## **Mục Lục**

1. Giới Thiệu
2. Linear Search
   * Ý tưởng thuật toán
   * Ưu điểm
   * Nhược điểm
   * Độ phức tạp
   * Cài đặt thuật toán (Python)
3. Binary Search
   * Ý tưởng thuật toán
   * Ưu điểm
   * Nhược điểm
   * Độ phức tạp
   * Cài đặt thuật toán (Python)
4. Bảng Tổng Kết So Sánh

## **2. Linear Search (Tìm kiếm tuyến tính)**

### **2.1 Ý tưởng thuật toán**

Linear Search là một phương pháp tìm kiếm tuần tự, trong đó chúng ta duyệt qua từng phần tử của danh sách và so sánh từng phần tử với giá trị cần tìm cho đến khi tìm thấy hoặc hết danh sách.

### **2.2 Ưu điểm**

* Đơn giản và dễ hiểu.
* Không yêu cầu danh sách phải được sắp xếp trước.
* Phù hợp với danh sách ngắn hoặc khi dữ liệu không được sắp xếp.

### **2.3 Nhược điểm**

* Hiệu suất kém với các danh sách dài, do phải kiểm tra từng phần tử.
* Độ phức tạp thời gian là tuyến tính.

### **2.4 Độ phức tạp**

* Thời gian: O(n), với n là số phần tử trong danh sách.
* Không gian: O(1), chỉ cần một vài biến để lưu trữ vị trí và giá trị cần tìm.

### **2.5 Cài đặt thuật toán (Python)**

python

def linear\_search(a, t):

for i in range(len(a)):

if a[i] == t:

return i

return -1

# Ví dụ sử dụng

a = [4, 2, 5, 1, 3]

t = 5

r = linear\_search(a, t)

print(r)

# Output: 2

## **3. Binary Search (Tìm kiếm nhị phân)**

### **3.1 Ý tưởng thuật toán**

Binary Search là một phương pháp tìm kiếm hiệu quả hơn, áp dụng trên danh sách đã được sắp xếp. Thuật toán này chia đôi danh sách và so sánh giá trị cần tìm với phần tử giữa. Nếu giá trị cần tìm nhỏ hơn, tiếp tục tìm kiếm trong nửa bên trái; nếu lớn hơn, tìm kiếm trong nửa bên phải. Lặp lại quá trình cho đến khi tìm thấy hoặc danh sách chỉ còn một phần tử.

### **3.2 Ưu điểm**

* Hiệu suất cao với các danh sách dài đã được sắp xếp.
* Độ phức tạp thời gian là logarithmic.

### **3.3 Nhược điểm**

* Yêu cầu danh sách phải được sắp xếp trước.
* Phức tạp hơn so với Linear Search về mặt cài đặt.

### **3.4 Độ phức tạp**

* Thời gian: O(log n), với n là số phần tử trong danh sách.
* Không gian: O(1), chỉ cần một vài biến để lưu trữ chỉ số và giá trị cần tìm.

### **3.5 Cài đặt thuật toán (Python)**

python

def binary\_search(a, t):

l, r = 0, len(a) - 1

while l <= r:

m = (l + r) // 2

if a[m] == t:

return m

elif a[m] < t:

l = m + 1

else:

r = m - 1

return -1

# Ví dụ sử dụng

a = [1, 2, 3, 4, 5]

t = 5

r = binary\_search(a, t)

print(r)

# Output: 4

## **4. Bảng Tổng Kết So Sánh**

| **Tiêu chí** | **Linear Search** | **Binary Search** |
| --- | --- | --- |
| **Ý tưởng thuật toán** | Duyệt tuần tự qua từng phần tử | Chia đôi danh sách và tìm kiếm |
| **Ưu điểm** | Đơn giản, không yêu cầu sắp xếp | Hiệu suất cao với danh sách sắp xếp |
| **Nhược điểm** | Hiệu suất kém với danh sách dài | Yêu cầu danh sách sắp xếp, cài đặt phức tạp hơn |
| **Độ phức tạp thời gian** | O(n) | O(log n) |
| **Độ phức tạp không gian** | O(1) | O(1) |