**Bài 6**

1. Phương pháp vét cạn để tìm điểm lớn nhất là so sánh tất cả các cặp điểm. Cho biết độ phức tạp nếu làm theo cách này.

-Nếu sử dụng phương pháp vét cạn để tìm điểm lớn nhất bằng cách so sánh tất cả các cặp điểm, độ phức tạp của thuật toán sẽ là O(n2)O(n2), với nn là số lượng điểm. Điều này xuất phát từ việc phải so sánh mỗi điểm với tất cả các điểm còn lại trong tập hợp để xác định liệu nó có lớn nhất hay không.

+Vòng lặp ngoài: Trong thuật toán, có một vòng lặp duyệt qua từng điểm trong danh sách điểm. Vòng lặp này thực hiện nn lần, với nn là số lượng điểm.

+Vòng lặp bên trong: Trong mỗi vòng lặp ngoài, có một vòng lặp bên trong duyệt qua từng điểm khác trong danh sách điểm. Vòng lặp bên trong cũng thực hiện nn lần cho mỗi vòng lặp ngoài.

+So sánh điểm: Trong vòng lặp bên trong, chúng ta cần thực hiện một số lần so sánh để kiểm tra xem điểm đang xem xét có lớn nhất không. Đối với mỗi điểm, chúng ta cần so sánh với tất cả các điểm khác trong danh sách. Do đó, số lần so sánh trong vòng lặp bên trong là n−1n−1.

=> Tổng cộng, số lần so sánh được thực hiện là n×(n−1)n×(n−1), tức là O(n^2)O(n^2). Điều này là do mỗi điểm trong danh sách cần được so sánh với tất cả các điểm còn lại để xác định xem nó có lớn nhất hay không. Do đó, độ phức tạp của thuật toán vét cạn là O(n2)O(n2).

1. Thiết kế một thuật toán chia để trị để tìm điểm lớn nhất.

\*Hàm để so sánh và trả về điểm lớn hơn giữa hai điểm

Function maxPoint(p1, p2):

maxP.x = max(p1.x, p2.x)

=> Lấy giá trị lớn nhất của hai điểm theo trục x

maxP.y = max(p1.y, p2.y)

=>Lấy giá trị lớn nhất của hai điểm theo trục y

return maxP

\*Hàm để tìm điểm lớn nhất trong một tập hợp điểm bằng cách chia để trị

Function findMax(points, left, right):

=> Trường hợp cơ bản: Nếu chỉ có một điểm

If left equals right:

Return points[left]

=>Trả về điểm duy nhất

End If

=> Chia mảng thành hai phần bằng cách tính chỉ số trung bình

mid = left + (right - left) / 2

=>Tìm điểm lớn nhất trong nửa trái và nửa phải của mảng

maxLeft = findMax(points, left, mid)

maxRight = findMax(points, mid + 1, right)

=>Tìm điểm lớn nhất giữa hai điểm lớn nhất của hai nửa mảng

maxPoint = maxPoint(maxLeft, maxRight)

Return maxPoint

\*Ví dụ sử dụng:

points = [ {1, 2}, {5, 3}, {7, 8}, {4, 6}, {9, 1} ]

=> Tập hợp các điểm

maxPoint = findMax(points, 0, points.size() - 1)

=> Tìm điểm lớn nhất trong tập hợp

Print "Điểm lớn nhất:", maxPoint.x, maxPoint.y

=>In ra điểm lớn nhất

1. Viết phương trình truy hồi cho độ phức tạp của thuật toán và độ phức tạp của nó:

Phương trình truy hồi:

T(n) = 2T(n/2) + O(n)

Giải thích:

T(n): Độ phức tạp thời gian để tìm điểm lớn nhất trong tập hợp n điểm.

2T(n/2): Độ phức tạp thời gian để thực hiện hai cuộc gọi đệ quy trên hai tập hợp con có n/2 điểm.

O(n): Độ phức tạp thời gian để chia tập hợp và so sánh hai điểm lớn nhất.

Độ phức tạp:

Giải phương trình truy hồi T(n) = 2T(n/2) + O(n), ta có: T(n) = O(n log n).