Order Of Play

The ability to sort information into order is essential to most programs, and there are many ways of doing it

Bubble Sort

This diagram illustrates the Bubble Sort for a reduced hand of nine cards (T is the Ten carc). The ordered part of the hand grows from the right-hand end with each pass. The 1 and 2 underneath the hand of cards indicates the two cards currently being compared

```
Begin Sort
2 8 9 3 T 5 K 6 7 Begin Pass 1
1 2
8 2 9 3 T 5 K 6 7
  1 2
8 9 2 3 T 5 K 6 7
    1 2
8 9 3 2 T 5 K 6 7
      1 2
8 9 3 T 2 5 K 6 7
        1 2
8 9 3 T 5 2 K 6 7
          1 2
8 9 3 T 5 K 2 6 7
            1 2
8 9 3 T 5 K 6 2 7
              1 2
8 9 3 T 5 K 6 7 2 End Pass 1
9 8 T 5 K 6 7 3 2 End Pass 2
9 T 8 K 6 7 5 3 2 End Pass 3
T 9 K 8 7 6 5 3 2 End Pass 4
T K 9 8 7 6 5 3 2 End Pass 5
K T 9 8 7 6 5 3 2 End pass 6
End Scrt
```

Insertion Sort

With the Insertion Sort, the ordered part of the list grows from the left-hand end. Cards are moved directly to their correct position in the list as they are inspected

```
Begin Sort
2 8 9 3 T 5 K 6 7
2 1
8 2 9 3 T 5 K 6 7
2   1
9 8 2 3 T 5 K 6 7
    2 1
9 8 3 2 T 5 K 6 7
2       1
T 9 8 3 2 5 K 6 7
      2   1
T 9 8 5 3 2 K 6 7
2           1
K T 9 8 5 3 2 6 7
        2     1
K T 9 8 6 5 3 2 7
        2       1
K T 9 8 7 6 5 3 2
End Scrt
```

Sorting is one of the most widely used computer operations, but it is a task at which computers are, by their own standards, highly inefficient. According to operational research, between 30 and 40 per cent of all computing time is spent in sorting, and if you add the associated tasks of merging data and searching for specific items, then the figure probably rises above 50 per cent.

Programmers have probably spent as much time inventing sort algorithms (general methods of solving problems) as computers have spent doing the actual sorting. Advanced sorting methods are extremely difficult to analyse, but it is quite easy to understand the simplest methods computers use to sort data with the aid of the example of sorting a pack of playing cards.

Lay 13 cards of the same suit on a table. Arrange them in a line, in no particular order, but the Ace and the Two should not be at the right-hand end of the line. The cards are to be sorted into descending order (King, Queen, Jack... Ace), starting at the left. This is an almost trivial task for us, and requires so little thought that it is difficult to describe exactly how we might do it. If, however, you were to specify that only one card can be moved at a time, that no card can be placed on top of another, and that the cards are to cover as little of the table as possible, the task becomes a lot less trivial, and an efficient method is hard to determine. In this analogy the cards are pieces of data, the maximum surface covered corresponds to the computer memory required, and you are the program. How do you solve the problem?

1) Put a coin below the leftmost card to act as a position marker and to remind you where you are in the sort. Compare the marked card with the card to its right. Are they in descending order? If they are not, swap their positions, leaving the coin where it is, and obeying the rule of only moving one card at a time and not placing cards on top of each other. Notice what you have to do to swap them.

2) When the two cards are in order, move the coin one place to the right and repeat Step 1. You are now in a loop that will end once you move the coin into the rightmost position. Reaching this position is called making a 'pass' through the cards.

3) At the end of the first pass look at the cards. The Ace, which is the lowest card in the suit, has found its way to the rightmost end of the line, and so is in its correct place. If you make a further pass through the cards, as detailed in Steps 1 and 2, the Two card will be moved to its correct place. This is repeated, through pass after pass, until the whole suit is in descending order.

You may have noticed several drawbacks to this method. It is very tedious; it is not economical, as simply exchanging the positions of two cards requires three different operations; and, above all, many of the comparisons made between different cards are unnecessary. For example, after one pass the Ace is in its correct place, so there's no point moving the coin into position 13 (where no comparison is possible, anyway). On the second pass, because the card on the right is in its correct place, there was no need to move the pointer to position 12. In general, each pass will end one place to the left of the endpoint of the previous pass.

Knowing where to stop is another problem. A computer will continue comparing cards indefinitely unless it is told to stop. The only sure rule is: stop after a pass with no swaps. In other words, if you've gone through the data without altering its order, then it must be in order.

The method of sorting we have investigated is called the 'Bubble Sort'. Its advantages include simple programming techniques, little use of extra memory and reasonable efficiency with small amounts of partially ordered data. These are the criteria by which a sort algorithm must be judged, although when the data to be sorted is extensive, speed may have to be sacrificed for economy of memory simply because computer memory may not accommodate both the raw data and a sorted copy. For this reason, we'll ignore algorithms that require taking data from one array and moving it to the sorted position in a second array. The second method of simple sorting is based more directly on the way that we would sort cards.

1) Lay the shuffled cards out again and place a penny coin beneath the second card from the left. Whichever card the penny is beneath at the beginning of each pass, we will call the 'penny card'.

2) Push the penny card out of the line, leaving a gap, and place a twopenny coin beneath the card's immediate left. Call this card the twopenny card.

3) Compare the penny card with the twopenny card. If they're in order, then push the penny card back into place and skip to Step 4. If they're not in order, then push the twopenny card into the gap and move the twopenny coin one place to the left to mark a new twopenny card (if the twopenny card is at the extreme left, this will not apply, so