Relabellings of Graphs and Rotation Systems

Isomorphisms

Automorphisms

Orbits

Canonical forms

One possible canonical form for a graph G:

For G: n vertices, data structure is an adjacency matrix.

Set lex_min_G = G.
For each of the n! permutations p of 0 to n-1 do
    Relabel the vertices of G using p to get H.
    If H < lex_min_G, set lex_min_G= H.

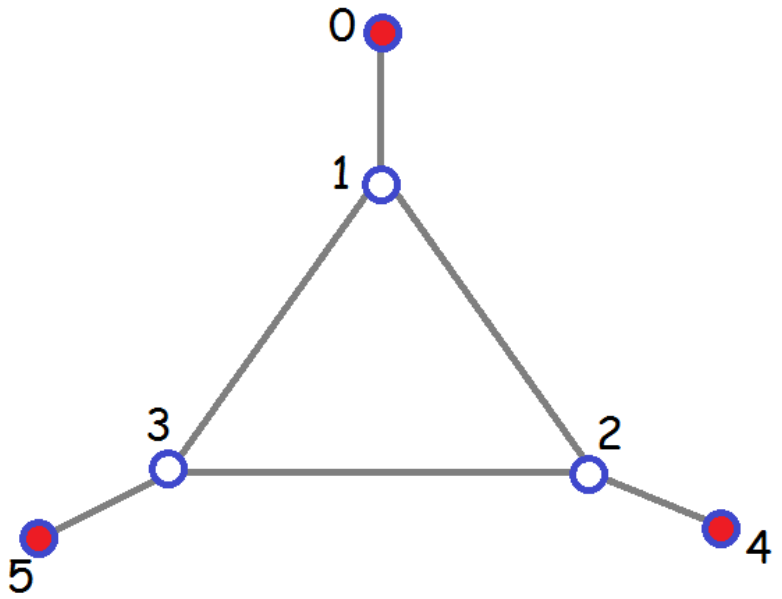
Important feature of a canonical form:

If two graphs are isomorphic they must have identical canonical forms.

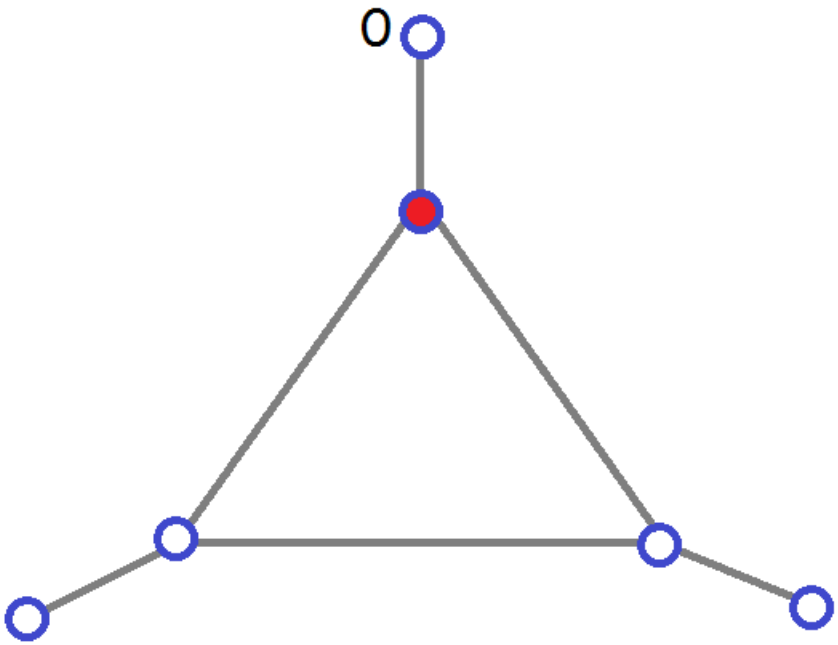
This one is not easy to compute but it satisfies the requirements.

Immediate speed up idea:
Suppose we insist that the vertices are sorted by degree in the canonical form.
Then we only have to test permutations that admit this property.

Any such graph property that does not change based on how you choose to label the vertices can be used.



We can be smarter than just looking at all permutations. For example, if vertex 0 is as labelled and we want a lex. min. adjacency matrix with vertices sorted by degree, what must the red vertex be labelled as?



One way to get all the automorphisms of lex_min_G:

Find lex_min_G.

For each of the n! permutations p of 0 to n-1 do
    Relabel the vertices of G using p to get H.
    If H = lex_min_G, p is an automorphism
    of lex_min_G.

To get all the automorphisms of G:

Let q be a permutation such that when q is applied to G it gives lex_min_G.

For each of the automorphisms p of lex_min_G,

the corresponding automorphism of G is: $q^* p * q^{-1}$

One possible canonical form for a rotation system:

For G: n vertices, data structure is a rotation system.

Set lex_min_G = G.

For each of the n! permutations p of 0 to n-1 do

    Relabel the rotation system G using p to get
    a rotation system H.
    If H < lex_min_G, set lex_min_G= H.

Flip G to get rotation system flip_G.

For each of the n! permutations p of 0 to n-1 do

    Relabel the rotation system flip_G using p to get
    a rotation system H.
    If H < lex_min_G, set lex_min_G= H.

Two important features that make a rotation system different from a graph:

1. The neighbours of each vertex are given in clockwise order.

2. The ordering is considered to be cyclic.
   We avoid considering all rotations in graphs without multiple edges by using the ordering that puts the smallest numbered neighbour first (our standard form). This makes comparing them easier.

0: 1 7 4 = 0: 7 4 1 = 0: 4 1 7

What is the standard form relabelled rotation system when the permutation

4 5 0 2 1 3

is applied to this rotation system?

0: 5

1: 2

2: 1 4 5 3

3: 2

4: 2

5: 0 2

Here we can make a massive speed up by being smarter.

Suppose we only consider relabelling permutations than can arise by applying clockwise-BFS to a graph or its flip.

Any other permutations that are automorphisms of the lex. min. subject to this condition would then also have to be a labelling that arises by applying a clockwise-BFS to the rotation system or its flip.

For a given choice of root r and first child f: choosing the BFS parent of a vertex is something that only depends on r and f, and NOT how the initial graph was labelled.

I suspect some students chose the smallest numbered vertex in the original labelling. This is not a good graph invariant as it does depend on how the graph was originally labelled.

Summary of important points:

If G and H are graphs: G is isomorphic to H if at least one of the n! relabellings p of the vertices of G gives the same adjacency matrix as H has.

For each graph G there is n! ways to label its vertices.
If one of the distinct labellings has exactly r automorphisms, then every distinct labelling as r automorphisms.

d= number of distinct labellings
r= number of automorphisms of one distinct labelling

Theorem: r * d = n!

If G and H are rotation systems: G is isomorphic to H if at least one of the n! relabellings p of the vertices of G or the flip of G gives the same rotation system as H.

For each rotation system G there is n! ways to label its vertices.If one of the distinct labellings has exactly r automorphisms, then every distinct labelling as r automorphisms.

d= number of distinct labellings
r= number of automorphisms of one distinct labelling

Theorem: r * d = n!

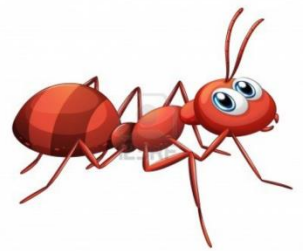
An automorphism of an object is an isomorphism from that object to itself.

The permutations that are automorphisms form a group:

1. The identity permutation is an automorphism.
2. If p is an automorphism then so is p^{-1}.
3. If p and q are automorphisms, then so is p * q.

Vertices u and v are in the same orbit if there is an automorphism that maps u to v. Each orbit is a subset of the vertices due to the properties of a group.

Vertices u and v are in the same orbit if there is an automorphism that maps u to v.



What it means to be in the same orbit:
An ant sitting at u and looking around at the unlabelled object will see exactly the same thing as an ant sitting at v.

Properties that do not depend on how vertices are labelled (e.g. the clockwise-BFS rotation systems that start at u) be the same with respect to vertices in the same orbit.

Ant: http://www.1decision1day.com/2013/05/21/what-i-learned-from-an-ant/

Let S and T be subsets of the vertices.
Examples: independent set, dominating set.

Subsets S and T are isomorphic for G if there is some automorphism of G that maps the vertices in S to the vertices in T.

g= group order for G.
d= number of ways to have a distinct subset of vertices so that it is isomorphic to S.
r= number of automorphisms mapping S to S.
Theorem: r * d = g

When considering labellings of G:
The set of permutations that are allowed labellings form a group called S_n which is just the group with all n! permutations. [GO= n!]
When considering subsets of the vertices, the set of permutations that are allowed are the g automorphisms of G. [GO= g]

d= number of distinct labelled objects you get by relabelling with an allowed labelling
r= number of allowed labellings mapping one distinct object to itself.
Both cases: r * d = GO

Both cases: r * d = GO

This is because each distinct labelled object has the same number of automorphisms mapping it to itself as each other labelled object.