

CHIẾN LƯỢC CHIA ĐỂ TRỊ

- Các đặc trưng cơ bản
- Các ví dụ minh họa

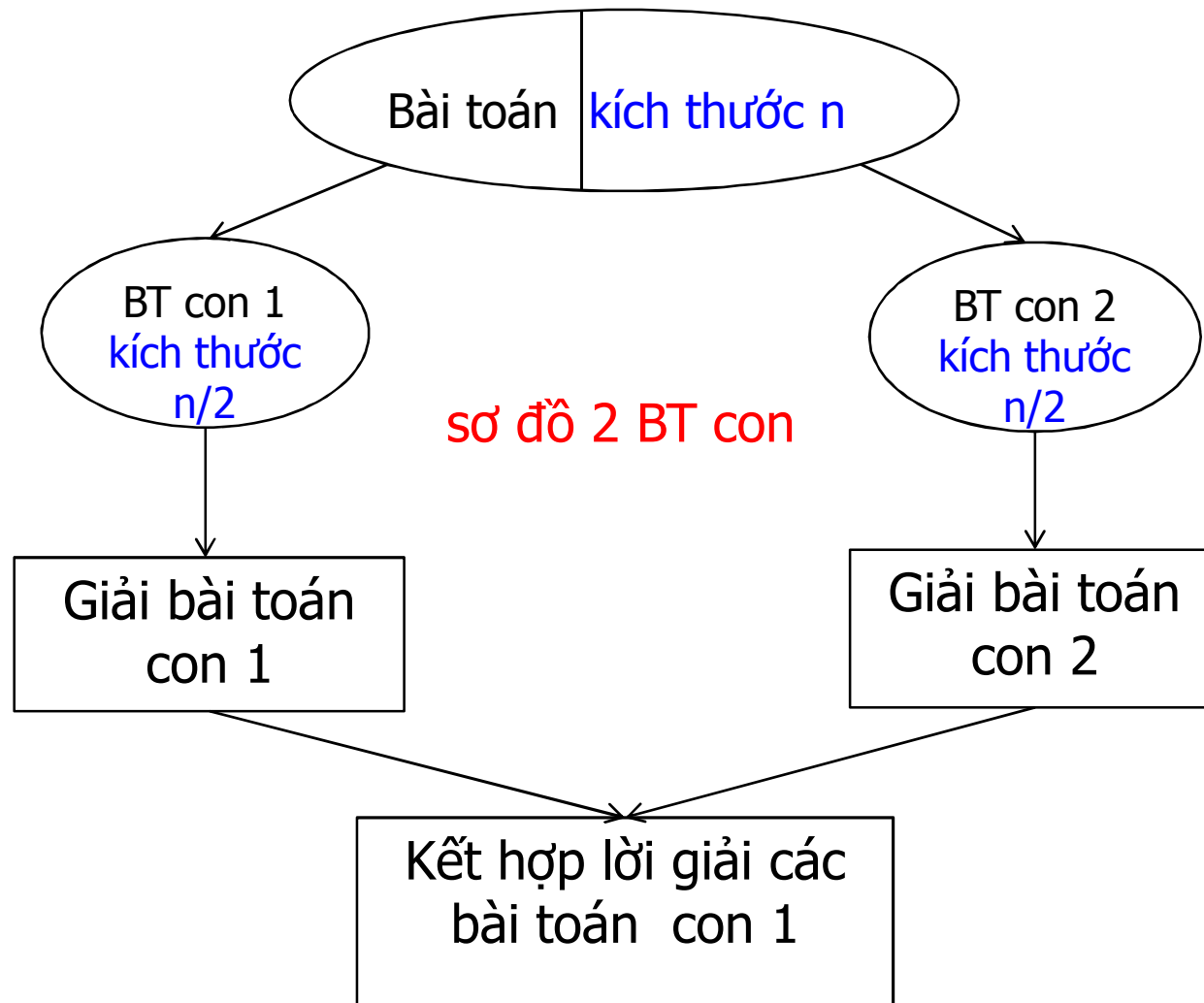
CÁC ĐẶC TRƯNG CƠ BẢN

- Là chiến lược thiết kế được áp dụng rất rộng rãi trong giải toán nói chung và thiết kế giải thuật nói riêng

CÁC ĐẶC TRƯNG CƠ BẢN

- Giải thuật giải một bài toán dựa trên chiến lược chia để trị được phát triển theo các bước:
 - Chia bài toán thành các bài toán con, thường là cùng kiểu
 - Giải các bài toán con (thường dùng kỹ thuật đệ quy)
 - Kết hợp lời giải các bài toán con để có lời giải bài toán ban đầu

CÁC ĐẶC TRƯNG CƠ BẢN



CÁC VÍ DỤ MINH HỌA

- Tìm kiếm nhị phân
- Giải thuật MergeSort
- Giải thuật QuickSort (đọc Levitin)
- Bài toán nhân các số nguyên lớn
- Bài toán tìm cặp điểm gần nhau nhất

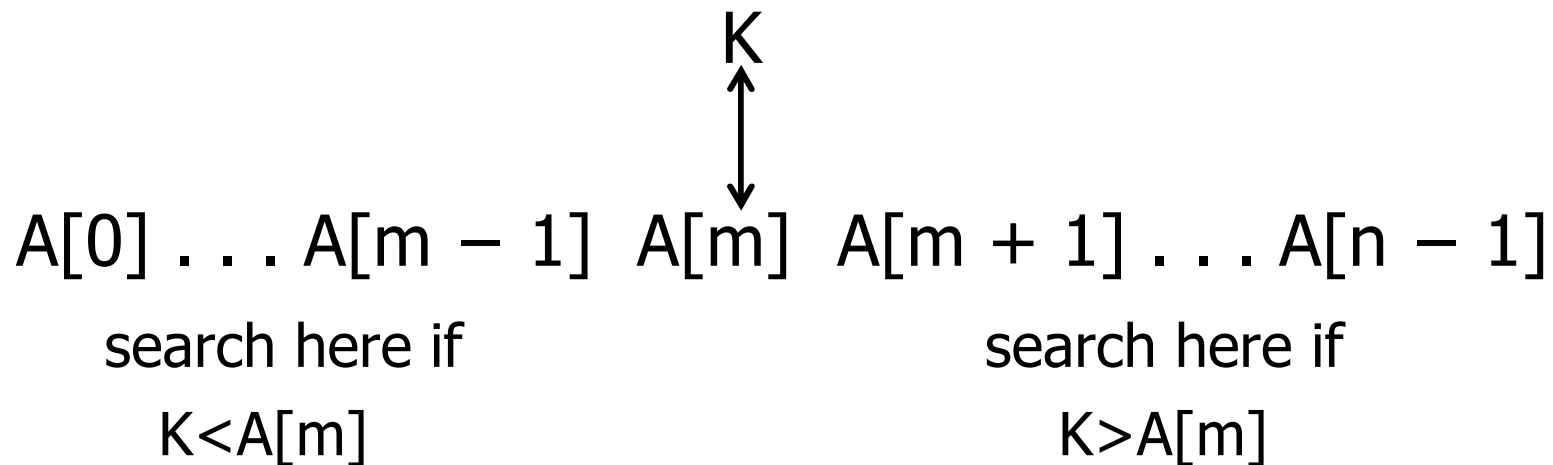
TÌM KIẾM NHỊ PHÂN

- Cho một mảng $A[0..n-1]$ các số được sắp theo thứ tự tăng và một số K , tìm trong mảng số có giá trị bằng K

TÌM KIẾM NHỊ PHÂN

- Giải thuật tìm kiếm dựa trên chiến lược chia để trị như sau
 - Chia mảng $A[0..n-1]$ thành 2 phần (thường chọn điểm chia m ở giữa mảng)
 - Nếu $A[m] = K$ thì kết thúc tìm kiếm
 - Nếu $K < A[m]$ tìm K trong mảng con bên trái (giải bài toán con)
 - Nếu $K > A[m]$ tìm K trong mảng con bên phải

TÌM KIẾM NHỊ PHÂN



TÌM KIẾM NHI PHÂN

ALGORITHM BinarySearch($A[0..n - 1]$, K)

```
1   $l \leftarrow 0$ ;  $r \leftarrow n - 1$ 
2  while  $l \leq r$  do
3       $m \leftarrow \lfloor (l + r) / 2 \rfloor$ 
4      if  $K = A[m]$  return  $m$ 
5      else if  $K < A[m]$   $r \leftarrow m - 1$ 
6          else  $l \leftarrow m + 1$ 
7  return  $-1$ 
```

TÌM KIẾM NHI PHÂN

- Tính độ phức tạp thời gian
 - Kích thước đầu vào là n
 - Thao tác cơ bản là so sánh trong lệnh 2 (giả sử thời gian là c)
 - Với kích thước đầu vào n , số phép so sánh phụ thuộc vào n và chính các phần tử của mảng vì vậy cần tính 3 trường hợp, tốt nhất, xấu nhất và trung bình

TÌM KIẾM NHI PHÂN

- Trường hợp tốt nhất
 - Khi $A[m]=K$, trong lần so sánh đầu tiên
 - $T(n) = c = \Theta(1)$ (hoặc $O(1)$)

TÌM KIẾM NHI PHÂN

- Trường hợp xấu nhất
 - Gọi s là số lần so sánh nhiều nhất, thì suy ra $n/2^{s-1} \geq 1$
 - Từ đó $n \geq 2^{s-1}$ hay $s \leq (\log_2 n) + 1$
 - $T(n) = O(\log_2 n)$

TÌM KIẾM NHI PHÂN

- Trường hợp trung bình
 - Số lần so sánh trung bình khi tìm thấy thành công là $s \approx \log_2 n - 1$ và không tìm thấy là $s \approx \log_2 n + 1$
 - $T(n) = O(\log n)$

GIẢI THUẬT MERGESORT

- Giải thuật MergeSort sử dụng chiến lược **chia để trị**, sắp xếp một mảng tăng dần như sau
 - Chia mảng $A[p..r]$ thành 2 mảng con $A[p..\lfloor (p+r)/2 \rfloor]$ và $A[\lfloor (p+r)/2 \rfloor + 1..r]$
 - **Sắp xếp** các mảng con
 - Kết hợp (**bằng cách trộn**) các phần tử của các mảng con thành mảng được sắp

GIẢI THUẬT MERGESORT

MERGE-SORT(A, p, r)

1 **if** $p < r$

2 **then** $q \leftarrow \lfloor (p + r)/2 \rfloor$

3 MERGE-SORT(A, p, q)

4 MERGE-SORT($A, q + 1, r$)

5 MERGE(A, p, q, r)

GIẢI THUẬT MERGESORT

```
MERGE( $A, p, q, r$ )
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  create arrays  $L[1 \dots n_1 + 1]$  and  $R[1 \dots n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
14         then  $A[k] \leftarrow L[i]$ 
15              $i \leftarrow i + 1$ 
16         else  $A[k] \leftarrow R[j]$ 
17              $j \leftarrow j + 1$ 
```


GIẢI THUẬT MERGESORT

- Đặt $n=r-p+1$
- Dễ thấy thời gian chạy của $\text{Merge}(A, p, q, r)$ là $O(n)$
- Gọi $T(n)$ là thời gian chạy của giải thuật $\text{Mergesort}(A, p, r)$ thì

$$T(n) = \begin{cases} O(1), n = 1 \\ 2T(n/2) + O(n), n > 1 \end{cases}$$

GIẢI THUẬT MERGESORT

Giải sử $n=2^k$, coi $O(1)=1$ và $O(n) = n$, thì

$$T(n) = 2T(n/2) + n = 2(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n$$

$$= 2^2(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n$$

$$= \dots$$

$$= 2^kT(n/2^k) + kn$$

$$= n + (\log_2 n) n = O(n \log_2 n)$$

GIẢI THUẬT MERGESORT

Áp dụng định lý Master (hệ thức truy hồi chia để trị), ta cũng có

$$T(n) = O(n \log_2 n)$$

BÀI TOÁN NHÂN CÁC SỐ NGUYÊN LỚN

- Cho 2 số nguyên a và b , mỗi số có n chữ số, n lớn và chẵn hãy tính tích $a*b$

BÀI TOÁN NHÂN CÁC SỐ NGUYÊN LỚN

- Mỗi số nguyên a có n chữ số (n chẵn) có thể biểu diễn

$$a = r_{n-1}10^{n-1} + r_{n-2}10^{n-2} + \dots + r_{n/2}10^{n/2} + r_{n/2-1}10^{n/2-1} + \dots + r_110 + r_0$$

$$= \underbrace{(r_{n-1}r_{n-2}\dots r_{n/2})}_{n/2 \text{ chữ số}} 10^{n/2} + \underbrace{r_{n/2-1}\dots r_1r_0}_{n/2 \text{ chữ số}}$$

$n/2$ chữ số

$n/2$ chữ số

Ký hiệu $a_1 = r_{n-1}r_{n-2}\dots r_{n/2}$ và $a_0 = r_{n/2-1}\dots r_1r_0$, thì

$$a = a_110^{n/2} + a_0$$

BÀI TOÁN NHÂN CÁC SỐ NGUYÊN LỚN

- Viết b dưới dạng $b = b_1 10^{n/2} + b_0$
$$c = a * b = (a_1 10^{n/2} + a_0) * (b_1 10^{n/2} + b_0)$$
$$= (a_1 * b_1) 10^n + (a_1 * b_0 + a_0 * b_1) 10^{n/2} + (a_0 * b_0)$$
$$= c_2 10^n + c_1 10^{n/2} + c_0$$

Trong đó

$$c_2 = a_1 * b_1$$

$$c_0 = a_0 * b_0$$

$$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$$

BÀI TOÁN NHÂN CÁC SỐ NGUYÊN LỚN

ALGORITHM Product(a, b, n)

```
1   $a \leftarrow r_{n-1}r_{n-2}\dots r_{n/2}r_{n/2-1}\dots r_1r_0$ ;  $b \leftarrow p_{n-1}p_{n-2}\dots p_{n/2}p_{n/2-1}\dots p_1p_0$ 
2  if  $n=1$  return  $r_0 * p_0$ 
3  else  $a_1 \leftarrow r_{n-1}r_{n-2}\dots r_{n/2}$ ;  $a_0 \leftarrow r_{n/2-1}\dots r_1r_0$ 
4          $b_1 \leftarrow p_{n-1}p_{n-2}\dots p_{n/2}$ ;  $b_0 \leftarrow p_{n/2-1}\dots p_1p_0$ 
5          $c_2 \leftarrow \text{Product}(a_1, b_1, n/2)$ ;  $c_0 \leftarrow \text{Product}(a_0, b_0, n/2)$ 
6          $A \leftarrow a_1 + a_0$ ;  $B \leftarrow b_1 + b_0$ ;
7          $c_1 \leftarrow \text{Product}(A, B, n/2) - (c_2 + c_0)$ 
8         return  $c_2 10^n + c_1 10^{n/2} + c_0$ 
```

BÀI TOÁN NHÂN CÁC SỐ NGUYÊN LỚN

- Gọi $T(n)$ là thời gian chạy của thuật giải $\text{Product}(a, b, n)$

$$T(n) = \begin{cases} O(1), n = 1 \\ 3T(n/2) + O(n), n > 1 \end{cases}$$

BÀI TOÁN NHÂN CÁC SỐ NGUYÊN LỚN

- Sử dụng định lý Master (đánh giá hệ thức truy hồi chia để trị) ta có

$$a=3, b=2, d=1$$

Vì $a=3 > 2^1=b^d$, nên $T(n)=O(n^{\log_b a})=O(n^{\log_2 3}) \approx O(n^{1.6})$

- Lưu ý: khi áp dụng chiến lược trực tiếp thì được giải thuật $O(n^2)$

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

- Cho P là một tập n điểm trong mặt phẳng Oxy, tìm cặp điểm có khoảng cách nhỏ nhất
- Lưu ý: Áp dụng chiến lược trực tiếp sẽ có giải thuật $O(n^2)$

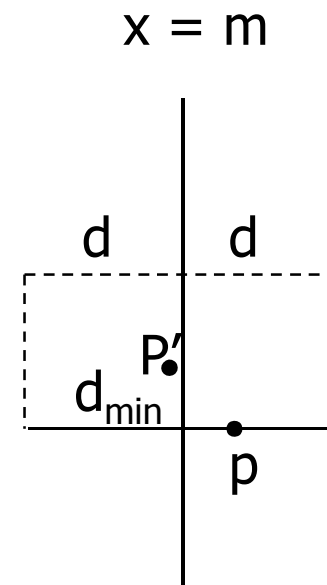
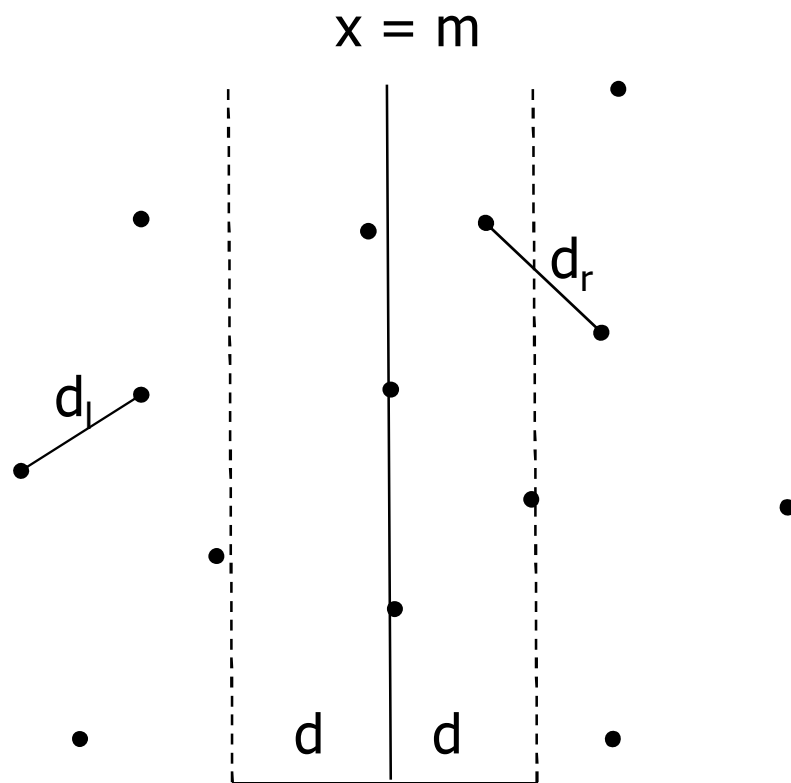
BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

- Áp dụng chiến lược chia để trị tìm cặp điểm gần nhau nhất như sau:
 - Chia tập điểm đã cho làm 2 tập
 - Tìm các cặp điểm gần nhau nhất trong mỗi tập
 - Tìm cặp điểm gần nhau nhất mà có các đỉnh (điểm) của nó tương ứng thuộc mỗi tập được chia
 - Tìm cặp nhỏ nhất trong 3 cặp trên

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

- Không mất tính tổng quát, giả sử P được sắp tăng theo hoành độ
- Gọi Q là tập điểm đã cho được sắp tăng theo tung độ y
- Chia P thành 2 tập con P_l và P_r tương ứng có $\lceil n/2 \rceil$ và $\lfloor n/2 \rfloor$ điểm bởi đường thẳng $x=m$
- Một số điểm của P_l và P_r có thể nằm trên đường thẳng $x=m$

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT



BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

- Gọi d_l và d_r là các khoảng cách ngắn nhất của các cặp tương ứng trong P_l và P_r , tìm $d = \min(d_l, d_r)$
- Tìm cặp điểm gần nhau nhất có 2 điểm tương ứng thuộc 2 phần đối nhau theo đường thẳng $x=m$ (các điểm này chỉ thuộc trong phần giữa 2 đường nét đứt), gọi khoảng cách giữa chúng là d_c
- Lấy $d_{\min} = \min(d, d_c)$

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

- Tìm d_c như sau:
 - Xét tập S các điểm nằm giữa 2 đường nét đứt, các điểm này sắp theo thứ tự tăng của tung độ y
 - Khởi đầu $d_{\min}=d$, gọi p là một điểm trong S , nếu có P' đi sau p mà hiệu các tung độ nhỏ hơn d_{\min} thì xác định lại d_{\min} là khoảng cách giữa p và p'

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

ALGORITHM EfficientClosestPair(P, Q, n)

```
1  if  $n \leq 3$ 
2      return the minimal distance found by the brute-force algorithm
3  else
4      copy the first  $\lceil n/2 \rceil$  points of  $P$  to array  $P_l$ 
5      copy the same  $\lceil n/2 \rceil$  points from  $Q$  to array  $Q_l$ 
6      copy the remaining  $\lfloor n/2 \rfloor$  points of  $P$  to array  $P_r$ 
7      copy the same  $\lfloor n/2 \rfloor$  points from  $Q$  to array  $Q_r$ 
8       $d_l \leftarrow \text{EfficientClosestPair}(P_l, Q_l, \lceil n/2 \rceil)$ 
9       $d_r \leftarrow \text{EfficientClosestPair}(P_r, Q_r, \lfloor n/2 \rfloor)$ 
10      $d \leftarrow \min\{d_l, d_r\}$ 
11      $m \leftarrow P[\lceil n/2 \rceil - 1].x$ 
```


BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

ALGORITHM EfficientClosestPair(P, Q, n) //tiếp theo

```
1  if  $n \leq 3$ 
2      return the minimal distance found by the brute-force algorithm
3  else
    -----
12     copy all the points of Q for which  $|x - m| < d$  into array  $S[0..num - 1]$ 
13      $dminsq \leftarrow d^2$ 
14     for  $i \leftarrow 0$  to  $num - 2$  do
15          $k \leftarrow i + 1$ 
16         while  $k \leq num - 1$  and  $(S[k].y - S[i].y)^2 < dminsq$ 
17              $dminsq \leftarrow \min((S[k].x - S[i].x)^2 + (S[k].y - S[i].y)^2, dminsq)$ 
18              $k \leftarrow k + 1$ 
19 return  $\sqrt{dminsq}$ 
```

BÀI TOÁN TÌM CẶP ĐIỂM GẦN NHAU NHẤT

- Gọi $T(n)$ là độ phức tạp của thuật giải, thì

$$T(n) = \begin{cases} O(1), n = 1 \\ 2T(n/2) + O(n), n > 1 \end{cases}$$

trong đó $O(n)$ là tổng chi phí của các lệnh copy và tìm d_{\min} theo tập S của các dòng lệnh 14-18

- Do $a=2$, $b=2$, $d=1$, $a=b^d$, nên

$$T(n) = O(n^d \log_2 n) = O(n \log_2 n)$$

BÀI TẬP VỀ NHÀ

- Đọc chương 5 (Divide -and -conquer), sách Levitin
- Đọc bài toán nhân **2 ma trận, QuickSort**
- Giải bài toán “tìm dãy con của một dãy có tổng lớn nhất” bằng **trực tiếp và chia để trị**
- **Làm bài tập về nhà đã cho trong DS bài tập**
- Bài thực Hành: MergeSort và tìm cặp điểm gần nhau nhất