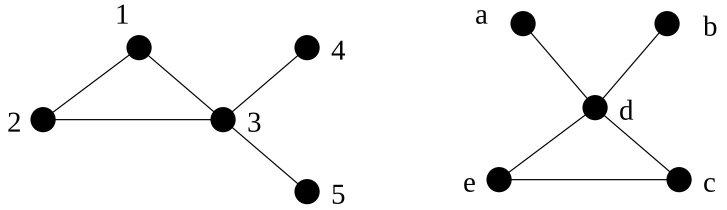
COMPUTING ISOMORPHISM

In this chapter, we will look at graph isomorphism (and automorphism), includying algorithms using invariants and certificates. We will also see isomorphism of other structures.

Graph Isomorphism

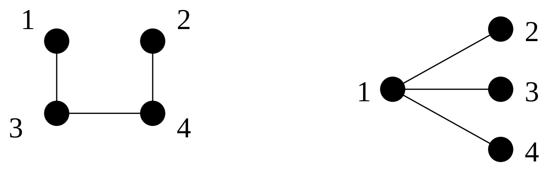
Example 1:



G_1 and G_2 are **isomorphic**, since there is a bijection $f : V_1 \to V_2$ that preserve edges:

$$\begin{array}{rcl} 1 & \to & c \\ 2 & \to & e \\ 3 & \to & d \\ 4 & \to & a \\ 5 & \to & b \end{array}$$

Example 2:



G_3 and G_4 are not isomorphic. Any bijection would not preserve edges since G_3 has no vertex of degree 3, while G_4 does (the degree sequence of a graph is invariant (in sorted order) under isomorphism).

DEFINITION. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there i a bijection $f : V_1 \to V_2$ such that

$$\{f(x), f(y)\} \in E_2 \iff \{x, y\} \in E_1.$$

The mapping f is said to be an *isomorphism* between G_1 and G_2.

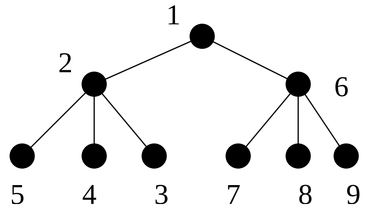
If f is a isomorphism from G to itself, it is called an *automorphism*. The set of all automosphisms of a graph is a permutation group (which is a group under the "composition o permutations" operation). See chapter 6 for more on permutation groups.

The problem of determining if two graphs are isomorphic is in general difficult, but most researchers believe it is not NP-complete.

Some special cases can be solved in polynomial time, such as: graphs with maximum degree bounded by a constant and trees.

Invariants

Let $DS = [deg(v_1), deg(v_2), \ldots, deg(v_n)]$ be the degree sequence of a graph. And let $SDS = [d_1, d_2, \ldots, d_n]$ be its degree sequence in sorted order.



$$DS = [2, 4, 1, 1, 1, 4, 1, 1, 1]$$
$$SDS = [1, 1, 1, 1, 1, 1, 2, 4, 4]$$

SDS is the same for all graphs that are isomorphic to G. So, SDS is *invariant* (under isomorphism).

DEFINITION. Let $\mathcal{F}$ be a family of graphs. An *invariant* on $\mathcal{F}$ is a function ϕ with domain $\mathcal{F}$ such that $\phi(G_1) = \phi(G_2)$ if G_1 is isomorphic to G_2.

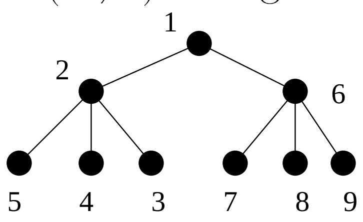
If $\phi(G_1) \neq \phi(G_2)$ we can conclude G_1 and G_2 are not isomorphic. If $\phi(G_1) = \phi(G_2)$, we still need to check whether they are isomorphic.

Definition. Let $\mathcal{F}$ be a family of graphs on the vertex set V. Let $D : \mathcal{F} \times V \to \{0, 1, \ldots, k\}$. Then, the *partition of V induced by D* is

$$B = [B[0], B[1], \ldots, B[k]]$$

where $B[i] = \{v \in V : D(G, v) = i\}$.
If $\phi_D(G) = [|B[0]|, |B[1]|, \ldots, |B[k]|]$ is an invariant, then we say that D is an invariant inducing function.

Example:
$D(G, u) =$ degree of vertex u in graph G.

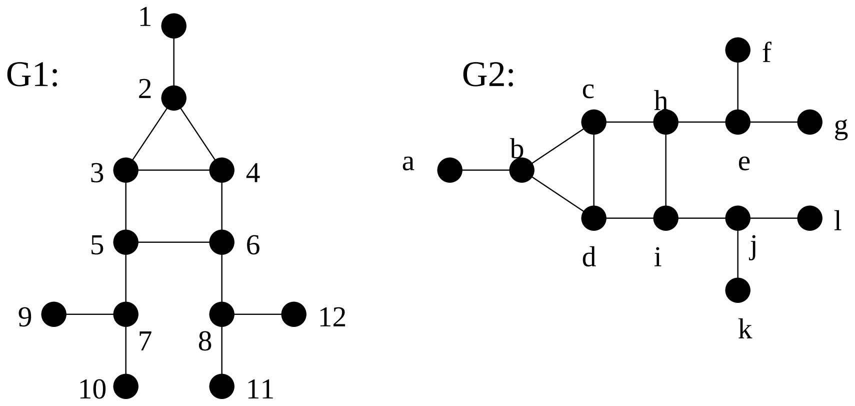


Ordered partition induced by D:

$$B = [\emptyset, \{3, 4, 5, 7, 8, 9\}, \{1\}, \emptyset, \{2, 6\}, \emptyset, \emptyset, \emptyset, \emptyset]$$

$$\phi_D(G) = [0, 6, 1, 0, 2, 0, 0, 0, 0]$$

$\phi_D(G)$ is an invariant for $\mathcal{F} =$ family of all graphs on V.
So, D is an invariant inducing function.



Initial partition:

$X_0(G_1) = \{1, 2, \ldots, 12\}$ $X_0(G_2) = \{a, b, \ldots, l\}$

1st invariant inducing function:

$D_1(G, v) = \#$ of neighbours for each degree

$$\begin{array}{r|r}
D_1(G_1,1) = [0010\cdots0] & D_1(G_2,a) = [0010\cdots0] \\
D_1(G_1,2) = [1020\cdots0] & D_1(G_2,b) = [1020\cdots0] \\
D_1(G_1,3) = [0030\cdots0] & D_1(G_2,c) = [0030\cdots0] \\
D_1(G_1,4) = [0030\cdots0] & D_1(G_2,d) = [0030\cdots0] \\
D_1(G_1,5) = [0030\cdots0] & D_1(G_2,e) = [2010\cdots0] \\
D_1(G_1,6) = [0030\cdots0] & D_1(G_2,f) = [0010\cdots0] \\
D_1(G_1,7) = [2010\cdots0] & D_1(G_2,g) = [0010\cdots0] \\
D_1(G_1,8) = [2010\cdots0] & D_1(G_2,h) = [0030\cdots0] \\
D_1(G_1,9) = [0010\cdots0] & D_1(G_2,i) = [0030\cdots0] \\
D_1(G_1,10) = [0010\cdots0] & D_1(G_2,j) = [2010\cdots0] \\
D_1(G_1,11) = [0010\cdots0] & D_1(G_2,k) = [0010\cdots0] \\
D_1(G_1,12) = [0010\cdots0] & D_1(G_2,l) = [0010\cdots0]
\end{array}$$

partition refinement of X_0 induced by D_1:

$X_1(G_1) = \{1, 9, 10, 11, 12\}, \{2\}, \{3, 4, 5, 6\}, \{7, 8\}$

$X_1(G_2) = \{a, f, g, k, l\}, \{b\}, \{c, d, h, i\}, \{e, f\}$

$X_1(G_1) = \{1, 9, 10, 11, 12\}, \{2\}, \{3, 4, 5, 6\}, \{7, 8\}$
$X_1(G_2) = \{a, f, g, k, l\}, \{b\}, \{c, d, h, i\}, \{e, f\}$

2nd invariant inducing function:
$D_2(G, v) = \#$ of triangles in G passing through v.

v	$D_2(G_1, v)$
1	0
2	1
3	1
4	1
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

v	$D_2(G_2, v)$
a	0
b	1
c	1
d	1
e	0
f	0
g	0
h	0
i	0
j	0
k	0
l	0

partition refinement of X_1 induced by D_2:

$X_2(G_1) = \{1, 9, 10, 11, 12\}, \{2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}$
$X_2(G_2) = \{a, f, g, k, l\}, \{b\}, \{c, d\}, \{h, i\}, \{e, f\}$

G_1 and G_2 are still compatible.

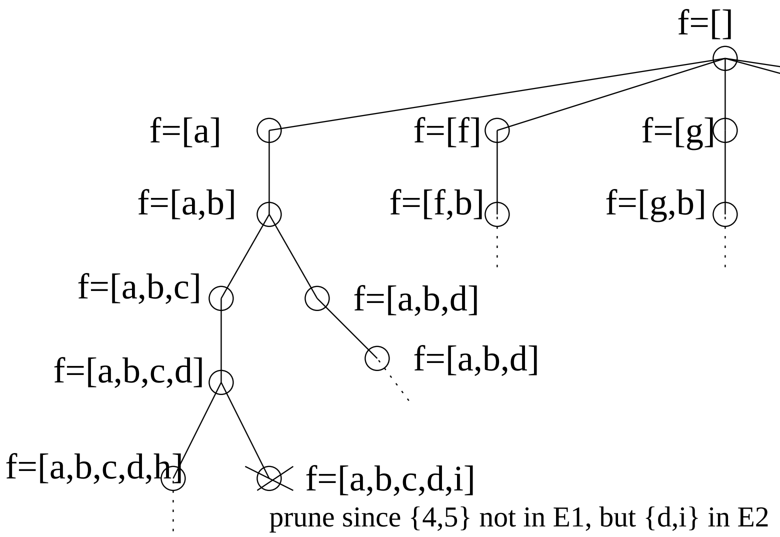
$X_2(G_1) = \{1, 9, 10, 11, 12\}, \{2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}$
$X_2(G_2) = \{a, f, g, k, l\}, \{b\}, \{c, d\}, \{h, i\}, \{e, f\}$

We only need to check bijections between the following sets:

$$\begin{array}{rcl} \{1, 9, 10, 11, 12\} & \leftrightarrow & \{a, f, g, k, l\} \\ \{2\} & \leftrightarrow & \{b\} \\ \{3, 4\} & \leftrightarrow & \{c, d\} \\ \{5, 6\} & \leftrightarrow & \{h, i\} \\ \{7, 8\} & \leftrightarrow & \{e, f\} \end{array}$$

\# of bijections to test: $5! \times 1! \times 2! \times 2! \times 2! = 40$.
Without partition refinement, we would have to test 12! bijections!

Backtracking algorithm to find all isomorphisms



Algorithm Iso($\mathcal{I}, G_1, G_2$) (global n, W, X, Y)

procedure GetPartitions()

  $X[0] \leftarrow V(G_1)$; $Y[0] \leftarrow V(G_2)$; $N \leftarrow 1$;

  for each $D \in \mathcal{I}$ do

    for $i \leftarrow 0$ to $N-1$ do

      Partition $X[i]$ into sets $X_1[i], X_2[i], \ldots, X_{m_i}[i]$,

        where $x, x' \in X_j[i] \iff D(x) = D(x')$

      Partition $Y[i]$ into sets $Y_1[i], Y_2[i], \ldots, Y_{n_i}[i]$,

        where $y, y' \in Y_j[i] \iff D(y) = D(y')$

      if $m_i \neq n_i$ then exit; (G_1 and G_2 are not isomorphic)

      Order $Y_1[i], Y_2[i], \ldots, Y_{m_i}[i]$ so that for all j

        $D(x) = D(y)$ whenever $x \in X_j[i]$ and $y \in Y_j[i]$

      if ordering is not possible then exit; (not isomorphic)

    Order the partitions so that:

      $|X[i]| = |Y[i]| \leq |X[i+1]| = |Y[i+1]|$ for all i

    $N \leftarrow N + m - 1$;

  return (N);

procedure FindIsomorphism(l)

  if $l = n$ then output (f);

  $j \leftarrow W[l]$;

  for each $y \in Y[j]$ do

    $OK \leftarrow true$;

    for $u \leftarrow 0$ to $l-1$ do

      if $(\{u, l\} \in E(G_1)$ and $\{f[u], y\} \notin E(G_2))$ or

       $(\{u, l\} \notin E(G_1)$ and $\{f[u], y\} \in E(G_2))$ then $OK \leftarrow false$;

    if OK then $f[l] \leftarrow y$; FindIsomorphism($l+1$);

main $N \leftarrow$GetPartitions();

  for $i \leftarrow 0$ to N do for eac $x \in X[i]$ do $W[x] \leftarrow i$;

  FindIsomorphism(0);

Computing Certificates

DEFINITION. A *certificate* $Cert()$ for a family $\mathcal{F}$ of graphs is a function such that for $G_1, G_2 \in \mathcal{F}$, we have

$$Cert(G_1) = Cert(G_2) \iff G_1 \text{ and } G_2 \text{ are isomorphic}$$

Certificates for Trees

We will compute certificates in polynomial time for the family of **trees**. Consequently, graph isomorphism for trees can be solved in polynomial time.
Algorithm to compute certificates for a tree:

1. Label all vertices with string 01.
2. While there are more than 2 vertices in G:
   for each non-leaft x of G do
   - 2.1. Let Y be the set of labels of the leaves adjacent to x and the label of x with initial 0 and trailing 1 deleted from x;
   - 2.2. Replace the label of x with the concatenation of the labels in Y, sorted in increasing lexicographic order, with a 0 prepended and a 1 appended.
   - 2.3. Remove all leaves adjacent to x.
3. If there is only one vertes x left, report x's label as the certificate.
4. If there are 2 vertices x and y left, concatenate x and y in increasing lexicographic order, and report it as the certificate.

Here, see slides with Examples 7.2,7.3,7.4,7.5 from the textbook.

Certificates for general graphs

Let $G = (V, E)$. Consider all permutations $\pi : V \to V$.
Each Π determines an adjacenc matrix:

$$A_\pi[u,v] \;=\; \begin{cases} 1, & \text{if } \{\pi(u), \pi(v) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Example: $G = (V = \{1, 2, 3\}, E = \{\{1, 2\}, \{1, 3\}\})$.

π :	A_π :	Num_π	π :	A_π :	Num_π
$[1,2,3]$	$\begin{array}{ccc} - & 1 & 1 \\ - & - & 0 \\ - & - & - \end{array}$	110	$[1,3,2]$	$\begin{array}{ccc} - & 1 & 1 \\ - & - & 0 \\ - & - & - \end{array}$	110
$[2,1,3]$	$\begin{array}{ccc} - & 1 & 0 \\ - & - & 1 \\ - & - & - \end{array}$	101	$[2,3,1]$	$\begin{array}{ccc} - & 0 & 1 \\ - & - & 1 \\ - & - & - \end{array}$	011
$[3,1,2]$	$\begin{array}{ccc} - & 1 & 0 \\ - & - & 1 \\ - & - & - \end{array}$	101	$[3,2,1]$	$\begin{array}{ccc} - & 0 & 1 \\ - & - & 1 \\ - & - & - \end{array}$	011

We could define the certificate to be

$$Cert(G) = min\{Num_\pi(G) : \pi \in Sym(V)\}.$$

$Cert(G)$ is difficult to compute. $Cert(G)$ has as many leading 0's as possible. So, k is as large as possible, where k is the number of all-zero columns above the diagonal. So, vertices $\{1, 2, \ldots, k\}$ form a maximum independent set in G (or equivalently a maximum clique in the complement graph $\overline{G}$).

So, computing $Cert(G)$ as defined before is NP-hard.

However, it is believed that determining if $G_1 \sim G_2$ (G_1 isomorph to G_2) is not NP-complete.

So, it is possible that the approach of computing $Cert(G)$ to solve the graph isomorphism problem is more work than necessary.
So, instead, we will define the certificate as follows:

$$Cert(G) = min\{Num_\pi(G) : \pi \in \Pi_G\},$$

where Π_G is a set of permutations determined by the structure of G but not by any particular ordering of V.

Discrete and equitable partitions

Definitions.
A partition B is a **discrete partition** if $|B[j]| = 1$ for all j, $0 \leq j \leq k$.
It is a **unit partition** if $|B| = 1$.

Let $G = (V, E)$ be a graph and $N_G(u) = \{x \in V : \{u, x\} \in E\}$.
A partition B is an **equitable partition** with respect to the graph G if for all i and j

$$|N_G(u) \cap B[j]| = |N_G(v) \cap B[j]|$$

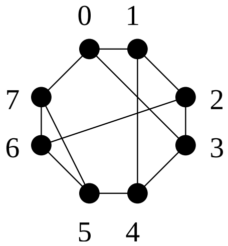
for all $u, v \in B[i]$.

Given B an ordered equitable partition with k parts, we can define M_B to be a $k \times k$ matrix where
$M_B[i, j] = |N(G(v) \cap B[j]|$ where $v \in B[i]$.

Since B is equitable any choice of v produces the same result.

Also define $Num(B) =$ sequence of $k(k-1)/2$ elements above diagonal written column by column.

Example:



$B = [\{0\}, \{2,4\}, \{5,6\}, \{7\}, \{1,3\}]$ is an equitable partition with respect to the graph above.

$$M_B = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \end{bmatrix}$$

and $Num(B) = [0,0,1,1,0,1,2,3,0,0]$.

If B is a **discrete** partition then B corresponds to a permutation $\pi : B[i] = \{\pi[i]\}$, in which case

$$Num(B) = Num_\pi(G),$$

adjusting so that $Num(B)$ is interpreted as the sequence of bits of a binary number.

Partition Refinement

Definition. An ordered partition B is a *refinement* of the ordered partition A if

1. every block $B[i]$ of B is contained in some block $A[j]$ of A; and
2. if $u \in A[i_1]$ and $v \in A[j_1]$ with $i_1 \leq j_1$, then $u \in B[i_2]$ and $v \in B[j_2]$ with $i_2 \leq j_2$.

The definition basically says that B must refine A and preserve its order.
Example:
$A = [\{0, 3\}, \{1, 2, 4, 5, 6\}]$
$B = [\{0, 3\}, \{1, 5, 6\}, \{2, 4\}]$ is a refinement of A,
but $B' = [\{1, 5, 6\}, \{2, 4\}, \{0, 3\}]$ is not a refinement of A because blocks are out of order with respect to A.
Let A be an ordered partition and T be any block of A. Define a function $D_T : V \to \{0, 1, \ldots, n-1\}$:

$$D_T(v) = |N_G(v) \cap T|$$

This function can be used to refine A.

1. Set B equal to A.
2. Let $\mathcal{S}$ be a list containing the blocks of B.
3. While $(\mathcal{S} \neq \emptyset)$ do
4.   remove a block T from the list $\mathcal{S}$
5.   for each block $B[i]$ of B do
6.   for each h, set $L[h] = \{v \in B[i] : D_T(v) = h\}$
7.   if there is more than pne non-empty block in L then
8.   replace $B[i]$ with he non-empty blcoks in L in order of the index h, $h = 0, 1, \ldots, n-1$.
9.   add the non-empty blocks in L to the end of the list $\mathcal{S}$

Notes: in step 5 we ignore blocks of $\mathcal{S}$ if the block has already been partitioned in B.
The procedure will produce an equitable partition.
The ordering at step 8 is chosen in order to make $Num(B)$ smaller.

Slide with a copy of Algortihm 7.5 page 256.
Slide with copy of example 7.8 pages 258-261.

Handwritten slides with explanation of the rest of the chapter.