Tên: Trần Hoàng Quân

Mssv: 2151050355

Trả lời câu hỏi 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?

-Trả lời:

Với kích thước dữ liệu đầu vào nhỏ(3000) nhìn chung tốc độ chênh lệch của các thuật toán là không rõ để nhận thấy.

Với mảng đã được sắp xếp, thì Bubble Sort cho tốc độ nhanh nhất do chi phí để biết được đây là mảng có thứ tự của 2 thuật toán trên là O(n).

Với mảng gần như đã được sắp xếp thì Insertion Sort và Binary Insertion Sort là những sự lựa chọn tốt nhất do số phép hoán đổi phải thực hiện ít.

Selection Sort cho tốc độ khá chậm trong đa số trường hợp do độ phức tạp luôn là O(n2), do đó Selection Sort chỉ nên dùng cho các trường hợp số lượng phần tử cần sắp xếp không quá nhiều.

Với mảng gần như đã được sắp xếp thì Shaker Sort cho tốc độ nhanh hơn đáng kể so với Bubble Sort, do thu hẹp được khoảng phải duyệt tiếp theo sau khi duyệt.

Heap Sort, Merge Sort và Quick Sort có tốc độ ổn định xuyên suốt cả 4 loại dữ liệu đầu vào.

**Câu 2:** Trong các 2 phương pháp tìm kiếm đã học,

trường hợp nào thì cả 02 phương pháp đều như

nhau? Giải thích tại sao?

-Trả lời:

**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.

-Trả lời:

**Thuật toán sắp xếp cocktail** là một cải tiến của [Bubble Sort](https://vi.wikipedia.org/wiki/S%E1%BA%AFp_x%E1%BA%BFp_n%E1%BB%95i_b%E1%BB%8Dt" \o "Sắp xếp nổi bọt). Sau khi đưa phần tử nhỏ nhất về đầu dãy, thuật toán sẽ giúp chúng ta đưa phần tử lớn nhất về cuối dãy. Do đưa các phần tử về đúng vị trí ở cả hai đầu nên thuật toán sắp xếp cocktail sẽ giúp cải thiện thời gian sắp xếp dãy số.

Lập trình:

void ShakerSort(int a[], int n){

int Left = 0;

int Right = n - 1;

int k = 0;

int i;

**while** (Left < Right){

**for** (i = Left; i < Right; i++){

**if** (a[i] > a[i + 1]){

swap(a[i], a[i + 1]);

k = i;

}

}

Right = k;

**for** (i = Right; i > Left; i--){

**if** (a[i] < a[i - 1]){

swap(a[i], a[i - 1]);

k = i;

}

}

Left = k;

}

}