# 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: So sánh từng cặp phần tử, nếu không đúng thứ tự thì đổi chỗ.

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: O(n²) do phải duyệt qua mọi cặp phần tử.

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

Ý tưởng: Tìm phần tử nhỏ nhất và đưa về đầu dãy.

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: O(n²) do phải tìm phần tử nhỏ nhất trong từng lần lặp.

## 3. Sắp xếp chèn trực tiếp (Insertion Sort)

Ý tưởng: Chèn từng phần tử vào đúng vị trí trong danh sách con đã 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: O(n²) trong trường hợp xấu nhất.

## 4. Sắp xếp nổi bọt (Bubble Sort)

Ý tưởng: So sánh và đổi chỗ các cặp phần tử liền kề cho đến khi mảng được sắp xếp.

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: O(n²) do phải so sánh tất cả các phần tử.