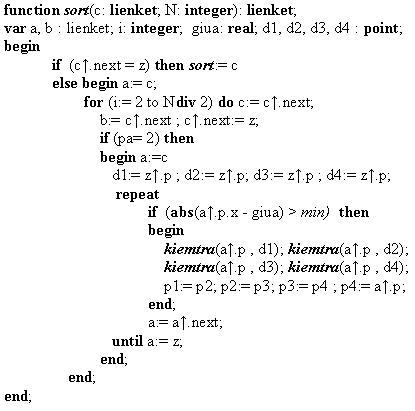


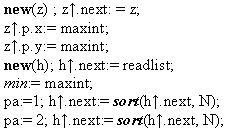
Nút rỗng x ở cuối danh sách làm điểm cầm canh với toạ độ x,y giả. Ta dung một thủ tục đơn giản khác để kiểm tra khoảng cách giữa hai điểm đã cho có nhỏ hơn biến toàn cục *min*hay không. Nếu nhỏ hơn, biến min sẽ lưu lại khoảng cách này và các điểm nút được lưu vào hai biến toàn cục g*p1* và *gp2*



Như vậy, *min* luôn chứa khoảng cách giữa gp1 và gp2, là 2 điểm gần nhất được thấy tới lúc này.

Bước tiếp theo là chúng ta cùng xét thủ tục Sort sau để tìm điểm gần nhất khi pa=2





Nếu pa:=1 thủ tục sẽ trả lại một danh sách kết nối các điểm được sắp theo hoành độ x. Cái hay của thủ tục này là khi pa:= 2, chương trình không chỉ sắp xếp theo y mà còn hoàn thành việc tính điểm gần nhất.

Đầu tiên chúng ta sắp xếp theo x, sau đó sắp xếp theo y rồi cặp điểm gần nhất bằng lời gọi ***sort*** như trên thủ tục trên

Trong thủ tục ***kiemtra***, hai biến toàn cục gp1, gp2 chứa cặp điểm gần nhất

điều then chốt trong thuật toán này là thao tác sort khi pa= 2. Trước khi thực hiện đệ quy, các điểm đã được sắp xếp theo x; trật tự này dùng để chia đôi các điểm và tìm hoành độ x của đường phân giới. Sau khi đệ quy, các điểm được sắp xếp theo y và khoảng cách giữa các cặp điểm trong mỗi nửa bên là lớn hơn *min*. Trật tự theo y được dùng để quét các điểm ở gần đường phân giới; giá trị *min* được dùng để giới hạn số điểm kiểm tra. mỗi điểm ở trong phạm vi *min* kể từ điểm phân giới được kiểm tra với từng điểm trong 4 điểm trước đó và 4 điểm này cũng ở trong phạm vi *min* kể từ đường phân giới. Ta biết rằng các điểm rơi vào một bên của đường phân giới cách nhau một khoảng tối thiểu là *min*, như vậy số các điểm rơi vào vòng tròn có bán kính *min*sẽ được giới hạn lại.

Nếu các bạn để ý sẽ nhận ra rằng, chúng ta đã không cài đặt thuần tuý thuật toán chia để trị. Ta không thực sự tính cặp gần nhất trong hai nửa bên, rồi lấy cặp gần hơn trong hai cặp có được. Thay vào đó, chúng ta chỉ đơn giản tính cặp gần hơn giữa hai cặp gần nhất bằng cách dùng một biến toàn cục *min* trong suốt việc tính toán đệ quy. Mỗi lần ta tìm được một cặp gần hơn, ta có thể xét một dải đứng hẹp hơn quanh đường phân giới đang xét.

Qua tất cả những điều chúng ta xét ở trên, đến lúc này chúng ta có thể thấy:

\*\*\* *Cặp điểm gần nhất trong một tập N điểm có thể được tìm trong NlogN bước.*

Như vậy, bài toán tìm cặp điểm gần nhất có thể dùng cho các bài toán hình học khác. Ví dụ như bài toán tìm “tất cả láng giềng gần nhất: Với mỗi điểm ta tìm điểm gần nhất của nó”. Bài toán này có thể giải bằng cách dùng một chương trình tương tự như trên với việc mở rộng xử lý theo đường phân giới: Với mỗi điểm xem có hay không một điểm ở bên kia đường phân giới gần với nó hơn là điểm gần nhất ở cùng bên. Một lần nữa, việc sắp xếp theo y giúp ích cho quá trình tính toán này

Bottom of Form