

**Tech Tips:**

- To print, click File>Print
- To edit, click the SIGN IN button in the upper, right corner. Then click File>Make a copy
- To download, click File>Download

**Course Standard & Goal****Resources & Activities**

Curriculum Standard  
*Learning Goal (Parent-friendly language)*

- [Unit 1: General Information](#) – Use this resource to review the general information about computers that you have covered in your study of computer science. . This information is used for our NOCTI Computer Programming credentialing exam.

Curriculum Standard  
*Learning Goal (Parent-friendly language)*

- [Unit 2: Searching Algorithms](#) – Use this resource to learn how **algorithms** search through data (in a step-by-step process that will always give the right solution for any input)

Curriculum Standard  
*Learning Goal (Parent-friendly language)*

- [Unit 3: Sorting Algorithms](#) – Use what you have learned about searching algorithms to take it a step further and investigate sorting algorithms.

- \*Must be turned in for course credit.**



# Unit 1: General Information

Computing Fundamentals covers a foundational understanding of computer hardware, software, operating systems, peripherals, and troubleshooting. It is designed to help you maximize the value and benefits of using computer technology. This section meets the requirements the **NOCTI Computer Programming Credentialing Exam, General Information and Concepts (24%)** section. This section also meets the **IC3 Global Standard 5 (GS5) of the IC3 Digital Literacy program**. The GS5 certification is comprised of three exams: Computing Fundamentals, Living Online, and Key Applications, and this section covers **100% of the Computing Fundamentals exam**.



## Fill in the Blanks

**Instructions:** Fill in the missing words according to the information presented. A word bank below has been provided for you.

200	HTTPS	Spotlight Search
business	iMac	Tablets
cellular	Internet	terabytes
clicking	logical	touchpad
Dock	Mac	USB cable
driver	Malware	view
Geography	password	virus
gigabytes	peripheral	viruses
Google Docs	smartphones	Windows
home	speed	

1. **(Hardware, Storage, and Connections)** A desktop computer is primarily used in home or \_\_\_\_\_ settings.
2. **(Hardware, Storage, and Connections)** The \_\_\_\_\_ is an example of a desktop computer that combines all of the external pieces of a traditional desktop into one single piece of hardware.
3. **(Hardware, Storage, and Connections)** A laptop computer utilizes a \_\_\_\_\_ in place of a mouse.

4. **(Hardware, Storage, and Connections)** \_\_\_\_\_ and smartphones offer even more portability than a laptop computer.
5. **(Hardware, Storage, and Connections)** Computer storage is often stored in gigabytes or \_\_\_\_\_.
6. **(Hardware, Storage, and Connections)** As you fill the storage for your device, the computer's \_\_\_\_\_ will decrease.
7. **(Hardware, Storage, and Connections)** Cameras, smartphones, external hard drives, and microphones are examples of \_\_\_\_\_ computer devices.
8. **(Hardware, Storage, and Connections)** A \_\_\_\_\_ is often used to connect a peripheral device and a computer together.
9. **(Hardware, Storage, and Connections)** A \_\_\_\_\_ needs to be installed before a printer and computer can communicate.
10. **(Hardware, Storage, and Connections)** You can connect to a network using \_\_\_\_\_, Wi-Fi, or wired connections.
11. **(Hardware, Storage, and Connections)** Wi-Fi signals rarely get higher than \_\_\_\_\_ feet.
12. **(Hardware, Storage, and Connections)** \_\_\_\_\_ plays a role in the range of service and the signal strength of your wireless networks.
13. **(Navigating Your Device)** You can move a file into a folder on a Windows desktop by \_\_\_\_\_ and dragging the file and placing it in the folder.
14. **(Navigating Your Device)** The \_\_\_\_\_ on a Mac computer is where you can access programs saved in the system.

15. **(Navigating Your Device)** Microsoft Word 2016 can be used on both \_\_\_\_\_ and Windows machines.
16. **(Navigating Your Device)** The Mac \_\_\_\_\_ feature lets you find items quickly on your computer.
17. **(Navigating Your Device)** One way to keep your information organized is to use \_\_\_\_\_ and meaningful names for your data.
18. **(Cloud Computing)** \_\_\_\_\_ is an example of a cloud computing app.
19. **(Cloud Computing)** Cloud computing services are available as long as you have an \_\_\_\_\_ connection.
20. **(Cloud Computing)** Setting a document to \_\_\_\_\_ only makes it impossible for people you share it with to edit the document.
21. **(Computer and Security)** A \_\_\_\_\_ is detrimental software designed to run on a computer.
22. **(Computer and Security)** \_\_\_\_\_ is a general term which refers to viruses, Trojan horses, spyware, and other harmful software.
23. **(Computer and Security)** It is important to use a \_\_\_\_\_ to secure your mobile devices.
24. **(Computer and Security)** Updates can help protect your devices from \_\_\_\_\_.
25. **(Computer and Security)** If a website has \_\_\_\_\_ in the address field, it means it is a secure website.



## Hardware and Storage

---

### Computers and Their Characteristics Description:

There are several types of computers: desktops, laptops, tablets, phones, and other mobile devices. Each type can perform similar functions, but some perform certain tasks better than others and some have advantages over others.

### Understanding Memory and Storage Description:

Computer memory and storage are two factors that affect a device's capability to perform tasks. Typically, the more memory the faster your computer will run. Computer storage is where you store applications, photos, videos, and other files—the more storage you have, the more files you can save to your computer.

Currently, memory is measured in gigabytes (GB) while storage is measured in both gigabytes (GB) and terabytes (TB) with terabytes being bigger (1 TB = 1024 GB).

1. Which of the following computers would be the fastest?

- 16 GB of memory with 500 GB of storage
- 12 GB of memory with 1 TB of storage
- 8 GB of memory with 750 GB of storage

2. Which of the following computers would store the most files?

- 16 GB of memory with 500 GB of storage
- 12 GB of memory with 1 TB of storage
- 8 GB of memory with 750 GB of storage



## Characteristics of Devices

---

Device Characteristics Bank:

Portable (2 possible)

More powerful (2 possible)

Uses mouse or touchpad (2 possible)

Uses gestures or voice (1 possible)

Usually made up of a tower, monitor, keyboard, and mouse (1 possible)

Prone to theft (1 possible)

Device - Desktops:

- Portable
- More powerful
- Uses mouse or touchpad
- Uses gestures or voice
- Usually made up of a tower, monitor, keyboard, and mouse
- Prone to theft

Device - Laptops:

- Portable
- More powerful
- Uses mouse or touchpad
- Uses gestures or voice
- Usually made up of a tower, monitor, keyboard, and mouse
- Prone to theft

Device - Mobile Phones:

- Portable
- More powerful
- Uses mouse or touchpad
- Uses gestures or voice
- Usually made up of a tower, monitor, keyboard, and mouse
- Prone to theft



**End of Unit 1: General Information**



## Unit 2: Searching Algorithms

**Searching** for a keyword, a value, or a specific piece of data (information) is the basis of many computing applications.

In this lesson, we will use these **algorithms** to search through data, in a step-by-step process that will always give the right solution for any input you give it as long as the process is followed exactly. If we check every value in a list, we are guaranteed to find the right number (as long as it is in the list!).



### Arrays

We often want to store lists of values when we're creating programs, and these lists are called **arrays**.

When we want to access a particular value in an array, we access it by referencing its "**index**" in the array, which represents its position. The first index in an array is usually "0", and if we perform a **find()** function/method, coding will either indicate the index number location of the value or an indicator it was not found (typically a "-1").

In our array of numbers, fill in the index values below:

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
5	9	10	2	90	4

What is the **length()** or how many values are listed in this array?

\_\_\_\_\_



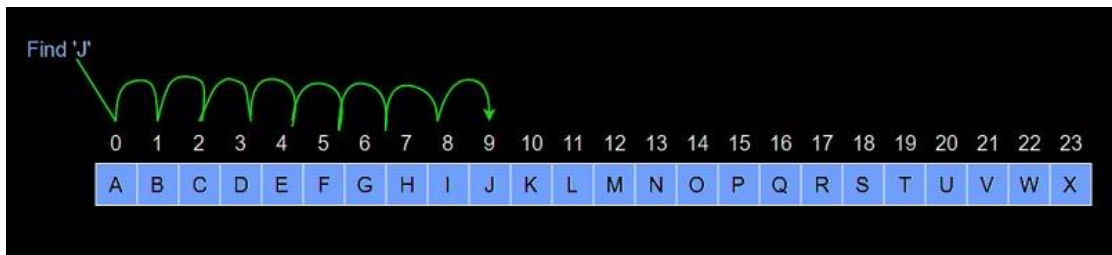
### Sequential Search

A **sequential search** is when you look at each piece of data, one by one, and don't stop until you find what you are looking for. You can use a sequential search on any data, whether it is sorted or unsorted (sorted means placed in order lowest to highest or highest to lowest – covered in Unit 3). Sequential search is used to search through data that is unsorted.

5	9	10	2	90	4
---	---	----	---	----	---

In the above array what is the index value when searching for the number 2?

\_\_\_\_\_

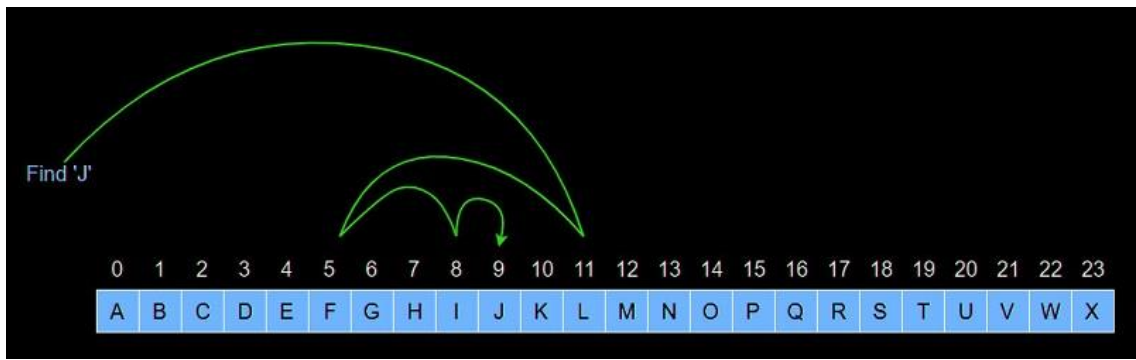


What is the value of the item found at index 9? \_\_\_\_\_



## Binary Search

A **binary search** is an algorithm that tells us how to efficiently find a specific value in an ordered (sorted) list. It is called 'binary' search because each time you look at a value in the list you divide the list into 2 parts, one is discarded and the other is kept. The word "binary" here just means something that has two parts (greater than the value being searched OR less than the value being searched).



Could you perform a binary search on the below list?

5	9	10	2	90	4
---	---	----	---	----	---

Yes \_\_\_\_\_ No \_\_\_\_\_



## Guessing Game

***("I'm thinking of a number between 1 and 100(maximum)")***

### Linear Search Algorithm

Here's a step-by-step description of using **linear search** to play the guessing game:

1. Let min = 1 and max = (integer limit)
2. Starting at min, guess every number in order until you guess number <= max.
3. If you guessed the number, stop. You found it!

This algorithm is relatively easy to program, but could potentially take a long time. Explain how a Linear Search for a guessing game could be extremely inefficient.

---



---



## Binary Search Algorithm

**Directions:** Complete the step-by-step description of using a **binary search** to play the guessing game (feel free to rewrite #2, #4, & #6 if you want to):

1. \_\_\_\_\_
2. Guess the average of max and min, rounded down so that it is an integer.
3. \_\_\_\_\_
4. If the guess was too low, set min to be one larger than the guess.
5. \_\_\_\_\_
6. Go back to step two.



## Maximum Number of Binary Guesses Possible

We know that linear search on an array of 100 elements might have to make as many as 100 guesses (called "worst case scenario"). Binary searches make fewer guesses than linear searches. The difference between the worst-case number of guesses for linear search and binary search becomes more striking as the array length increases. Let's see how to analyze the maximum number of guesses that binary search makes.

There's a mathematical function which calculates the number of times we repeatedly halve, starting at the max number, until we get the guessed number:

base-2 logarithm of max ( **$\log_2(\text{max})$** )

Directions: Complete the chart of the maximum number of binary guesses possible:

max	$\log_2(\text{max})$	max	$\log_2(\text{max})$
2	1	100	at most 6
4		128	
8		256	
10	at most 3	512	9
16		1024	
32		1,048,576	
64		2,097,152	21



## Testing Your Knowledge

**Question 1:** 32 conferences qualified for the 2020 March Madness Conference Bracket (although the conference was cancelled ☹). If the names of the teams were arranged in sorted order (an array), how many items in the array would a binary search have to examine to find the location of a particular team in the array, in the worst case? Circle the correct answer.

- A. At most, 32
- B. At most, 1
- C. At most, 6
- D. At most. 16

**Question 2:** In 2013, there were 193 member states in the United Nations. If the names of these states were sorted alphabetically in an array, about how many names would binary search examine to locate a particular name in the array, in the worst case? Circle the correct one.

- A. No more than 193.
- B. No more than 64.
- C. No more than 4.
- D. No more than 8.
- E. No more than 128.

1 2 3 4 5 6 7 8 9  
↑ low    ↑ mid    ↑ high

End of Unit 2: Searching Algorithms



## Unit 3: Sorting Algorithms

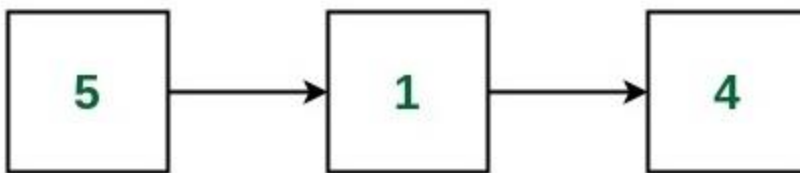
As consumers we expect computers to process information quickly so programs run faster, websites load faster, and we don't have to watch progress bars and the spinning wheels of processing information. One way to increase the speed of a computer is to write programs with fewer computational steps.

**Sorting Algorithms** sort a list of items into ascending or descending order can help either a human or a computer find items on that list quickly with algorithms like binary searches. Some **Sorting Algorithms** can have more than one comparison happening at the same time, meaning the time required will be the same as what one computer by itself would take to make multiple comparison steps. Parallel Sorting Algorithms are much faster to sort values into order because they make simultaneous comparisons.

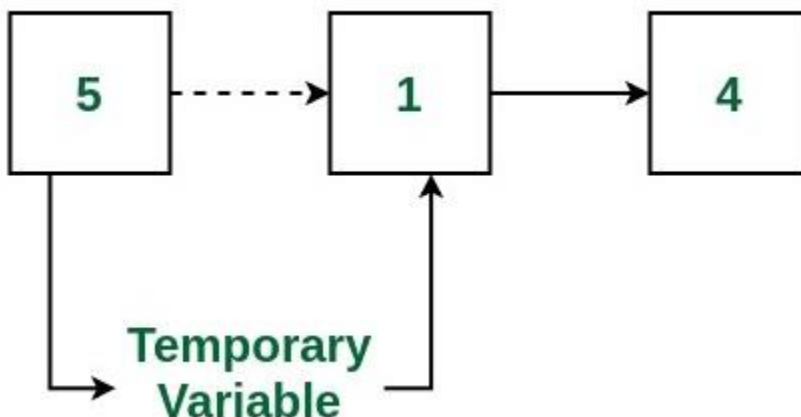


### Swap() Function using a Temporary Variable

A key step in many sorting algorithms is **swapping** the location of two items in an array, which is easy for our brains to process, but harder to program for computers, because computers do not have the concept of physical space. This function is further complicated because only one value can occupy an index value at a time. For example if we wanted to sort the below list in ascending order (Go from 5,1,4 ultimately to 1,4,5) and first needed to swap the 5 and 1 values:



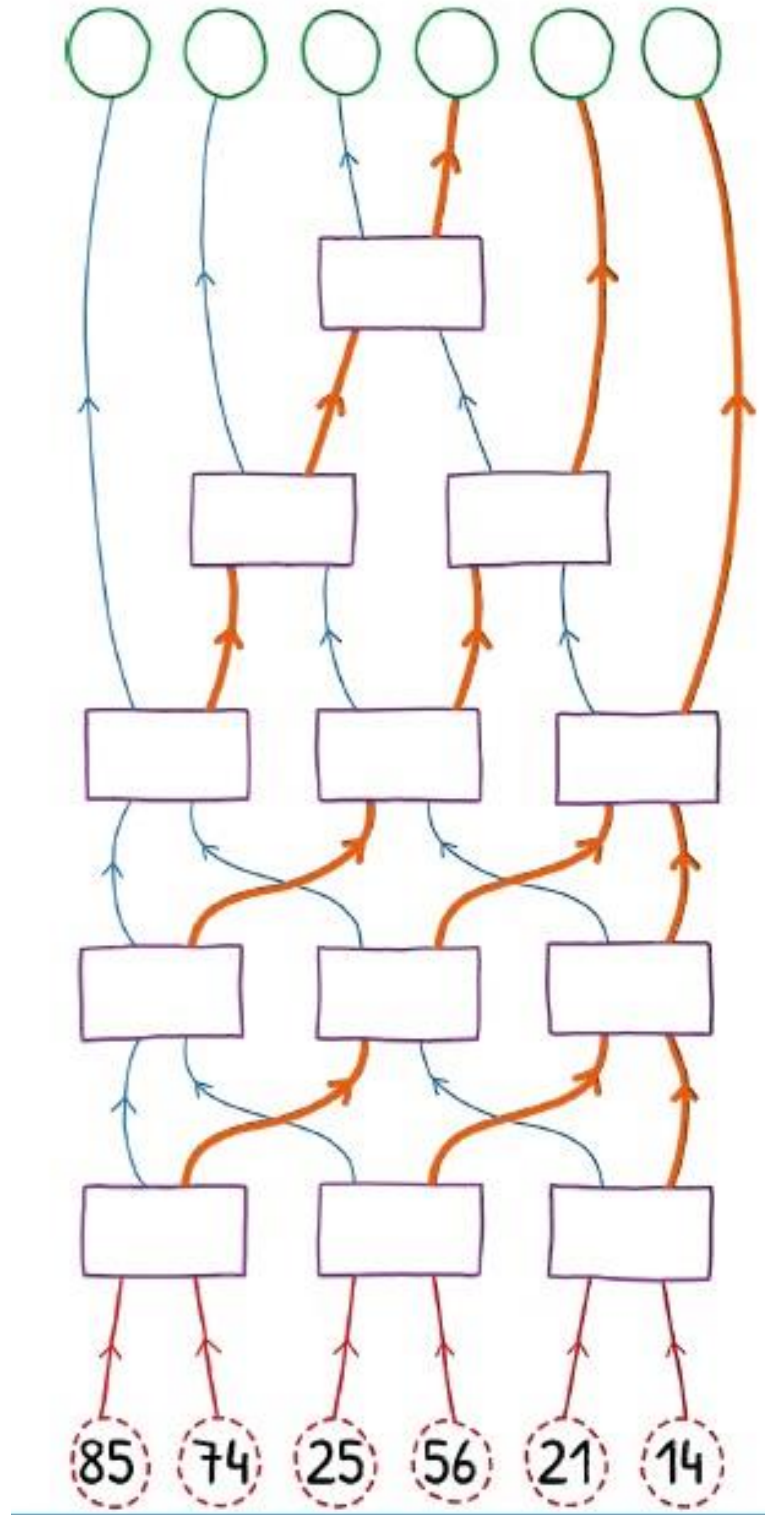
We would have to introduce a temporary variable so either value is not lost in the swap.



Once **swap()** is programmed correctly we have many ways to sort our lists.

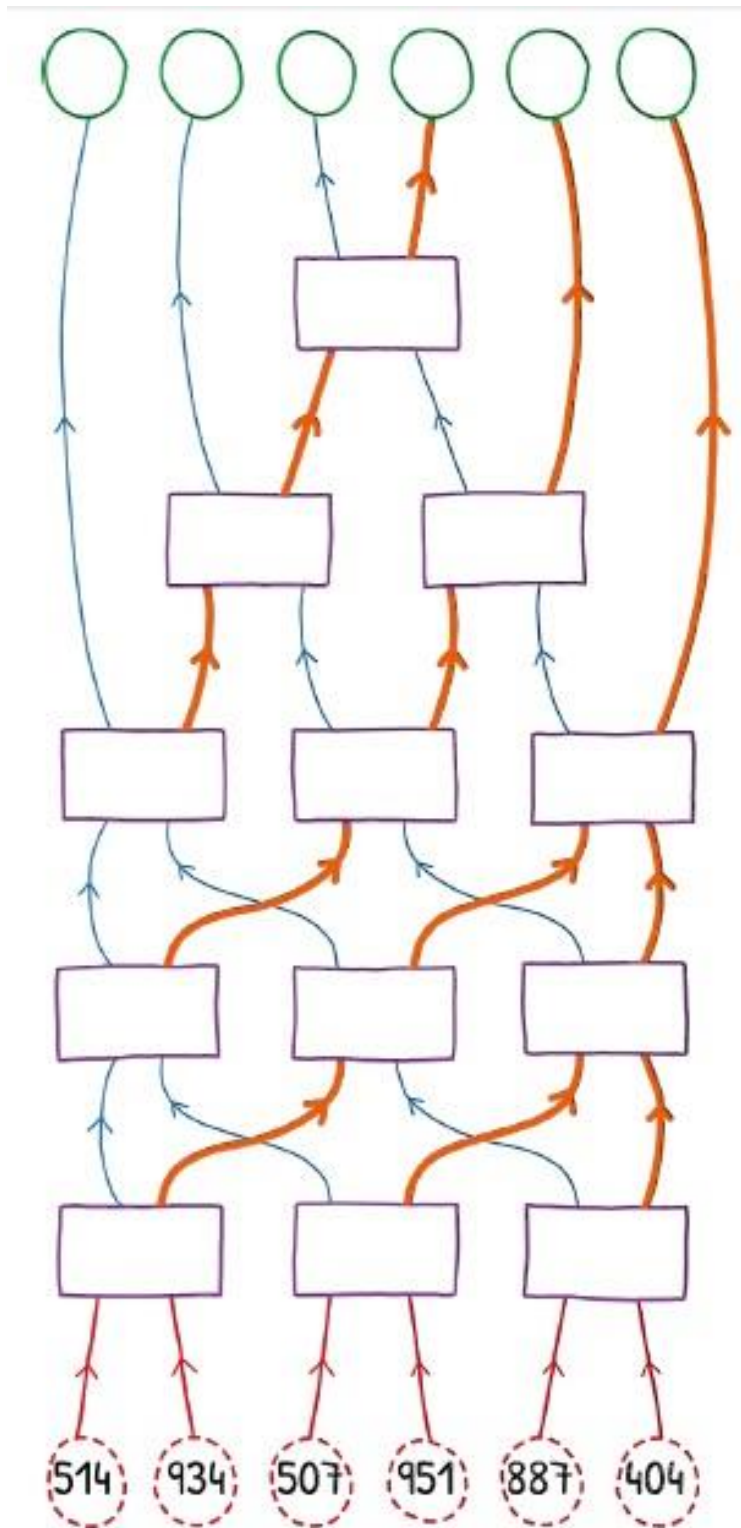
# Practice Sorting Exercise 1

Sort the values in ascending order according to the chart below:



## Practice Sorting Exercise 2

Sort the values in ascending order according to the chart below:

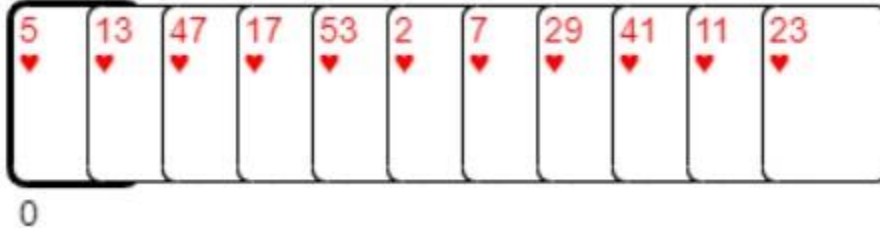


## ✓ Selection Sort

---

The Selection Sort algorithm repeatedly selects the next-smallest element and swaps it into place. Here is the pseudocode for sorting cards using selection sort:

1. Find the smallest card. Swap it with the first card.
2. Find the second-smallest card. Swap it with the second card.
3. Find the third-smallest card. Swap it with the third card.
4. Repeat finding the next-smallest card, and swapping it into the correct position until the array is sorted.



Using the above algorithm, which two cards from below will be sorted first?  
Pass 1: card \_\_\_\_\_ ♥ and card \_\_\_\_\_ ♥ swap

Pass 2: Is there a swap in the 2<sup>nd</sup> pass?

\_\_\_\_\_ yes          \_\_\_\_\_ no



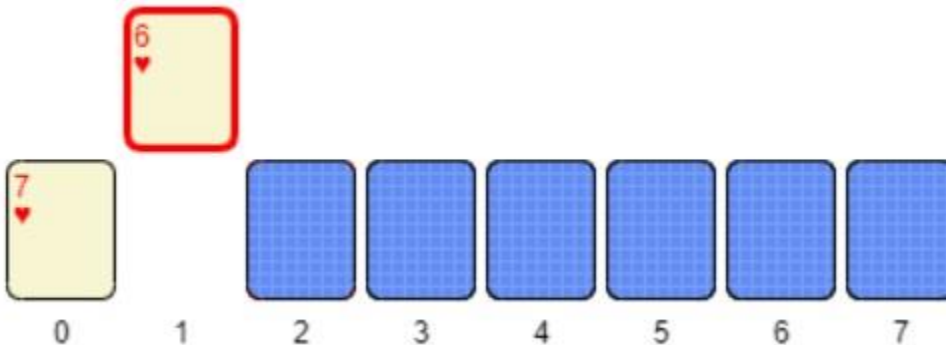
## Insertion Sort

---

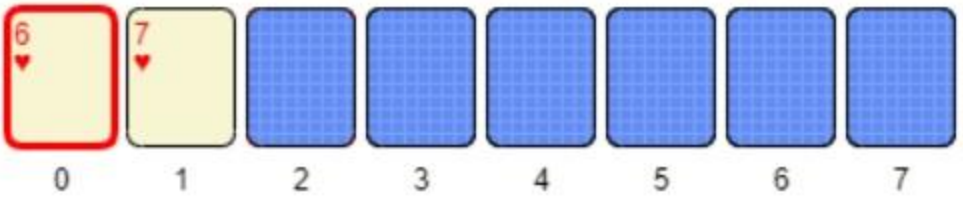
**Insertion sort** is a simple sorting algorithm that builds the final sorted array (or list) one item at a time.

In insertion sort, each new card could be smaller than some of the cards you're already holding, and so you go down the line, comparing the new card against each card in your hand, until you find the place to put it. You do not have to see the entire set of cards (like Selection Sort). You insert the new card in the right place. Then the dealer gives you another card, and you repeat the same procedure.

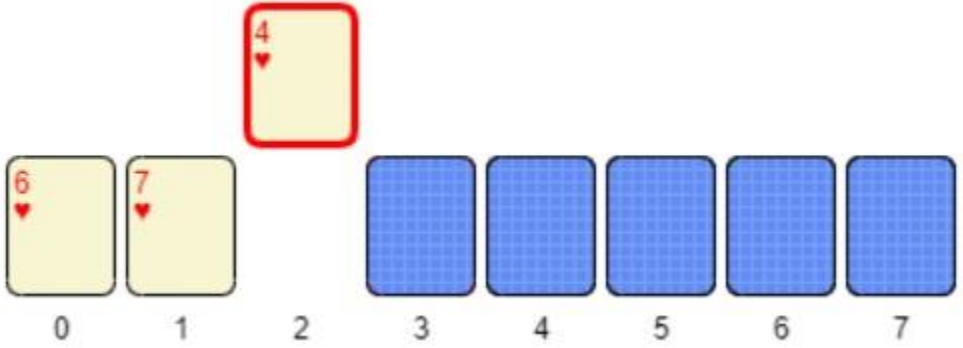
**Pass 1:**



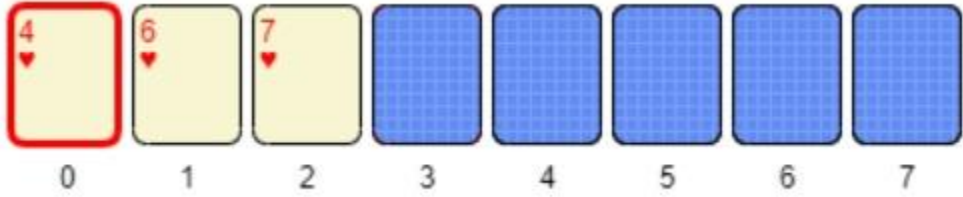
In pass 1 on the previous page, cards 7♥ and card 6♥ were compared, and card 6♥ was inserted before 7♥



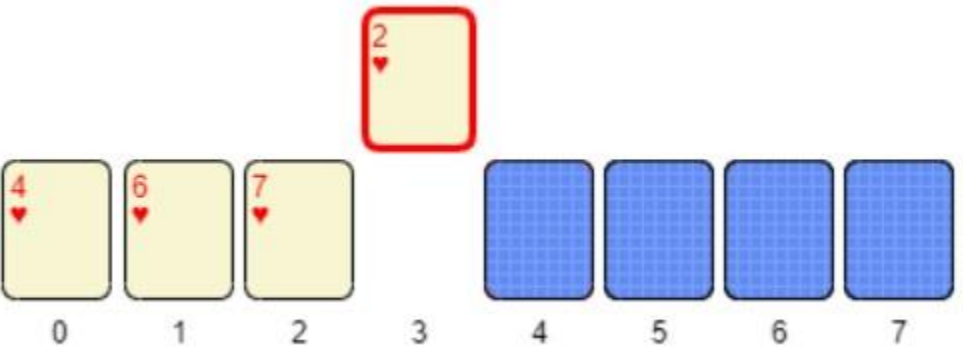
**Pass 2:**



In pass 2, card 6♥ and card 4♥ were compared, and card 4♥ was inserted before card 6♥



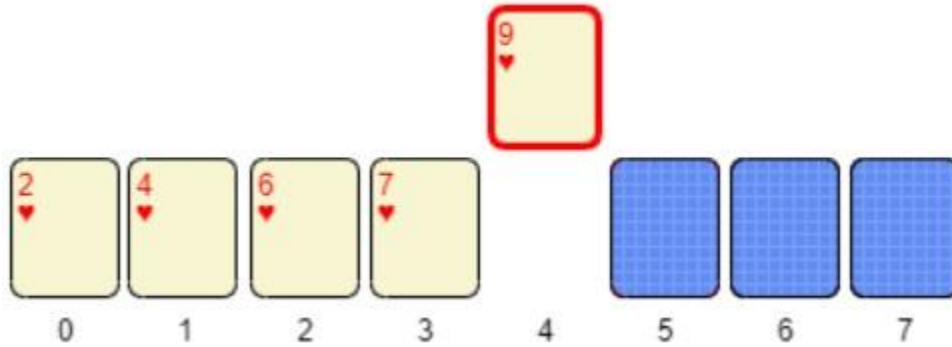
**Pass 3:**



In pass 3, does a swap or insertion occur?

\_\_\_\_\_ yes      \_\_\_\_\_ no

**Pass 4:**



In pass 4, does a swap or insertion occur?

\_\_\_\_\_ yes          \_\_\_\_\_ no

 **Analysis of Sorting Algorithms - Big  $\theta$**

---

In Unit 2 – Searching Algorithms we discussed worst-case scenarios where searching would take up a maximum amount of passes and time. This concept is called Big-O notation or

**Big- $\theta$  Notation**

Big O notation is used in Computer Science to describe the performance or complexity of an algorithm.

For Insertion Sort the best case scenario would be denoted as:

Best case:  **$\theta(n)$**

This could be described as if the array was already sorted, all passes would have zero swaps.

For Insertion Sort the worst case scenario would be denoted as:

Worst case:  **$\theta(n^2)$**


**Directions:** Describe an array which would fit a worst-case scenario for Insertion Sort:

---

---

---

---

 **End of Unit 3: Sorting Algorithms**