UNIVERSITY OF WATERLOO

Faculty of Mathematics
Waterloo, Ontario N2L 3G1

Centre for Education in
Mathematics and Computing

Grade 6 Math Circles

February 14/15, 2017

Sorting - Solutions

Sort it out!

You have in front of you five shapes. In groups of two to four, sort them.

What features of the shapes did you decide to sort the shapes by? Size? Colour? Letter on the shape? Number of sides?

How did you go about sorting the shapes? Did you look in the unsorted pile for the "biggest" or the "smallest" shape first? Did you take shapes from your unsorted pile at random and put them into a sorted line one by one? Or did you do something else entirely?

Sorting

Sorting is used every day by almost everyone in some way or another. Do you have a list of favourite colours, movies, books, or TV shows? How did you sort them into that order? Books in a library are sorted by genre and the author's last name. You can sort your cards when playing Uno or Go Fish to make pairs easier to see.

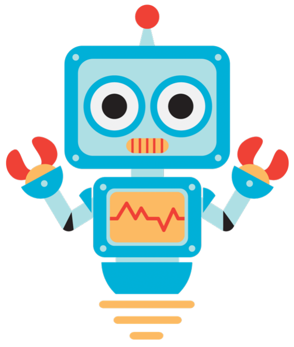
When you were sorting shapes just now, you probably thought it was no problem to sort the first five and it may have taken just a little longer when there were more shapes to sort. Look at the list below and find the smallest number:

3 9 1 5

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 scientist have come up with many ways to sort large amounts of data quickly and efficiently. These sorting **algorithms** are what we will look at today.

Algorithms



An algorithm is a set of instructions where each step is clear. If you have ever solved a Rubik's Cube, made a paper airplane, or baked cookies, chances are you followed an algorithm. We tell computers what to do by creating algorithms for them to follow. The key to many algorithms is repetition until a certain requirement is met.

Exercise: Sort by Birthday

Let's play a game! I will divide you into two teams and the teams will compete to sort yourselves in a line by birthday.

Here are the rules:

1. Absolutely no talking! As soon as you get out of your seats, you may not talk until your team is done sorting.
2. The year you were born does **not** matter.
3. No running, pushing, or hitting others to get their attention.
4. When you're done sorting, everyone put your hand up in the air.

If you break any of these rules, your **entire team is disqualified.**

Insertion Sort

The first sorting algorithm we will look at today is called insertion sort. 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, they are compared to each element in the sorted list one by one until their proper place is found.

The algorithm we follow is:

1. Move first element from the unsorted list to the beginning of the sorted list
2. Compare that element with the next element in the sorted list. If the next element is lesser, then switch.
3. Repeat step 2 until the element cannot be swapped with the next element (in other words, the next element is greater or equal to it).
4. Repeat steps 1-3 until the unsorted list is empty.

Example 1

Unsorted	Sorted	
3 9 1 5		*1. Start with an unsorted list and an empty sorted list.*
9 1 5	$\underline{\mathbf{3}}$	*2. Move first element from unsorted list to sorted.*
9 1 5	3	*3. There is no next element in the sorted list to compare 3 to, so we move on.*
1 5	$\underline{\mathbf{9}}$ 3	*4. Move first element from unsorted list.*
1 5	3 $\underline{\mathbf{9}}$	*5. $3 < 9$ so swap the two numbers.*
1 5	3 9	*6. There is no next element in the sorted list so we move on.*
5	$\underline{\mathbf{1}}$ 3 9	*7. Move first element from unsorted list*
5	1 3 9	*8. $3 \geq 1$ so we do not swap the two and we move on.*
	$\underline{\mathbf{5}}$ 1 3 9	*9. Move first element from unsorted list.*
	1 $\underline{\mathbf{5}}$ 3 9	*10. $1 < 5$ so swap the two.*
	1 3 $\underline{\mathbf{5}}$ 9	*11. $3 < 5$ so swap the two.*
	1 3 5 9	*12. $9 \geq 5$ so do not swap the two. And there are no more elements left in the unsorted list so we stop here.*

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 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.

Exercise: Insertion Sort

Sort the following list using the insertion sort algorithm:

12 53 91 8 11

Show your work in a T-chart like in the example we just looked at. You do not need to include the steps where no elements move positions.

Unsorted	**Sorted**
12 53 91 8 11	
53 91 8 11	**<u>12</u>**
91 8 11	**<u>53</u>** 12
91 8 11	12 **<u>53</u>**
8 11	**<u>91</u>** 12 53
8 11	12 **<u>91</u>** 53
8 11	12 53 **<u>91</u>**
11	**<u>8</u>** 12 53 91
	<u>11</u> 8 12 53 91
	8 **<u>11</u>** 12 53 91

Bubble Sort

Unlike insertion sort, bubble sort does move elements one at a time from an unsorted list to a sorted one. The bubble sort algorithm goes through the list repeatedly and moves larger elements to the end until the list is sorted.

The algorithm we follow is:

1. Look at the first pair of elements.
2. If they are not in order, then swap them.
3. Now, look at the next pair. (The last element of the previous pair should be the first element of this pair).
4. Repeat steps 2 and 3 until you reach the end of the list.
5. Repeat steps 1-4 until these steps can be completed without any swaps.

Example 3

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

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

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

We have just completed steps 1 to 4.
Because two swaps happened, we repeat.

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

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

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

We have just completed steps 1 to 4 again.
Because one swap happened, we repeat.

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

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

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

We have just completed steps 1 to 4 again.
No swaps happened so our list is sorted!

Notice that in the last 5 steps, the list is already in order! The reason bubble sort must go over them again is that the algorithm does not know any way to recognize a sorted list than to compare all pairs in the list and make sure they are in the correct order.

We have now sorted this list of numbers with two different sorting algorithms. How do you think they compare?

Let's test it out!

Exercise: Tournament of Algorithms

Find a partner. One person will use insertion sort and the other will use bubble sort to sort:

2 4 6 0 1

On the next page, show the steps for sorting this list with the algorithm you were assigned. Count how many steps (number of lines) it took and write these in the chart below. For insertion sort, don't write down repeated lines.

Insertion sort	**Bubble sort**
10	20

Insertion sort

Unsorted	**Sorted**
2 4 6 0 1	
4 6 0 1	**2**
6 0 1	**4** 2
6 0 1	2 **4**
0 1	**6** 2 4
0 1	2 **6** 4
0 1	2 **6** 4
0 1	2 4 **6**
1	**0** 2 4 6
	1 0 2 4 6
	0 **1** 2 4 6

Bubble sort

$$\underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 6\quad 0\quad 1 \quad\longrightarrow\quad \underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 6\quad 0\quad 1$$

$$2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}}\quad 0\quad 1 \quad\longrightarrow\quad 2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}}\quad 0\quad 1$$

$$2\quad 4\quad \underline{\mathbf{6}}\quad \underline{\mathbf{0}}\quad 1 \quad\longrightarrow\quad 2\quad 4\quad \underline{\mathbf{0}}\quad \underline{\mathbf{6}}\quad 1$$

$$2\quad 4\quad 0\quad \underline{\mathbf{6}}\quad \underline{\mathbf{1}} \quad\longrightarrow\quad 2\quad 4\quad 0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{6}}$$

We have just completed steps 1 to 4. Because two swaps happened, we repeat.

$$\underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 0\quad 1\quad 6 \quad\longrightarrow\quad \underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 0\quad 1\quad 6$$

$$2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{0}}\quad 1\quad 6 \quad\longrightarrow\quad 2\quad \underline{\mathbf{0}}\quad \underline{\mathbf{4}}\quad 1\quad 6$$

$$2\quad 0\quad \underline{\mathbf{4}}\quad \underline{\mathbf{1}}\quad 6 \quad\longrightarrow\quad 2\quad 0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{4}}\quad 6$$

$$2\quad 0\quad 1\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}} \quad\longrightarrow\quad 2\quad 0\quad 1\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}}$$

We have just completed steps 1 to 4 again. Because two swaps happened, we repeat.

$$\underline{\mathbf{2}}\quad \underline{\mathbf{0}}\quad 1\quad 4\quad 6 \quad\longrightarrow\quad \underline{\mathbf{0}}\quad \underline{\mathbf{2}}\quad 1\quad 4\quad 6$$

$$0\quad \underline{\mathbf{2}}\quad \underline{\mathbf{1}}\quad 4\quad 6 \quad\longrightarrow\quad 0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{2}}\quad 4\quad 6$$

$$0\quad 1\quad \underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 6 \quad\longrightarrow\quad 0\quad 1\quad \underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 6$$

$$0\quad 1\quad 2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}} \quad\longrightarrow\quad 0\quad 1\quad 2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}}$$

We have just completed steps 1 to 4 again. Because two swaps happened, we repeat.

$$\underline{\mathbf{0}}\quad \underline{\mathbf{1}}\quad 2\quad 4\quad 6 \quad\longrightarrow\quad \underline{\mathbf{0}}\quad \underline{\mathbf{1}}\quad 2\quad 4\quad 6$$

$$0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{2}}\quad 4\quad 6 \quad\longrightarrow\quad 0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{2}}\quad 4\quad 6$$

$$0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{2}}\quad 4\quad 6 \quad\longrightarrow\quad 0\quad \underline{\mathbf{1}}\quad \underline{\mathbf{2}}\quad 4\quad 6$$

$$0\quad 1\quad \underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 6 \quad\longrightarrow\quad 0\quad 1\quad \underline{\mathbf{2}}\quad \underline{\mathbf{4}}\quad 6$$

$$0\quad 1\quad 2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}} \quad\longrightarrow\quad 0\quad 1\quad 2\quad \underline{\mathbf{4}}\quad \underline{\mathbf{6}}$$

We have just completed steps 1 to 4 again. No swaps happened so our list is sorted!

Exercise: Sort 40 numbers!

In groups of four or five sort 40 numbers! Before you start, discuss with your group mates what the most efficient way to sort these numbers would be. You do not need to use one of the sorting algorithms we've already looked at.

After sorting the numbers, write down what your strategy was and if you felt it was successful or not. If you could do it again, what would you do differently. Discuss this with your group mates as well.

Keep this exercise in mind, we will come back to it in a bit!

Answers may vary.

The key to this exercise is to realize that that fastest way to sort 40 numbers is to divide them among your group members and each sort your own numbers and then combining these smaller sorted sublists to make the complete sorted lists.

Merge Sort

The last sorting algorithm we are going to learn today is well suited for dealing with sorting large lists because it breaks lists into smaller lists that we call **sublists** and then puts them back together in order. This is sometimes called the **Divide and Conquer strategy.**

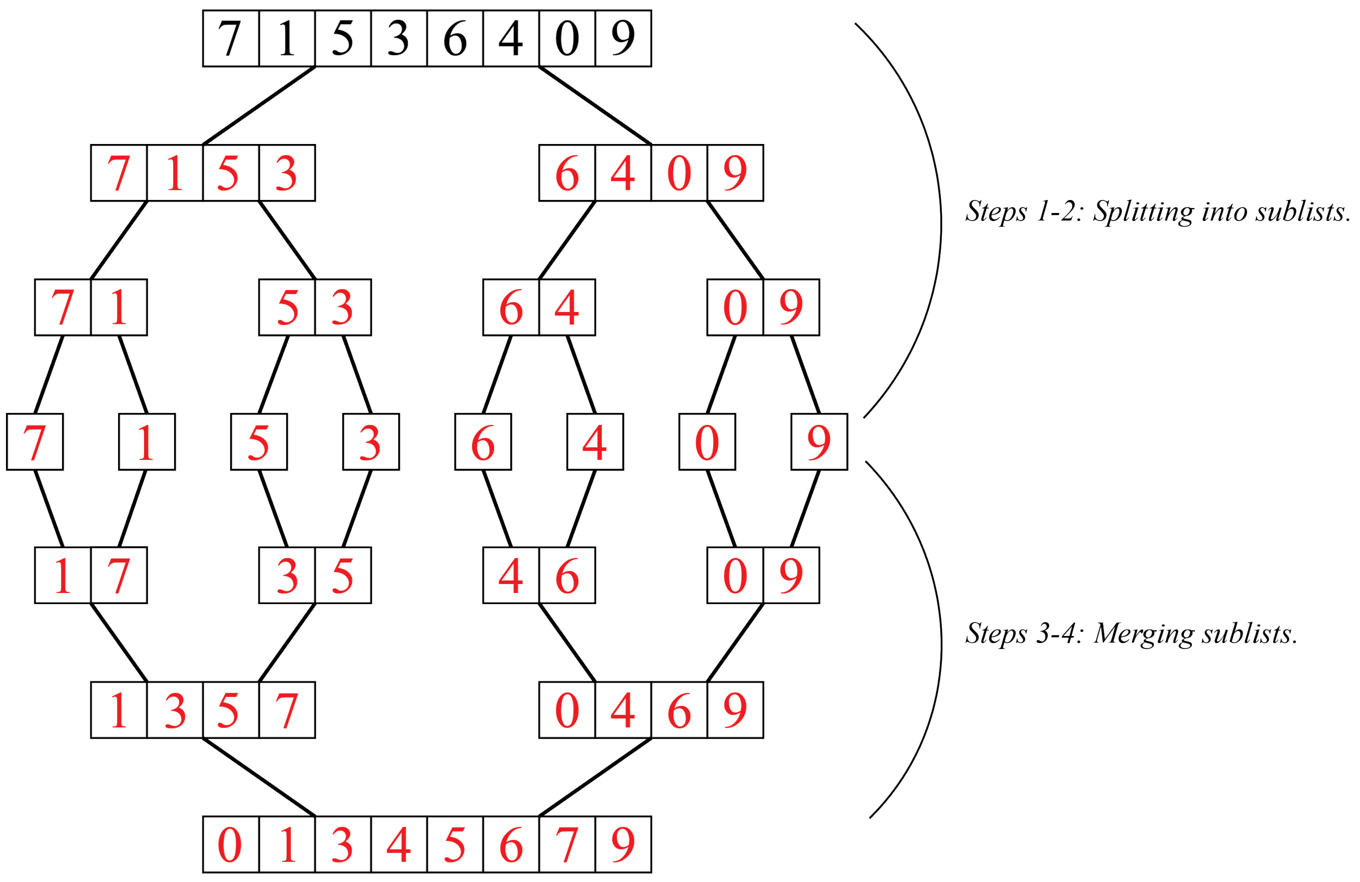
Merge sort splits a list in half over and over again until all elements are in their own sublist. Then, the sublists are paired, sorted, and merged again until the entire list is sorted! Because you have broken the list into one element sublists, they are easy to sort. And because the resulting sublists are already sorted, merging a pair only requires comparing the first values of the two sublists.

The algorithm we follow is:

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

This algorithm can be visualized well as a tree diagram. That is a diagram with each level connected to the next by branches showing how the sublists are split and merged again.

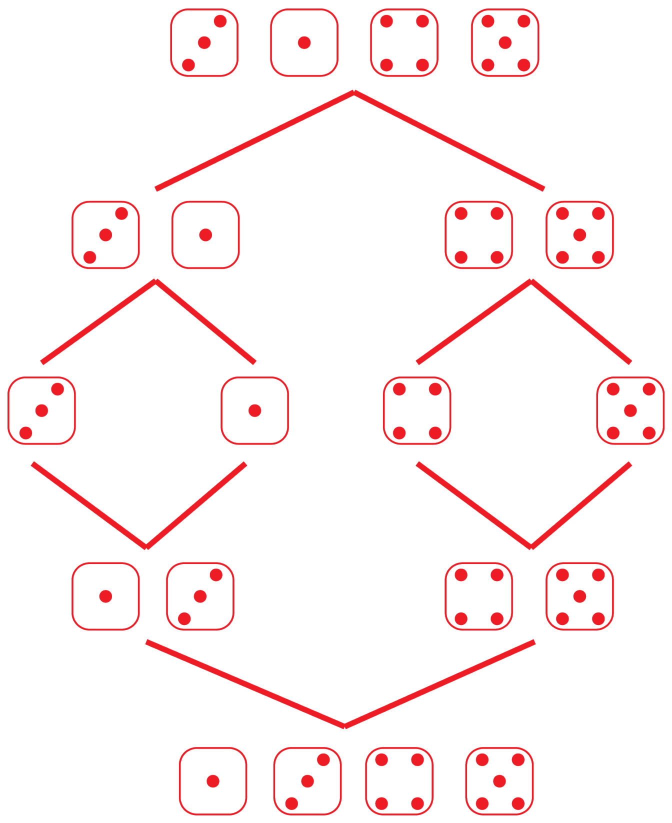
Example 4



Exercise: Merge Sort

Use merge sort to sort the following dice: ⚂ ⚀ ⚃ ⚄

Draw a tree diagram to show your work.



Exercise: Sort 40 numbers again! Get back with your groups from before. Sort the 40 numbers again, this time using merge sort. You do not need to show any work. Compare merge sort with your original strategy. What if you used one of the first two algorithms we learned to sort these 40 numbers? Can you think of an even better sorting algorithm?

We have looked at three different sorting algorithms today. In your Beaver Computing Competition Math Circle last Fall, you practiced the type of logical reasoning used in computer science. This is a taste of that type of thinking being applied to solve a specific problem.

Problem Set

Show all steps in sorting algorithms unless the question says otherwise.

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

   Answers will vary.

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

C A T D O G

(a) Use insertion sort.

Unsorted	**Sorted**
C A T D O G	
A T D O G	**C**
T D O G	**A** C
D O G	**T** A C
D O G	A **T** C
D O G	A C **T**
O G	**D** A C T
O G	A **D** C T
O G	A C **D** T
G	**O** A C D T
G	A **O** C D T
G	A C **O** D T
G	A C D **O** T
	G A C D O T
	A **G** C D O T
	A C **G** D O T
	A C D **G** O T

(b) Use bubble sort.

$$\underline{\mathbf{C}}\quad \underline{\mathbf{A}}\quad \mathrm{T}\quad \mathrm{D}\quad \mathrm{O}\quad \mathrm{G} \longrightarrow \underline{\mathbf{A}}\quad \underline{\mathbf{C}}\quad \mathrm{T}\quad \mathrm{D}\quad \mathrm{O}\quad \mathrm{G}$$

$$\mathrm{A}\quad \underline{\mathbf{C}}\quad \underline{\mathbf{T}}\quad \mathrm{D}\quad \mathrm{O}\quad \mathrm{G} \longrightarrow \mathrm{A}\quad \underline{\mathbf{C}}\quad \underline{\mathbf{T}}\quad \mathrm{D}\quad \mathrm{O}\quad \mathrm{G}$$

$$\mathrm{A}\quad \mathrm{C}\quad \underline{\mathbf{T}}\quad \underline{\mathbf{D}}\quad \mathrm{O}\quad \mathrm{G} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \underline{\mathbf{D}}\quad \underline{\mathbf{T}}\quad \mathrm{O}\quad \mathrm{G}$$

$$\mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \underline{\mathbf{T}}\quad \underline{\mathbf{O}}\quad \mathrm{G} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \underline{\mathbf{O}}\quad \underline{\mathbf{T}}\quad \mathrm{G}$$

$$\mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \mathrm{O}\quad \underline{\mathbf{T}}\quad \underline{\mathbf{G}} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \mathrm{O}\quad \underline{\mathbf{G}}\quad \underline{\mathbf{T}}$$

4 swaps were made so we repeat.

$$\underline{\mathbf{A}}\quad \underline{\mathbf{C}}\quad \mathrm{D}\quad \mathrm{O}\quad \mathrm{G}\quad \mathrm{T} \longrightarrow \underline{\mathbf{A}}\quad \underline{\mathbf{C}}\quad \mathrm{D}\quad \mathrm{O}\quad \mathrm{G}\quad \mathrm{T}$$

$$\mathrm{A}\quad \underline{\mathbf{C}}\quad \underline{\mathbf{D}}\quad \mathrm{O}\quad \mathrm{G}\quad \mathrm{T} \longrightarrow \mathrm{A}\quad \underline{\mathbf{C}}\quad \underline{\mathbf{D}}\quad \mathrm{O}\quad \mathrm{G}\quad \mathrm{T}$$

$$\mathrm{A}\quad \mathrm{C}\quad \underline{\mathbf{D}}\quad \underline{\mathbf{O}}\quad \mathrm{G}\quad \mathrm{T} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \underline{\mathbf{D}}\quad \underline{\mathbf{O}}\quad \mathrm{G}\quad \mathrm{T}$$

$$\mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \underline{\mathbf{O}}\quad \underline{\mathbf{G}}\quad \mathrm{T} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \underline{\mathbf{G}}\quad \underline{\mathbf{O}}\quad \mathrm{T}$$

$$\mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \mathrm{G}\quad \underline{\mathbf{O}}\quad \underline{\mathbf{T}} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \mathrm{G}\quad \underline{\mathbf{O}}\quad \underline{\mathbf{T}}$$

1 swap was made so we repeat.

$$\underline{\mathbf{A}}\quad \underline{\mathbf{C}}\quad \mathrm{D}\quad \mathrm{G}\quad \mathrm{O}\quad \mathrm{T} \longrightarrow \underline{\mathbf{A}}\quad \underline{\mathbf{C}}\quad \mathrm{D}\quad \mathrm{G}\quad \mathrm{O}\quad \mathrm{T}$$

$$\mathrm{A}\quad \underline{\mathbf{C}}\quad \underline{\mathbf{D}}\quad \mathrm{G}\quad \mathrm{O}\quad \mathrm{T} \longrightarrow \mathrm{A}\quad \underline{\mathbf{C}}\quad \underline{\mathbf{D}}\quad \mathrm{G}\quad \mathrm{O}\quad \mathrm{T}$$

$$\mathrm{A}\quad \mathrm{C}\quad \underline{\mathbf{D}}\quad \underline{\mathbf{G}}\quad \mathrm{O}\quad \mathrm{T} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \underline{\mathbf{D}}\quad \underline{\mathbf{G}}\quad \mathrm{O}\quad \mathrm{T}$$

$$\mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \underline{\mathbf{G}}\quad \underline{\mathbf{O}}\quad \mathrm{T} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \underline{\mathbf{G}}\quad \underline{\mathbf{O}}\quad \mathrm{T}$$

$$\mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \mathrm{G}\quad \underline{\mathbf{O}}\quad \underline{\mathbf{T}} \longrightarrow \mathrm{A}\quad \mathrm{C}\quad \mathrm{D}\quad \mathrm{G}\quad \underline{\mathbf{O}}\quad \underline{\mathbf{T}}$$

No swaps were made so sorting is completed.

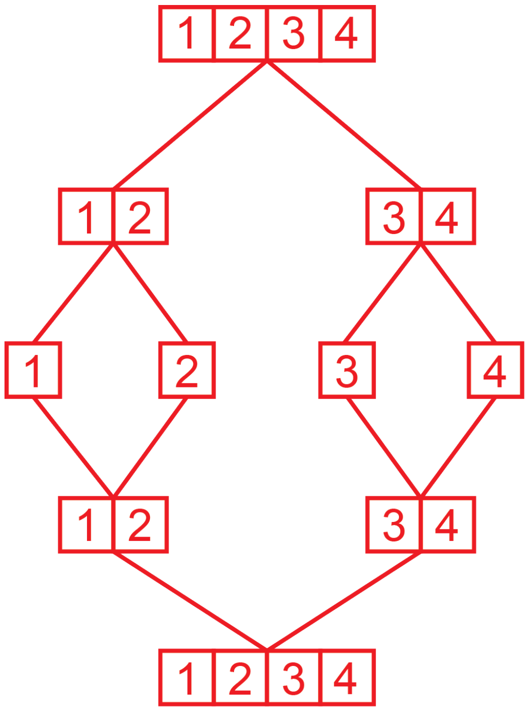
3. Tommy accidentally knocks Chuckie's number blocks off the table. Tommy wants to put the blocks back on the table in order like Chuckie had them before. However, he is just a baby so he does not notice that the blocks are already in the correct order! Help Tommy decide which sorting algorithm to use to sort these blocks:

1 2 3 4

(a) Use insertion sort.

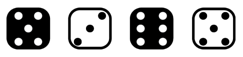
Unsorted	**Sorted**
1 2 3 4	
2 3 4	$\underline{\mathbf{1}}$
3 4	$\underline{\mathbf{2}}$ 1
3 4	1 $\underline{\mathbf{2}}$
4	$\underline{\mathbf{3}}$ 1 2
4	1 $\underline{\mathbf{3}}$ 2
4	1 2 $\underline{\mathbf{3}}$
	$\underline{\mathbf{4}}$ 1 2 3
	1 $\underline{\mathbf{4}}$ 2 3
	1 2 $\underline{\mathbf{4}}$ 3
	1 2 3 $\underline{\mathbf{4}}$

(b) Use merge sort.

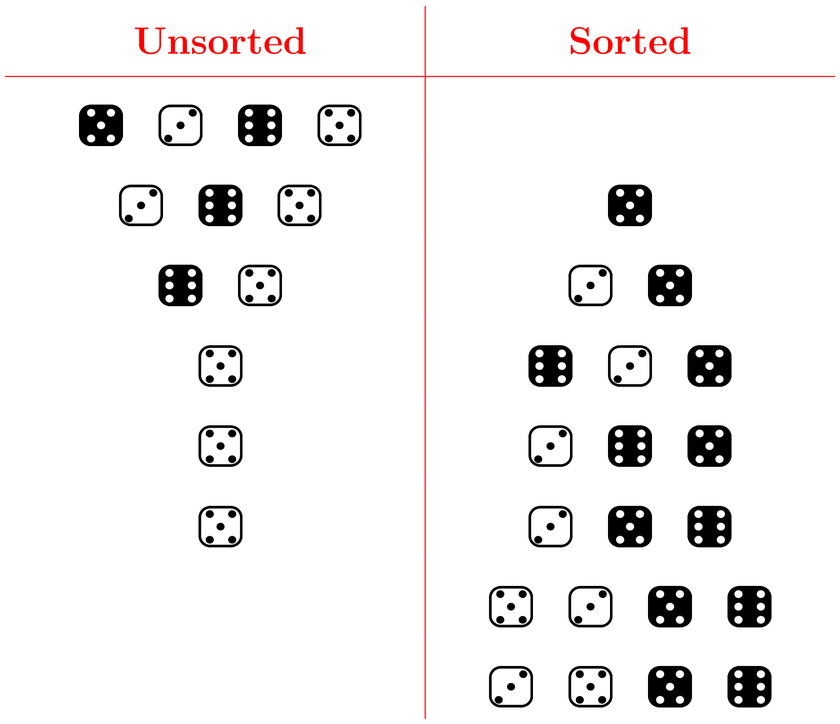


(c) How do the two algorithms compare? Which should Tommy use to sort the already sorted list and why?

Answers may vary.
Merge sort is the best algorithm for Tommy to use because it involves the least amount of steps and comparisons. Remember that in general, merge sort is faster than insertion and bubble sort. If the algorithm for insertion sort was changed so that elements are moved to the end of the sorted list instead of the beginning, insertion sort would be the best for this specific situation because Tommy would just need to move elements from the beginning of the unsorted list to the end of the sorted list one by one.

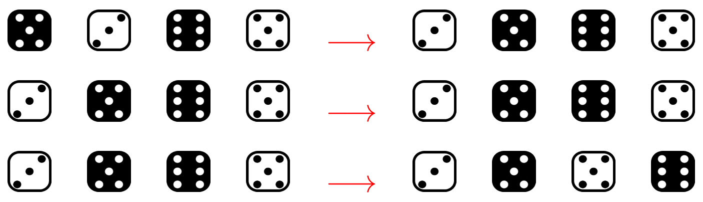
4. Sort the following dice using the algorithm specified below: 
Make sure the show which colour each die is.

(a) Use insertion sort. Remember that in the algorithm for inspection sort, we stop

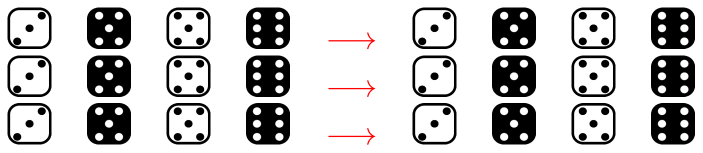


swapping elements in the sorted list when the next element is greater than or equal to the one being moved along the list.

(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 ⚄ ?)
Insertion sort is unstable and bubble sort is stable.

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

$\downarrow$

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

$\downarrow$

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

$\downarrow$

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

$\downarrow$

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

$\downarrow$

101 **2**99 **3**65 **4**01 **4**62 **5**02 **5**60 **9**02

1. Sort the list the third digit (ones digit).
2. Sort the list by the second digit (tens digit).
3. Sort the list by the third digit (hundreds 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.

6. This question compares insertion sort and radix sort which we looked at in the previous question.

   (a) Sort the following lists of numbers with both insertion and radix sort:

      i. 256 702 524 816

Insertion sort

Unsorted	**Sorted**
256 702 524 816	
702 524 816	**256**
524 816	**702** 256
524 816	256 **702**
816	**524** 256 702
816	256 **524** 702
	816 256 524 702
	256 **816** 524 702
	256 524 **816** 702
	256 524 702 **816**

Radix sort

$$\begin{array}{c}
256 \quad 702 \quad 524 \quad 816 \\
\downarrow \\
70\underline{\mathbf{2}} \quad 52\underline{\mathbf{4}} \quad 25\underline{\mathbf{6}} \quad 81\underline{\mathbf{6}} \\
\downarrow \\
7\underline{\mathbf{0}}2 \quad 8\underline{\mathbf{1}}6 \quad 5\underline{\mathbf{2}}4 \quad 2\underline{\mathbf{5}}6 \\
\downarrow \\
\underline{\mathbf{2}}56 \quad \underline{\mathbf{5}}24 \quad \underline{\mathbf{7}}02 \quad \underline{\mathbf{8}}16
\end{array}$$

ii. 256 702 524 816 130

Insertion sort

Unsorted	Sorted
256 702 524 816 130	
702 524 816 130	$\underline{\mathbf{256}}$
524 816 130	$\underline{\mathbf{702}}$ 256
524 816 130	256 $\underline{\mathbf{702}}$
816 130	$\underline{\mathbf{524}}$ 256 702
816 130	256 $\underline{\mathbf{524}}$ 702
130	$\underline{\mathbf{816}}$ 256 524 702
130	256 $\underline{\mathbf{816}}$ 524 702
130	256 524 $\underline{\mathbf{816}}$ 702
130	256 524 702 $\underline{\mathbf{816}}$
	$\underline{\mathbf{130}}$ 256 524 702 816

Radix sort

$$\begin{array}{c}
256 \quad 702 \quad 524 \quad 816 \quad 130 \\
\downarrow \\
13\underline{\mathbf{0}} \quad 70\underline{\mathbf{2}} \quad 52\underline{\mathbf{4}} \quad 25\underline{\mathbf{6}} \quad 81\underline{\mathbf{6}} \\
\downarrow \\
7\underline{\mathbf{0}}2 \quad 8\underline{\mathbf{1}}6 \quad 5\underline{\mathbf{2}}4 \quad 1\underline{\mathbf{3}}0 \quad 2\underline{\mathbf{5}}6 \\
\downarrow \\
\underline{\mathbf{1}}30 \quad \underline{\mathbf{2}}56 \quad \underline{\mathbf{5}}24 \quad \underline{\mathbf{7}}02 \quad \underline{\mathbf{8}}16
\end{array}$$

iii. 256 702 524 816 130 888

Insertion sort

Unsorted	Sorted
256 702 524 816 130 888	
702 524 816 130	$\underline{\mathbf{256}}$
524 816 130 888	$\underline{\mathbf{702}}$ 256
524 816 130 888	256 $\underline{\mathbf{702}}$
816 130 888	$\underline{\mathbf{524}}$ 256 702
816 130 888	256 $\underline{\mathbf{524}}$ 702
130 888	$\underline{\mathbf{816}}$ 256 524 702
130 888	256 $\underline{\mathbf{816}}$ 524 702
130 888	256 524 $\underline{\mathbf{816}}$ 702
130 888	256 524 702 $\underline{\mathbf{816}}$
888	$\underline{\mathbf{130}}$ 256 524 702 816
	$\underline{\mathbf{888}}$ 130 256 524 702 816
	130 $\underline{\mathbf{888}}$ 256 524 702 816
	130 256 $\underline{\mathbf{888}}$ 524 702 816
	130 256 524 $\underline{\mathbf{888}}$ 702 816
	130 256 524 702 $\underline{\mathbf{888}}$ 816
	130 256 524 702 816 $\underline{\mathbf{888}}$

Radix sort

256 702 524 816 130 888

↓

13$\underline{\mathbf{0}}$ 70$\underline{\mathbf{2}}$ 52$\underline{\mathbf{4}}$ 25$\underline{\mathbf{6}}$ 81$\underline{\mathbf{6}}$ 88$\underline{\mathbf{8}}$

↓

7$\underline{\mathbf{0}}$2 8$\underline{\mathbf{1}}$6 5$\underline{\mathbf{2}}$4 1$\underline{\mathbf{3}}$0 2$\underline{\mathbf{5}}$6 8$\underline{\mathbf{8}}$8

↓

$\underline{\mathbf{1}}$30 $\underline{\mathbf{2}}$56 $\underline{\mathbf{5}}$24 $\underline{\mathbf{7}}$02 $\underline{\mathbf{8}}$16 $\underline{\mathbf{8}}$88

(b) Which algorithm took less steps to complete for the three lists?

In general, radix sort uses the least number of steps.

(c) What is the relationship between the length of the lists and the number of steps it takes to sort them for both algorithms.

For radix sort, only 3 new "steps" were added for every additional element (putting the new element into the appropriate order for each digit). For insertion sort however, there is generally a significant increase in the number of steps/comparisons needed to sort the list with the added element.

7. Squidward Tentacles gets a part time job at an art gallery where he has to move paintings. Because some of the paintings are very heavy and Squidward has no bones (or work ethic), it is very hard for him to move them. His boss, Squilliam Fancyson, insists that he move the paintings according to a sorting algorithm so they are ordered from smalleset to largest.

The paintings and their area in cm squared are numbered below in their current sequence.

(1) Mona Lisa (204)
(2) Starry Night (681)
(3) The Scream (673)
(4) Guemica (27 117)
(5) The Persistence of Memory (79)
(6) A Sunday Afternoon on the Island of La Grande Jatte (6406)
(7) Whistler's Mother (2333)
(8) The Night Watch (15 863)

(a) What algorithm would you suggest for Squidward and why?

Answers may vary. In this question you should pay attention to not only the number of steps and comparisons that an algorithm goes through but also how many times and how far elements are moved (for the sake of poor Squidward). While radix sort may theoretically take the least number of steps, it forces Squidward to move the paintings around several times before they are sorted. While bubble sort may take many steps on the other hand, Squidward would only have to swap paintings which are next to each other when they are in the wrong order and every swap would move the painting closer to its appropriate spot. Insertion and merge sort would not be the best choices because they involve moving the paintings far from their appropriate location multiple times during the sorting process.

(b) Show the steps for sorting the sculptures with the algorithm you chose.

Answers will vary depending on the sorting algorithm chosen.

(c) Since under the sea long distance calls are very expensive, you only had a short amount of time to tell Squidward your plan for how to sort the sculptures. In one or two sentences, what would you say to Squidward?

Answers will vary depending on the sorting algorithm chosen.

8. **Challenge question.** For every algorithm we looked at in Math Circles and the problem set, create a "worst case" list from the numbers 1, 2, 3, 4, 5, 6, 7, 8. The worst case is the sequence of the eight numbers which will take the most steps to sort.

   Insertion sort: 1, 2, 3, 4, 5, 6, 7, 8
   Bubble sort: 8, 7, 6, 5, 4, 3, 2, 1
   Merge sort: 5, 1, 7, 3, 6, 2, 8, 4
   Radix sort: The initial sequence does not matter since these are all 1 digit numbers.

9. **Challenge question.** When we use bubble sort, we go through and compare each pair in the list one last time at the end of the algorithm to make sure every element is sorted into its correct place. However, during these last comparisons, the list is already sorted! How can we change the algorithm so it does not need to go through the already sorted list one last time to check that every element is in the correct order.

   Go back and look at the previous examples of bubble sort and see if you can notice a pattern when you are carrying out the steps of sorting.

   Let's take a look at the Tournament of Algorithms example. Notice that after going through every pair in the list the first time, the largest number has been moved to its correct spot at the end of the list. After going through the entire list a second time, the second largest number has been moved to its correct spot as well. If you use the worst case of Bubble sort from the previous question, you can see that this pattern will continue until the whole list is sorted. You can change the bubble sort algorithm so that every time when it finishes going through the entire list, it cuts off the last element because it is already in the correct place and ignores it for the rest of the sorting. This means that is some cases we can ignore the last step of checking if everything is in the right order because we know that to sort n elements, bubble sort will need to go through the entire list at most $n-1$ times and that each time, there will be one less comparison you have to make.