# 选择排序

## A.代码

public void selectSort(T[] a) {

T tmp = null;

int i = 0;

int j = 0;

int index = 0;

for (i = 0; i < a.length - 1; i++) {

index = i;

for (j = i + 1; j < a.length; j++) {

if (a[j].compareTo(a[index]) < 0)

index = j;

}

if (index != i) {

tmp = a[index];

a[index] = a[i];

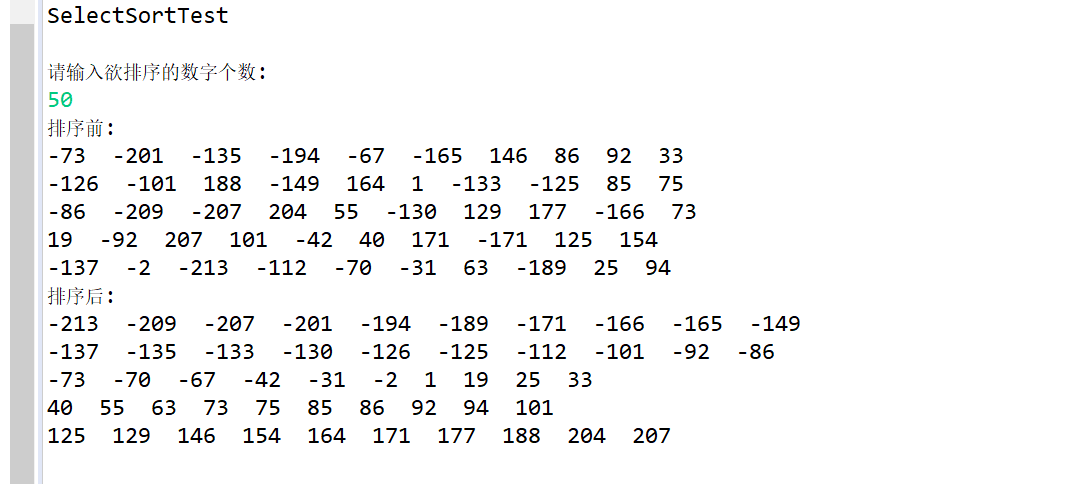
a[i] = tmp;

}

}

}

## B.运行结果



# 冒泡排序

## A.代码

public void bubbleSortsort(T[] a) {

T tmp;

for (int i = 0; i < a.length - 1; i++)

for (int j = 0; j < a.length - i - 1; j++) {

if (a[j].compareTo(a[j + 1]) > 0) {

tmp = a[j];

a[j] = a[j + 1];

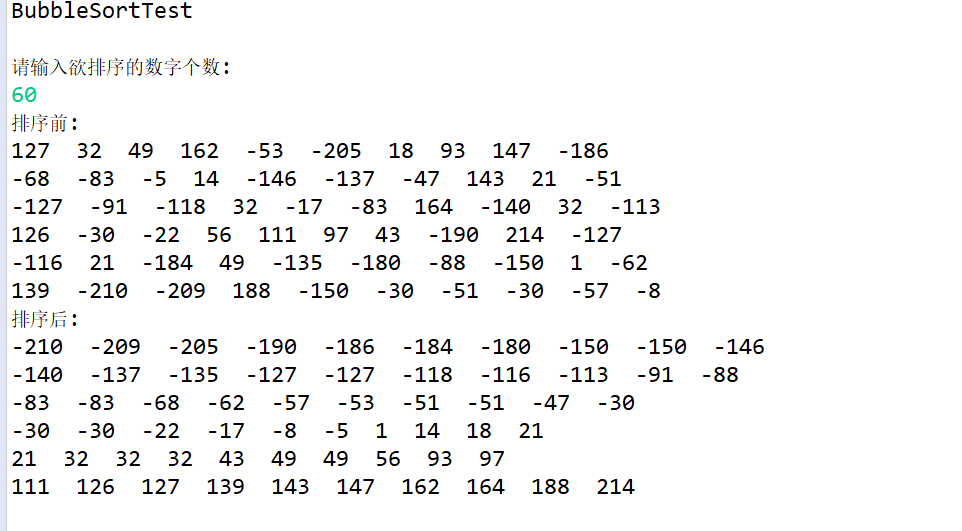
a[j + 1] = tmp;

}

}

}

## B.运行结果



# 插入排序

## A.代码

public void insetrSort(T[] a) {

// TODO Auto-generated method stub

T tmp;

int j;

for (int i = 1; i < a.length; i++) {

tmp = a[i];

for (j = i - 1; j >= 0; j--) {

if (a[j].compareTo(tmp) > 0)

a[j + 1] = a[j];

else {

break;

}

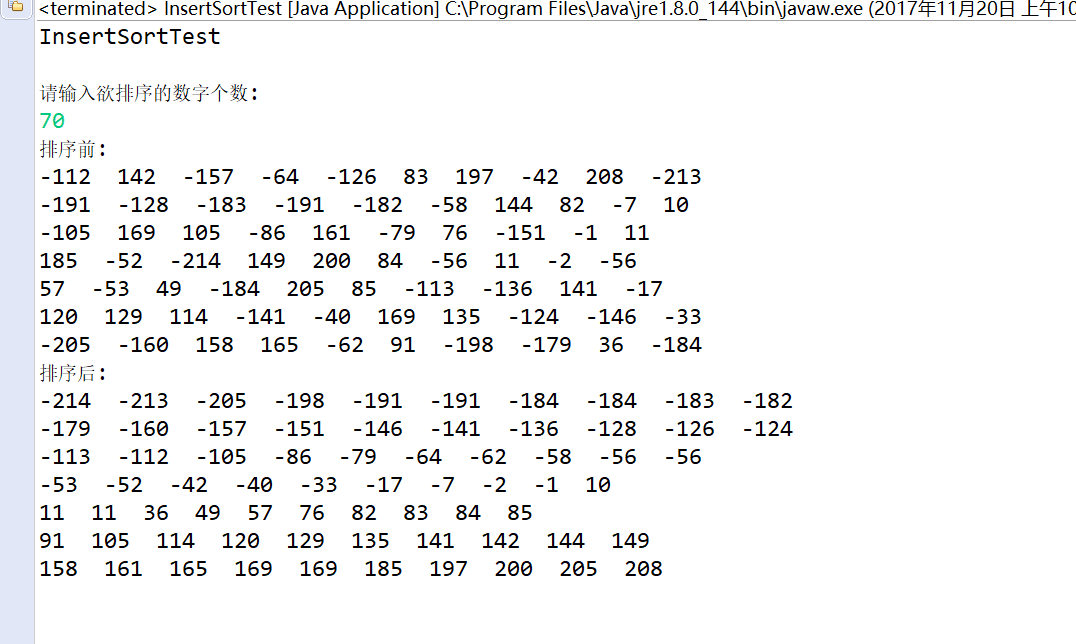
}

a[j + 1] = tmp;

}

}

## B.运行结果



# 快速排序

## A.代码

public void quicklySort(T[] a) {

quicklySort(a, 0, a.length - 1);

}

private void quicklySort(T[] a, int left, int right) {

if (left + CUTOOF <= right) {

T pivot = median(a, left, right);

int i = left;

// change

int j = right;

for (;;) {

while (a[++i].compareTo(pivot) < 0) {

}

while (a[--j].compareTo(pivot) > 0) {

}

if (i < j) {

swap(a, i, j);

} else {

break;

}

}

// change

swap(a, i, right);

quicklySort(a, left, i - 1);

quicklySort(a, i + 1, right);

} else {

insetrSort(a);

}

}

private void swap(T[] a, int m, int n) {

T tmp = a[m];

a[m] = a[n];

a[n] = tmp;

}

private T median(T[] a, int left, int right) {

int center = (left + right) / 2;

if (a[center].compareTo(a[left]) < 0) {

swap(a, left, center);

}

if (a[right].compareTo(a[left]) < 0) {

swap(a, left, right);

}

if (a[right].compareTo(a[center]) < 0) {

swap(a, center, right);

}

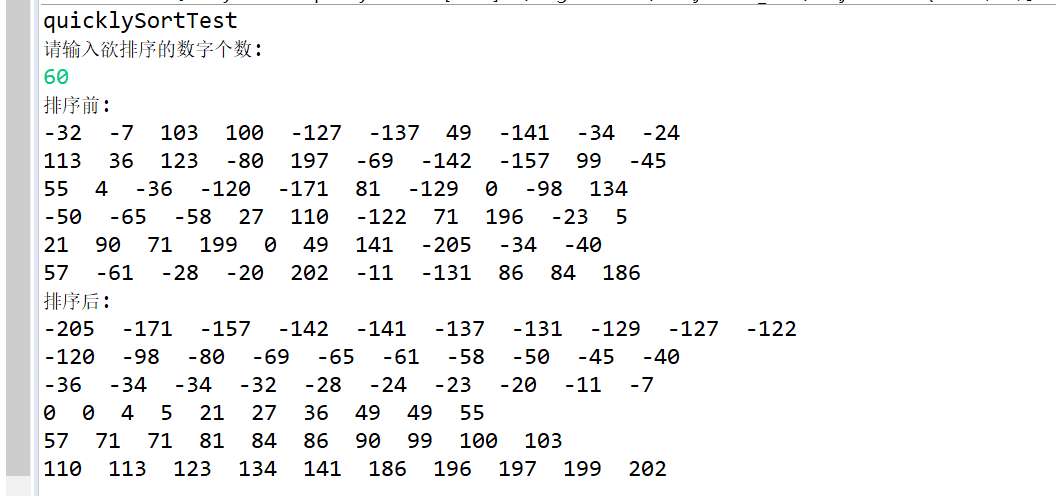
// Place pivot at position right - 1

swap(a, center, right);

return a[right];

}

## B.运行结果



# 归并排序

## A.代码

public void mergeSort(T[] a) {

@SuppressWarnings("unchecked")

T[] tmpArray = (T[]) new Comparable[a.length];

mergeSort(a, tmpArray, 0, a.length - 1);

}

private void mergeSort(T[] a, T[] tmpArray, int left, int right) {

if (left < right) {

int center = (left + right) / 2;

mergeSort(a, tmpArray, left, center);

mergeSort(a, tmpArray, center + 1, right);

merge(a, tmpArray, left, center + 1, right);

}

}

private void merge(T[] a, T[] tmpArray, int leftPos, int rightPos,

int rightEnd) {

int leftEnd = rightPos - 1;

int elementNums = rightEnd - leftPos + 1;

int index = leftPos;

for (; leftPos <= leftEnd && rightPos <= rightEnd;) {

if (a[rightPos].compareTo(a[leftPos]) < 0) {

tmpArray[index++] = a[rightPos++];

} else {

tmpArray[index++] = a[leftPos++];

}

}

while (rightPos <= rightEnd) {

tmpArray[index++] = a[rightPos++];

}

while (leftPos <= leftEnd) {

tmpArray[index++] = a[leftPos++];

}

for (int k = 0; k < elementNums; k++, rightEnd--) {

a[rightEnd] = tmpArray[rightEnd];

}

}

B.运行结果

