Cho một tập điểm trong mặt phẳng, và bài toán là tìm cặp điểm gần nhất trong tập điểm đó. Bài toán này phát sinh trong một số ứng dụng. Ví dụ, trong kiểm soát không lưu, chúng ta cần theo dõi các máy bay mà chúng di chuyển quá gần nhau, vì điều này có thể dẫn đến va chạm.

Chúng ta có công thức tính khoảng cách giữa hai điểm *p* và *q* như sau:

.

Với chiến lược trực tiếp, chúng ta sẽ tính khoảng cách giữa từng cặp điểm và trả về khoảng cách ngắn nhất, độ phức tạp theo thời gian của chiến lược trực tiếp là *O(n2)*. Chúng ta có thể tính khoảng cách ngắn nhất với độ phức tạp *O(nlogn)* bằng cách sử dụng chiến lược chia để trị.

**Giải thuật**

**Input:** Mảng *P[]* gồm n điểm.

**Output:** Khoảng cách ngắn nhất giữa hai điểm trong tập điểm.

Bước tiền tiến trình, mảng**input**  được sắp xếp theo hoành độ.

1. Tìm điểm chính giữa trong mảng đã sắp xếp, tức lấy điểm *P[n/2]*.
2. Chia mảng thành hai phần. Một nửa gồm các điểm từ *P[0]* đến *P[n/2]*. Nửa thứ hai gồm các điểm từ *P[n/2+1]* đến *P[n-1]*.
3. Đệ qui tìm các khoảng cách ngắn nhất trong các mảng con, gọi các khoảng cách này là *dl* và *dl*. Chọn giá trị nhỏ nhất giữa *dl* và *dr* và đặt là *d.*
4. Từ ba bước trên, chúng ta có một cận trên của khoảng cách ngắn nhất là *d*. Chúng ta cần xem xét các cặp điểm mà một điểm nằm ở mảng trái và một điểm nằm ở mảng phải. Xem xét đường thẳng đứng đi qua *P[n/2]* và tìm tất cả các điểm có hoành độ tới đường thẳng đó nhỏ hơn *d* và xây dựng thành mảng, đặt tên là *strip.*
5. Sắp xếp mảng *strip* tăng dần theo tung độ. Bước này là *O(nlogn)*. Nó có thể được tối ưu thành *O(n)* bằng cách dùng đệ qui *merge sort*.
6. Tìm khoảng cách ngắn nhất trong mảng *strip*.
7. Cuối cùng trả về khoảng cách ngắn nhất *d*, là khoảng cách đã được tính ở bước 6.

**Caution**

**-**Điểm trùng nhau, khoảng cách bằng 0 -> bỏ qua