Defn: A graph is acyclic if it has no cycle.

Tree: Connected acyclic graph

Leaf: vtx of degree 1 in a tree.

Lemma: The following are equivalent:

(a) G is a tree

(b) G is minimally connected (mc)

(c) G is maximally acyclic (ma)

(d) $\forall u, v \in G \; \exists!$ u-v path.

mc : G is connected but $G - xy$ isn't, $\forall\, xy \in E(G)$.

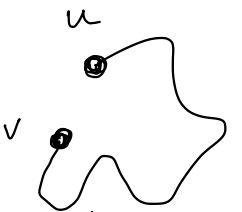
ma : $G + xy \supseteq$ cycle $\forall\, xy \notin E(G)$, $x, y \in V(G)$.

proof :

(a) $\Rightarrow$ (b). Suppose not.

$\exists\, uv \in E(G)$ st $G - uv$ is connected.

Let P be a u-v path in $G - uv$



$G - uv$

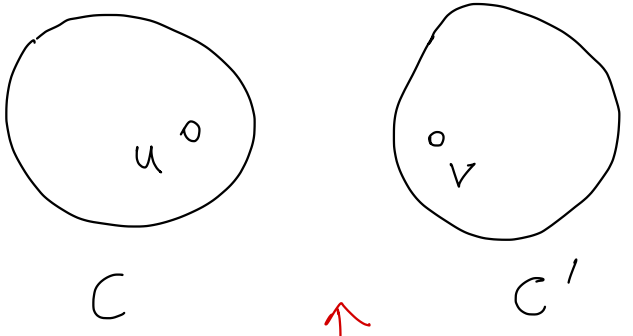
$G \supseteq uv \cup P$, which is a cycle. A contradiction.

(b) $\Rightarrow$ (c)

Suppose for contradiction that $\exists$ cycle $C \subseteq G$.

Let $u, v \in V(C)$ (and consecutive).

G minimally connected $\Rightarrow$ $G - uv$ is disconnected.



C, C' connected components of $G - uv$ st $C \cap C' = \phi$.

↑ contradiction.

Let $x, y \in V(G)$ (distinct)

G connected $\Rightarrow \exists$ x-y path P in G.

Note that $xy \notin E(G) \Rightarrow xy \cup P$ is a cycle.

(c) $\Rightarrow$ (d)

Let $x, y \in V(G)$.

Claim: $\exists \geq 1$ x-y path in G.

proof: done if $xy \in E(G)$.

If $xy \notin E(G)$, then $G \cup xy \supseteq$ cycle.

$\Rightarrow \exists$ x-y path in G ——— ▨

Now recall from L2:

Proposition 2. If $\exists$ distinct paths P and Q in G with same endpoints, then G contains a cycle.

$\Rightarrow \exists \leq 1$ x-y path in G.

(d) $\Rightarrow$ (a) G is connected & acyclic!

Oth. we'd have ≥ 2 paths between a pair of vtxs. ——— ▨

Basic properties about trees.

Defn: A tree $T \subseteq G$ is spanning if $V(G) = V(T)$.

Lemma 2: G connected $\Rightarrow$ $G \supseteq$ a spanning tree.

proof: Let $T \subseteq G$ with $V(T) = V(G)$ and minimaly connected.

By (6), T is a tree. ▨

Lemma 3: Let T be a connected graph.

$$T \text{ is a tree} \iff e(T) = v(T) - 1.$$

proof: ($\Rightarrow$) Induction on $v(T)$.

- True when $v(T) = 1$.
- Take $n \geq 1$.

I.H: Let H be a tree with $v(H) = n$. Then, $e(H) = n-1$.

Let T be a tree with $v(T) = n+1$.

Let $\ell \in V(T)$ be a leaf.

$T - \ell$ is a tree on n vtxs. Hence, $e(T-\ell) = n-1$

$$e(T) = e(T-\ell)+1 = n = (n+1)-1.$$

($\Leftarrow$) Let T be connected with $e(T) = v(T)-1$.

Let $T' \subseteq T$ spanning tree. By "($\Rightarrow$)" we have

$$e(T') = v(T')-1 = v(T)-1$$

$$\Rightarrow T' = T.$$

Algorithms

Defn: An algorithm consists of a set of valid inputs and instructions such that for each valid input the computation of the algorithm is a uniquely defined finite series of elementary steps which produces a certain output.

Core elementary steps:

- Arithmetic operations
- Comparisons
- Variable updates
- Array access
- Evaluating one iteration of a loop.
- Evaluating a if condition and taking a branch.

Input : usually a 0-1 matrix, or a binary string.

Size of an input : # coordinates, size of the string.

Defn : Let A be an algorithm which accepts input from a finite set X. For $x \in X$, let $E : X \to \mathbb{N}$ be defined as

$$E(x) = \#\text{ elementary steps to compute } A(x)$$

The running time (or time complexity) is the function $T : \mathbb{N} \to \mathbb{N}$ given by $T(n) := \max\{E(x) : x \in X \text{ st } \text{size}(x) = n\}$.

Example:

ALGORITHM SUM

Input: $v \in \{0,1\}^n$ for some $n \in \mathbb{N}$.

Output: integer s

$s \leftarrow 0$

for $i \in \{1, \dots, n\}$, do

    $s \leftarrow s + v[i]$

return s.

RUNNING TIME

$Cn + C'$

$i \leq n$: +1

$i \leftarrow i+1$: +2

+1

Variable update: +1

Reading $v[i]$ +1

Summing: +1

Function growth: Let $f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$.

① $f = O(g)$ if $\exists\ C > 0$ such that $f(n) \leq C g(n)\ \forall\ n \in \mathbb{N}$.

② $f = \Omega(g)$ ——————— $f(n) \geq C g(n)\ \forall\ n \in \mathbb{N}$.

③ $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$.

④ $f = o(g)$ if $f(n)/g(n) \to 0$ (Alternative notation: $f \ll g$)

⑤ $f = \omega(g)$ if $f(n)/g(n) \to \infty$ (——————— $f \gg g$)

Defn: Let A be an algorithm and T its running time. We say that the running time of A is $O(f(n))$ if $T(n) = O(f(n))$.

If $\exists\ k \in \mathbb{N}$ such that $T(n) = O(n^k)$, we say that is a polynomial time algorithm.

If $k = 1$, then A is a linear-time algorithm.

Decision problem: problems that have a YES or NO answer. Formally, it is a BF: $f: X \to \{0,1\}$.

Defn: A DP $f: X \to \{0,1\}$ is in the class **P** if $\exists$ an algorithm that for every $x \in X$ outputs $f(x)$ in time polynomial in $|x|$.

Defn: A DP $f: X \to \{0,1\}$ is in **NP** if $\exists$ a verifier $V \in \mathbf{P}$ and a polynomial P such that $\forall\ x \in X$:

$$f(x) = 1 \iff \exists \text{ witness } w \text{ with } |w| \leq p(|x|) \text{ and } V(x,w) = 1.$$