**Sorting**

Sorting is, without doubt, the most fundamental algorithmic problem

1. Supposedly, 25% of all CPU cycles are spent sorting
2. Sorting is fundamental to most other algorithmic problems, for example binary search.
3. Many different approaches lead to useful sorting algorithms, and these ideas can be used to solve many other problems.

What is sorting? It is the problem of taking an arbitrary permutation of *n* items and rearranging them into the total order,

**Issues in Sorting**

**Increasing or Decreasing Order**? - The same algorithm can be used by both all we need do is change tex2html_wrap_inline271to tex2html_wrap_inline273in the comparison function as we desire.

**What about equal keys?** - Does the order matter or not? Maybe we need to sort on secondary keys, or leave in the same order as the original permutations.

**What about non-numerical data?** - Alphabetizing is sorting text strings, and libraries have very complicated rules concerning punctuation, etc. Is Brown-Williams before or after Brown America before or after Brown, John?

We can ignore all three of these issues by assuming a comparison function which depends on the application. Compare (a,b) should return ``<'', ``>'', or ''=''.

**LINEAR SORT**

**Keep one end fix on each pass and increment linearly next pass. Compare it will all other one by one.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **32** | **51** | **27** | **85** | **66** | **23** | **13** | **57** |

**Pass1:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **32** | **51** | 27 | 85 | 66 | 23 | 13 | 57 |
| **32** | 51 | **27** | 85 | 66 | 23 | 13 | 57 |
| **27** | 51 | 32 | **85** | 66 | 23 | 13 | 57 |
| **27** | 51 | 32 | 85 | **66** | 23 | 13 | 57 |
| **27** | 51 | 32 | 85 | 66 | **23** | 13 | 57 |
| **23** | 51 | 32 | 85 | 66 | 27 | **13** | 57 |
| **13** | 51 | 32 | 85 | 66 | 27 | 23 | **57** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **51** | **32** | **85** | **66** | **27** | **23** | **57** |

**Pass 2:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **51** | 32 | 85 | 66 | 27 | 23 | 57 |
| **13** | **32** | 51 | 85 | 66 | 27 | 23 | 57 |
| **13** | **32** | 51 | 85 | 66 | 27 | 23 | 57 |
| **13** | **32** | 51 | 85 | 66 | 27 | 23 | 57 |
| **13** | **27** | 51 | 85 | 66 | 32 | 23 | 57 |
| **13** | **23** | 51 | 85 | 66 | 32 | 27 | 57 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **51** | **85** | **66** | **32** | **27** | **57** |

**Pass 3:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | 23 | **51** | 85 | 66 | 32 | 27 | 57 |
| **13** | 23 | **51** | 85 | 66 | 32 | 27 | 57 |
| **13** | 23 | **32** | 85 | 66 | 51 | 27 | 57 |
| **13** | 23 | **32** | 85 | 66 | 51 | 27 | 57 |
| **13** | 23 | **27** | 85 | 66 | 51 | 32 | 57 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | 85 | 66 | 51 | 32 | 57 |

**Pass 4:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | 23 | 27 | 85 | 66 | 51 | 32 | 57 |
| **13** | 23 | 27 | 66 | 85 | 51 | 32 | 57 |
| **13** | 23 | 27 | 51 | 85 | 66 | 32 | 57 |
| **13** | 23 | 27 | 32 | 85 | 66 | 51 | 57 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | 85 | 66 | 51 | 57 |

**Pass 5:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | 23 | 27 | 32 | 85 | 66 | 51 | 57 |
| **13** | 23 | 27 | 32 | 66 | 85 | 51 | 57 |
| **13** | 23 | 27 | 32 | 51 | 85 | 66 | 57 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | 85 | 66 | 57 |

**Pass 6:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | 23 | 27 | 32 | 51 | 85 | 66 | 57 |
| **13** | 23 | 27 | 32 | 51 | 66 | 85 | 57 |
| 13 | 23 | 27 | 32 | 51 | 57 | 85 | 66 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | **57** | 85 | 66 |

**Pass 7:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | 23 | 27 | 32 | 51 | 57 | 85 | 66 |
| 13 | 23 | 27 | 32 | 51 | 57 | 66 | 85 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | **57** | **66** | **85** |

Sorted Array:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | **57** | **66** | **85** |

**LINEAR SORT**

Complexity : n2 (‘n’ no. of comparisons to sort one element x ‘n’ no. of element to be sorted)

Algorithm:

**(LINEAR Sort) LSORT (DATA, N)**

**Here DATA is an array with N elements. This algorithm sorts the elements. The algorithms sort the elements in DATA.**

1. **Repeat step 2and 3 for K=1 to N-1**
2. **Set PTR :=K+1 [ Initialize pass pointer PTR]**
3. **Repeat while PTR ≤ N [Execute Pass]**

**(a). If DATA[K] > DATA[PTR], then:**

**Interchange DATA [K] and DATA [PTR]**

**[End of If structure]**

**(b) Set PTR: = PTR+1**

**[End of inner loop]**

**[End of step 1 outer loop]**

1. **Exit**

**Selection Sort**

**Procedure: MIN (A, K, N, LOC)**

**An Array is in Memory.**

**This procedure finds the location LOC of the smallest element among A [K], A [K+1], ---- A [N].**

**1. Set MIN: = A [K] and LOC: =K [Initialize pointers]**

**2. Repeat for J=K+1, K+2, -------- N**

**If MIN > A [J], then Set MIN: = A [J] and**

**LOC: = A [J] and LOC: =J**

**[End of loop]**

**3. Return**

**The selection sort Algorithm can now be easily stated:**

**Algorithm: (Selection Sort) SELECTION (A, N)**

**This algorithm sorts the array A with N elements.**

1. **Repeat Step 2 and 3 for K=1 to N-1**
2. **Call MIN (A, K, N, LOC)**
3. **[Interchange A[K] and A[LOC]**

**Set TEMP: =A[K], A[K]:=A[LOC] and A[LOC]:=TEMP**

**[End of Step 1 loop]**

1. **Exit**

**Selection Sort**

**Suppose an array A with n elements A [1], A [2], ------- A [N] is in memory.**

**The Selection sort algorithm for sorting A works as follows:**

* **First find the smallest in the list and put it in the first position.**
* **Then find the second smallest element in the list and put it in the second position.**

**Start with first element and compare with others one by one till element less than first one is found. Then compare the smaller one with the rest until you are able to locate the smallest value.**

**Swap that value with the first element in row then ignoring the first element now.**

**Repeat the procedure again to find the next min value and then swapping it with second element and then third element and then so on…….**

**Smaller values will accumulate on the left,**

|  |  |
| --- | --- |
| **Pass 1:** | **Find the location LOC of the smallest in the list of N elements A[1], A[2], -----, A[N], and then interchange A[LOC] and A[1]. Then: A[1] is sorted.** |
| **Pass 2:** | **Find the location LOC of the smallest in the sublist of N-1 elements A[2], A[3], ----- , A[N], and then interchange A[LOC] and A[2]. Then: A[1], A[2] is sorted, since A[1] ≤ A[2].** |
| **Pass 3:** | **Find the location LOC of the smallest in the sublist of N-2 elements A[3], A[4], ----- , A[N], and then interchange A[LOC] and A[3]. Then: A[1], A[2], A[3] is sorted, since A[2] ≤ A[3].** |
| ……. | ………………………………………………….. |
| **Pass N-1:** | **Find the location LOC of the smallest of the elements A[N-1], A[N], and then interchange A[LOC] and A[N-1]. Then: A[1], A[2], A[3],---- A[N] is sorted, since A[N-1] ≤ A[N].** |

**Thus A is sorted after N-1 passes.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **32** | **51** | **27** | **85** | **66** | **23** | **13** | **57** |

**Pass 1:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Min32** | **51** | **27** | **85** | **66** | **23** | **13** | **57** |
| **32** | **51** | **Min 27** | **85** | **66** | **23** | **13** | **57** |
| **32** | **51** | **Min 27** | **85** | **66** | **23** | **13** | **57** |
| **32** | **51** | **Min 27** | **85** | **66** | **23** | **13** | **57** |
| **32** | **51** | **27** | **85** | **66** | **Min 23** | **13** | **57** |
| **32** | **51** | **27** | **85** | **66** | **23** | **Min 13** | **57** |
| **32** | **51** | **27** | **85** | **66** | **23** | **Min 13** | **57** |

**Swap Min value with First element**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **32** | **51** | **27** | **85** | **66** | **23** | **Min 13** | **57** |
| **32** | **51** | **27** | **85** | **66** | **23** | **Min 13** | **57** |

**Pass 2:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **Min 51** | **27** | **85** | **66** | **23** | **32** | **57** |
| **13** | **51** | **Min 27** | **85** | **66** | **23** | **32** | **57** |
| **13** | **51** | **Min 27** | **85** | **66** | **23** | **32** | **57** |
| **13** | **51** | **Min 27** | **85** | **66** | **23** | **32** | **57** |
| **13** | **51** | **27** | **85** | **66** | **Min 23** | **32** | **57** |
| **13** | **51** | **27** | **85** | **66** | **Min 23** | **32** | **57** |
| **13** | **Min 23** | **27** | **85** | **66** | **51** | **32** | **57** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **85** | **66** | **51** | **32** | **57** |

**Pass 3**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **Min 27** | **85** | **66** | **51** | **32** | **57** |
| **13** | **23** | **Min 27** | **85** | **66** | **51** | **32** | **57** |
| **13** | **23** | **Min 27** | **85** | **66** | **51** | **32** | **57** |
| **13** | **23** | **Min 27** | **85** | **66** | **51** | **32** | **57** |
| **13** | **23** | **Min 27** | **85** | **66** | **51** | **32** | **57** |
| **13** | **23** | **Min 27** | **85** | **66** | **51** | **32** | **57** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **85** | **66** | **51** | **32** | **57** |

**Pass 4:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **Min 85** | **66** | **51** | **32** | **57** |
| **13** | **23** | **27** | **85** | **Min 66** | **51** | **32** | **57** |
| **13** | **23** | **27** | **85** | **66** | **Min 51** | **32** | **57** |
| **13** | **23** | **27** | **85** | **66** | **51** | **Min 32** | **57** |
| **13** | **23** | **27** | **Min 32** | **66** | **51** | **85** | **57** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **66** | **51** | **85** | **57** |

**Pass 5:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **Min 66** | **51** | **85** | **57** |
| **13** | **23** | **27** | **32** | **66** | **Min 51** | **85** | **57** |
| **13** | **23** | **27** | **32** | **66** | **Min 51** | **85** | **57** |
| **13** | **23** | **27** | **32** | **66** | **Min 51** | **85** | **57** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **Min 51** | **66** | **85** | **57** |
| **13** | **23** | **27** | **32** | **51** | **66** | **85** | **57** |

**Pass 6:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | **Min 66** | **85** | **57** |
| **13** | **23** | **27** | **32** | **51** | **Min 66** | **85** | **57** |
| **13** | **23** | **27** | **32** | **51** | **Min 57** | **85** | **66** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | **57** | **85** | **66** |

**Pass 7:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **13** | **23** | **27** | **32** | **51** | **57** | **Min 85** | **66** |
| **13** | **23** | **27** | **32** | **51** | **57** | **Min66** | **85** |
| **13** | **23** | **27** | **32** | **51** | **57** | **Min 85** | **66** |

# Insertion Sort

* One of the simplest methods to sort an array is an insertion sort.
* An example of an insertion sort occurs in everyday life while playing cards.
* To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in the correct place. This process is repeated until all the cards are in the correct sequence.
* Both average and worst-case time is ***O***(*n*2 ).

## Theory

* Starting near the top of the array in Figure 2-1(a), we extract the 3.
* Then the above elements are shifted down until we find the correct place to insert the 3.
* This process repeats in Figure 2-1(b) with the next number.
* Finally, in Figure 2-1(c), we complete the sort by inserting 2 in the correct place.
* Assuming there are *n* elements in the array, we must index through *n* - 1 entries.
* For each entry, we may need to examine and shift up to *n* - 1 other entries, resulting in a ***O***(*n*2) algorithm.
* The insertion sort is an *in-place* sort. That is, we sort the array in-place.
* No extra memory is required.
* The insertion sort is also a *stable* sort.
* Stable sorts retain the original ordering of keys when identical keys are present in the input data.

**InsertionSort(LA, N)**

1. **Set A[0] = -∞**
2. **Repeat Step 3-5 for k:=2-N**
3. **TEMP:=A[k] and PTR:=k-1**
4. **Repeat while TEMP < A[PTR]**
   1. **Set A[PTR+1]:=A[PTR]**
   2. **Set PTR:=PTR-1**

**[End of Step 4 loop]**

1. **Set A[PTR+1]:=TEMP**

**[End of Step 2 loop]**

**6. Return**

### s_fig21 Figure 2-1: Insertion Sort

**Bubble Sort**

**BubbleSort(DATA, N)**

1. Repeat step 2 -3 for K:=1 to N-1
2. Set PTR :=1
3. Repeat While PTR≤N-K
   1. If DATA[PTR] > DATA[PTR+1], then

Interchange DATA[PTR] and DATA[PTR+1]

[End of if structure]

* 1. Set PTR:= PTR+1

[End of Step 3 loop]

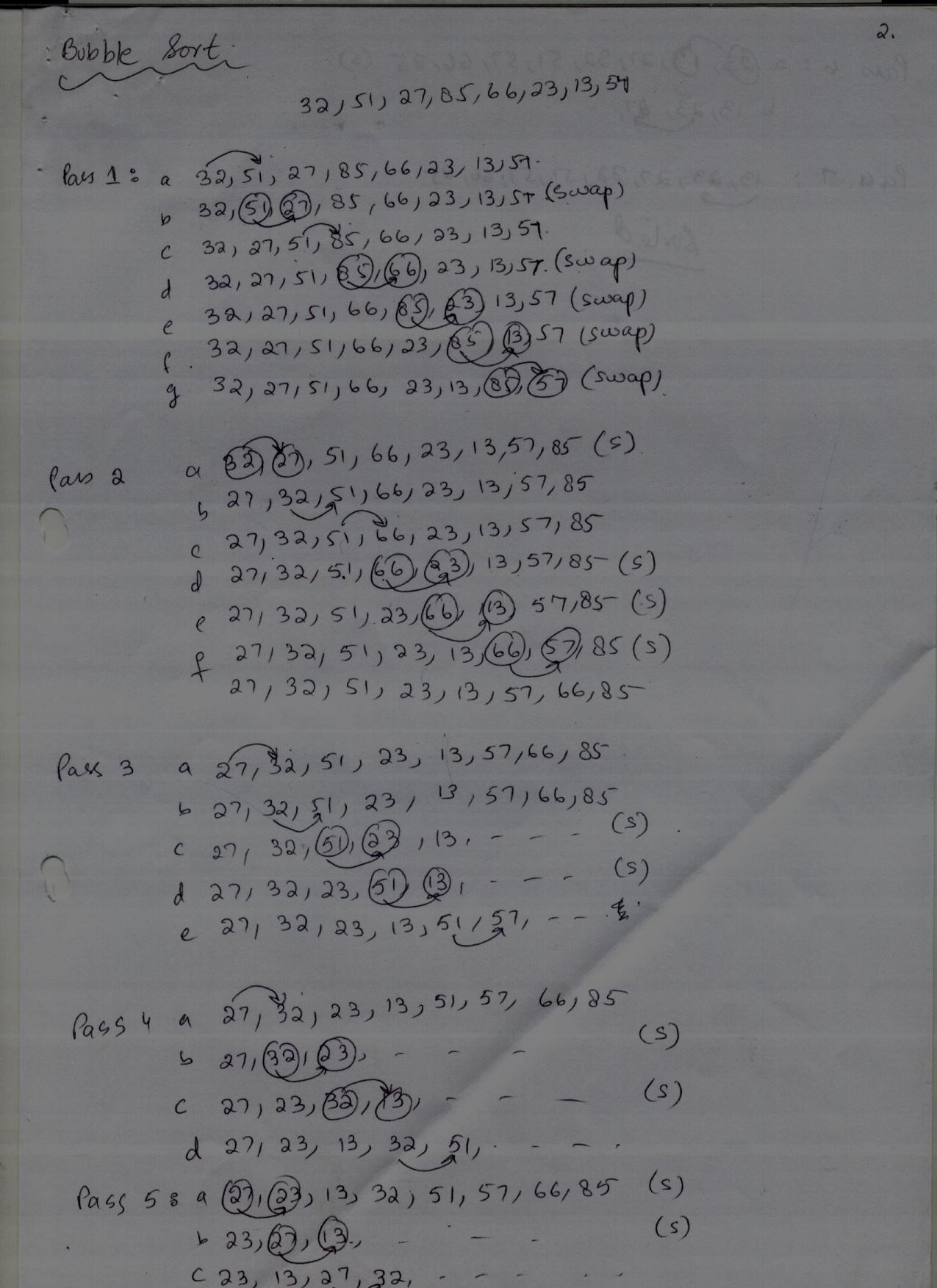
[End of Step 1 loop]

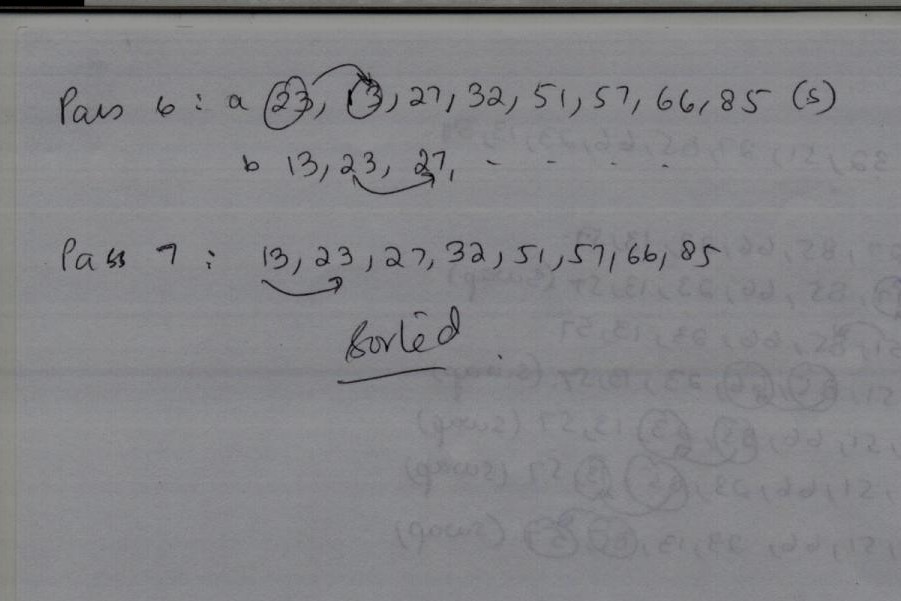
1. Exit

**Example of Bubble Sort**

Let suppose DATA : 32, 51, 27, 85, 66, 23, 13, 54

Following are the steps involved in bubble sort

****



**ARRAY CLASS ( SORTING )**

|  |  |
| --- | --- |
| **Class Name : Array** | **Super Class: Java.util Package** |
| **Responsibilities** | **Collaborations** |
| [**static void sort(char[] a)**](http://www.tutorialspoint.com/java/util/arrays_sort_char.htm)  This method sorts the specified array of chars into ascending numerical order. |  |
| [**static void sort(char[] a, int fromIndex, int toIndex)**](http://www.tutorialspoint.com/java/util/arrays_sort_char_index.htm)  This method sorts the specified range of the specified array of chars into ascending numerical order. |  |
| [**static void sort(double[] a)**](http://www.tutorialspoint.com/java/util/arrays_sort_double.htm)  This method sorts the specified array of doubles into ascending numerical order. |  |
| [**static void sort(double[] a, int fromIndex, int toIndex)**](http://www.tutorialspoint.com/java/util/arrays_sort_double_index.htm)  This method sorts the specified range of the specified array of doubles into ascending numerical order. |  |
| [**static void sort(float[] a)**](http://www.tutorialspoint.com/java/util/arrays_sort_float.htm)  This method sorts the specified array of floats into ascending numerical order. |  |
| [**static void sort(float[] a, int fromIndex, int toIndex)**](http://www.tutorialspoint.com/java/util/arrays_sort_float_index.htm)  This method sorts the specified range of the specified array of floats into ascending numerical order. |  |
| [**static void sort(int[] a)**](http://www.tutorialspoint.com/java/util/arrays_sort_int.htm)  This method sorts the specified array of ints into ascending numerical order. |  |
| [**static void sort(int[] a, int fromIndex, int toIndex)**](http://www.tutorialspoint.com/java/util/arrays_sort_int_index.htm)  This method sorts the specified range of the specified array of ints into ascending numerical order. |  |