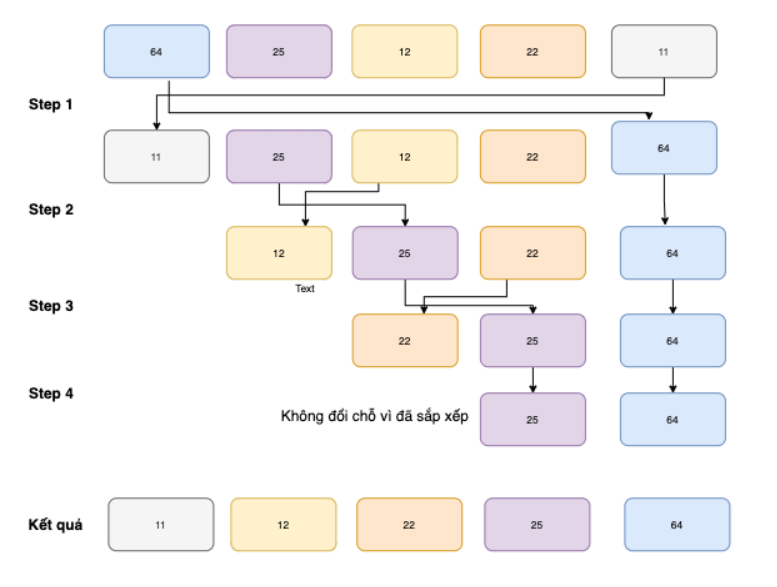
1. Selection Sort là gì? Hãy mô tả cách hoạt động của thuật toán này.
2. So sánh Selection Sort với Bubble Sort. Có điểm tương đồng và khác biệt nào giữa hai thuật toán này?
3. Thuật toán Selection Sort có thể được cài đặt như thế nào trong Java? Hãy cung cấp một đoạn mã mẫu.
4. Selection Sort có hiệu suất tốt không? Giải thích.
5. Selection Sort hoạt động như thế nào với một mảng đã sắp xếp trước đó? Có cách nào để tối ưu hóa thuật toán trong trường hợp này không?
6. Selection Sort có thời gian chạy là bao nhiêu? Giải thích về độ phức tạp thời gian của thuật toán này.
7. Có cách nào để cải thiện hiệu suất của Selection Sort không? Nếu có, hãy nêu ra và giải thích cách thực hiện nó.
8. Selection Sort có thể được sử dụng để sắp xếp các kiểu dữ liệu khác nhau, không chỉ số nguyên. Đúng hay sai? Giải thích.

Dưới đây là trả lời ngắn gọn cho các câu hỏi liên quan đến thuật toán Selection Sort trong Java:

1. Selection Sort là một thuật toán sắp xếp đơn giản. Nó hoạt động bằng cách tìm kiếm phần tử nhỏ nhất trong mảng và đưa nó vào vị trí đầu tiên. Sau đó, nó tìm phần tử nhỏ nhất trong phần còn lại của mảng và đưa vào vị trí thích hợp. Quá trình này được lặp lại cho đến khi mảng được sắp xếp hoàn toàn.



1. Selection Sort và Bubble Sort đều là thuật toán sắp xếp đơn giản. Tuy nhiên, Selection Sort hoán đổi phần tử ít lần hơn so với Bubble Sort. Trong Selection Sort, mỗi lần tìm kiếm phần tử nhỏ nhất, chỉ có một hoán đổi phần tử xảy ra. Trong khi đó, Bubble Sort có thể hoán đổi phần tử nhiều lần trong một vòng lặp.
2. Dưới đây là đoạn mã mẫu để triển khai Selection Sort trong Java:

java

Copy

public class SelectionSort {

public static void selectionSort(int[] arr) {

int n = arr.length;

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[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

}

}

public static void main(String[] args) {

int[] arr = {5, 2, 8, 12, 1};

selectionSort(arr);

System.out.println("Mảng sau khi sắp xếp:");

for (int num : arr) {

System.out.print(num + " ");

}

}

}

1. Hiệu suất của Selection Sort không tốt trong trường hợp mảng lớn, vì số lần so sánh và hoán đổi phần tử là tương đối nhiều. Độ phức tạp thời gian của thuật toán này là O(n^2), có nghĩa là số lần so sánh và hoán đổi phần tử tăng lên theo bình phương số phần tử trong mảng.
2. Khi mảng đã sắp xếp trước đó, Selection Sort vẫn phải tìm kiếm phần tử nhỏ nhất trong phần còn lại của mảng và đưa nó vào vị trí thích hợp. Tuy nhiên, có thể sử dụng một biến kiểm tra để xác định xem có cần hoán đổi phần tử trong một vòng lặp hay không. Nếu không có hoán đổi nào xảy ra, có thể kết thúc quá trình sắp xếp sớm.
3. Selection Sort có độ phức tạp thời gian là O(n^2), với n là số phần tử trong mảng. Điều này có nghĩa là thời gian thực thi của thuật toán tăng theo bình phương số lượng phần tử.
4. Selection Sort không có cách nào để cải thiện hiệu suất trong trường hợp tổng quát. Thuật toán này luôn phải tìm kiếm phần tử nhỏ nhất trong phần còn lại của mảng và hoán đổi nó, dẫn đến độ phức tạp n^2.
5. Selection Sort có thể được sử dụng để sắp xếp các kiểu dữ liệu khác nhau, không chỉ số nguyên. Điềuđó đúng. Thuật toán Selection Sort có thể được áp dụng cho các kiểu dữ liệu so sánh được. Trong Java, để sắp xếp các kiểu dữ liệu không phải nguyên (ví dụ: String), bạn cần đảm bảo rằng kiểu dữ liệu đó có thực hiện giao diện Comparable hoặc sử dụng một trình tự so sánh Comparator. Bạn có thể thay đổi phần so sánh trong thuật toán Selection Sort để phù hợp với kiểu dữ liệu mà bạn muốn sắp xếp.