Billy robinson

Contents

[**How is this topic important within Computer Science?** 2](#_Toc27146447)

[**History of this topic within computer science** 2](#_Toc27146448)

[**Examples of this topic being implemented in computer science** 2](#_Toc27146449)

[**Summary of what I’ve learned from this topic.** 3](#_Toc27146450)

[**Reference List** 4](#_Toc27146451)

# **How is this topic important within Computer Science?**

Graph theory is important to computer science because it is functional in a lot of the major systems that we use. The theory is utilized in many ways within computer science itself. A graph contains many nodes and edges (edges being lines connecting nodes, whilst nodes are circle like points that typically represent a piece of data, for example, a website). They are often essential for planning out programs or navigational diagrams, thus meaning that they are highly used by many programmers, network engineers, and other industry professionals. This is also extremely important to computer science because of the ease at which graph theory allows us to do a lot of core functions that a computer does. For example, sorting and searching (discussed in more detail later), thus being essential to how computers themselves operate. This is used in multiple areas within the industry, without the theory, lots of common or everyday programs would not be able to exist.

A great example of how this could represent something that is used every day are relational databases, with nodes representing the data and edges representing the edges. It allows us to understand where information is coming from within the structure, thus allowing for easy planning and visualization of how a database works from a designer’s point of view, which can result in easier troubleshooting if any issues occur with that database in the future. This means that it can lead to a more effective solution than other, less efficient solutions.

# **History of this topic within computer science**

Graph theory was first introduced into the world by Leonhard Eule (Wilson, William Owen James, and Lloyd, 1976). He introduced the concept by introducing the Konigsberg bridge problem. This problem tried to get people to figure out if they could visit every part of the city once whilst only crossing one bridge. Bridges representing the edges of a graph and parts of the city representing the nodes of the graph. The theorem allowed people to simplify complex pieces of data down into a very simple graph format, which would allow them to do calculations much easier without any extra unneeded information.

In terms of computer science, graph theory has been widely adopted within data representation and is often used for planning programs, planning networks or general planning. Because of the nature of graphs, they allow people to easily represent data whilst making it simpler for humans to understand the data that is being represented (Slutsky, 2014). This has resulted in widespread adoption in using graphs to represent many different forms of data, from data flow in programs up to complex network security diagrams as well as showing how artificial intelligence calculates its output from a given input.

# **Examples of this topic being implemented in computer science**

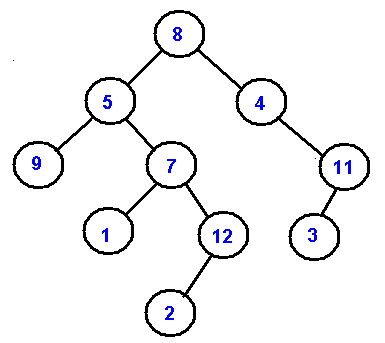
A great example of how this has been used within computer science in the real world is “Binary trees”, Binary trees are a data structure, in which each node has at most two children (nodes further down the line). These are used in computing as each node is assigned a value (Makinson, 2012), because of the structure of a binary tree, it makes sorting and searching for more efficient and quicker. Because the computer can either go down one route where the number is large or another where the number is small, thus getting to a specific number quickly as it goes down a shorter path, rather than checking if each number in a row is the number the computer is searching for.

Figure 1, A binary tree (Cmu.edu, 2009)

Another way in which graph theory is used within computer science is the optimization of data transmission via networks. Graph theory is “often used to represent the network.” (A. Ahmat, 2013). Allowing diagrams to be made which connect nodes together, allowing for people to plan extremely large networks. They can then plan out the quickest route that data should travel from one node to another node, thus allowing them to calculate how many transmissions boosters they may need to put in place in order to ensure that the entire network is covered.

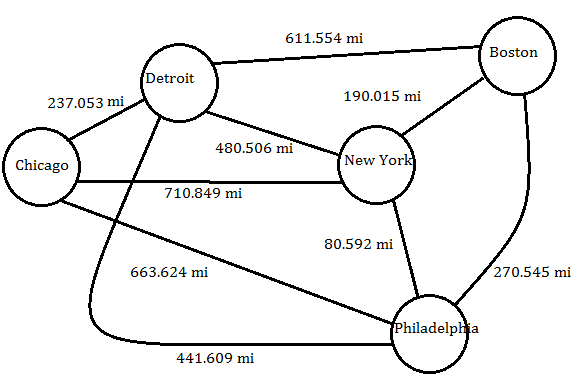
Finally, another example allows us to see how we use graph theory within computer science daily, being Google maps. Google maps use hundreds of nodes along the way to allow the program to quickly calculate the final distance (Cornell University, 2011), allowing the program to analyze which is the quickest way to get to a destination, then presenting this to a user.

Figure 2, Google map representation (Cornell University, 2011)

# **Summary of what I’ve learned from this topic.**

Within this topic, I have learned a lot about representing data which allows people to easily see what you’re trying to represent without being confused. Simple data representation is something that can be a major issue on larger projects and thanks to this topic I have discovered multiple methods of representing data, even outside of graphs. This topic has increased my knowledge surrounding the data sciences field and made me more interested in reading around topics relating to data presentation.

In addition, after researching and learning about this topic, I have developed a few skills from this topic, which will hopefully be used to assist in future topics within computer science. An example of a skill that I have developed is critical thinking, within the session that taught us the basics around this theory, we were shown and asked to work on the Konigsberg bridge problem, the problem isn’t possible. It’s important not to always think “there’s a solution” and instead be critical of something I’m being shown if a solution cannot be found. Another example of a soft skill that has been ­developed within this topic is mathematical thinking. Before this session, I wouldn’t have thought of a graph as something that could have a mathematical formula and as a result, I now understand how to turn data presentation into mathematical formulae.

# **Reference List**

* Wilson, R.J., William Owen James and Lloyd, K. (1976). *Graph theory, 1736-1936*.
* Cprogramming.com. (2015). *Cprogramming.com Article: Graphs in Computer Science - Cprogramming.com*. [online] Available at: http://www.cprogramming.com/tutorial/computersciencetheory/graphtheory.html [Accessed 11 Dec. 2019].
* ‌Codechef.com. (2019). *Tutorial on Graph Theory - part 1 | CodeChef*. [online] Available at: http://www.codechef.com/wiki/tutorial-graph-theory-part-1.
* ‌Kent.edu. (2019). *Algorithmic Graph Theory*. [online] Available at: http://www.personal.kent.edu/~rmuhamma/GraphTheory/graphTheory.htm [Accessed 11 Dec. 2019].
* Makinson, D. (2012). *Sets, logic, and maths for computing*. London: Springer.
* Cmu.edu. (2009). *Binary Trees*. [online] Available at: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html [Accessed 12 Dec. 2019].
* A. Ahmat, K. (2013). *Graph Theory and Optimization Problems for Very Large Networks*. [online] Available at: https://www.researchgate.net/publication/45862845\_Graph\_Theory\_and\_Optimization\_Problems\_for\_Very\_Large\_Networks [Accessed 12 Dec. 2019].
* Cornell University. (2011). *Google maps-It’s Just One Big Graph*. [online] Available at: https://blogs.cornell.edu/info2040/2011/09/14/google-maps-its-just-one-big-graph/. [Accessed 12 Dec. 2019].
* Slutsky, D. (2014). The Effective Use of Graphs. *Journal of Wrist Surgery*, [online] 03(02), pp.067–068. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4078179/.

Billy robinson

Contents

[**How is this topic important within Computer Science?** 7](#_Toc27146467)

[**History of this topic within computer science** 7](#_Toc27146468)

[**Examples of this topic being implemented in computer science** 8](#_Toc27146469)

[**Summary of what I’ve learned from this topic.** 8](#_Toc27146470)

[**Reference List** 9](#_Toc27146471)

# **How is this topic important within Computer Science?**

Logical reasoning is used by Developers and programs daily. It allows developers within the computer science field to create everything ranging from extremely basic algorithms with singular paths to paths with hundreds of potential paths and outcomes. When writing programs, developers must use logical reasoning in order to create conditional statements that will determine the output and flow of their programs. Without logical reasoning, a computer wouldn’t be able to perform actions based upon conditions and instead would continue following the same path within an algorithm forever without deviation. Logical reasoning has a large role within many places within computer science, one of the major areas being artificial intelligence. Ertel, Black and Mast (Cited in 2017) stated Its usage in artificial intelligence allows the computer to judge if a provided input is similar to previously seen training data.

Every single program that requires “if” statements use logical reasoning and as a result, it means that Logical reasoning has a significant impact on computer science and is used constantly. A great example of how logical reasoning works within computer science can be seen when designing a program using truth tables.

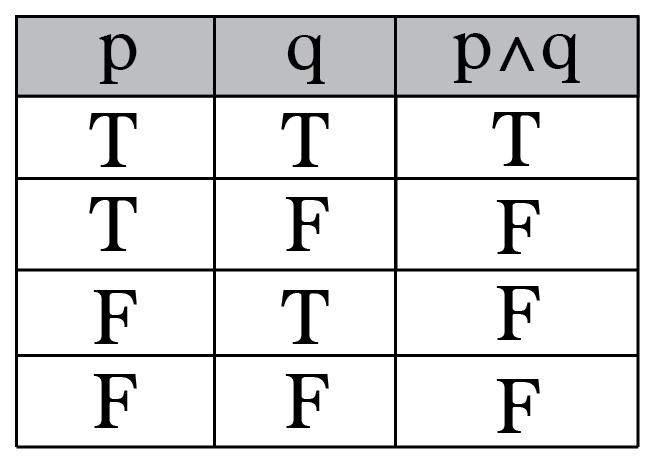
The truth tables are used by developers in order to plan out how an algorithm will flow if given specific input and allows for developers to easily debug a program and consider the logical flow of their program before writing it. It allows developers to take sometimes rather complicated problems and work out how their program would handle it and which flow their program would adopt.

Figure 1, A truth table (Berry, 2017).

# **History of this topic within computer science**

Logical reasoning has always been a thing within the human mind, allowing people to make thousands of logical choices per second without ever consciously thinking of the outcome or flow of these choices, transferring this from human knowledge into something that computers can understand and process to make logical choices has been the difficult part. The main part of history that led to the development of logical reasoning within computer science is the Turing machine.

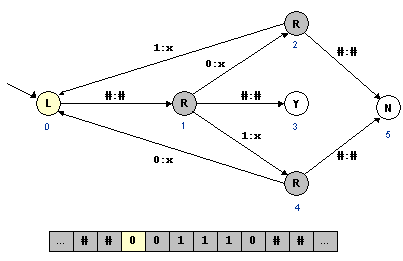
The Turing machine works by changing the direction and state of the machine depending on the value of the current cell (B  Jack Copeland, 2004). This is extremely similar to how modern programs work, by testing the value of a variable and if it’s true, following through by completing an algorithm that can update the cell’s current value and then move on so that it will be checked again in the future.

Figure 2, A Turing machine diagram (Sedgewick, 2015).

It’s important, however, to note that sometimes statements can be malformed. This means that the statement can be neither proven nor disproven. This is discussed in more depth within Gödel's incompleteness theorem. This means that if a piece of code is unchecked, it can create a logic error that will prevent the code from running correctly whilst still being perfectly valid from the point of view of the compiler.

# **Examples of this topic being implemented in computer science**

Within the computer science industry, Logical reasoning has been used in pretty much every major program, website and computer’s themselves, as a result, it’s very easy to get a large pool of examples that will demonstrate how logical reasoning has impacted and been used within the field of computer science.

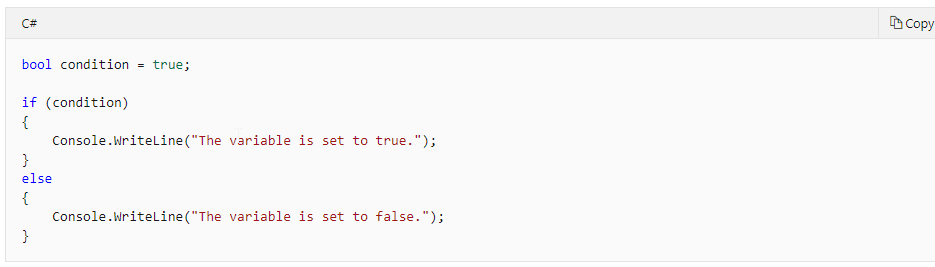


Figure 3, If else statement in C# (BillWagner, 2015).

As shown in the above code snippet, logical reasoning has been used to create multiple flows, thus allowing the program to go a different way from the normal route (the ‘else’ statement is the normal route). This meaning that instead of what would have just been a straight path with no possible deviation, instead, we can go down multiple based on a pre-made condition, thus making computers highly useful machines that can chain actions based upon pre-conditions or a set of rules that allow the program to behave in a specific way if conditions are met.

Another great example of how logical reasoning is represented within computer science are flow charts, flow charts are great examples of logical reasoning representation because within a flowchart you have logical diamond decisions, every diamond has a yes or a no-flow and allows you to see what will happen further down the line, similar to a logic table it allows you to plan out programs and allows you to troubleshoot with more ease.

# **Summary of what I’ve learned from this topic.**

Within this topic, I’ve learned a decent amount about Logic within computer science and logic in general, I’ve gained a lot of understanding about engineering concepts and about their usage within computer science Before researching and reading into this topic I hadn’t come across logic gates in this much detail before, as a result, I put a lot more research into logic gates and their usage within computer science, trying to relate them to programming with examples.

From this topic, I’ve learned a lot about thinking logically when it comes to developing and building programs, especially those in sequence driven languages such as C# and python. Because of how engrained logical reasoning is within developing programs, it means that every time I develop a program I will have to use the skills gained from this project within my work. In terms of what thinking skills have been developed via the research and knowledge gained on this topic, I’d personally say I’ve learned a lot about thinking logically. This has allowed me to develop my programming skills and helped me to develop flow charts that will allow me to effectively design and plan a program. This will help a lot in the future with troubleshooting programs to ensure that they work as expected, using tools such as truth tables in order to construct such testing and design materials.

# **Reference List**

* Berry, B. (2017). *Intro To Truth Tables & Boolean Algebra*. [online] Medium. Available at: https://medium.com/i-math/intro-to-truth-tables-boolean-algebra-73b331dd9b94 [Accessed 3 Dec. 2019].
* B  Jack Copeland (2004). *The essential Turing : seminal writings in computing, logic, philosophy, artificial intelligence, and artificial life plus the secrets of an enigma*. Oxford ; New York: Oxford University Press.
* Sedgewick, R. (2015). Universality. [online] Princeton.edu. Available at: https://introcs.cs.princeton.edu/java/53universality/ [Accessed 10 Dec. 2019].
* Ertel, W., Black, N. and Mast, F. (2017). *Introduction to Artificial Intelligence*. Cham, Switzerland: Springer.
* BillWagner (2015). *if-else - C# Reference*. [online] Microsoft.com. Available at: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/if-else [Accessed 10 Dec. 2019].
* Dalton, D.W. and Goodrum, D.A. (1991). The Effects of Computer Programming on Problem-Solving Skills and Attitudes. *Journal of Educational Computing Research*, [online] 7(4), pp.483–506. Available at: https://journals.sagepub.com/doi/abs/10.2190/762V-KV6T-D3D1-KDY2.

‌