Simple Sorting II

Lecture 5

Prof. Eric Polizzi
Summary previous lecture

1- Bubble Sort
2- Selection Sort
3- Insertion Sort
Summary previous lecture

- Example: sort in ascending order

### Bubble Sort (main step):

- Start from the left

  ![Initial Array]

- Compare neighbors: “swap” and iterate left to right

  ![Sorted Array]

**O(N²) comparisons and swaps**

```java
public void bubbleSort()
{
  int in, out;
  int temp;
  for(out=N-1; out>0; out--)
  // outer loop (backward)
    for(in=0; in<out; in++)
    // inner loop (forward)
      if(array[in] > array[in+1])
      // out of order? swap them
        {
          temp = array[in];
          array[in] = array[in+1];
          array[in+1] = temp;
        }
  // end bubbleSort()
```
Example: sort in ascending order

**Selection Sort (main step):**

- Find smallest item
  
  ![Selection Sort Step 1](image)

- Swap it with item at 1st position
  
  ![Selection Sort Step 2](image)

**O(N^2)** comparisons; **O(N)** swaps

```
public void selectionSort()
{
    int in, out, min;
    int temp; // temp variable
    for (out = 0; out < N - 1; out++) // outer loop
    {
        // find the minimum item between [out+1, N-1]
        min = out; // initialize minimum index
        for (in = out + 1; in < N; in++) // inner loop
        {
            // if array[in] < array[min]
            min = in; // update minimum index
            // swap item "array[min]" with item "array[out]"
            temp = array[out];
            array[out] = array[min];
            array[min] = temp
        } // end inner loop
    } // end outer loop
} // end selectionSort()
```
Simple Sorting Algorithms

1- Bubble Sort
2- Selection Sort
3- Insertion Sort
3- Insertion Sort- overview

Easier to understand if we start in the middle of the process

- Players are partially sorted at the left of the marked player

- The marked player is removed from the list. The players from the left that are taller than the marked player, shift up.

- The marked player is inserted into the empty spot on the left.

A new marked player is selected.
And so on...
2- Insertion Sort - overview

- Example: sort in ascending order

All Steps for 5 items

- Select 2\textsuperscript{nd} item (key)

- Swap it with the 1\textsuperscript{st} item if not in order

- Select 3\textsuperscript{rd} item (key)

- Insert it inside the ordered array on the left

- Select 4\textsuperscript{th} item (key)

- Insert it inside the ordered array on the left

- Select 5\textsuperscript{th} item (key)

- Insert it inside the ordered array on the left

- The end
3- Insertion Sort- Examples

- For fun: https://www.youtube.com/watch?v=ROalU379l3U

- To do: Test Java applet InsertSort.html
The basic insertionSort method is few lines long (example below uses array of integer for simplicity)

```java
public void insertionSort()
{
    int in,out;
    int temp; // temp variable
    for(out=1; out<N; out++) // outer loop – select key
    {
        temp=array[out]; // save in memory select key item
        in=out; // start shifting at out
        while(in>0 && array[in-1]>=temp)//shift until key-item in position
        {
            array[in] = array[in-1]; // shift up
            in--; // go down one position
        }
        array[in]=temp; // insert select key item
    } // end outer loop
} // end insertionSort()
```
3- Insertion Sort- Complexity Analysis

- Complexity analysis: (two loops) so it is still a $O(N^2)$

- Max number of comparisons
  
  $1+2+3+...+(N-1)=N(N-1)/2$

  - However, only half this number in average $N(N-1)/4$
  - Half the time of BubbleSort

- Number of shifts (copies) is also equal in average to $N(N-1)/4$

  - However, a shift is not as time consuming as a swap
3- Insertion Sort- Complexity Analysis

- **For random data**
  - Insertion Sort should run twice faster than Bubble Sort
  - Insertion Sort should also run faster than Selection Sort

- **For data arranged in inverse order**
  - Every possible comparisons and shifts take place
  - No faster than Bubble Sort

- **For data that is already sorted or almost sorted**
  - Insertion sort runs in $O(N)$ (the while loop is never true)
  - Efficient way to order arrays that are slightly out of order
  - Often use as the final stage of more sophisticated algorithm such as quicksort
Simple Sorting

- **In theory**, Bubble Sort, Selection Sort and Insertion Sort are all $O(N^2)$; they are also all 'in-place' memory efficient algorithms.

- **In practice**, Insertion sort is the best bet of the three in most situations.

- It runs in $O(N)$ for 'almost sorted data'; for 'random data', efficiency may be improved using a binary search to insert the key.
Complement: Sorting Object

- Sorting algorithms can be applied to sort various fields of an object
- Example: Object "Person" that includes LastName, FirstName, Age
  - One can sort by age (ascending integer)
  - One can sort by LastName (alphabetical order)- String comparisons

```java
String str1,str2;
str1=Tesla;
str2=Edison;
if(str1.compareTo(str2)==0)
   System.out.println(str1+" is lexicographically equal to "+ str2);
if(str1.compareTo(str2)>0) // that's the one for this example
   System.out.println(str1+" is lexicographically greater than "+ str2);
if(str1.compareTo(str2)<0)
   System.out.println(str1+" is lexicographically less than"+ str2);
```

- **Stability**: Algorithm is said to be stable if it only sorts what needs to be sorted. Example:
  - If two identical LastName are present in the input data, the order of inputs is unchanged
Complement: 'Unsorting' algorithm

- Need to randomly shuffle items of an 'ordered' array?

Fisher-Yates/Knuth shuffle algorithm to generate random permutations

**All Steps for 5 items**

- Random select one item in the [0-4] index range
  - Swap it with the 5\textsuperscript{th} item..............................

- Random select one item in the [0-3] index range
  - Swap it with the 4\textsuperscript{th} item..............................

- Random select one item in the [0-2] index range
  - Swap it with the 3\textsuperscript{rd} item..............................

- Random select one item in the [0-1] index range
  - Swap it with the 2\textsuperscript{nd} item..............................

- The end.....................................................................

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>20</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>20</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>20</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>20</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>20</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>20</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
Complement: 'Unsorting' algorithm

- Durstenfeld modern 'in place' implementation of the Fisher-Yates algorithm

```java
public void shuffleArray()
{
    Random rnd = new Random();
    int out, index;
    int temp;
    for(out=N-1; out>0; out--)
    { // outer loop (backward)
        index = rnd.nextInt(out+1); // select random number in [0:out]
        // simple swap
        temp = array[index];
        array[index] = array[out];
        array[out] = temp;
    }
} // end shuffleArray()
```

- Complexity analysis: O(N)