Selection, Insertion and Bubble Sort

Posted on December 29, 2012 by theoryapp

Sorting — arranging items in order — is the most fundamental task in computation. Sorting enables efficient searching algorithms such as binary search.

Selection, insertion and bubble sort are easily understandable and also similar to each other, but they are less efficient than merge sort or quick sort. The basic ideas are as below:

- **Selection sort**: repeatedly pick the smallest element to append to the result.
- **Insertion sort**: repeatedly add new element to the sorted result.
- **Bubble sort**: repeatedly compare neighbor pairs and swap if necessary.

Here we wish to sort an array of integers in Java; this can be easily extended to other data types and collections.

Selection Sort

Selection sort is to repetitively pick up the smallest element and put it into the right position:

- Find the smallest element, and put it to the first position.
- Find the next smallest element, and put it to the second position.
- Repeat until all elements are in the right positions.

A loop through the array finds the smallest element easily. After the smallest element is put in the first position, it is fixed and then we can deal with the rest of the array. The following implementation uses a *nested loop* to repetitively pick up the smallest element and swap it to

search here …

Go

Recent Posts

- A Double Sum of Binomials
- Chernoff Bound
- Store compressed data in database using PHP
- Format numbers with leading and trailing zeros in PHP
- Sperner Theorem on Maximal Antichains
- Kolmogorov Complexity
- Unix Commands for Large Files
- Get or Remove the

its final position. The `swap()` method exchanges two elements in an array.

```
public static void selectionSort(int[] arr)
{
    // find the smallest element starting from pos
    for (int i = 0; i < arr.length - 1; i++)
    {
        int min = i;  // record the position of th
        for (int j = i + 1; j < arr.length; j++)
        {
            // update min when finding a smaller e
            if (arr[j] < arr[min])
                min = j;
        }

        // put the smallest element at position i
        swap(arr, i, min);
    }
}
public static void swap (int[] arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

The following tracks the code on an array with elements `{38, 27, 43, 3, 9, 82, 10}`.

```
{38, 27, 43,  3,  9, 82, 10},   i: 0,   min: 3,   m
{ 3, 27, 43, 38,  9, 82, 10},   i: 1,   min: 4,   m
{ 3,  9, 43, 38, 27, 82, 10},   i: 2,   min: 6,   m
{ 3,  9, 10, 38, 27, 82, 43},   i: 3,   min: 4,   m
{ 3,  9, 10, 27, 38, 82, 43},   i: 4,   min: 4,   m
{ 3,  9, 10, 27, 38, 82, 43},   i: 5,   min: 6,   m
{ 3,  9, 10, 27, 38, 43, 82},   i: 6
```

Suppose the input array has *n* elements. The outer loop runs *n-1* rounds, roughly one for each position. Inside each outer loop, the inner loop goes through the unsorted part of the array. On average, each inner loop scans through *n/2* elements. The total time is roughly *t(n) = (n – 1) * (n/2) *k = O(n^2)*, where *k* denotes the number of basic operations inside each inner loop; the constants are absorbed in the big-Oh notion. Note that in the big-Oh notion, we only care about the dominating factor (which is *n^2* in this case).

Insertion Sort

Insertion sort maintains a sorted sub-array, and repetitively inserts new elements into it. The process is as following:

- Take the first element as a sorted sub-array.
- Insert the second element into the sorted sub-array (shift elements if needed).

- Last Char in Strings in PHP
- Decision Tree Complexity
- Read Excel XLS Files in Java

Categories

- Algorithm
- BigData
- Combinatorics
- HowTo
- Java
- JavaScript
- MySQL
- PHP
- Probability
- SQL
- Theory
- Uncategorized
- Web

Tags

abstract class abstract method ArrayList boolean function chernoff bound class CSS derandomization divide and conquer dom endswith expression FIFO fourier analysis generic type geocoding google maps html inheritance interface java java collection JavaScript lifo loop mysql object-oriented parser PHP polymorphism probability programming pseudorandom generator

- Insert the third element into the sorted sub-array.
- Repeat until all elements are inserted.

The following `insertionSort()` method implements insertion sort. It uses a nested loop to repetitively insert elements into the sorted sub-array.

```java
public static void insertionSort(int[] arr)
{
    for (int i = 1; i < arr.length; i++)
    {
        // a temporary copy of the current element
        int tmp = arr[i];
        int j;

        // find the position for insertion
        for (j = i; j > 0; j--)
        {
            if (arr[j - 1] < tmp)
                break;
            // shift the sorted part to right
            arr[j] = arr[j - 1];
        }

        // insert the current element
        arr[j] = tmp;
    }
}
```

The following tracks the code on an array with elements `{38, 27, 43, 3, 9, 82, 10}`.

```
{38, 27, 43,  3,  9, 82, 10}
{27, 38, 43,  3,  9, 82, 10},   i: 1
{27, 38, 43,  3,  9, 82, 10},   i: 2
{ 3, 27, 38, 43,  9, 82, 10},   i: 3
{ 3,  9, 27, 38, 43, 82, 10},   i: 4
{ 3,  9, 27, 38, 43, 82, 10},   i: 5
{ 3,  9, 10, 27, 38, 43, 82},   i: 6
```

Suppose the array length is n. The outer loop runs roughly n times, and the inner loop on average runs n/2 times. The total time is about t(n) = n * (n/2) = O(n^2). In terms of the efficiency, this is the same as selection sort.

Bubble Sort

Bubble sort repetitively compares adjacent pairs of elements and swaps if necessary.

- Scan the array, swapping adjacent pair of elements if they are not in relative order. This bubbles up the largest element to the end.
- Scan the array again, bubbling up the second largest element.
- Repeat until all elements are in order.

random random walk
recursion
regular expression rss seo
sql startswith store locator
string twitter xml

Archives

- February 2015
- December 2014
- August 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- February 2013
- January 2013
- December 2012

Meta

- Log in
- Entries RSS
- Comments RSS
- WordPress.org

The following `bubbleSort()` method implements bubble sort. It uses a nested loop to repetitively swap elements and bubble up the largest elements one by one.

```java
public static void bubbleSort (int[] data)
{
    for (int i = data.length - 1; i >= 0; i--)
    {
        // bubble up
        for (int j = 0; j <= i - 1; j++)
        {
            if (data[j] > data[j + 1])
                swap(data, j, j + 1);
        }
    }
}
```

The following tracks the code on an array with elements `{38, 27, 43, 3, 9, 82, 10}` for three rounds of bubbling.

```
{38, 27, 43,  3,  9, 82, 10}
i: 6
{27, 38, 43,  3,  9, 82, 10},   j: 0
{27, 38, 43,  3,  9, 82, 10},   j: 1
{27, 38,  3, 43,  9, 82, 10},   j: 2
{27, 38,  3,  9, 43, 82, 10},   j: 3
{27, 38,  3,  9, 43, 82, 10},   j: 4
{27, 38,  3,  9, 43, 10, 82},   j: 5
i: 5
{27, 38,  3,  9, 43, 10, 82},   j: 0
{27,  3, 38,  9, 43, 10, 82},   j: 1
{27,  3,  9, 38, 43, 10, 82},   j: 2
{27,  3,  9, 38, 43, 10, 82},   j: 3
{27,  3,  9, 38, 10, 43, 82},   j: 4
i: 4
{3,  27,  9, 38, 10, 43, 82},   j: 0
{3,  9,  27, 38, 10, 43, 82},   j: 1
{3,  9,  27, 38, 10, 43, 82},   j: 2
{3,  9,  27, 10, 38, 43, 82},   j: 3
```

Suppose the array length is n. The outer loop runs roughly n times, and the inner loop on average runs n/2 times. The total time is about *t(n) = n * (n/2) = O(n^2)*. In terms of the efficiency, this is the same as selection sort and insertion sort.

Related Posts

- Merge Sort
- Quick Sort
- Algorithm Analysis

References

The following links give visualizations on selection, insertion and bubble sort:
http://www.sorting-algorithms.com/selection-sort

http://www.sorting-algorithms.com/insertion-sort
http://www.sorting-algorithms.com/bubble-sort

Comments

0 comments

‹ Algorithm Analysis Merge Sort ›

Tagged with: bubble sort, insertion sort, selection sort, sort
Posted in Algorithm, Java