Các Giải Thuật Sắp Xếp Nâng Cao

Mục tiêu

Nội dung các bài thực hành dưới đây nhằm củng cố kiến thức cho sinh viên về các giải thuật sắp xếp nâng cao. Ngoài ra, bài thực hành sẽ giúp mở rộng tư duy, khái quát hóa bài toán sắp xếp trên các kiểu dữ liệu khác nhau trong thực tế.

Nội dung

Câu 1, 2, 3, 4: Sinh viên tự thực hiện

Câu 5: Cho một danh sách các phần tử ngày/tháng/năm. Hãy viết chương trình sắp xếp lại danh sách này theo thứ tự giảm dần bằng các giải thuật:
a/ Giải thuật Shellsort
b/ Giải thuật Heapsort
c/ Giải thuật Quicksort
Sau cùng, hãy in từng danh sách đã sắp xếp ra màn hình theo qui tắc mỗi phần tử nằm trên một dòng.

Hướng dẫn:

Trong bài thực hành này, ta thấy dữ liệu nhập vào là một danh sách các giá trị tuân theo định dạng "ngày/tháng/năm" và được biểu diễn dưới dạng chuỗi (String), nên trong phần đọc dữ liệu INPUT của chương trình ta sẽ dùng biến kiểu chuỗi để lưu dữ liệu.

Trong phần xử lý của chương trình, nếu để các phần tử "ngày/tháng/năm" này dưới dạng chuỗi và đi sắp xếp thì thực ra ta chỉ sắp xếp được các chuỗi theo độ dài và theo thứ tự xuất hiện của các kí tự trong bảng mã ASCII chứ không phải là sắp xếp theo thứ tự về mặt thời gian của nó. Vì vậy, ta phải phân tách các chuỗi này ra thành các giá trị số nguyên tương ứng với ngày, tháng và năm được biểu diễn của nó. Và để dễ dàng cho việc lưu trữ và quản lý các giá trị ngày, tháng, năm này thì ta sẽ đi xây dựng một lớp MyDate nhận ngày, tháng, năm là các giá trị thuộc tính của lớp. Cách thức để tách các giá trị ngày, tháng, năm trong chuỗi "ngày/tháng/năm" ra thành các số nguyên là ta sẽ sử dụng phương thức split() có sẵn trên lớp String (Tham khảo: https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#split(java.lang.String)) để cắt chuỗi lớn thành các chuỗi con thông qua kí tự phân tách '/' và sau đó dùng phương thức parseInt() của lớp Integer

(Tham khảo: https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html#parseInt(java.lang.String)) để chuyển đổi các chuỗi con này thành các giá trị số nguyên.

Ở giai đoạn sắp xếp thì để việc sắp xếp đảm bảo theo đúng thứ tự về mặt thời gian của giá trị ta phải sắp xếp các đối tượng MyDate theo năm trước, rồi đến tháng và sau cùng là ngày.

Sau cùng, ta xuất ra mảng lưu trữ các đối tượng MyDate đã được sắp xếp ở trên theo thứ tự giảm dần và lần lượt với các thuật toán như ShellSort, HeapSort và QuickSort.

Chương trình:

Trong bài này để thuận tiện cho việc kiểm tra chương trình với nhiều dữ liệu INPUT, ta sẽ nhập xuất dữ liệu từ file (tập tin bên ngoài) thay vì từ môi trường Console như vẫn làm trước đây.

```
package Week4;
import java.util.Scanner;
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.io.FileWriter; // Ghi dữ liệu ra file

public class LabEx_Cau1 {

    private static class MyDate{
        private int day; // Ngày
        private int month; // Tháng
        private int year; // Năm

        // Phương thức dùng để so sánh giá trị ngày/tháng/năm của một đối tượng lớp
MyDate với một đối tượng khác thuộc lớp MyDate
        private int CompareTo(MyDate B) {
            if (this.year > B.year) return 1;
            else if (this.year < B.year) return -1;
            else // Trường hợp 2 năm bằng nhau ==> xét tiếp tới tháng
            {
                if (this.month > B.month) return 1;
                else if (this.month < B.month) return -1;
                else // Trường hợp 2 tháng bằng nhau ==> xét tiếp tới ngày
                {
                    if (this.day > B.day) return 1;
                    else if (this.day < B.day) return -1;
                    else return 0;
                }
            }
        }

        private String Display() {
            return this.day + "/" + this.month + "/" + this.year;
        }
    }
```

```
    private static MyDate[] list;

    public static void main(String[] args) {
        // Nhập dữ liệu từ file
        Scanner input = null;
        try {
            // Mở luồng nhập
            input = new Scanner(new File("D:\\data.input"));
        } catch (FileNotFoundException e) {
            e.printStackTrace(); // Xuất Exception nếu không tìm thấy file
        }

        int n = input.nextInt(); // Đọc số phần tử trong danh sách

        // Tạo mảng các đối tượng kiểu MyDate để lưu danh sách các phần tử
ngày/tháng/năm
        list = new MyDate[n];

        // Đọc từng phần tử trong danh sách
        String strDate = "";
        String[] arrDate;
        for (int i=0; i<n; i++) {
            strDate = input.next();

            // Cắt chuỗi ra thành các chuỗi con
            arrDate = strDate.split("/");

            // Vừa đọc, vừa xử lý và lưu trữ dữ liệu
            // Nếu cắt được thành 3 chuỗi con thì ta sẽ xử lý
            if (arrDate.length == 3) {
                list[i] = new MyDate();
                list[i].day = Integer.parseInt(arrDate[0].trim()); // Chuyển đổi
chuỗi thành số nguyên
                list[i].month = Integer.parseInt(arrDate[1].trim());
                list[i].year = Integer.parseInt(arrDate[2].trim());
            }
        }
        // Đóng luồng nhập
        input.close();

        // Sắp xếp các phần tử ngày/tháng/năm trong mảng list
        ShellSort(list, n);
        HeapSort(list, n);
        QuickSort(list, 0, n-1);

        // Xuất danh sách đã sắp xếp ra file OUTPUT
        try {
            // Mở luồng xuất
            FileWriter writer = new FileWriter("D:\\data.output");
            for (MyDate myDate : list) {
                writer.write(myDate.Display());
                writer.write("\n"); // Câu lệnh này để xuống dòng
            }
            // Đóng luồng xuất
            writer.close();
        } catch (Exception Ex) {
            Ex.printStackTrace();
        }
```

```
        System.out.println("Done!");
    }

    // Sắp xếp theo chiều giảm dần
    private static void ShellSort(MyDate[] list, int N) {
        int h = 1; // Độ dài bước nhảy
        while (h < N / 3) h = h * 3 + 1;
        int j;
        MyDate x;
        while (h > 0) {
            for (int i=h; i<N; i++) {
                x = list[i];
                j = i - h;
                // Giá trị của phương thức A.CompareTo(B) < 0 nghĩa là A < B
                while (j >= 0 && list[j].CompareTo(x) < 0) {
                    list[j + h] = list[j];
                    j = j - h;
                }
                list[j + h] = x;
            }
            h = (h - 1) / 3;
        }
    }

    // Sắp xếp theo chiều giảm dần
    public static void HeapSort(MyDate[] list, int N) {// Sắp xếp
        for (int i=N/2-1; i>=0; i--) {
            Heapify(list, N, i);
        }
        for (int i=N-1; i>=0; i--)
        {
            MyDate temp = list[0];
            list[0]= list[i];
            list[i] = temp;
            Heapify(list, i, 0);
        }
    }

    public static void Heapify(MyDate[] list, int N, int i) {// Vun đống
        int min = i;
        int left = 2 * i + 1;
        int right = 2 * i + 2;
        if (left < N && list[left].CompareTo(list[min]) < 0) min = left;
        if (right < N && list[right].CompareTo(list[min]) < 0) min = right;
        if (min != i)
        {
             MyDate temp = list[i];
             list[i]= list[min];
             list[min] = temp;
             Heapify(list, N, min);
        }
    }

    // Sắp xếp theo chiều giảm dần
    public static void QuickSort(MyDate[] A, int left, int right) {// Sắp xếp
        int i = left, j = right;
        int k = (left + right) / 2;
        MyDate x = A[k], temp;
```

```
        do {
            while (A[i].CompareTo(x) > 0) i++;
            while (A[j].CompareTo(x) < 0) j--;
            if (i <= j) {
                temp = A[i];
                A[i] = A[j];
                A[j] = temp;
                i++; j--;
            }
        } while (i < j);

        if (left < j) QuickSort(A, left, j);
        if (i < right) QuickSort(A, i, right);
    }
}
```

Câu 6: Tương tự như câu 1 nhưng các phần tử của danh sách có đầy đủ ngày/tháng/năm giờ:phút:giây. Hãy viết chương trình sắp xếp lại danh sách này theo thứ tự tăng dần bằng giải thuật SelectionSort hoặc BubleSort.

Hướng dẫn:

Cũng tương tự như câu 1, tuy nhiên ở câu 2 chuỗi thời gian của chúng ta có thêm giá trị giờ:phút:giây nên ta phải xây dựng một lớp mới có tên là MyTime gồm có 6 thuộc tính: (ngày, tháng, năm, giờ, phút, giây). Ngoài ra, ở thao tác cắt chuỗi, ta cần phải cắt 2 chuỗi ngày/tháng/năm và giờ:phút:giây dựa vào kí tự khoảng trắng trước, sau đó ta mới cắt từng chuỗi con ngày/tháng/năm dựa vào kí tự "/" và chuỗi con giờ:phút:giây dựa vào kí tự ":". Bên cạnh đó, khi so sánh giá trị của 2 đối tượng MyTime thì ta phải so sánh theo năm, tháng, ngày trước rồi nếu bằng nhau thì so sánh tiếp đến giờ, phút và giây.

Chương trình:

```
package Week4;
import java.util.Scanner;
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.io.FileWriter; // Ghi dữ liệu ra file

public class LabEx_Cau2 {

    private static class MyTime{
        private int dd; // Ngày
        private int MM; // Tháng
        private int yyyy; // Năm

        private int HH; // Giờ
        private int mm; // Phút
        private int ss; // Giây
```

```
        // Phương thức dùng để so sánh giá trị ngày/tháng/năm của một đối tượng lớp
MyDate với một đối tượng khác thuộc lớp MyDate
        private int CompareTo(MyTime B) {
            if (this.yyyy > B.yyyy) return 1;
            else if (this.yyyy < B.yyyy) return -1;
            else // Trường hợp 2 năm bằng nhau ==> xét tiếp tới tháng
            {
                if (this.MM > B.MM) return 1;
                else if (this.MM < B.MM) return -1;
                else // Trường hợp 2 tháng bằng nhau ==> xét tiếp tới ngày
                {
                    if (this.dd > B.dd) return 1;
                    else if (this.dd < B.dd) return -1;
                    else // Trường hợp 2 ngày bằng nhau ==> xét tiếp tới giờ
                    {
                        if (this.HH > B.HH) return 1;
                        else if (this.HH < B.HH) return -1;
                        else // Trường hợp 2 giờ bằng nhau ==> xét tiếp tới phút
                        {
                            if (this.mm > B.mm) return 1;
                            else if (this.mm < B.mm) return -1;
                            else // Trường hợp 2 phút bằng nhau ==> xét tiếp tới giây
                            {
                                if (this.ss > B.ss) return 1;
                                else if (this.ss < B.ss) return -1;
                                else return 0;
                            }
                        }
                    }
                }
            }
        }

        private String Display() {
            return this.dd + "/" + this.MM + "/" + this.yyyy + " " + this.HH + ":" +
this.mm + ":" + this.ss;
        }
    }

    private static MyTime[] list;

    public static void main(String[] args) {
        // Nhập dữ liệu từ file
        Scanner input = null;
        try {
            // Mở luồng nhập
            input = new Scanner(new File("D:\\data.input"));
        } catch (FileNotFoundException e) {
            e.printStackTrace(); // Xuất Exception nếu không tìm thấy file
        }

        int n = input.nextInt(); // Đọc số phần tử trong danh sách
        input.nextLine(); // thêm câu lệnh này để lấy kí tự xuống dòng của dòng đầu
tiên

        // Tạo mảng các đối tượng kiểu MyDate để lưu danh sách các phần tử
ngày/tháng/năm
        list = new MyTime[n];
```

```
        // Đọc từng phần tử trong danh sách
        String strDate = "";
        String[] arrFull, arrDate, arrTime;
        for (int i=0; i<n; i++) {
            strDate = input.nextLine();

            // Cắt chuỗi ra thành các chuỗi con
            arrFull = strDate.split(" "); // VD: 22/12/2020 22:10:07
            arrDate = arrFull[0].split("/"); // VD: 22/12/2020
            arrTime = arrFull[1].split(":"); // VD: 22:10:07

            // Vừa đọc, vừa xử lý và lưu trữ dữ liệu
            // Nếu cắt được thành 3 chuỗi con thì ta sẽ xử lý
            if (arrDate.length == 3) {
                list[i] = new MyTime();
                list[i].dd = Integer.parseInt(arrDate[0].trim()); // Chuyển đổi chuỗi
thành số nguyên
                list[i].MM = Integer.parseInt(arrDate[1].trim());
                list[i].yyyy = Integer.parseInt(arrDate[2].trim());

                list[i].HH = Integer.parseInt(arrTime[0].trim());
                list[i].mm = Integer.parseInt(arrTime[1].trim());
                list[i].ss = Integer.parseInt(arrTime[2].trim());
            }
        }
        // Đóng luồng nhập
        input.close();

        // Sắp xếp các phần tử ngày/tháng/năm trong mảng list
        SelectionSort(list, n);
        BubleSort(list, n);

        // Xuất danh sách đã sắp xếp ra file OUTPUT
        try {
            // Mở luồng xuất
            FileWriter writer = new FileWriter("D:\\data.output");
            for (MyTime myDate : list) {
                writer.write(myDate.Display());
                writer.write("\n"); // Câu lệnh này để xuống dòng
            }
            // Đóng luồng xuất
            writer.close();
        } catch (Exception Ex) {
            Ex.printStackTrace();
        }

        System.out.println("Done!");
    }

    // Sắp xếp theo chiều tăng dần
    private static void SelectionSort(MyTime[] A, int N) {
        int index;
        MyTime trungGian;
        for (int i=0; i<N; i++) {
            index = i;
            for (int j=i+1; j<N; j++)
            if (A[j].CompareTo(A[index]) < 0)
                index = j;
                // Hoán vị A[i] và A[index]
```

```
                trungGian = A[i];
                A[i] = A[index];
                A[index] = trungGian;
            }
        }

        // Sắp xếp theo chiều tăng dần
        private static void BubleSort(MyTime[] A, int N) {
            MyTime trungGian;
            for (int i=0; i<N; i++)
                for (int j=N-1; j>i; j--)
                if (A[j-1].CompareTo(A[j]) > 0) {
                    trungGian = A[j-1];
                    A[j-1] = A[j];
                    A[j] = trungGian;
                }
        }
}
```

Câu 7: Khoa có 1 dãy số gồm n số nguyên. Khoa muốn sắp xếp các phần tử của dãy số này theo các yêu cầu sau:

Từ trái qua phải:

- Các số nguyên dương xuất hiện theo giá trị tăng dần
- Các số nguyên âm xuất hiện theo giá trị giảm dần
- Không thay đổi vị trí của phần tử mang giá trị 0.
- Không thay đổi tính chất ở mỗi vị trị (nghĩa là nếu trước khi sắp xếp, vị trí i có giá trị nguyên âm thì sau khi sắp xếp vị trí i cũng phải mang giá trị âm. Tương tự, với vị trí i mang giá trị dương.

Cho trước một dãy số nguyên A, hãy sắp xếp các số trong A theo cách của Khoa. Sau cùng, in dãy đã sắp xếp ra màn hình, sau mỗi phần tử có đúng một khoảng trắng.

Hướng dẫn

Ý tưởng thực hiện bài này có thể được làm như sau:

Bước 1: duyệt tuần tự qua các phần tử trong mảng ban đầu, nếu gặp phần tử là số nguyên dương thì ta bỏ phần tử này qua một mảng số nguyên dương khác, nếu gặp phần tử là số nguyên âm thì ta đưa phần tử này qua một mảng số nguyên âm khác, gặp số 0 thì giữ nguyên.

Bước 2: Sắp xếp theo thứ tự tăng dần các phần tử trong mảng chứa số nguyên dương

Bước 3: Sắp xếp theo chiều giảm dần các phần tử trong mảng chứa số nguyên âm

Bước 4: duyệt trở lại các phần tử trên mảng ban đầu, nếu gặp phần tử là số nguyên > 0 thì ta sẽ lấy phần tử chưa xét trên mảng số nguyên dương thay vào, còn nếu gặp phần tử là số nguyên < 0 thì ta lấy phần tử chưa xét trên mảng số nguyên âm thế vào.

Bước 5: Xuất ra mảng ban đầu sau khi đã lấp hết các phần tử vào theo đúng thứ tự và kết thúc

Chương trình

```
package Week4;
import java.util.Scanner;

public class LabEx_Cau3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Nhập vào số lượng phần tử N: ");
        int N = input.nextInt();

        System.out.println("Nhập vào từng phần tử của dãy số: ");

        int[] A = new int[N];
        for (int i=0; i<N; i++) {
            System.out.print("A[" + i + "] = ");
            A[i] = input.nextInt();
        }

        int[] positiveNumber = new int[N]; // Mảng chứa các số nguyên dương
        int[] negativeNumber = new int[N]; // Mảng chứa các số nguyên âm

        // Duyệt qua từng phần tử của mảng để tách số nguyên dương và số nguyên âm
        int posLength = -1, negLength = -1;
        for (int i=0; i<N; i++) {
            if (A[i] > 0) {
                posLength++;
                positiveNumber[posLength] = A[i];
            } else if (A[i] < 0) {
                negLength++;
                negativeNumber[negLength] = A[i];
            }
        }

        SortASC(positiveNumber, 0, posLength); // Sắp xếp tăng dần
        SortDESC(negativeNumber, 0, negLength); // Sắp xếp giảm dần

        // Đưa các phần tử trở lại mảng A theo đúng thứ tự
        int j = -1, k = -1;
        for (int i=0; i<N; i++) {
            if (A[i] > 0) {
                j++;
                A[i] = positiveNumber[j];
            } else if (A[i] < 0) {
                k++;
                A[i] = negativeNumber[k];
            }
        }
```

```
        // Xuất ra mảng đã sắp xếp theo thứ tự của đề bài
        System.out.println("Mảng đã sắp xếp: ");
        for (int i=0; i<N; i++) {
            System.out.print(A[i] + " ");
        }
    }

    private static void SortASC(int[] A, int left, int right) {
        if (left > right) return;
        int i = left, j = right;
        int k = (left + right) / 2;
        int x = A[k];
        int temp;

        do {
            while (A[i] < x) i++;
            while (A[j] > x) j--;
            if (i <= j) {
                temp = A[i];
                A[i] = A[j];
                A[j] = temp;
                i++; j--;
            }
        } while (i < j);
        if (left < j) SortASC(A, left, j);
        if (i < right) SortASC(A, i, right);
    }

    private static void SortDESC(int[] A, int left, int right) {
        if (left > right) return;
        int i = left, j = right;
        int k = (left + right) / 2;
        int x = A[k];
        int temp;

        do {
            while (A[i] > x) i++;
            while (A[j] < x) j--;
            if (i <= j) {
                temp = A[i];
                A[i] = A[j];
                A[j] = temp;
                i++; j--;
            }
        } while (i < j);
        if (left < j) SortDESC(A, left, j);
        if (i < right) SortDESC(A, i, right);
    }
}
```