SC/MATH 1019: Discrete Mathematics for Computer Science

Some introductory motivating examples

About this course

This course is a *mathematics* course where topics are chosen to be relevant to students studying computer science. This means we will learn mathematical *abstraction* and *rigour* as well as how to *prove* things. The mathematics will be our focus, but we also wish to keep in mind the many ways the mathematics we study can be applied. These slides contain some examples the content of the course along with some of the applications to computer science.

Simplifying if statements

Consider the following example.

$$\vdots$$

$$\textbf{if } \Big((x < y \textbf{ and } y < z) \textbf{ or } (x \geq y \textbf{ and } y < z)\Big) \textbf{ then } \text{ [do something]}$$

$$\vdots$$

Simplifying if statements

Consider the following example.

$$\vdots$$

$$\textbf{if } \Big((x < y \textbf{ and } y < z) \textbf{ or } (x \geq y \textbf{ and } y < z)\Big) \textbf{ then } \text{ [do something]}$$

$$\vdots$$

Can this example by simplified?

Simplifying if statements (cont.)

The previous example can be simplified as follows.

$$\vdots$$

$$\textbf{if } (y < z) \textbf{ then } \text{[do something]}$$

$$\vdots$$

Simplifying if statements (cont.)

The previous example can be simplified as follows.

$$\vdots$$

$$\textbf{if } (y < z) \textbf{ then} \quad \text{[do something]}$$

$$\vdots$$

What are the benefits of making such a simplication?

Simplifying if statements (cont.)

The previous example can be simplified as follows.

$$\begin{array}{c} \vdots \\ \textbf{if } (y < z) \textbf{ then } \text{ [do something]} \\ \vdots \end{array}$$

What are the benefits of making such a simplication?

- Speed/efficiency
- Readability (maintenance, debugging, etc.)

Proposition logic

In Chapter 1 we will learn about *Proposition logic*. We will cover various rules for manipulating and simplifying expressions involving **and**, **or**, and **not** which evaluate to either **true** or **false**.

Proposition logic

In Chapter 1 we will learn about *Proposition logic*. We will cover various rules for manipulating and simplifying expressions involving **and**, **or**, and **not** which evaluate to either **true** or **false**.

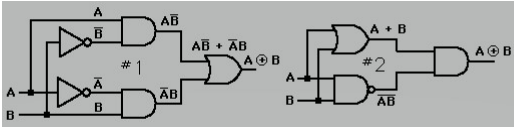
$$
\begin{aligned}
(p \land q) \lor (\neg p \land q) &\equiv ((p \land q) \lor \neg p) \land ((p \land q) \lor q) \\
&\equiv (p \lor \neg p) \land (q \lor \neg p) \land q \\
&\equiv T \land q \\
&\equiv q
\end{aligned}
$$

Boolean algebra

A related topic to propositional logic is *Boolean algebra* which is in Chapter 12 of the textbook. We will not cover Boolean algebra this semester, but after this course you will be able to understand Boolean algebra if you so desire. Boolean algebra also has applications (for example, in logic gates and minimization of circuits).

A related topic to propositional logic is *Boolean algebra* which is in Chapter 12 of the textbook. We will not cover Boolean algebra this semester, but after this course you will be able to understand Boolean algebra if you so desire. Boolean algebra also has applications (for example, in logic gates and minimization of circuits).

$$A\overline{B} + \overline{A}B = (A + B)\overline{(AB)}$$



Computing runtime/complexity

Consider the following example.

$$
\begin{array}{l}
i = 1 \\
j = 1 \\
\textbf{while } j < n \\
\quad \textbf{if } i < j \textbf{ then } i = i + 1 \\
\quad \textbf{else } j = j + 1, i = 1
\end{array}
$$

Computing runtime/complexity

Consider the following example.

$$
\begin{array}{l}
i = 1 \\
j = 1 \\
\textbf{while } j < n \\
\quad \textbf{if } i < j \textbf{ then } i = i + 1 \\
\quad \textbf{else } j = j + 1, i = 1
\end{array}
$$

How many times does this loop run?

Computing runtime/complexity (cont.)

$$
\begin{array}{l}
i = 1 \\
j = 1 \\
\textbf{while } j < n \\
\quad \textbf{if } i < j \textbf{ then } i = i + 1 \\
\quad \textbf{else } j = j + 1, i = i + 1
\end{array}
$$

The above loop runs for the following pairs (i, j).

$$
\begin{array}{l}
(1,1) \\
(1,2),(2,2) \\
(1,3),(2,3),(3,3) \\
\quad\vdots \\
(1,n),(2,n),\ldots,(n,n)
\end{array}
$$

Computing runtime/complexity (cont.)

Looking at the pairs

$$\begin{array}{l} (1,1) \\ (1,2),(2,2) \\ (1,3),(2,3),(3,3) \\ \quad\vdots \\ (1,n),(2,n),\ldots,(n,n) \end{array}$$

we see that in total we have

$$1+2+3+\cdots+n=\sum_{k=1}^{n} k=\frac{n(n+1)}{2}$$

pairs which is $O(n^2)$ (i.e. a quadratic function in n).

Summation, big-O, and recurrence relations

We will learn about *summation* and and $\sum$-*notation* in Chapter 2. In Chapter 3 we will cover *big-O notation* and. These are important and useful concepts in the analysis of algorithms.

Summation, big-O, and recurrence relations

We will learn about *summation* and and $\sum$-*notation* in Chapter 2. In Chapter 3 we will cover *big-O notation* and. These are important and useful concepts in the analysis of algorithms.

In Chapter 8 we will study *recurrence relations* which are another important tool in understanding algorithms.

Summation, big-O, and recurrence relations

We will learn about *summation* and and $\sum$-*notation* in Chapter 2. In Chapter 3 we will cover *big-O notation* and. These are important and useful concepts in the analysis of algorithms.

In Chapter 8 we will study *recurrence relations* which are another important tool in understanding algorithms.

$$F_0 = 0, F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, n \geq 2$$

$$0, 1, 1, 2, 3, 5, 8, 13, \ldots$$

Connecting points

Question: Given n points, or vertices, what is the minimum number of lines, or edges, needed so that we can get from any point to any other point by traveling along the lines? Here a given line must connect exactly two points.

Connecting points

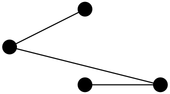
Question: Given n points, or vertices, what is the minimum number of lines, or edges, needed so that we can get from any point to any other point by traveling along the lines? Here a given line must connect exactly two points.

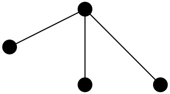
Try for $n = 3$, 4, and 5 points and see if you can find a pattern.

Connecting points

Question: Given n points, or vertices, what is the minimum number of lines, or edges, needed so that we can get from any point to any other point by traveling along the lines? Here a given line must connect exactly two points.

Try for $n = 3$, 4, and 5 points and see if you can find a pattern.

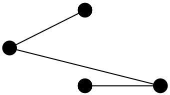


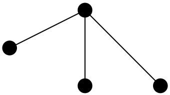


Connecting points

Question: Given n points, or vertices, what is the minimum number of lines, or edges, needed so that we can get from any point to any other point by traveling along the lines? Here a given line must connect exactly two points.

Try for $n = 3$, 4, and 5 points and see if you can find a pattern.





Answer: $n - 1$ lines

Graphs and trees

In Chapter 10 we will learn about *graphs* and in Chapter 11 we will learn about *trees*. Graph are objects which model many things, and trees are graphs which are minimally connected. Both these objects have many applications (networks, parsing, data structures, etc.).

Graphs and trees

In Chapter 10 we will learn about *graphs* and in Chapter 11 we will learn about *trees*. Graph are objects which model many things, and trees are graphs which are minimally connected. Both these objects have many applications (networks, parsing, data structures, etc.).