Data Structures and Algorithm Analysis

Lab 5, Elementary Sort.

Contents

Selection Sort

The algs4 Selection Sort:

https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Selection.java.html

Selection Sort Implementation

A simple implementation of selection sort:

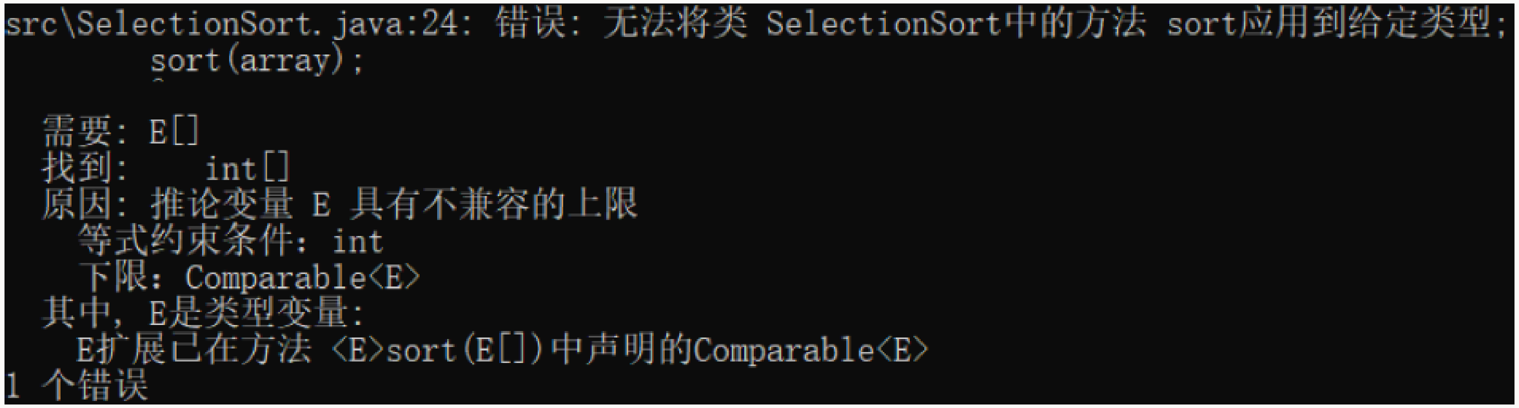
```
static <E> void exchange( E[] array, int i, int j ) {
  E tmp = array[i];
  array[i] = array[j];
  array[j] = tmp;
}
static<E extends Comparable <E>> void sort( E[] array ) {
  for( int i = 0; i < array.length; ++ i ) {
    int min = i;
    for( int j = i+1; j < array.length; ++ j ) {
      int compare = array[min].compareTo(array[j]);
      if( compare > 0 )
        min = j;
    }
    exchange(array, i, min);
  }
}
```

Selection Sort Implementation

However, when you call above code like this:

```
public static void main( String[] args ) {
  int[] array = new int[]{ 1, 5, 2, 4, 3 };
  sort(array);
}
```

You get compilation error like:



```
src\SelectionSort.java:24: 错误: 无法将类 SelectionSort中的方法 sort应用到给定类型;
        sort(array);
        ^
  需要: E[]
  找到:    int[]
  原因: 推论变量 E 具有不兼容的上限
    等式约束条件: int
    下限: Comparable<E>
  其中, E是类型变量:
    E扩展已在方法 <E>sort(E[])中声明的Comparable<E>
1 个错误
```

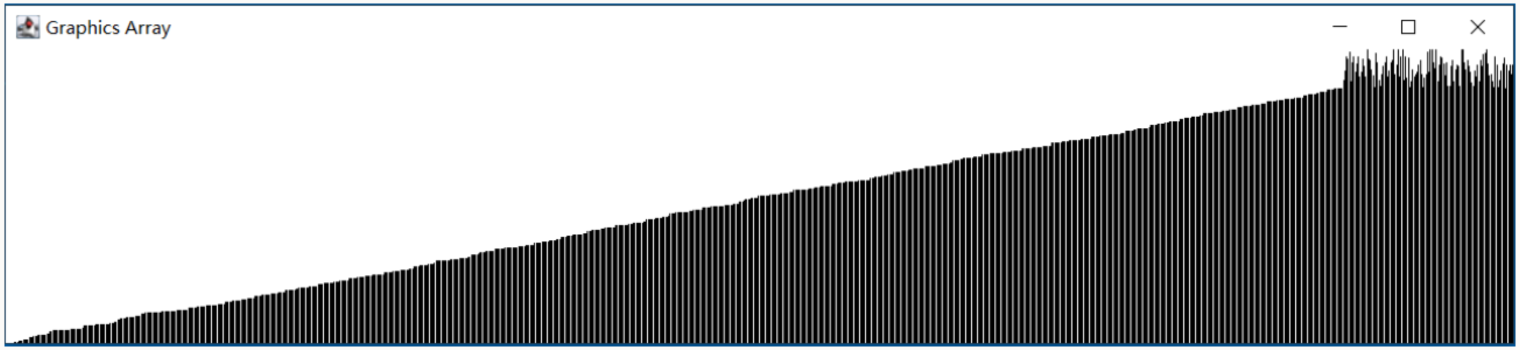
Selection Sort Implementation

The reason is that generic types are classes. In order to use int[], an integer sorting should also be implemented.

```
private static void exchange( int[] array, int i, int j ) {
  int tmp = array[i];
  array[i] = array[j];
  array[j] = tmp;
}
public static void sort( int[] array ) {
  for( int i = 0; i < array.length; ++ i ) {
    int min = i;
    for( int j = i+1; j < array.length; ++ j ) {
      if( array[min] > array[j] )
      min = j;
    }
    exchange (array, i, min );
  }
}
```

Visualize the Sorting Process

See SelectionSort.java and ArrayPanel.java in lab material.



Insertion Sort

Below is a simple implementation of insertion sort:

```
public static void sort( int[] array ) {
  for( int i = 1; i < array . length ; ++ i ) {
    for( int j = i; j > 0 && array[j] < array[j-1]; -- j )
      exchange (array , j -1, j);
  }
}
```

The advantage of insertion sort is that when array is partially sorted, the time spent on insertion sort could be close to linear.

Measuring the Performance of Sorting Algorithm

What was talked before about measuring performance:

- Try to generate some random data.
- Try to generate larger data size.
- ……?

What’s more?

Exercise 1: Dequeue Sort

The above algorithms is based on some assumptions, including that "random access of array data is very fast". Now you don't have this, try finish the exercise 2.1.14 in "Algorithm, 4th":

Explain how you would sort a deck of cards, with the restriction that the only allowed operations are to look at the values of the top two cards, to exchange the top two cards, and to move the top card to the bottom of the deck.

Download material from sakai. Write your code in "sort". Write your test in "main".

Exercise 2: Expensive exchange

Try to solve 2.1.15 in "Algorithm, 4th":

A clerk at a shipping company is charged with the task of rearranging a number of large crates in order of the time they are to be shipped out. Thus, the cost of compares is very low (just look at the labels) relative to the cost of exchanges (move the crates). The warehouse is nearly full—there is extra space sufficient to hold any one of the crates, but not two. What sorting method should the clerk use?