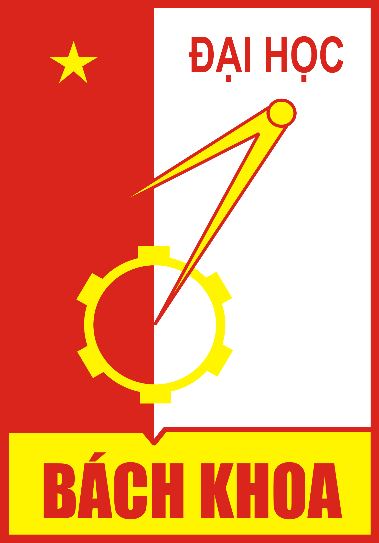
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**



**VIỆN CÔNG NGHỆ THÔNG TIN – TRUYỀN THÔNG**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

****

**BÁO CÁO MÔN HỌC  
PROJECT II**

**ĐỀ TÀI**

**Tìm hiểu và cài đặt bài toán tìm cặp điểm gần nhất**

**(Closest Pair)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sinh viên thực hiện:** | | | |
| **HỌ VÀ TÊN** | **MSSV** | **LỚP** | **KHÓA** |
| **NGUYỄN TẤT HÒA** | **20131536** | **CNTT2.01** | **K58** |

**Giảng viên hướng dẫn: PGS TS. Nguyễn Đức Nghĩa**

**HÀ NỘI, THÁNG 4/2016**

MỤC LỤC

[**Lời mở đầu** 3](#_Toc451351520)

[**Phần 1: Giới thiệu bài toán** 4](#_Toc451351521)

[**1.** **Phát biểu bài toán** 4](#_Toc451351522)

[**2.** **Lịch sử giải quyết bài toán.** 4](#_Toc451351523)

[**Phần 2: Các thuật toán** 4](#_Toc451351524)

[**1.** **Giải thuật vét cạn.** 4](#_Toc451351525)

[**2.** **Giải thuật chia để trị.** 6](#_Toc451351526)

[**Phần 3: Phân tích và thiết kế chương trình** 10](#_Toc451351527)

[**1.** **main()** 10](#_Toc451351528)

[**2.** **bruteForce()** 10](#_Toc451351529)

[**3.** **generatePoint()** 10](#_Toc451351530)

[**4.** **checkPoint()** 10](#_Toc451351531)

[**5.** **ClosestPair()** 10](#_Toc451351532)

[**6.** **stripClosest()** 11](#_Toc451351533)

[**7.** **readFile()** 12](#_Toc451351534)

[**8.** **savePoint()** 12](#_Toc451351535)

[**9.** **result()** 12](#_Toc451351536)

[**10.** **compareY()** 12](#_Toc451351537)

[**Phần 4: Thực nghiệm.** 12](#_Toc451351538)

[**1. Phương pháp sinh dữ liệu.** 13](#_Toc451351539)

[**2. Tiêu chí đánh giá** 13](#_Toc451351540)

[**3.** **Cấu hình máy thực nghiệm** 13](#_Toc451351541)

[**4.** **Kết quả so sánh.** 14](#_Toc451351542)

[**5.** **Hướng dẫn cài đặt và sử dụng chương trình.** 16](#_Toc451351543)

[**Phần 5: Source code** 18](#_Toc451351544)

[**Phần 6: Tài liệu tham khảo** 33](#_Toc451351545)

# **Lời mở đầu**

Thời đại hiện nay, công nghệ thông tin càng ngày càng phát triển, trong đó Khoa học máy tính là một trong những bộ môn quan trọng trong sự phát triển đó. Khoa học máy tính là nền tảng của mọi lĩnh vực liên quan đến CNTT, trong đó có Công nghệ Phần mềm. Ngành này giúp người học nắm vững kiến thức thuật toán, hiểu biết đầy đủ về các lĩnh vực đa dạng của máy tính.

Tìm cặp điểm gần nhất là một bài toán kinh điển trong khoa học máy tính. Hiện nay, có nhiều thuật toán để giải bài toán này. Trong bài báo cáo này sẽ trình bày hai thuật toán cơ bản để giải quyết vần đề đó là phương pháp vét cạn và phương pháp chia để trị, với từng phương pháp, bản báo cáo sẽ trình bày rõ ý tưởng, giải thuật, cài đặt giải thuật chi tiết, đưa ra được thời gian để giải quyết vấn đề trong từng trường hợp cụ thể cũng như chứng minh tính đúng đắn của thuật toán. Cuối cùng đưa ra đánh giá, kết luận về tính hiệu quả và tính tối ưu để giải quyết được bài toán tìm cặp điểm gần nhất.

Do kiến thức còn hạn hẹp, chắc hẳn bản báo cáo vẫn còn những sai sót và hạn chế vì vậy em mong nhận được sự góp ý, hướng dẫn và sửa chữa để bản báo cáo được dần dần hoàn thiện hơn.

Em xin chân thành cảm ơn PGS TS Nguyễn Đức Nghĩa đã giúp em có được những kiến thức cơ bản trong quá trình hướng dẫn để hoàn thành được bản báo cáo này.

Hà Nội, ngày 1 tháng 4 năm 2016.

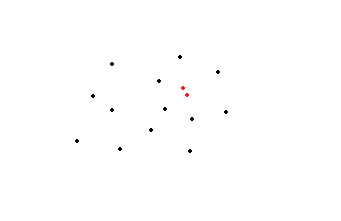
**Sinh viên thực hiện**:

Nguyễn Tất Hòa

## **Phần 1: Giới thiệu bài toán**

1. **Phát biểu bài toán**

Các cặp điểm gần nhất là một vấn đề của hình học tính toán và là một trong những bài toán kinh điển của bộ môn khoa học máy tính: Cho tập P gồm n điểm trong một không gian d chiều, tìm một cặp điểm thuộc n điểm đó sao cho khoảng cách giữa hai điểm đó là ngắn nhất. Trong bản báo cáo này sẽ tập trung phát biểu và cài đặt thuật toán trên không gian hai chiều.

1. **Lịch sử giải quyết bài toán.**
   * Giải thuật vét cạn: giải thuật này không có tác giả. Độ phức tạp của giải thuật này là O(n2).
   * Giải thuật chia để trị: giải thuật này do Bentley và Shamos trình bày năm 1976. Độ phức tạp của giải thuật là O(*n*log*n*).
   * Giải thuật cải tiến từ Chia để trị của Bentley và Shamos: giải thuật này do Mohammad Zaidul Karim và Nargis Akter đề xuất. Giải thuật này chia bài toán thành n phần thay vì 2 phần. Giải thuật này trong nhiều trường hợp có độ phức tạp nhỏ hơn O(n log n).

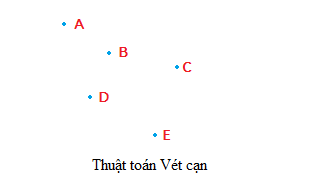
## **Phần 2: Các thuật toán**

1. **Giải thuật vét cạn.**
   1. **Ý tưởng.**

Ý tưởng giải thuật vét cạn gồm các bước cụ thể như sau:

* + - Tính tất cả các khoảng cách giữa các điểm trong tập P.
    - Xác định khoảng cách có giá trị nhỏ nhất trong số các khoảng cách vừa tìm được.
    - Cặp điểm có khoảng cách ngắn nhất là cặp điểm gần nhất.

Ví dụ:



Trong hình trên gồm một tập 5 điểm, chúng ta cần tính tất cả khoảng cách có thế có giữa năm điểm ấy. Bắt đầu từ A có 4 khoảng cách với 4 điểm B, C, D, E. Từ điểm B có 3 khoảng cách tới điểm C, D, E do khoảng cách tới điểm A đã tính trước đó. Tương tự như vậy với các điểm C, D, E vậy có tất cả 4 + 3 + 2 + 1 = 10 khoảng cách cần tính toán. Tổng quát với n điểm thì tổng số phép tính khoảng cách giữa các điểm đó là

* 1. **Giải thuật**

Thuật toán tìm cặp điểm gần nhất bằng vét cạn:

1. **procedure** **BruteForce**(P, n)

2. *min* = ∞

3. **for** i = 1 to n − 1 **do**

4. **for** j = i + 1 to n **do**

5. d = **dist**(Pi, Pj)

6. **if** (d < *min*) **then**

7. *min* = d

8. **closestPair** = (Pi, Pj)

9. **end if**

10. **end for**

11. **end for**

12. **return** closestPair

13. **end procedure**

* 1. **Độ phức tạp.**

Độ phức tạp của thuật toán được tính như sau:

* + - Độ phức tạp của chương trình con dist (phép tính căn và bình phương) là O(1).
    - Độ phức tạp của các lệnh 2, 4, 6, 7, 8 là O(1).
    - Độ phức tạp của lệnh 4 là O(n-i).
    - Độ phức tạp của lệnh 3 sẽ là O(n­­­2­­­) như đã tính toán phía trên

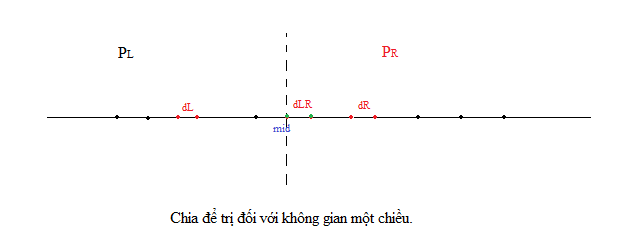
Vậy độ phức tạp của thuật toán vét cạn cho bài toán tìm cặp điểm gần nhất là T(n) = O(n­­2). Vì thuật toán chỉ đơn giản là xét duyệt tất cả các điểm và khoảng cách nên việc cài đặt là như nhau đối với mọi không gian.

1. **Giải thuật chia để trị.**
   1. **Đối với không gian một chiều**

Ý tưởng của giải thuật chia để trị gồm các bước như sau:

* + - Chia tập P thành 2 tập con có số lượng phần tử bằng nhau. Điểm ở giữa là pmid
    - Tìm cặp điểm có khoảng cách nhỏ nhất trong nửa trái P­­L­ là d­L.
    - Tìm cặp điểm có khoảng cách nhỏ nhất trong nửa phải P­­R­ là d­R.
    - ­Tìm khoảng cách nhỏ nhất đối với các điểm xung quanh vị trí p­mid­ gọi là d­mid­.
    - So sánh d­L­, d­R­, d­mid­ sẽ tìm ra được cặp điểm có khoảng cách nhỏ nhất.
  + Dùng đệ quy để tìm được hai điểm có khoảng cách nhỏ nhất thuộc mỗi tập con bên trái và tập con bên phải. Từ đó ta tìm được d­L­. d­R­.  
     Còn d­mid­ giả thiết rằng d­mid­ là khoảng cách nhỏ nhất cần tìm thì chắc chắn trong hai điểm tìm được một điểm sẽ thuộc P­L­ và một điểm sẽ thuộc P­R­ vì nếu vi phạm điều trên thì mẫu thuẫn với d­L­, d­R­ là khoảng cách nhỏ nhất của mỗi nữa trái, phải. Chính vì vậy nên chỉ cần xét khoảng cách giữa hai điểm gần phải nhất đối với tập P­L­ và gần trái nhất đối với tập P­R­ gọi đó là d­mid­.

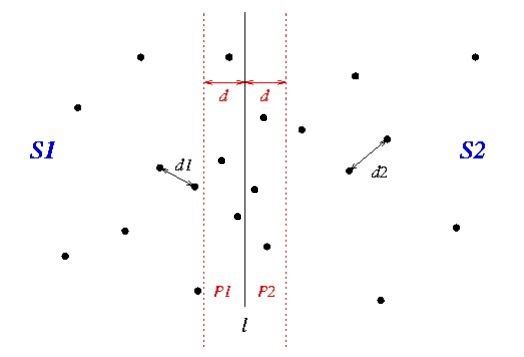
Cuối cùng so sánh d­L­, d­R­, d­mid­ rồi đưa ra hai điểm có khoảng cách nhỏ nhất thuộc tập P. Do trong bản báo cáo này tập trung vào không gian hai chiều nên sẽ không đi sâu vào việc cài đặt thuật toán cũng như so sánh hiệu năng của giải thuật trên không gian một chiều.



* 1. **Đối với không gian hai chiều.**

1. **Ý tưởng.**

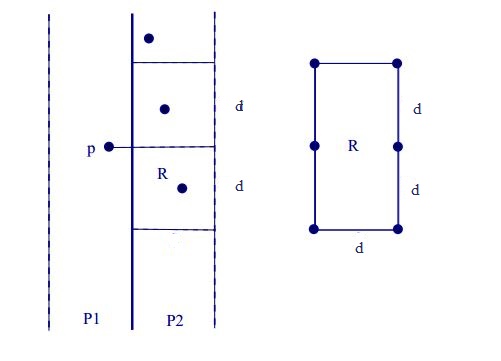
Ý tưởng của thuật toán hoàn toàn tương tự như trong không gian một chiều nhưng việc tìm được khoảng cách d­mid­ lại không hề đơn giản bởi lúc này không phải chỉ có hai điểm cần được xét.



Trên thực tế, trong không gian hai chiều, những điểm xung quanh p­mid­ có thể là tất cả các điểm trong tập hợp đang xét dẫn đến độ phức tạp của bài toán sẽ là O(n2) mà mục tiêu của thuật toán là O(*n*log*n*).

Chú ý rằng với mỗi cặp ứng cử viên (p, q) xung quanh p­mid với p S1 và q S2 thì p, q phải nằm trong khoảng cách *l (l* là giải phân cách đi qua điểm chính giữa p­mid ) một đoạn là d kéo về hai phía với d = min(dL, dR). Như đã nói ở trên thì số điểm cần xét đến có thể là tất cả các điểm.

Và thật may mắn ta có một cách để giải quyết vấn đề này. Xét mỗi điểm p S1, tất cả các điểm thuộc S2 mà cách l một khoảng bằng d phải nằm trong một hình chữ nhật R có kích thước d x 2d. Bởi vì những điểm mà nằm bên ngoài hình chữ nhật R sẽ không thể đạt được khoảng cách tới p mà lại nhỏ hơn d được.



Trong không gian hai chiều thì số điểm thỏa mãn điều kiện đó nhiều nhất là 6 điểm. Do đó chúng ta chỉ cần thực hiện nhiều nhất là 6 x n phép so sánh khoảng cách chứ không còn phải n2. Tất cả những gì chúng ta phải làm là sắp xếp những điểm xung quanh p­mid lại theo chiều tăng của tung độ và kiểm tra các điểm đó theo thứ tự với nhiều nhất sáu điểm láng giềng. Thuật toán sắp xếp cùng cần có độ phức tạp là O(*n*log*n*), trong bài này thuật toán được lựa chọn là Merge Sort.

* 1. **­Giải thuật.**  
     Thuật toán tìm cặp điểm gần nhất bằng cách **chia để trị**:

1**. proceduce ClosestPair**(P, n){

2. **if** n<3 **then**

3. **return** **bruteForce**(P,n)

4. **end** **if**

5. *mid* = n/2; midPoint = P[*mid*];

6. dL = **ClosestPair**(P,mid); dR = **ClosestPair**(P+mid, n-mid);

7. d = **min**(dL, dR);

8. pLR = { P | abs(p[i].x – midPoint.x) <= d};

9. **return** **min**(**stripClosest**(pLR, pLR.count, d), d);

10.**end** **proceduce.**

Thuật toán tìm khoảng cách nhỏ nhất của tập pLR:

1. **proceduce stripClosest**(pLR, k, d){
2. *min* = d;
3. **for** i: 1 to k **do**
4. **for** j: i+1 to k **&&** pLR[j].y - pLR[i].y < min **do**
5. **if** (**dist**(pLR[i],pLR[j]) < min) **then**
6. *min* = **dist**(pLR[i],pLR[j]);
7. **if** (*min* < dmin) **then**
8. dmin = *min*;
9. Closest Pair = (pLR[i], pLR[j])
10. **end if**
11. **end if**
12. **end for**
13. **end for**
14. **return** *min*
15. **end proceduce**
    1. **Độ phức tạp.**

Độ phức tạp của thuật toán chia để trị ở từng bước sẽ là:

* + - Bước 2: O(1)
    - Bước 3: O(25)
    - Bước 5: O(n)
    - Bước 6: O(*n*log*n*)
    - Bước 8: O(n)
    - Bước 9: O(n) (như đã chứng minh phía trên)

Do ở việc sắp xếp ở bước 9 độ phức tạp là O(*n*log*n*) và điều này được lặp lại ở mỗi lần đệ quy nên độ phức tạp của thuật toán sẽ là O(*n*log­2*n*). Ta có thể giải quyết vấn đề này để giải thuật có thể đạt được độ phức tạp O(*n*log*n*) bằng cách sắp xếp theo chiều tăng của tọa độ y ngay từ đầu chương trình thay vì sắp xếp trong hàm **stripClosest** như đã nói ở trên.

Cuối cùng thuật toán đã đạt được độ phức tạp là O(*n*log*n*).

## **Phần 3: Phân tích và thiết kế chương trình**

Theo những ý tường và hướng giải quyết đã nêu ở phần trên thì chương trình sẽ bao gồm những hàm như sau:

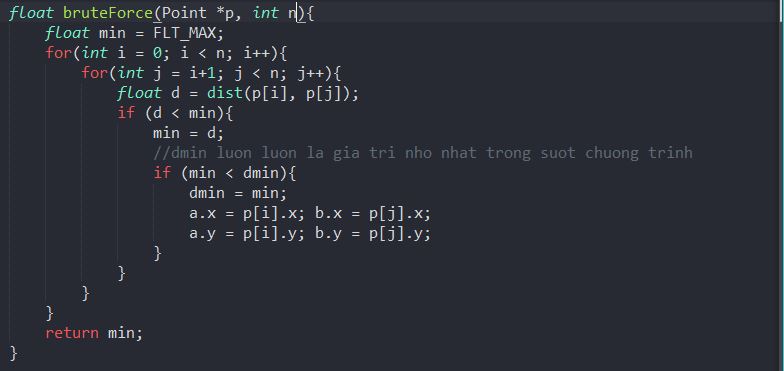
1. **main()**

Hàm khởi tạo khi chương trình được chạy, tại đây khai báo các biến và từ đó chạy tới các hàm. Trong chương trình này hàm **main()** sẽ thực hiện các **switch-case** để người dùng có thể lựa chọn từng chức năng chính của chương trình.

1. **bruteForce()**

Hàm được truyền vào mảng chứa không gian dữ liệu và số điểm trong không gian là n.

Hàm sử dụng thuật toán vét cạn để tìm ra cặp điểm có khoảng cách nhỏ nhất trong tập không gian dữ liệu.



1. **generatePoint()**

**Đối số truyền vào:**

**Point a[]:** Mảng chứa dữ liệu để các điểm đã được tạo từ đó đối chiếu để tìm điểm khác biệt.

**int n:** Số điểm cần sinh ngẫu nhiên.

**char fileName[]:** Tên file để lưu dữ liệu

Hàm sinh ngẫu nhiên tập không gian để kiểm chứng hiệu năng của từng thuật toán.

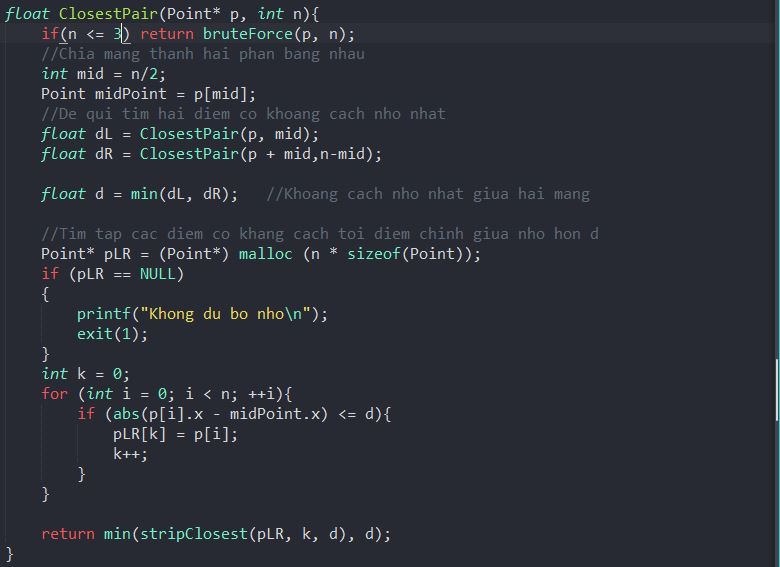
1. **checkPoint()**

Kiểm tra xem các điểm có vừa sinh ra đã tồn tại trong tập dữ liệu chưa, nếu có trả về **true** ngược lại trả về **fasle,** hàm này hỗ trợ cho **generatePoint().**

1. **ClosestPair()**

Hàm chính để tìm cặp điểm có khoảng cách nhỏ nhất trong tập không gian dữ liệu bằng phương pháp chia để trị, được truyền vào một con trỏ p có kiểu dữ liệu **Point\*** chính là tập không gian dữ liệu đã được sắp xếp theo tọa độ y khi mà người dùng lựa chọn giải quyết bài toán bằng phương pháp chia để trị và một biến số nguyên n là số điểm có trong không gian dữ liệu.

Hàm thực hiện đệ quy tìm ra khoảng cách nhỏ nhất giữa hai nửa của tập dữ liệu từ đó tìm ra số điểm



1. **stripClosest()**

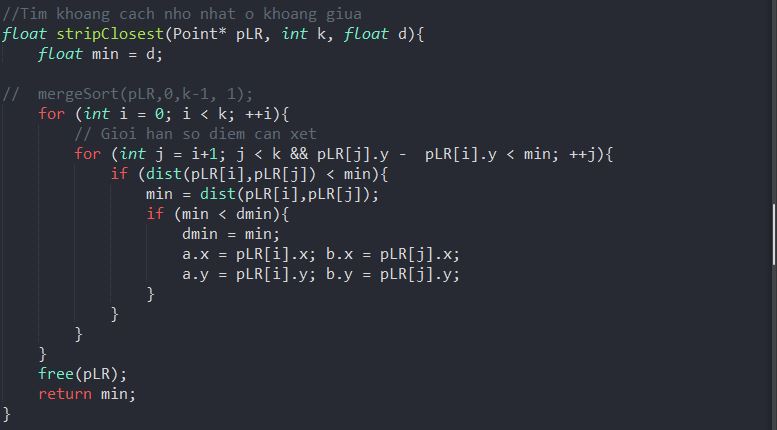
***Đối số truyền vào:***

**Point\* pLR**: Các điểm nằm trong vùng lân cận điểm chính giữa ở mỗi lần đệ quy.

**int k**: Số điểm trong tập pLR.

**float d**: khoảng cách nhỏ nhất trong hai nửa trái phải.

Hàm tìm và so sánh hai khoảng cách đệ quy với khoảng cách giữa các điểm xung quanh vị trí p­mid­ để phục vụ cho hàm ClosestPair().



1. **readFile()**

Hàm truyền vào biến số kiểu **char fileName[]** là tên file được người dùng nhập vào từ bàn phím để đọc dữ liệu.

Hàm đọc dữ liệu đầu vào trong chương trình. Thực hiện đọc và gán các điểm cho mảng được cấp phát động từ file có cầu trúc được đinh sẵn.

1. **savePoint()**

Lưu lại tập không gian dữ liệu do người dùng nhập từ bàn phím vào file. Trong đó:

**Point \*p**: Tập không gian dữ liệu

**int n**: số điểm

**char fileName[]**: tên file

Là các đối số truyền vào.

1. **result()**

Lưu kết quả vào file khi người dùng giải quyết xong bài toán và muốn lưu lại kết quả

1. **compareY()**

compareY() giúp định nghĩa lại cho phương thức so sánh của hàm **qsort()** sẵn có trong thư viện stdlib.h. Nó hỗ trợ cho **qsort()** để có thể sắp xếp được với **struct**, sắp xếp lại các điểm ngay trước khi hàm **ClosestPair()** được gọi

## **Phần 4: Thực nghiệm.**

Theo lí thuyết thì khi giải bài toán tìm cặp điểm gần nhất thì thuật toán vét cạn có độ phức tạp là O(n­­2) còn thuật toán chia để trị thì có độ phức tạp là O(*n*log*n*), tức là thuật toán Chia để trị tốt hơn thuật toán Vét cạn. Như vậy, cần cài đặt hai giải thuật Vét cạn và Chia để trị . Áp dụng hai giải thuật này lên nhiều tập dữ liệu khác nhau với các tiêu chí đánh giá hợp lý nhằm kiểm tra tính đúng đắn của lý thuyết.

### **1. Phương pháp sinh dữ liệu.**

* + Sinh ngẫu nhiên tập P
  + Các điểm trong cùng một tập không được trùng nhau
  + Sinh đủ số lượng yêu cầu
  + Các điểm trong mỗi lần sinh ra là khác nhau

Cách sinh dữ liệu để không tồn tại 2 điểm trùng nhau trong tập:

Dùng một mảng đủ lớn (kích thước bằng với kích thước của tập P) để lưu giá trị của tất cả các điểm đã được tạo ra trước đó. Sau mỗi lần sinh ngẫu nhiên một điểm mới kiểm tra xem điểm đó đã tồn tại chưa nếu tồn tại thì tiếp tục sinh ngẫu nhiên cho đến khi không tồn tại thì lưu lại vào tập P. Thuật toán kết thúc khi ghi đủ số lượng phần tử.

Thuật toán:

1. **proceduce** **generatePoint**(a[], n)
2. Open file point.txt
3. Write value n to file point.txt in first line
4. x = *random*(10000); y = *random*(100000);
5. **for** i: 1 to n **do**
6. **while** exist in a[] **do**
7. x = random(10000); y = random(100000);
8. **end** **while**
9. Write x, y to file point in a line;
10. **end** **for**
11. **end** **proceduce**

Do x, y lần lượt là hoành độ và tung độ của điểm trong P và được sinh ngẫu nhiên trong khoảng (0,100000) nên độ lớn của không gian dữ liệu có thể lên tới 1010 điểm, đủ lớn để có thể thấy được sự khác biệt giữa hai thuật toán

### **2. Tiêu chí đánh giá**

* Kết quả của hai thuật toán phải giống nhau trong các trường hợp
* Trong đa số trường hợp thì thuật toán Chia để trị có thời gian chạy máy ít hơn Vét cạn.
* Khi dữ liệu lớn dần thì cả hai thuật toán đều có thời gian chạy máy tăng lên, nhưng thuật toán Vét cạn có thời gian chạy máy tăng rất nhanh (O(n2) tức là cho kết quả chậm khi dữ liệu lớn), còn chia để trị thì tăng không đáng kể (O(*n*log*n*)).

1. **Cấu hình máy thực nghiệm**

Trong quá trình cài đặt và thực nghiệm có sử dụng máy tính với các thông số như sau:

* CPU: Intel, Core i5, 4200U, 1.60 GHz
* RAM: DDR3L 8GB 1600 MHz
* Ổ cứng: HDD, 500GB

### **Kết quả so sánh.**

Sau đây là kết quả thực nghiệm khi chạy chương trình với trục tung có đơn vị là giây(s) và trục hoành có đơn vị là điểm.

Do thời gian giải quyết bài toán của thuật toán Vét cạn tăng rất nhanh theo sự tăng của không gian dữ liệu nên bên dưới sẽ chỉ trình bày biểu đồ thuật toán Chia để trị với không gian dữ liệu lớn.

**Kết luận:**

* Về lý thuyết:
  + Brute Force: O(n­­2).
  + Devide and Conquer: O(*n*log*n*)
* Về thực nghiệm: dựa vào kết quả đồ thị.
  + Khi tập dữ liệu nhỏ thì thời gian chạy của hai thuật toán không chênh lệch, nhưng khi dữ liệu đủ lớn thì giải thuật Chia để trị chạy nhanh hơn giải thuật Vét cạn.
  + Khi tập dữ liệu lớn dần thì cả hai giải thuật đều có thời gian chạy máy tăng nhưng:
    - Vét cạn: tăng rất nhanh.
    - Chia để trị: tăng rất chậm.
    - Các biểu đồ thứ 2 và thứ 3 thể hiện rất rõ điều đó
    - Khi tập điểm là bội số của 10000 thì thuật toán Vét cạn cho thời gian trả về kết quả là rất lâu lên tới tầm bảy phút khi tập P có 100000 điểm, cũng với tập đó thì thuật toán chia để trị trả về kết quả khi chưa đến 0.12(s).

Từ các tính toán trên lý thuyết và thực tế cho thấy để giải quyết bài toán Tìm cặp điểm gần nhất thì thuật toán Chia để trị đạt được hiệu năng cao hơn thuật toán Vét cạn. Trong một không gian dữ liệu đủ lớn việc lựa chọn thuật toán Chia để trị để giải quyết bài toán tìm cặp điểm gần nhất là hợp lí.

1. **Hướng dẫn cài đặt và sử dụng chương trình.**

Chương trình giải quyết bài toán tìm cặp điểm gần nhất được viết bởi ngôn ngữ lập trình C hoàn toàn dể sử dụng đối với những người chưa bao giờ lập trình, để chạy được chương trình chỉ cần bất cứ chương trình có thể biên dịch và chạy ngôn ngữ C.

Khi chạy chương trình người dùng sẽ có các lựa chọn:

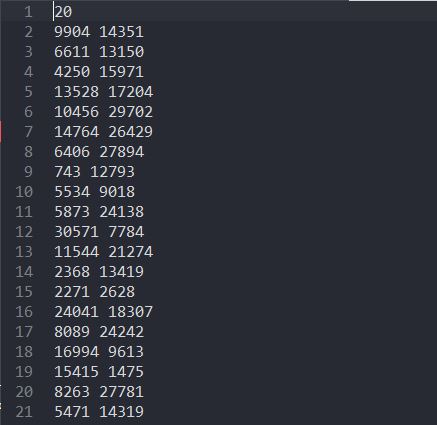
* + - Đọc dữ liệu đầu vào.
    - Giải quyết bài toán.
    - In danh sách các điểm ra màn hình.
    - Thoát khỏi chương trình

**Đọc dữ liệu đầu vào**: Có các lựa chọn như:

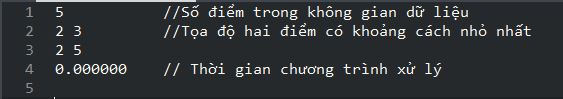
* + - * Đọc từ file (tên file cho người dùng nhập vào từ bàn phím, đuôi file phải là “.in”, theo cấu trúc định sẵn).
      * Nhập vào từ bàn phím
      * Sinh điểm ngẫu nhiên

Trong cả ba trường hợp không gian dữ liệu đều được lưu trong file do người dùng đã đặt tên trước đó và được lưu trong cùng một thư mục với chương trình, chứa vị trí các điểm trong không gian có cấu trúc như sau:

* + - Dòng đầu là số các điểm có trong tập P
    - Các dòng tiếp theo là vị trí theo tọa độ (x, y) của các điểm trong đó x là hoành độ, y là tung độ.



**Giải quyết bài toán**: Có hai lựa chọn cho người dùng đó là dùng thuật toán Vét cạn hoặc dùng chia để trị để giải quyết bài toán với tập không gian dữ liệu đã nhập, nếu trước đó người dùng không nhập dữ liệu chương trình sẽ thông báo lỗi và yêu cầu người dùng nhập vào dữ liệu. Sau khi giải quyết bài toán xong chương trình sẽ hỏi người dùng có muốn lưu lại kết quả không, nếu có thì kết quả sẽ được lưu vào file có tên trùng tên file dữ liệu với đuôi file là “.out”. Cấu trúc file kết quả:



## **Phần 5: Source code**

Dưới đây là toàn bộ chương trình giải quyết bài toán Closest Pair:

**File ClosestPair.cpp**

#include <stdio.h>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <time.h>

#include <float.h>

// #include <windows.h>

#include <string.h>

#include <math.h>

#include "Timer.cpp"

#include <set>

// #define number 100000

using namespace std;

int count = 0;

char fileName[50]="input";

struct Point{

int x, y;

};

Point\* p;

Point a, b;

float dmin = FLT\_MAX; //dmin la gia tri nho nhat trong suot chuong trinh

void readFile(char\*);

void savePoint(Point\*, int, char\*);

void printPoint(Point\*);

void setPoint(Point\*, Point\*);

void generatePoint(int);

void mergeSort(Point\*, int, int, int);

// sap xep tron theo x voi flag = 0, theo y voi flag = 1

void merge(Point\*, int, int ,int, int);

void result(double, double);

float min(float, float);

float stripClosest(Point\*, int, float);

float dist(Point, Point); // khoang cach giua hai diem p, q

float bruteForce(Point\*, int); // Thuat toan vet can

int compareY(const void\*, const void\*);

//Doc gia tri dau vao

void readFile(char fileName[]){

char fileread[50];

strcpy(fileread,fileName);

strcat(fileread, ".in"); //Gan phan mo rong cho file

FILE\* f = fopen(fileread,"rt");

if(f==NULL) {

printf("Khong mo duoc file\n");

exit(0);

}

fscanf(f, "%d", &count);

p = (Point\*) malloc(count \* sizeof(Point));

int x, y;

for(int i = 0; i < count; i++){

fscanf(f,"%d",&x);

fscanf(f,"%d",&y);

p[i].x = x;

p[i].y = y;

}

fclose(f);

}

//Luu cac diem dau vao nhap tu ban phim

void savePoint(Point \*p, int count, char fileName[]){

FILE \*f;

char fileread[50];

strcpy(fileread,fileName);

strcat(fileread,".in");

f = fopen(fileread,"w");

if (f == NULL)

{

printf("Khong mo duoc file\n");

exit(1);

}

fprintf(f, "%d\n", count);

for (int i = 0; i < count; ++i)

{

fprintf(f, "%d %d\n",p[i].x, p[i].y);

}

fclose(f);

}

// Sinh ngau nhien diem tren mat phang

bool checkPoint(Point \*a, int count, int x, int y){

if (count == 0) return 0;

for(int i = 0; i< count; i++){

if (a[i].x == x && a[i].y == y) return 1;

}

return 0;

}

//Sinh diem ngau nhien

void generatePoint(Point a[], int n, char fileName[]){

FILE \*f;

char generateFile[30];

strcpy(generateFile,fileName);

strcat(generateFile,".in");

f = fopen(generateFile,"w");

if (f == NULL)

{

printf("Khong mo duoc file\n");

exit(1);

}

fprintf(f, "%d\n", n);

int x; int y;

x = rand() %100000;

y = rand() %100000;

for(int i = 0; i < n; i++){

while(checkPoint(a,i,x,y)){

x = rand() %10000;

y = rand() %10000;

}

a[i].x = x;

a[i].y = y;

fprintf(f, "%d %d\n",x, y);

}

fclose(f);

}

//Luu lai ket qua

void result(Point A, Point B, double y, char fileName[]){

char fileResult[50];

strcpy(fileResult,fileName);

strcat(fileResult, ".out");

FILE\* f1 = fopen(fileResult,"a+");

if(f1 == NULL){

printf("Khong mo duoc file");

exit(1);

}

fprintf(f1,"%d\n%d %d\n%d %d\n%f\n\n",count,A.x, A.y, B.x, B.y, y);

fclose(f1);

}

//Gan gia tri cua p cho q

void setPoint(Point\* p, Point\* q){

for(int i = 0; i < count; i++){

q[i] = p[i];

}

}

//In ra man hinh toa do cac diem

void printPoint(Point\* p){

printf("Danh sach cac diem:\n");

for(int i = 0; i< count; i++){

printf("P%d(%d,%d) ",i+1, p[i].x, p[i].y);

if (i%5 == 4){

printf("\n");

}

}

}

//Tinh khoang cach giua hai diem

float dist(Point p, Point q){

return sqrt(pow(abs(p.x-q.x),2) + pow(abs(p.y-q.y),2));

}

//Tra ve gia tri nho nhat

float min(float x, float y){

if (x > y) return y;

else return x;

}

//Thuat toan vet can

float bruteForce(Point \*p, int n){

float min = FLT\_MAX;

for(int i = 0; i < n; i++){

for(int j = i+1; j < n; j++){

float d = dist(p[i], p[j]);

if (d < min){

min = d;

//dmin luon luon la gia tri nho nhat trong suot chuong trinh

if (min < dmin){

dmin = min;

a.x = p[i].x; b.x = p[j].x;

a.y = p[i].y; b.y = p[j].y;

}

}

}

}

return min;

}

//Sap xep so sanh theo chieu tang toa do Y su dung qsort() san co trong thu vien stdlib;

int compareY(const void\* a, const void\* b){

Point \*p1 = (Point\*) a;

Point \*p2 = (Point\*) b;

return (p1->y - p2->y);

}

void merge(Point\* p, int L, int M, int R, int flag){

int i =L; int j = M+1;

Point\* tmp = (Point\*) malloc((R + 1) \* sizeof(Point));

for(int k = L; k <= R; k++){

if(i > M){

tmp[k] = p[j];

j++;

}

else if(j>R){

tmp[k] = p[i];

i++;

}

else {

int s1;

int s2;

if (flag == 0){

s1 = p[i].x;

s2 = p[j].x;

}

else{

s1 = p[i].y;

s2 = p[j].y;

}

if (s1 < s2){

tmp[k] = p[i];

i++;

}

else {

tmp[k] = p[j];

j++;

}

}

}

for(int k = L; k <=R; k++){

p[k] = tmp[k];

}

free(tmp);

}

void mergeSort(Point\* p, int L, int R, int flag){

if(L < R){

int M = (L+R)/2;

mergeSort(p, L, M, flag);

mergeSort(p, M+1, R, flag);

merge(p,L, M, R, flag);

}

}

//Tim khoang cach nho nhat o khoang giua

float stripClosest(Point\* pLR, int k, float d){

float min = d;

// mergeSort(pLR,0,k-1, 1);

for (int i = 0; i < k; ++i){

// Gioi han so diem can xet

for (int j = i+1; j < k && pLR[j].y - pLR[i].y < min; ++j){

if (dist(pLR[i],pLR[j]) < min){

min = dist(pLR[i],pLR[j]);

if (min < dmin){

dmin = min;

a.x = pLR[i].x; b.x = pLR[j].x;

a.y = pLR[i].y; b.y = pLR[j].y;

}

}

}

}

free(pLR);

return min;

}

float ClosestPair(Point\* p, int n){

if(n <= 3) return bruteForce(p, n);

//Chia mang thanh hai phan bang nhau

int mid = n/2;

Point midPoint = p[mid];

//De qui tim hai diem co khoang cach nho nhat

float dL = ClosestPair(p, mid);

float dR = ClosestPair(p + mid,n-mid);

float d = min(dL, dR); //Khoang cach nho nhat giua hai mang

//Tim tap cac diem co khang cach toi diem chinh giua nho hon d

Point\* pLR = (Point\*) malloc (n \* sizeof(Point));

if (pLR == NULL)

{

printf("Khong du bo nho\n");

exit(1);

}

int k = 0;

for (int i = 0; i < n; ++i){

if (abs(p[i].x - midPoint.x) <= d){

pLR[k] = p[i];

k++;

}

}

return min(stripClosest(pLR, k, d), d);

}

int main(int argc, char \*argv[]){

srand(time(NULL));

bool checkRead = false; // Bien kiem tra su ton tai cua du lieu

int choose; //Bien kiem tra lua chon dau vao lan 1

do{

// double y1, y2;

// system("clear");

system("cls");

printf("1. Doc du lieu dau vao\n");

printf("2. Giai quyet bai toan\n");

printf("3. Danh sach cac diem\n");

printf("4. Thoat\n");

do{

printf("Nhap vao lua chon cua ban: ");

scanf("%d", &choose);

}while(choose != 1 && choose !=2 && choose != 3 && choose != 4);

switch(choose){

case 1:{

// system("clear");

system("cls");

int chooseIn;

//Doc du lieu dau vao trong chuong trinh

printf("1. Doc du lieu tu file\n");

printf("2. Nhap du lieu tu ban phim\n");

printf("3. Sinh diem ngau nhien\n");

do{

printf("Xin moi lua chon: ");

scanf("%d", &chooseIn);

}

while(chooseIn != 1 && chooseIn != 2 && chooseIn != 3);

switch(chooseIn){

case 1:{

fflush(stdin);

printf("Nhap ten file du lieu dau vao: ");

scanf("%s",fileName);

readFile(fileName);

checkRead = true;

printf("Nhap thanh cong...");

getch();

break;

}

// Doc du lieu tu ban phim

case 2:{

printf("Nhap vao so diem: ");

scanf("%d", &count);

p = (Point\*) malloc (count \* sizeof(Point));

for (int i = 0; i < count; ++i)

{

printf("x[%d]: ",i+1);

scanf("%d", &p[i].x);

printf("y[%d]: ",i+1);

scanf("%d", &p[i].y);

}

fflush(stdin);

printf("Nhap ten file de luu: ");

scanf("%s",fileName);

savePoint(p, count, fileName);

//Bien kiem tra trang thai dau vao

checkRead = true;

printf("Nhap thanh cong...");

getch();

break;

}

case 3:{

printf("Nhap vap so diem: ");

scanf("%d", &count);

printf("Nhap ten file de luu: ");

scanf("%s",fileName);

p = (Point\*) malloc (count \* sizeof(Point));

srand(time(NULL));

generatePoint(p, count, fileName);

checkRead = true;

printf("Nhap thanh cong...");

getch();

break;

}

default: {

break;

}

}

break;

}

case 2:{

// system("clear");

system("cls");

if (checkRead == false)

{

printf("Chua doc du lieu dau vao...\n");

getch();

break;

}

int chooseAlgorithm;

printf("1. Brute Force\n");

printf("2. Divide and Conquer\n");

do{

printf("Xin moi lua chon thuat toan so: ");

scanf("%d", &chooseAlgorithm);

}while (chooseAlgorithm != 1 && chooseAlgorithm != 2);

// mergeSort(p,0,count-1, 1);

qsort(p, count, sizeof(Point), compareY);

bool checkSolution = false;

Timer ti;

double y;

switch(chooseAlgorithm){

case 1:{

bruteForce(p, count);

y = ti.getElapsedTime();

printf("Khoang cach gan nhat la: %3.2f\n", dmin);

printf("Hai diem gan nhau nhat la:\nA(%d, %d), B(%d, %d)\n",a.x,a.y,b.x,b.y);

printf("Thoi gian chay: %f",y);

checkSolution = true;

break;

}

case 2:{

ClosestPair(p, count);

y = ti.getElapsedTime();

printf("Khoang cach gan nhat la: %3.2f\n", dmin);

printf("Hai diem gan nhau nhat la:\nA(%d, %d), B(%d, %d)\n",a.x,a.y,b.x,b.y);

printf("Thoi gian chay: %f",y);

printf("\n");

checkSolution = true;

break;

}

default: {

printf("Ban khong lua chon thuat toan\n");

getch();

break;

}

}

if(checkSolution == true){

char saveResult[10] = "";

printf("\nBan co muon luu lai ket qua khong? c/k? ");

scanf("%s", saveResult);

if(strcmp(saveResult,"c") == 0 || strcmp(saveResult,"C") == 0 || strcmp(saveResult,"co") == 0){

char fileOut[30];

printf("Nhap ten file de luu ket qua: ");

scanf("%s", fileOut);

result(a, b, y, fileOut);

printf("Da luu ket qua vao file %s.out...", fileOut);

}

}

getch();

break;

}

case 3:{

// system("clear");

system("cls");

if (checkRead == false)

{

printf("Chua co danh sach! Hay tao du lieu dau vao...\n");

getch();

break;

}

printf("Co %d diem tren mat phang\n", count);

printPoint(p);

getch();

break;

}

}

}while(choose != 4);

free(p);

return 0;

}

**Thư viện Timer.h**

/\*

\* Main author(s) : Quang Dung PHAM

\*

\* email: quang.pham@uclouvain.be

\*

\* Copyright 2011 Quang Dung PHAM

\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\*/

#ifndef TIMER\_H

#define TIMER\_H

using namespace std;

#include <sys/time.h>

class Timer{

public:

timeval \_start\_t;

Timer();

~Timer();

double getElapsedTime();

};

#endif

**File Timer.cpp**

/\*

\* Main author(s) : Quang Dung PHAM

\*

\* email: quang.pham@uclouvain.be

\*

\* Copyright 2011 Quang Dung PHAM

\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\*/

using namespace std;

#include "Timer.h"

#include <cstddef>

Timer::Timer(){

gettimeofday(&\_start\_t,NULL);

}

Timer::~Timer(){

}

double Timer::getElapsedTime(){

timeval t;

gettimeofday(&t,NULL);

double elapsedtime = (t.tv\_sec - \_start\_t.tv\_sec) \* 1000.0;

elapsedtime += (t.tv\_usec - \_start\_t.tv\_usec) / 1000.0;

elapsedtime = elapsedtime\*0.001;

/\*double elapsedtime = (t.tv\_sec - \_start\_t.tv\_sec);

elapsedtime += (t.tv\_usec - \_start\_t.tv\_usec) / 1000000.0;\*/

return elapsedtime;

}

## **Phần 6: Tài liệu tham khảo**

Trong quá trình tìm hiểu và cài đặt thuật toán em có tham khảo một số tài liệu sau:

* Tài liệu của PGS TS Nguyễn Đức Nghĩa.
* Slide bài giảng môn học “Cấu trúc dữ liệu và giải thuật” của thầy Phạm Quang Dũng trường DHBKHN.
* Cuốn sách “Algorithm Design” của Jon KleinBerg.

Và một số tài liệu trên Internet như:

* <https://www.cs.ucsb.edu/~suri/cs235/ClosestPair.pdf>
* <http://www.geeksforgeeks.org/closest-pair-of-points>
* <https://rosettacode.org/wiki/Closest-pair_problem>