HDSE 222

Data Structures and Algorithms Practice

P2 – Basic sorting Algorithm implementation

By,
Dr. Tharinda Nishantha Vidanagama.
Senior Lecturer,
CMIS.

Ordered array class

- As before create a class for the ordered array

```
class OrderArray {
    private double[] a; // ref to array a
    private int nElems; // number of data items

   public OrderArray(int max){ // constructor
       a = new double[max]; // create the array
       nElems = 0; // no items yet
   }
   public int size(){
      return nElems; }
}
```

Ordered Array Insert

- Elements are inserted in ascending order in the array

```
public void insert(double value)  // put element into array
    {
    int j;
    for(j=0; j<nElems; j++)       // find where it goes
       if(a[j] > value)           // (linear search)
          break;
    for(int k=nElems; k>j; k--)   // move bigger ones up
       a[k] = a[k-1];
    a[j] = value;                 // insert it
    nElems++;                     // increment size
    }  // end insert()
```

Java – Binary Search

- This method search for the element by repeatedly dividing in half the range of array elements to be considered!
- lowerBound and upperBound variables are set to the first and last elements of the array.
- If you cant find the element you can return some value that is not an array index.

```
public int find(double searchKey)
   {
   int lowerBound = 0;
   int upperBound = nElems-1;
   int curIn;

   while(true)
      {
      curIn = (lowerBound + upperBound ) / 2;
      if(a[curIn]==searchKey)
         return curIn;                  // found it
      else if(lowerBound > upperBound)
         return nElems;                 // can't find it
      else                              // divide range
         {
         if(a[curIn] < searchKey)
            lowerBound = curIn + 1; // it's in upper half
         else
            upperBound = curIn - 1; // it's in lower half
         }  // end else divide range
      }  // end while
   }  // end find()
```

Ordered Delete

- Use the binary search method to find the element for deletion
- When an element is deleted the higher elements are shifted down one position
- Display method is same as before.

```
public boolean delete(double value)
    {
    int j = find(value);
    if(j==nElems)                       // can't find it
       return false;
    else                                // found it
       {
       for(int k=j; k<nElems; k++) // move bigger ones down
          a[k] = a[k+1];
       nElems--;                        // decrement size
       return true;
       }
    }  // end delete()
```

Ordered Array Application

- Create an instance of the order array
- Insert some elements

```
class OrderApp
   {
   public static void main(String[] args)
      {
      int maxSize = 100;              // array size
      OrderArray arr;                   // reference to array
      arr = new OrderArray(maxSize);    // create the array

      arr.insert(77);                   // insert 10 items
      arr.insert(99);
      arr.insert(44);
      arr.insert(55);
      arr.insert(22);
      arr.insert(88);
      arr.insert(11);
      arr.insert(00);
      arr.insert(66);
      arr.insert(33);
```

Ordered Array Application (ii)

- Search for an item
- Delete some items
- Display the array contents.

```
      int searchKey = 55;                  // search for item
      if( arr.find(searchKey) != arr.size() )
         System.out.println("Found " + searchKey);
      else
         System.out.println("Can't find " + searchKey);

      arr.display();                       // display items

      arr.delete(00);                      // delete 3 items
      arr.delete(55);
      arr.delete(99);

      arr.display();                       // display items again
      }  // end main()
   }  // end class OrderedApp
```

Java - Bubble sort

- Create a base class as before (not the ordered array)
- Add these extra methods for sorting the array of elements.

```
private void swap(int one, int two) {

    double temp = a[one];

    a[one] = a[two];

    a[two] = temp;

}
```

Java - Bubble sort (ii)

- Sorting the bubbles!, The algorithm

```
public void bubbleSort() {

int out, in;

for(out=nElems-1; out>1; out--)    // outer loop (backward)

  for(in=0; in<out; in++)          // inner loop (forward)

    if( a[in] > a[in+1] )          // out of order?

        swap(in, in+1);            // swap them
}  // end bubbleSort()
```

Java – Selection Sort

- Previous “swap” method is also used.

```
public void selectionSort() {

    int out, in, min;

    for (out = 0; out < nElems - 1; out++) {// outer loop

        min = out; // minimum

        for (in = out + 1; in < nElems; in++)   // inner loop

            if (a[in] < a[min]) // if min greater,

                min = in; // we have a new min

        swap(out, min); // swap them

    } // end for(outer)
} // end selectionSort()
```