Nguyễn Trung Sơn – 2154050266 – IT2101 – Chương 3

**CÂU HỎI BÀI TẬP CHƯƠNG 3**

Câu 1: Trong các phương pháp xếp thứ tự đã học, phương pháp nào tối ưu nhất, và kém tối ưu nhất? Tại sao?

* Phướng pháp tối ưu nhất: Bubble sort, Insertion sort.

Vì nó độ phức tạp là O(n)

* Phương pháp kém tối ưu nhất: Selection sort

VÌ nó độ phức tạp là O(n2)

Câu 2: Trong các phương pháp tìm kiếm đã học, trường hợp nào thì cả 02 phương pháp đều như nhau? Giải thích tại sao?

* Trong 2 phương pháp tìm kiếm đã được học. Trường hợp cả 2 phương pháp đều như nhau là khi giá trị trong mảng có 1 giá trị hoặc mảng đó rỗng.

Câu 3: Ngoài các phương pháp xếp thứ tự đã học, hãy tìm hiểu thêm một phương pháp xếp thứ tự khác, giới thiệu sơ và giải thích.

* Tìm kiếm nội suy

Tìm kiếm nội suy (interpolation search) là một thuật toán cải tiến từ thuật toán tìm kiếm nhị phân. Thay vì xác định điểm chính giữa của danh sách, thuật toán tìm kiếm nội suy xác định điểm gần với vị trí của phần tử cần tìm, do đó tối ưu được thời gian hơn so với thuật toán tìm kiếm nhị phân. Độ phức tạp thời gian cũng vì thế mà tốt hơn là O(log(log(n))).

Tuy nhiên, thuật toán tìm kiếm nhị phân luôn ổn định với độ phức tạp thời gian là O(log(n)), thuật toán tìm kiếm nội suy lại không như vậy. Trong những trường hợp xấu nhất như dãy tăng/giảm phân bố không đều, thuật toán tìm kiếm này đạt độ phức tạp là O(n), không khác gì dùng thuật toán tìm kiếm tuyến tính cả. Do đó, bạn nên sử dụng thuật toán tìm kiếm nhị phân để đảm bảo được độ phức tạp O(log(n)).

Vẫn là mảng A, vẫn n phần tử bắt đầu từ 0 và tăng dần. Tìm x trong mảng này dùng thuật toán tìm kiếm nội suy như sau:

1. Gán left = 0, right = n – 1.
2. Gán mid = left + (right – left) \* (x – a[left]) / (a[right] – a[left]):
   * Nếu như A[mid] == x:
     + Dừng lại và trả về giá trị của mid.
   * Nếu như A[mid] > x:
     + right = mid – 1
   * Nếu như A[mid] < x:
     + left = mid + 1
3. Nếu left <= right và x >= A[left] và x <= A[right] (x còn nằm trong đoạn [A[left]; A[right]]):
   * Đúng thì quay lại bước 2.
   * Sai thì dừng và trả về kết quả -1 (không tìm thấy x)

Và tương tự với mảng giảm dần, bạn chỉ cần sửa lại:

* A[mid] < x:
  + right = mid – 1
* A[mid] > x:
  + left = mid + 1

Giờ hãy cùng xem cài đặt thuật toán tìm kiếm nội suy trong C++ với mảng tăng nha:

int InterpolationSearch(int A[], int n, int x)

{

int left = 0;

int right = n - 1;

int mid;

while (left <= right && x >= A[left] && x <= A[right])

{

mid = left + (right - left) \* (x - A[left]) / (A[right] - A[left]);

if (A[mid] == x)

return mid;

if (A[mid] > x)

right = mid - 1;

else if (A[mid] < x)

left = mid + 1;

}

return -1; // Không tìm thấy x

}