<u>The Pumping lemma</u>: if L is a regular language, Then there is a number p (The pumping length) where if w is any word in L of length at least p, Then $w = xuy$ for words x, y, u satisfying

(1) $u \neq \varepsilon$ (i.e. $|u| > 0$, The length of u is positive)

(2) $|xu| \leq p$

(3) $xu^n y \in L \quad \forall n \geq 0$.

<u>Remark</u> p can be taken to equal the number of states of a

deterministic finite state acceptor that recognizes L (we know such a finite state acceptor exists because L is regular).

Sketch of proof The name of the lemma comes from the fact that if L is regular, then all of its words can be pumped through a finite state acceptor that recognizes L. We assume this acceptor is deterministic and has p states. We will show the pumping lemma is a consequence of the pigeonhole principle we studied in the unit on functions. If a word w has length l, then the finite state acceptor must process l pieces of information ($w = a_1 a_2 \cdots a_l$, where $a_k \in A$ $\forall k$, $1 \le k \le l$) $\Rightarrow$ it passes through $l+1$ states starting w/ the initial state. In the hypotheses of the lemma, we assume $|w| = l \ge p$, but $p = \#$(states of the acceptor) $\Rightarrow$ the acceptor passes through $l+1 \ge p+1$ states to process w, and therefore at least one state is repeated among the first $p+1$. Let $s_1, \ldots, s_{l+1}$ be the sequence of states. $|w| = l \ge p$ implies $s_i = s_j$ with $i < j \le p+1$. Now we set x to be the part of w that makes the acceptor pass through states $s_1, s_2, \ldots, s_i$, i.e. $x = a_1 a_2 \cdots a_{i-1}$ (the first $i-1$ letters in w). We set u to be the part of w that makes the acceptor pass through states $s_i, s_{i+1}, s_{i+2}, \ldots, s_j$. In other words, $u = a_i a_{i+1} \cdots a_{j-1}$. Since $i < j$, $|u| \ge 1 \Rightarrow u \ne \varepsilon$. Finally, set y to be the part of w (the tail end) that makes the acceptor pass through states $s_j, s_{j+1}, \ldots, s_{l+1}$, i.e. $y = a_j a_{j+1} \cdots a_l$. Since $j \le p+1$, $j-1 \le p$, so $|xu| = |a_1 a_2 \cdots a_{j-1}| = j-1 \le p$ as needed. Furthermore, $s_i = s_j$, so at the beginning of u and at its end the acceptor is in the same state $s_i = s_j \Rightarrow x u^n y$ is accepted for every $n \ge 0 \Rightarrow x u^n y \in L$ as needed. We have obtained conditions (1) – (3).

(q.e.d.)

Applications of The pumping lemma

As a statement, the pumping lemma is the implication $P \to Q$ w/ P being the sentence "L is a regular language" and Q being the decomposition of every w, $|w| \geq p$ as $w = xuy$. We use the contrapositive $\neg Q \to \neg P$ (tautologically equivalent to $P \to Q$) as our criterion for detecting nonregular languages.

Examples ① $L = \{0^m 1^m \mid m \in \mathbb{N}, m \geq 0\}$ is not regular.

Let $w = 0^m 1^m$. We cannot decompose w as $w = xuy$ because whatever we let u be, we get a contradiction to $xu^n y \in L \ \forall n \geq 0$.

If $u \in 0^*$ (string of 0's), $x \in 0^*$ and $y \in 1^*$ (string of 1's). There are values of n for which $xu^n y \notin L$.

If $u \in 1^*$ we get a contradiction the same way.

If $u = 0^k 1^k$ for $k \geq 1$ $xu^2 y \notin L \ \forall x, y$ words!

② $L = \{0^m \mid m \text{ is prime}\}$ is not regular.

Since $w = 0^m$, x, u, y can consist only of 0's, so then $x = 0^i$, $u = 0^j$, $y = 0^k$. If $xu^n y \in L \ \forall n \geq 0$ then $i + nj + k$ is prime $\forall n \geq 0$, which is impossible.

Set $n = i + 2j + k + 2$, then $i + nj + k = i + (i + 2j + k + 2)j + k$

$= i + ij + 2j^2 + jk + 2j + k = i(j+1) + 2j(j+1) + k(j+1)$

$= (j+1)(i + 2j + k)$, where $|u| > 0$ so $j \geq 1$. Therefore, $n = (j+1)(i+2j+k)$ is not prime!

Practice at home weitz.de/pump (on Edi Weitz's website)

The pumping game - online game to help you understand the pumping lemma.

8.5 Applications of Formal Languages and Grammars as well as Automata Theory

1. Compiler architecture uses context-free grammars
2. Parsers - recognise if commands comply with the syntax of a language
3. Pattern matching and data mining - guess the language from a given set of words (applied in CS, genetics, etc.)
4. Natural language processing - example in David Wilkins' notes pp.40-44
5. Checking proofs by computers/automatic theorem proving - simpler example of this kind in David Wilkins' notes pp.45-57 that pertains to propositional logic
6. The theory of regular expressions enables
   (a) grep/awk/sed in Unix
   (b) More efficient coding (avoiding unnecessary detours in your code)
7. Biology - John Conway's game of life is a cellular automaton
8. Modelling of AI characters in games uses the finite state automation idea. Our character can choose among different behaviours based on stimuli - like a finite state automation reacting to input
9. Strategy and tactics in games - teach the opposition to recognise certain patterns, then suddenly change them to gain an advantage and score - used in football, fencing, etc.
10. Learning a sport/a numerical instrument/a new field or subject - split the information into blocks and learn how to combine them into meaningful patterns - uses notions from context-sensitive grammars.
11. Finite state automata and probability - chaos theory, financial mathematics.

etc...

9 Graph Theory

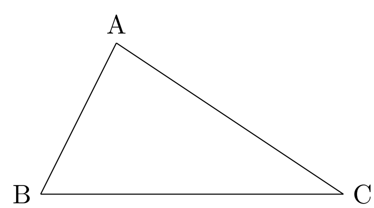
Task: Introduce terminology related to graphs; understand different types of graphs; learn how to put together arguments involving graphs.
An undirected graph consists of:

1. A finite set of points V called vertices
2. A finite set E of edges joining two distinct vertices of the graph.

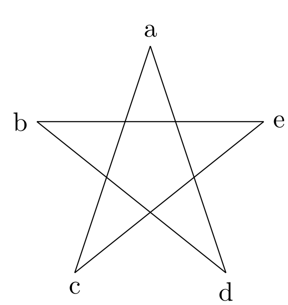
Understand the meaning of an edge better: Let V be the set of vertices. Consider $P(V)$, the power set of V. Let $V_2 \subseteq P(V)$ consist of all subsets of V containing exactly 2 points, **i.e.** $V_2 = \{A \in P(V) \mid \#(A) = 2\}$
Identify each element in V_2 with the edge joining the two points. In other words, if $\{a, b\} \in V_2$, then we can let ab be the edge corresponding to $\{1, b\}$.

Examples:

1. A triangle is an undirected graph.
   $V = \{A, B, C\}$

   3 possible 2 element subsets of V: $\{A, B\} \to AB$
   $\{A, C\} \to AC$
   $\{B, C\} \to BC$
   $E = \{AB, AC, BC\}$
2. A pentagram is an example of an undirected graph.
   $V = a, b, c, d, e$

   $E = \{ac, ad, be, ce, bd\}$

Convention: The set of vertices cannot be empty, **i.e.** $V \neq 0$.

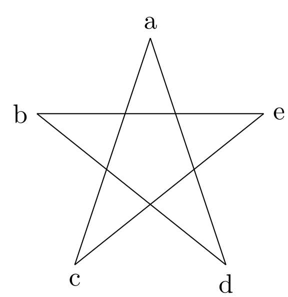
Q: If $V \neq \emptyset$, what is the simplest possible undirected graph?

A: A graph consisting of a single point, **i.e.** with one vertex and zero edges.

Definition: A graph is called trivial if it consists of one vertex and zero edges.
Next, study how vertices and edges relate to each other.

Definition: If v is a vertex of some graph, if e is an edge of that graph, and it $e = vv'$ for v' another vertex, then the vertex v is called incident to the edge e and the edge e is called incident to the vertex v.

Example:
b is incident to edges be and bd
be is incident to vertices b and e



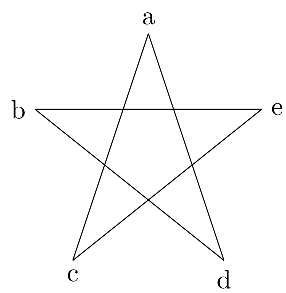
Definition: Let (V, E) be an undirected graph. Two vertices $A, B \in V \quad A \neq B$ are called adjacent if $\exists$ edge $AB \in E$.
We represent the incidence relations among the vertices V and edges E of an undirected graph via:

1. An incidence table
2. An incidence matrix

Legend:
1 an incidence relation holds
0 an incidence relation does not hold

From the pentagram:
$V = \{a, b, c, d, e\}$
$E = \{ac, ad, be, bd, ce\}$

The incidence table is:



	ac	ad	be	bd	ce
a	1	1	0	0	0
b	0	0	1	1	0
c	1	0	0	0	1
d	0	1	0	1	0
e	0	0	1	0	1

Correspondingly, the incidence matrix is:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Note that for the incidence matrix to make sense, we need to know that vertices were considered in the order $\{a, b, c, d, e\}$ and edges in the order $\{ac, ad, be, bd, ce\}$. If we shuffle either set, the incidence matrix changes.