We ~~saw~~ proved last week that we can solve the Tower of Hanoi puzzle recursively. Moving a stack of n discs took $s(n)$ steps, where

7-2

$\boxed{L11}$

$$s(1) = 1$$
$$s(k+1) = 2 \cdot s(k) + 1$$

How do we find a closed expression for $s(n)$?

- First, calculate a few steps:

$$s(1) = 1$$
$$s(2) = 2 \cdot s(1) + 1 = 2 \cdot 1 + 1 = 3$$
$$s(3) = 2 \cdot s(2) + 1 = 2 \cdot 3 + 1 = 7$$
$$s(4) = 2 \cdot s(3) + 1 = 2 \cdot 7 + 1 = 15$$

- Now make a <u>guess</u>: $s(n) \stackrel{?}{=} 2^n - 1$
- Try to prove your guess, using induction.

<u>Base Case</u> $(n=1)$: $s(1) = 1 = 2 - 1 = 2^1 - 1$ ✓

<u>Inductive Hypothesis</u>: Assume that $s(n) = 2^n - 1$, for all $1 \leq n \leq k$.

<u>Inductive Step</u>: We need to prove that $s(k+1) = 2^{k+1} - 1$.

$$\begin{aligned} s(k+1) &= 2 \cdot s(k) + 1 && \text{(Def. of } s(n)\text{)} \\ &= 2 \cdot (2^k - 1) + 1 && \text{(By induction)} \\ &= 2 \cdot 2^k - 2 + 1 \\ &= 2^{k+1} - 1 && ✓ \end{aligned}$$

Therefore $s(n) = 2^n - 1$.

Recursive functions can use multiple previous values:

$$f(0) = 0$$
$$f(1) = 1$$
$$f(k+2) = f(k+1) + f(k)$$

The~~se~~is function gives the Fibonacci ~~sequen~~ numbers:

$$f(0) = 0$$
$$f(1) = 1$$
$$f(2) = f(1) + f(0) = 1 + 0 = 1$$
$$f(3) = f(2) + f(1) = 1 + 1 = 2$$
$$f(4) = f(3) + f(2) = 2 + 1 = 3$$
$$f(5) = f(4) + f(3) = 3 + 2 = 5$$

$\vdots$ 8

13

21

$\vdots$

How quickly does f grow? We can show that it is $O(2^n)$:

Proof: We need to find c and n_0 such that $f(n) \leq c \cdot 2^n$, for all $n \geq n_0$. We will try ~~$n_0=0$~~ to prove this by induction for $n_0 = 0$.

Base Case: $f(0) = 0 \leq 1 = 2^0$, $f(1) = 1 \leq 2 = 2^1$ } Both work for $c \geq 1/2$.

Inductive Hypothesis: Assume that $f(n) \leq c \cdot 2^n$ for all $0 \leq n \leq k$, with $k \geq 1$.

Inductive Step: We need to show that
$f(k+1) \leq c \cdot 2^{k+1}$.

$$
\begin{aligned}
f(k+1) &= f(k) + f(k-1) && \text{(Def. of } f(n)\text{)} \\
&\leq c \cdot 2^k + c \cdot 2^{k-1} && (\text{IH} \wedge k \geq 1) \\
&\leq c \cdot 2^k + 2 \cdot c \cdot 2^{k-1} \\
&= c \cdot 2^k + c \cdot 2^k \\
&= 2 \cdot c \cdot 2^k \\
&= c \cdot 2^{k+1}
\end{aligned}
$$

Therefore $f(n)$ is $O(2^n)$ with $c = 1/2$ and $n_0 = 0$.

(This is not tight. There is a bonus question in the tutorial to find a tight $O(\alpha^n)$ bound.)

Graphs

A graph ~~G = (V, E)~~ represents a set ~~V~~ of things, ~~V, called nodes or vertices,~~ and a set ~~E~~ of connections between these things, ~~which are called edges~~.

Formally, a graph G ~~=~~ is a pair (V, E), where V is a set of nodes or vertices, representing the things, and E is a set of edges, representing the connections.

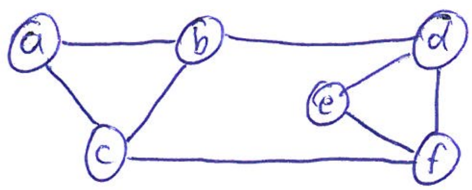
Each edge is a ~~n~~ pair of distinct vertices.
$(E \subseteq V \times V)$

Examples:

1) Road Network

   V = cities / interesting places

   E = roads connecting them



2) Facebook

   V = all people on FB

   E = ~~edge~~ all pairs of people who are FB-friends

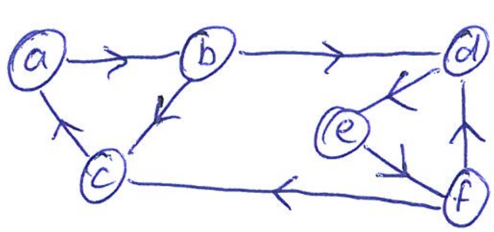
3) Internet

   V = all websites

   E = links between them.

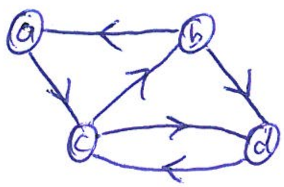
Edges can either be unordered pairs or ordered pairs. If edges are ordered ($(a,b) \neq (b,a)$), we say that the graph is <u>directed</u>.

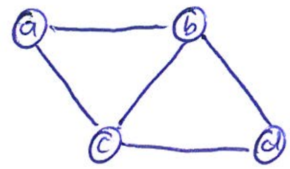
<u>Example</u>: Road network with 1-way streets



We can represent a graph in three ways:

1) As a drawing:





2) As an <u>adjacency list</u>:

a : c	a : b, c
b : a, d	b : a, c, d
c : b, d	c : a, b, d
d : c	d : b, c

3) As an <u>adjacency matrix</u>:

$$\begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

$$(a,b) = \begin{cases} 1 & \text{if } (a,b) \in E \\ 0 & \text{if } (a,b) \notin E \end{cases}$$

Drawings are generally easier for people to work with, but the other two are more useful when you want to program a computer to work with graphs.

Terminology

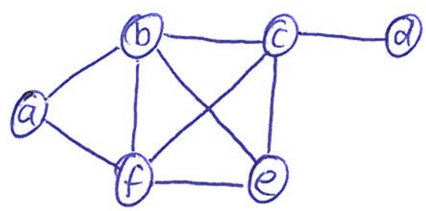
Given an undirected graph $G=(V,E)$,

- vertices u and v are <u>adjacent</u> if $\{u,v\}\in E$
- the <u>degree</u> of a vertex u is the number of adjacent vertices it has.

<u>Claim:</u> $\sum_{u\in V} \text{degree}(u) = 2\cdot|E|$

<u>Proof:</u> When adding up all the degrees, each edge is counted twice: once at each endpoint.

<u>Example:</u>



degree: a − 2, b − 4, c − 4, d − 1, e − 3, f − 4, g − 0

sum: $2+4+4+1+3+4+0=18$

edges: 9

<u>Claim:</u> Every undirected graph G has an <u>even</u> number of vertices of <u>odd degree</u>.

<u>Proof:</u>

$$2\cdot|E| = \sum_{u\in V} \text{degree}(u)$$

$$\underbrace{2\cdot|E|}_{\text{even}} = \underbrace{\sum_{u\in V_{even}} \text{degree}(u)}_{\text{even}} + \sum_{u\in V_{odd}} \text{degree}(u)$$

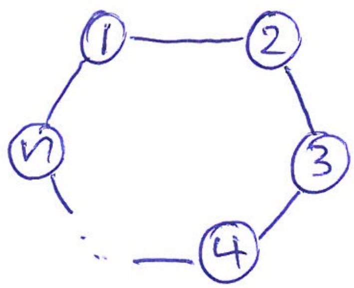
So $\sum_{u\in V_{odd}} \text{degree}(u)$ is even, but each degree itself is odd. Then $|V_{odd}|$ <u>must</u> be even.

Special Graphs

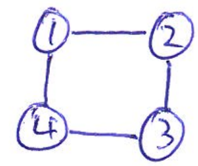
Path on n vertices (P_n):

P_4

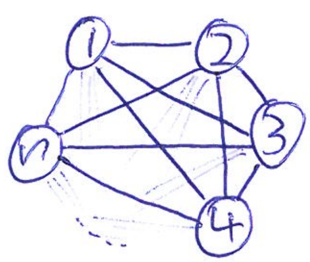
Cycle on n vertices (C_n):



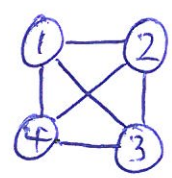
C_4



Complete graph on n vertices (K_n):



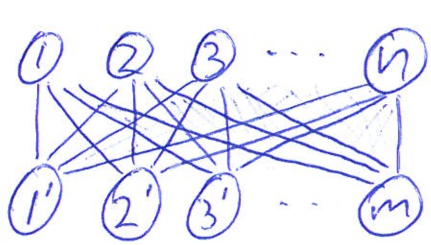
K_4



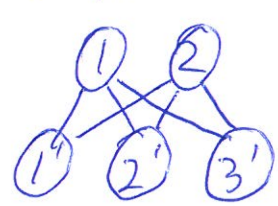
K_n has an edge between **all** pairs of vertices.

Complete Bipartite graph ($K_{n,m}$):

~~A graph is~~



$K_{2,3}$



A graph is **bipartite** if ...

End of L11