UNIVERSITY OF
WATERLOO

Faculty of Mathematics
Waterloo, Ontario N2L 3G1

Centre for Education in
Mathematics and Computing

Grade 7/8 Math Circles

November 11, 2020

Algorithms - Solutions

What is an Algorithm?

Algorithms are everywhere. Whether you know it or not, you use algorithms on a daily basis. Do you follow the same routine every morning to get ready for school? Then you are following an algorithm. If you've ever solved a Rubik's Cube, made a paper airplane, baked a cookie or cooked a specific recipe, you followed an algorithm. **An algorithm is a set of steps to accomplish a task.** In computer science, we tell computers what to do by creating algorithms for them to follow. Consider the following example.

Example: Adding Two-digit Numbers

Let's think about how we add two-digit numbers together. When we add two numbers, usually we place one number above the other and add each of their columns together. Recall that these columns are called place values and have names such as the ones column, tens column, hundreds column, and so on. Thus to add two-digit numbers, we add the ones column and then the tens column. We can use this information to create an algorithm.

Algorithm for Adding Two-digit Numbers:

1. Add the digits in the ones column. Write this sum in the ones column of your total sum.
2. Add the digits in the tens column. Write this sum in the tens column of your total sum.

Now adding two-digit numbers is easy! We just need to follow our algorithm to get the correct answer. Let's test our algorithm on a few examples:

$$\begin{array}{lrr||lrrr||lrrr}
\text{(a)} & & 9 & \text{(b)} & & 1 & 0 & \text{(c)} & & 1 & 8 \\
& + & 9 & & + & 1 & 8 & & + & 2 & 7 \\ \hline
& & 18 & & & 2 & 8 & & & 3 & 15
\end{array}$$

Wait, $18 + 27$ is not equal to 315. But I followed each step of our algorithm perfectly! Why did our algorithm produce a wrong answer?

In this case, we forgot to include the carry over rule and now our algorithm has an error! Sometimes algorithms will produce errors and that's why we should always test a few examples. Even if we believe our algorithms are perfect there could be a single example that breaks it! Let's fix our algorithm.

Algorithm for Adding Two-digit Numbers:

1. Add the digits in the ones column.
2. If their sum is greater than 9, then write the ones digit of this sum in the ones column of the total sum and then carry the tens digit of this sum to the next column.
3. Add the digits in the tens column and the amount we carried. Write this sum in the tens column of your total sum.

Try to convince yourself that this algorithm will work for any example where we add two-digit numbers.

Exercise: The word generalize means to make something more widespread. Our algorithm only adds two-digit numbers with a ones and tens column. Can you generalize our algorithm to something that works for more than just two-digit numbers? Try three digits? What about any number of digits?

Sorting

Sorting is used every day by almost everyone in one way or another. Books in a library are sorted by genre and then by author. At school, you may sort your notes and homework by subject. Your clothes at home may be sorted by type with your socks in one drawer and your pants in another drawer.

Consider the following list and **find the smallest number**:

$$\mathbf{8 \quad 1 \quad 5 \quad 13}$$

When you looked for the smallest number, did it feel like you took in all four numbers at once rather than looking at them one at a time? Our brains do an amazing job of processing small groups of data like this and quickly coming up with an answer. Computers on the other hand have to look through data one unit at a time and often have to deal with more data than humans can handle!

Because of this, computer scientists have come up with many ways to sort large amounts of data quickly and efficiently. There are many different **sorting algorithms** and we will look at some today.

Insertion Sort

The first sorting algorithm we will look at is called **insertion sort**. This algorithm is used to sort a list of numbers into ascending (increasing) order. It involves taking one number (we will call it an **element**) at a time from the front of an unsorted list and creating a new sorted list. As we move elements over from the unsorted list, we compare them to each element in the sorted list one by one until their proper place is found.

Insertion Sort Algorithm:

1. Create an empty list for the sorted numbers.
2. Move the first element from the unsorted list to the beginning of the sorted list.
3. If there is more than 1 element in the sorted list, compare that element with the next element in the sorted list. If the next element is smaller, then swap the two elements.
4. Repeat step 3 until the newly inserted element cannot be swapped with the next element (in other words, the next element is greater or equal to it).
5. Repeat steps 2-4 until the unsorted list is empty.

Example: Use insertion sort to sort the following list of numbers:

$$\mathbf{9 \quad 4 \quad 3 \quad 6 \quad 7}$$

Video Solution: *https://youtu.be/TcW6KjAoNC4*

To practice the insertion sort algorithm try out this activity: *https://www.geogebra.org/m/k36vvw4c*.

How come we have to go through all those steps? Why can't we just drop a number in the right place in the sorted list?

While we can easily look at a small list and find where a number fits, computers cannot do this and have to go through a list one by one. Even for us, it is not easy to quickly insert a number into its correct place if a list has millions of elements.

Selection Sort

Selection sort uses the following algorithm to sort a list of numbers into ascending (increasing) order. Just like insertion sort this algorithm works by maintaining two lists - the original unsorted list, and a new sorted list.

Selection Sort Algorithm:

1. Create an empty list for the sorted numbers.
2. Let the first number in the unsorted list be the current smallest number.
3. Find the smallest number in the unsorted list by comparing it to the current smallest number. Update the current smallest number each time a smaller number is found.
4. Move the current smallest number to the end of the sorted list.
5. Repeat steps 2-4 until the unsorted list is empty.

Example: Use selection sort to sort the following list of numbers:

14 6 11 5 20

Video Solution: *https://youtu.be/kjLT91U0uKE*

To practice the selection sort algorithm try out this activity: *https://www.geogebra.org/m/gcxxcmcc*.

What happens if a number is repeated in our unsorted lists? Will this algorithm sort it into ascending order correctly? Yes! - can you see why?

Bubble Sort

Unlike the sorting algorithms mentioned above, **bubble sort** does not move elements one at a time from an unsorted list to a sorted list. The algorithm repeatedly goes through the list and compares and then possibly swaps each pair until the largest number is moved to the end of the list. Then it repeats these steps untill the list is sorted into ascending (increasing) order.

Bubble Sort Algorithm:

1. Look at the first pair of numbers.
2. If they are not in increasing order, swap them.
3. Now look at the next pair. The last element of the previous pair should be the first element of the new pair.
4. Repeat steps 2 and 3 until you reach the end of the list.
5. Repeat steps 1-4 until you can go through the list without any swaps being made.

Example: Use bubble sort to sort the following list of numbers:

$$\mathbf{3 \quad 9 \quad 1 \quad 5}$$

Video Solution: *https://youtu.be/tGCCkmRVEmM*

Notice that in the last 5 steps, the list was already in sorted order! The reason bubble sort must go over them again is because the algorithm doesn't have a way to recognize that the list is sorted until it compares all the pairs in the list to make sure they are in the correct order.

To practice the bubble sort algorithm try out this activity: *https://www.geogebra.org/m/ssqzmy6k*.

Example: Use bubble sort to sort the following letters in alphabetically ascending order:

$$\mathbf{A \quad E \quad D \quad C \quad B}$$

Video Solution: *https://youtu.be/8c-pB9fmpZ4*

Another common approach to sorting (and problem solving in general) is to break the problem down into smaller parts. Often these smaller parts are easier to solve. We can then combine the smaller solutions to solve the whole problem. We'll see this in the sorting algorithm below.

Merge Sort

Merge sort takes an unsorted list and breaks that list in half. It keeps breaking the resulting lists in half until there is only one element in each list. These lists are very easy to sort! The single element lists are then paired up, with each 2 element list getting sorted. This merging process continues until we have one list again. This is sometimes called the **Divide and Conquer strategy.**

Merge Sort Algorithm:

1. Split the list in half.
2. Repeat step 1 on each sublist until each sublist has one element.
3. Merge and sort all pairs of sublists on the same level.
4. Repeat step 3 on the next levels until all sublists are merged into one.

Example: Use merge sort to sort the following list of numbers:

$$\mathbf{7 \quad 1 \quad 5 \quad 3 \quad 6 \quad 4 \quad 0 \quad 9}$$

Video Solution: *https://youtu.be/G5tzgzb-MEo*

To practice the merge sort algorithm try out this activity: *https://www.geogebra.org/m/mgufcpar*.

While the idea of breaking the list into smaller and smaller pieces seems to make sense, how does the merging process actually work?

At each merging step the algorithm uses the fact that each list it's merging together, is already sorted. This means it only has to compare the first element in each list (i.e. if the smallest element in list A is smaller than the smallest element in list B, then it must be the smallest in both).

To see exactly how this works consider the following example:

Video Solution: *https://youtu.be/61izwg-NsOA*

Problem Set - Solutions

1. Give an example of when you have used an algorithm in your own life.

   Answers vary.

2. Write a set of steps (an algorithm) on how to subtract (with borrowing) so someone who does not know how to subtract by borrowing can learn.

   Solutions may vary. Example solution:

   Algorithm for Subtracting Numbers:

   (a) Write the number that we are subtracting right below the number it is being subtracted from. Line them up so that the ones columns, tens column, etc. are lined up.

   (b) Look at the first column (ones column), we'll call this column **"current"**.

   (c) If the digit on top is smaller than the digit below it, look at the digit in the next column of the number on top. (Do i, ii, etc. if this condition is met.)

      i If this digit is not greater than zero, look at the digit in the next column of the number on top. Repeat this step until the digit on top is greater than zero.

      ii "Borrow" from this digit by subtracting 1 from it, and then add 10 to the digit on top in the previous column.

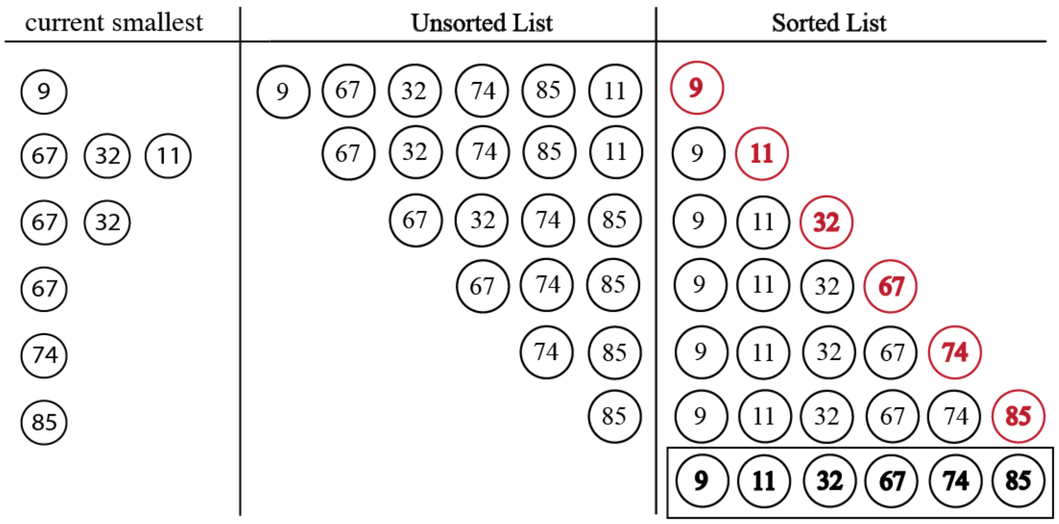
      iii If this column is not the "current" column, subtract 1 from it and then add 10 to the digit on top in the previous column. Repeat this step until 10 is added to the "current" column.

   (d) Subtract the digit below from the number on top in the "current" column. Write this difference in the "current" column of the total difference.

   (e) Let the next column be "current". If the "current" column has only one digit on top and no digit below it, write this digit in the "current" column of the the total difference. Repeat steps (c)-(e) until there is no next column.

3. Sort the following list of numbers using selection sort.

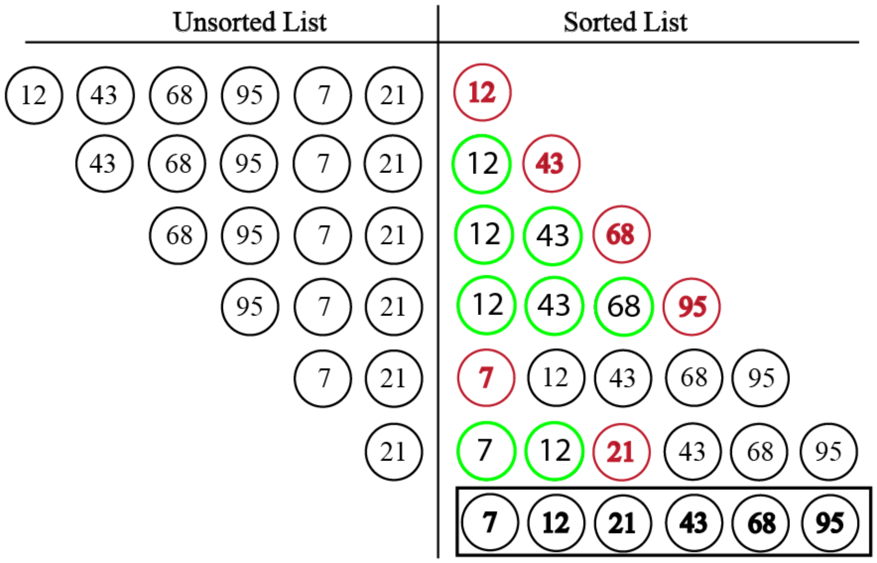
   9 67 32 74 85 11



4. Sort the following list of numbers using insertion sort.

   The green circled element indicates that the inserted element has been swapped with it in the sorted list.

$$12 \quad 43 \quad 68 \quad 95 \quad 7 \quad 21$$



5. Sort the following list of numbers using bubble sort.

41 84 14 79 26 53

Steps 1-4:

$\underline{\mathbf{41}}$	$\underline{\mathbf{84}}$	14	79	26	53	$\rightarrow$	$\underline{\mathbf{41}}$	$\underline{\mathbf{84}}$	14	79	26	53
41	$\underline{\mathbf{84}}$	$\underline{\mathbf{14}}$	79	26	53	$\rightarrow$	41	$\underline{\mathbf{14}}$	$\underline{\mathbf{84}}$	79	26	53
41	14	$\underline{\mathbf{84}}$	$\underline{\mathbf{79}}$	26	53	$\rightarrow$	41	14	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$	26	53
41	14	79	$\underline{\mathbf{84}}$	$\underline{\mathbf{26}}$	53	$\rightarrow$	41	14	79	$\underline{\mathbf{26}}$	$\underline{\mathbf{84}}$	53
41	14	79	26	$\underline{\mathbf{84}}$	$\underline{\mathbf{53}}$	$\rightarrow$	41	14	79	26	$\underline{\mathbf{53}}$	$\underline{\mathbf{84}}$

Repeat steps 1-4:

$\underline{\mathbf{41}}$	$\underline{\mathbf{14}}$	79	26	53	84	$\rightarrow$	$\underline{\mathbf{14}}$	$\underline{\mathbf{41}}$	79	26	53	84
14	$\underline{\mathbf{41}}$	$\underline{\mathbf{79}}$	26	53	84	$\rightarrow$	14	$\underline{\mathbf{41}}$	$\underline{\mathbf{79}}$	26	53	84
14	41	$\underline{\mathbf{79}}$	$\underline{\mathbf{26}}$	53	84	$\rightarrow$	14	41	$\underline{\mathbf{26}}$	$\underline{\mathbf{79}}$	53	84
14	41	26	$\underline{\mathbf{79}}$	$\underline{\mathbf{53}}$	84	$\rightarrow$	14	41	26	$\underline{\mathbf{53}}$	$\underline{\mathbf{79}}$	84
14	41	26	53	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$	$\rightarrow$	14	41	26	53	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$

Repeat steps 1-4:

$\underline{\mathbf{14}}$	$\underline{\mathbf{41}}$	26	53	79	84	$\rightarrow$	$\underline{\mathbf{14}}$	$\underline{\mathbf{41}}$	26	53	79	84
14	$\underline{\mathbf{41}}$	$\underline{\mathbf{26}}$	53	79	84	$\rightarrow$	14	$\underline{\mathbf{26}}$	$\underline{\mathbf{41}}$	53	79	84
14	26	$\underline{\mathbf{41}}$	$\underline{\mathbf{53}}$	79	84	$\rightarrow$	14	26	$\underline{\mathbf{41}}$	$\underline{\mathbf{53}}$	79	84
14	26	41	$\underline{\mathbf{53}}$	$\underline{\mathbf{79}}$	84	$\rightarrow$	14	26	41	$\underline{\mathbf{53}}$	$\underline{\mathbf{79}}$	84
14	26	41	53	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$	$\rightarrow$	14	26	41	53	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$

Repeat steps 1-4:

$\underline{\mathbf{14}}$	$\underline{\mathbf{26}}$	41	53	79	84	$\rightarrow$	$\underline{\mathbf{14}}$	$\underline{\mathbf{26}}$	41	53	79	84
14	$\underline{\mathbf{26}}$	$\underline{\mathbf{41}}$	53	79	84	$\rightarrow$	14	$\underline{\mathbf{26}}$	$\underline{\mathbf{41}}$	53	79	84
14	26	$\underline{\mathbf{41}}$	$\underline{\mathbf{53}}$	79	84	$\rightarrow$	14	26	$\underline{\mathbf{41}}$	$\underline{\mathbf{53}}$	79	84
14	26	41	$\underline{\mathbf{53}}$	$\underline{\mathbf{79}}$	84	$\rightarrow$	14	26	41	$\underline{\mathbf{53}}$	$\underline{\mathbf{79}}$	84
14	26	41	53	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$	$\rightarrow$	14	26	41	53	$\underline{\mathbf{79}}$	$\underline{\mathbf{84}}$

6. Think about the Selection Sort algorithm.

   (a) What guarantees that the end result will be a properly sorted list?

   At each step we have one less number to sort. Eventually we will run out of numbers, thus when the unsorted list is empty this guarantees that the end result will be a properly sorted list.

   (b) What would we have to change about the selction sort algorithm algorithm to get the list in descending order?

   Instead of finding the smallest number each time, we would find the largest number and add it to the end of the sorted list.

7. Sort the following list using selection sort, insertion sort, and bubble sort.

   25 13 6 9 1 38

   (a) How many steps did the selection sort take?
   (Each time you compare a number to the current smallest and each time you move a number is a step)

   For the first pass through the list we had 6 comparisons and 1 move (moving current smallest to the sorted list). For the second pass through the list we had 5 comparisons and 1 move. For the third pass through the list we had 4 comparisons and 1 move. For the fourth pass through the list we had 3 comparisons and 1 move. For the fifth pass through the list we had 2 comparisons and 1 move. For the last pass through the list we had 1 comparisons and 1 move. This gives us a total of **27 steps.**

   (b) How many steps did the insertion sort take?
   (Each time you move a number and each time you compare it to the numbers in the sorted list till it is in the right spot is a step)

   We move each number once to the sorted list so thats 6 moves. There were 6 swaps in total. This gives us a total of **12 steps**.

   (c) How many steps did the bubble sort take?
   (The comparison of each pair is a step whether you swap them or not)

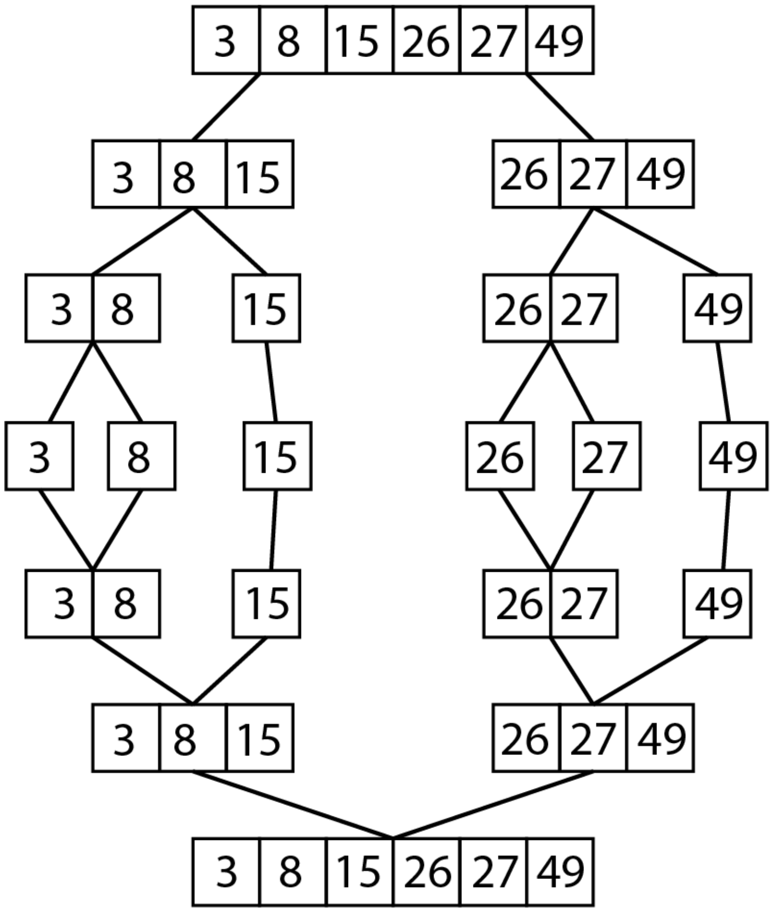
   Using the bubble sort algorithm we make 5 comparisons each pass thorugh the list. We make a total of 5 passes through the list. This gives us a total of **25 steps**.

(d) Which of these methods is the most efficient or the fastest?

In order, the fastest sorts are the insertion sort, the bubble sort, and then the selection sort.

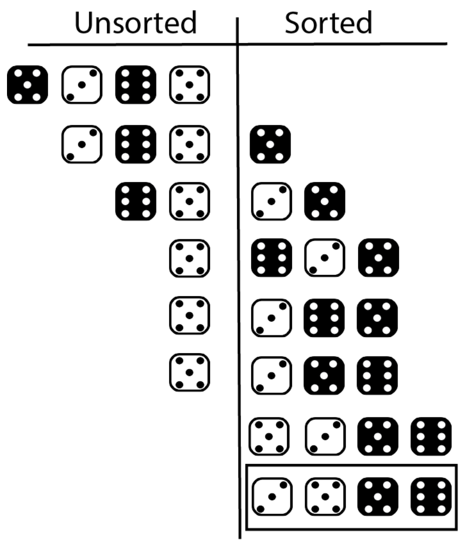
8. As humans, we can quickly observe that the following list is sorted and does not need to be sorted any further. However, computers can't see this and given this list, they will still complete the steps to any of the sorting algorithms. Go through the following list using **merge sort** as a computer would.

3 8 15 26 27 49



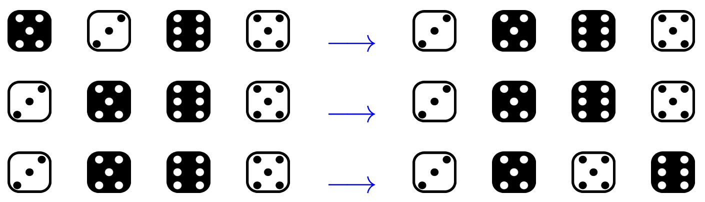
9. Sort the following dice using the algorithm specified below: ⚄ ⚂ ⚅ ⚄
Make sure the show which colour each die is.

   (a) Use insertion sort.

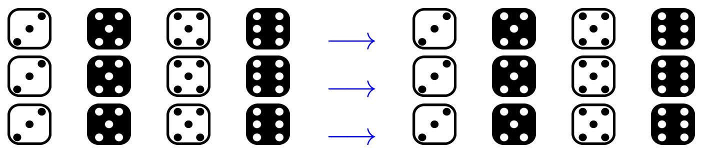


Remember that in the algorithm for insertion sort, we stop swapping elements in the sorted list when the next element is greater than or equal to the one being moved along the list.

   (b) Use bubble sort.



2 swaps were made so repeat.



No swaps were made so sorting is complete.

   (c) You probably noticed that there were two repeated elements in this list. We say a sorting algorithm is **stable** if repeated elements stay in the same order before and after a list is sorted. Which of the two algorithms we just looked at are stable? (Which algorithms resulted in a sorted list where ⚄ is before ⚄ ?)

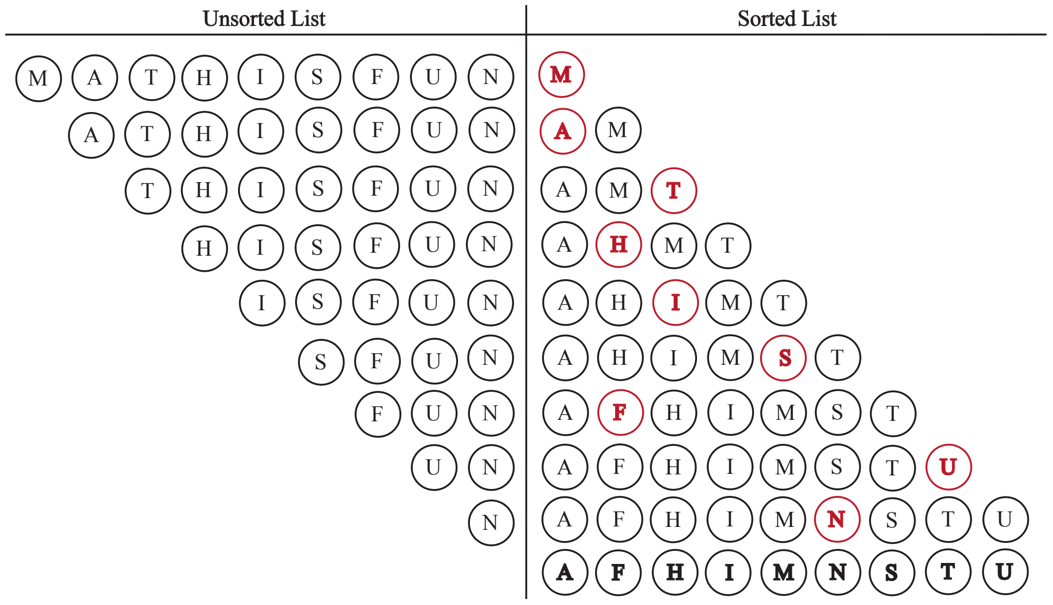
In this case, the insertion sort is **not** stable and the bubble sort is stable.

10. Sort the following letters in alphabetical order using the algorithms specified below.

M A T H I S F U N

(a) Use insertion sort.

We omit showing the swaps where the elements in the sorted list are greater than or equal to the one being moved along the list (the inserted element).



Unsorted List
Sorted List
M A T H I S F U N | **M**
A T H I S F U N | **A** M
T H I S F U N | A M **T**
H I S F U N | A **H** M T
I S F U N | A H **I** M T
S F U N | A H I M **S** T
F U N | A **F** H I M S T
U N | A F H I M S T **U**
N | A F H I M **N** S T U
A F H I M N S T U

(b) Use bubble sort.

You should end up with the same result as above.

11. Based on the following steps showing a list of 3 digit numbers being sorted with an algorithm called **Radix Sort**, write the down what you think the algorithm is.

365 502 560 299 101 462 401 902

$$36\underline{\mathbf{5}} \quad 50\underline{\mathbf{2}} \quad 56\underline{\mathbf{0}} \quad 29\underline{\mathbf{9}} \quad 10\underline{\mathbf{1}} \quad 46\underline{\mathbf{2}} \quad 40\underline{\mathbf{1}} \quad 90\underline{\mathbf{2}}$$

$$\downarrow$$

$$56\underline{\mathbf{0}} \quad 10\underline{\mathbf{1}} \quad 40\underline{\mathbf{1}} \quad 50\underline{\mathbf{2}} \quad 46\underline{\mathbf{2}} \quad 90\underline{\mathbf{2}} \quad 36\underline{\mathbf{5}} \quad 29\underline{\mathbf{9}}$$

$$\downarrow$$

$$5\underline{\mathbf{6}}0 \quad 1\underline{\mathbf{0}}1 \quad 4\underline{\mathbf{0}}1 \quad 5\underline{\mathbf{0}}2 \quad 4\underline{\mathbf{6}}2 \quad 9\underline{\mathbf{0}}2 \quad 3\underline{\mathbf{6}}5 \quad 2\underline{\mathbf{9}}9$$

$$\downarrow$$

$$1\underline{\mathbf{0}}1 \quad 4\underline{\mathbf{0}}1 \quad 5\underline{\mathbf{0}}2 \quad 9\underline{\mathbf{0}}2 \quad 5\underline{\mathbf{6}}0 \quad 4\underline{\mathbf{6}}2 \quad 3\underline{\mathbf{6}}5 \quad 2\underline{\mathbf{9}}9$$

$$\downarrow$$

$$\underline{\mathbf{1}}01 \quad \underline{\mathbf{4}}01 \quad \underline{\mathbf{5}}02 \quad \underline{\mathbf{9}}02 \quad \underline{\mathbf{5}}60 \quad \underline{\mathbf{4}}62 \quad \underline{\mathbf{3}}65 \quad \underline{\mathbf{2}}99$$

$$\downarrow$$

$$\underline{\mathbf{1}}01 \quad \underline{\mathbf{2}}99 \quad \underline{\mathbf{3}}65 \quad \underline{\mathbf{4}}01 \quad \underline{\mathbf{4}}62 \quad \underline{\mathbf{5}}02 \quad \underline{\mathbf{5}}60 \quad \underline{\mathbf{9}}02$$

(a) Sort the list by the third digit (ones column digit).

(b) Sort the list by the second digit (tens column digit).

(c) Sort the list by the first digit (hundreds column digit).

What makes this algorithm efficient is that instead of sorting the numbers themselves, each step places the associated digit into its correct category. You can think of each step as the computer having different buckets for the digits 0 to 9 and, depending on what that digit is in each element, the algorithm puts the element in its appropriate bucket. This way, you are not comparing the numbers to each other but individually putting each in a bucket three times based on its three digits.