# Bài Tập 1: Sắp Xếp Dãy Số Nguyên

Cho dãy n số nguyên sau: 39, 8, 5, 1, 3, 6, 9, 12, 4, 7, 10  
a. Mô phỏng các bước sắp xếp tăng dần dãy số trên bằng các giải thuật:  
- Sắp xếp đổi chỗ trực tiếp  
- Sắp xếp chọn trực tiếp  
- Sắp xếp chèn trực tiếp  
- Sắp xếp nổi bọt  
b. Cài đặt hoàn chỉnh các giải thuật trên bằng C++.  
c. Chứng minh độ phức tạp của các giải thuật.

## 1. Sắp xếp đổi chỗ trực tiếp (Interchange Sort)

Ý tưởng: Thuật toán Bubble Sort hoạt động bằng cách lặp đi lặp lại qua các phần tử trong mảng và hoán đổi các cặp phần tử liền kề nếu chúng không theo đúng thứ tự. Quá trình này lặp lại cho đến khi mảng được sắp xếp hoàn toàn.

Các bước mô phỏng:  
Bước 1: (39, 8, 5, 1, 3, 6, 9, 12, 4, 7, 10)  
Bước 2: (1, 8, 5, 39, 3, 6, 9, 12, 4, 7, 10)  
Bước 3: (1, 3, 5, 39, 8, 6, 9, 12, 4, 7, 10)  
Bước 4: (1, 3, 4, 39, 8, 6, 9, 12, 5, 7, 10)  
Bước 5: (1, 3, 4, 5, 8, 6, 9, 12, 39, 7, 10)  
Bước 6: (1, 3, 4, 5, 6, 8, 9, 12, 39, 7, 10)  
Bước 7: (1, 3, 4, 5, 6, 7, 9, 12, 39, 8, 10)  
Bước 8: (1, 3, 4, 5, 6, 7, 8, 12, 39, 9, 10)  
Bước 9: (1, 3, 4, 5, 6, 7, 8, 9, 39, 12, 10)  
Bước 10: (1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 39)

Code C++:  
void interchangeSort(int arr[], int n) {  
 for (int i = 0; i < n - 1; i++) {  
 for (int j = i + 1; j < n; j++) {  
 if (arr[i] > arr[j]) {  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
 }  
 }  
}

Độ phức tạp: Giả sử mảng có n phần tử, ta sẽ thực hiện:

Ở lần lặp thứ nhất: Duyệt qua toàn bộ mảng, thực hiện n-1 phép so sánh.

Ở lần lặp thứ hai: Thực hiện n-2 phép so sánh.

…

Ở lần lặp cuối cùng: Thực hiện 1 phép so sánh.

Tổng số phép so sánh là:

(n-1) + (n-2) + ….+ 1 = (n(n-1))/2

Đây là biểu thức bậc hai theo n, nên độ phức tạp thời gian trong trường hợp xấu nhất là:O(n^2)

Kết luận: Trường hợp tốt nhất (mảng đã sắp xếp): O(n).

Trường hợp trung bình và xấu nhất: O(n^2).

## 2. Sắp xếp chọn trực tiếp (Selection Sort)

Ý tưởng: Thuật toán sắp xếp chọn trực tiếp (Selection Sort) hoạt động bằng cách tìm phần tử nhỏ nhất (hoặc lớn nhất) trong mảng và đưa nó về đúng vị trí. Quá trình này lặp lại cho phần còn lại của mảng cho đến khi mảng được sắp xếp hoàn toàn.

Các bước mô phỏng:  
Bước 1: (1, 8, 5, 39, 3, 6, 9, 12, 4, 7, 10)  
Bước 2: (1, 3, 5, 39, 8, 6, 9, 12, 4, 7, 10)  
Bước 3: (1, 3, 4, 39, 8, 6, 9, 12, 5, 7, 10)  
Bước 4: (1, 3, 4, 5, 8, 6, 9, 12, 39, 7, 10)  
Bước 5: (1, 3, 4, 5, 6, 8, 9, 12, 39, 7, 10)  
Bước 6: (1, 3, 4, 5, 6, 7, 9, 12, 39, 8, 10)  
Bước 7: (1, 3, 4, 5, 6, 7, 8, 12, 39, 9, 10)  
Bước 8: (1, 3, 4, 5, 6, 7, 8, 9, 39, 12, 10)  
Bước 9: (1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 39)

Code C++:  
void selectionSort(int arr[], int n) {  
 for (int i = 0; i < n - 1; i++) {  
 int minIndex = i;  
 for (int j = i + 1; j < n; j++) {  
 if (arr[j] < arr[minIndex]) {  
 minIndex = j;  
 }  
 }  
 int temp = arr[i];  
 arr[i] = arr[minIndex];  
 arr[minIndex] = temp;  
 }  
}

Độ phức tạp:

Ở lần lặp đầu tiên: Tìm phần tử nhỏ nhất trong n phần tử, thực hiện n-1 phép so sánh.

Ở lần lặp thứ hai: Tìm phần tử nhỏ nhất trong n-1 phần tử, thực hiện n-2 phép so sánh.

…

Ở lần lặp cuối cùng: Chỉ cần thực hiện 1 phép so sánh.

Tổng số phép so sánh là:

(n-1) + (n-2) + …+ 1 = (n(n-1))/2

Độ phức tạp thời gian là: O(n^2)

Kết luận: Trường hợp tốt nhất, trung bình và xấu nhất đều là O(n^2).

3. Sắp xếp chèn trực tiếp (Insertion Sort)

Ý tưởng: Thuật toán sắp xếp chèn trực tiếp (Insertion Sort) sắp xếp từng phần tử bằng cách chèn nó vào vị trí thích hợp trong danh sách con đã được sắp xếp.

Các bước mô phỏng:  
Bước 1: (39, 8, 5, 1, 3, 6, 9, 12, 4, 7, 10)  
Bước 2: (8, 39, 5, 1, 3, 6, 9, 12, 4, 7, 10)  
Bước 3: (5, 8, 39, 1, 3, 6, 9, 12, 4, 7, 10)  
Bước 4: (1, 5, 8, 39, 3, 6, 9, 12, 4, 7, 10)  
Bước 5: (1, 3, 5, 8, 39, 6, 9, 12, 4, 7, 10)  
Bước 6: (1, 3, 5, 6, 8, 39, 9, 12, 4, 7, 10)  
Bước 7: (1, 3, 5, 6, 8, 9, 39, 12, 4, 7, 10)  
Bước 8: (1, 3, 5, 6, 8, 9, 12, 39, 4, 7, 10)  
Bước 9: (1, 3, 4, 5, 6, 8, 9, 12, 39, 7, 10)  
Bước 10: (1, 3, 4, 5, 6, 7, 8, 9, 12, 39, 10)  
Bước 11: (1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 39)

Code C++:  
void insertionSort(int arr[], int n) {  
 for (int i = 1; i < n; i++) {  
 int key = arr[i];  
 int j = i - 1;  
 while (j >= 0 && arr[j] > key) {  
 arr[j + 1] = arr[j];  
 j--;  
 }  
 arr[j + 1] = key;  
 }  
}

Độ phức tạp:

Vòng lặp ngoài chạy n-1 lần.

Vòng lặp trong tìm vị trí chèn phần tử, trong trường hợp xấu nhất sẽ chạy i lần với mỗi i từ 1 đến n-1.

Tổng số phép so sánh là:

1 + 2 + 3 + …+ (n-1) = (n(n-1))/2

Đây là một biểu thức bậc hai theo n, do đó độ phức tạp thời gian trong trường hợp xấu nhất là:O(n^2)

Tuy nhiên, trong trường hợp mảng gần như đã sắp xếp, số lần so sánh sẽ giảm xuống và độ phức tạp là O(n).

Kết luận:

Trường hợp xấu nhất: O(n^2).

Trường hợp tốt nhất: O(n).

4. Sắp xếp nổi bọt (Bubble Sort)

Ý tưởng: Thuật toán sắp xếp nổi bọt thực chất là một dạng cụ thể của sắp xếp đổi chỗ trực tiếp, nơi các phần tử lớn hơn “nổi” lên vị trí cuối mảng sau mỗi lần lặp.

Các bước mô phỏng:  
Bước 1: (8, 5, 1, 3, 6, 9, 12, 4, 7, 10, 39)  
Bước 2: (5, 1, 3, 6, 8, 9, 4, 7, 10, 12, 39)  
Bước 3: (1, 3, 5, 6, 8, 4, 7, 9, 10, 12, 39)  
Bước 4: (1, 3, 5, 6, 4, 7, 8, 9, 10, 12, 39)  
Bước 5: (1, 3, 5, 4, 6, 7, 8, 9, 10, 12, 39)  
Bước 6: (1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 39)

Code C++:  
void bubbleSort(int arr[], int n) {  
 for (int i = 0; i < n-1; i++) {  
 for (int j = 0; j < n-i-1; j++) {  
 if (arr[j] > arr[j+1]) {  
 int temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = temp;  
 }  
 }  
 }  
}

Độ phức tạp:

Vòng lặp ngoài chạy n-1 lần.

Vòng lặp trong chạy n-1, n-2, \dots, 1 lần.

Tổng số phép so sánh là:

(n-1) + (n-2) + …+ 1 = (n(n-1))/2

Bỏ qua hệ số và bậc thấp hơn, ta có độ phức tạp là:

O(n^2)

Kết luận:Trường hợp tốt nhất: O(n) (khi mảng đã sắp xếp).

Trường hợp trung bình và xấu nhất: O(n^2).