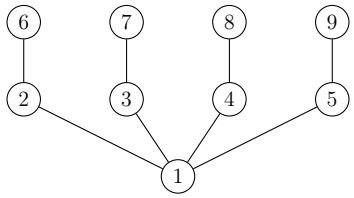
392013 Exercises Algorithmic Cheminformatics

Exercise 04: Stabilizers in McKay-Type Search Trees. May 4, 2026

How to use this sheet. The questions below are the exercises. The rest of the sheet is written as guided worked material: it introduces the needed concepts, applies them to one running graph, and then compares the hand calculation with the JSON trace produced by `graph-canon`. While reading, keep returning to the questions and fill in the missing arguments.

Exercises

We use the graph G shown here, the target-cell rule "choose the first non-trivial cell", depth-first search, and no implicit automorphisms.



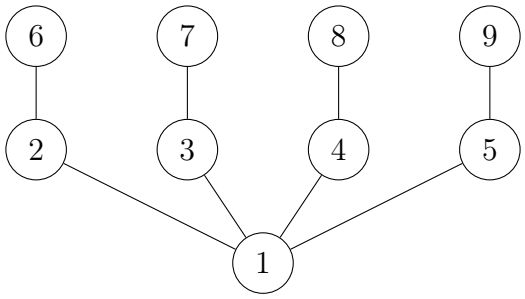
Trace ids such as `ID 0` are `graph-canon` node ids; they are introduced in the worked trace below.

1. Determine the root partition $\pi_{()}$ and the first target cell W_0.
2. Determine the automorphism group $\mathrm{Aut}(G)$. Show that the four root children (6), (7), (8), and (9) lie in one orbit under this group.
3. Reconstruct the explored search tree created by the method and compare it with the trace ids below. For each created node, give
   - the node id,
   - the individualization sequence τ,
   - the target cell index,
   - and the ordered partition π_τ.
4. For the root, the node after choosing 6, and the node after choosing $6, 9$ (called `ID 0`, `ID 1`, and `ID 2` in the trace), determine the relevant groups
   $$\mathrm{Aut}(G), \qquad \mathrm{Stab}(6), \qquad \mathrm{Stab}(6,9),$$
   and compute the orbit decomposition of the current target cell in each case.
5. The three explicit automorphisms discovered in the trace are
   $$\alpha = (3\ 4)(7\ 8), \qquad \beta = (3\ 4\ 5)(7\ 8\ 9), \qquad \gamma = (2\ 3\ 5)(6\ 7\ 9).$$
   State at which leaf each of these automorphisms is found, and at which node it is then used for pruning.
6. Explain why the root has only two created children, even though the first target cell has four elements. In particular, explain why the root child corresponding to 8 is never individualized.
7. Reproduce the JSON trace with `graph-canon`. Check that the node ids, target-cell indices, partitions, and discovered automorphisms match the trace discussed in this sheet.

The rest of the sheet develops the definitions and calculations needed to answer these questions.

Example Graph

For reference, here is the same graph G again. This time we also record the root partition that starts the search.



The graph has four isomorphic branches hanging from the central vertex 1. The equitable degree partition at the root is

$$\pi_0 = [\,6\,7\,8\,9 \mid 2\,3\,4\,5 \mid 1\,].$$

If we always choose the first non-trivial cell as target cell, then the first target cell is

$$W_0 = \{6, 7, 8, 9\}.$$

Concepts Needed for Stabilizer Pruning

We assume that equitable partitions and refinement are already known. What we need additionally is the group-theoretic language that explains why certain children in the search tree are equivalent.

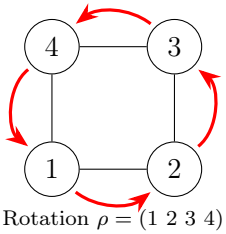
1. Automorphism and Automorphism Group

An automorphism of a graph G is a bijection

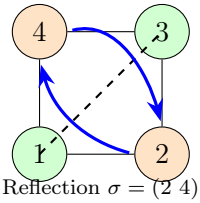
$$\gamma \colon V(G) \to V(G)$$

that preserves adjacency and non-adjacency. The set of all automorphisms forms a group under composition, called the automorphism group and denoted by $\mathrm{Aut}(G)$.

Example. Let $C_4 = (1 - 2 - 3 - 4 - 1)$. Two automorphisms are shown below: the rotation $\rho = (1\ 2\ 3\ 4)$ and the reflection $\sigma = (2\ 4)$. Both preserve all edges of the square, hence both belong to $\mathrm{Aut}(C_4)$.



Rotation $\rho = (1\ 2\ 3\ 4)$



Reflection $\sigma = (2\ 4)$

Main running example. For the graph G used in the rest of this sheet, the automorphism group can be described very explicitly. The central vertex 1 is the only vertex of degree 4, so every automorphism must fix 1. The graph then consists of the four isomorphic branches

$$\{2, 6\}, \qquad \{3, 7\}, \qquad \{4, 8\}, \qquad \{5, 9\}$$

attached to this centre. Hence an automorphism may permute these four branches arbitrarily, and conversely every permutation of the four branches gives an automorphism of G. More explicitly, for each permutation π of the branch labels $\{2,3,4,5\}$, there is an automorphism γ_π defined by

$$\gamma_\pi(1) = 1, \qquad \gamma_\pi(i) = \pi(i), \qquad \gamma_\pi(i+4) = \pi(i) + 4 \quad \text{for } i \in \{2,3,4,5\}.$$

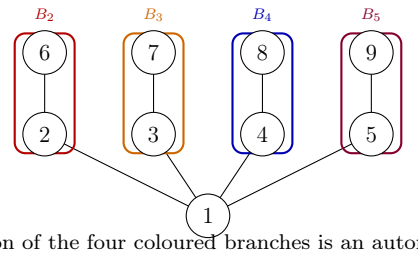
Thus

$$\mathrm{Aut}(G) = \{\gamma_\pi : \pi \text{ is a permutation of } \{2,3,4,5\}\}.$$

For example, swapping neighbouring branches gives automorphisms such as

$$(2\ 3)(6\ 7), \qquad (3\ 4)(7\ 8), \qquad (4\ 5)(8\ 9).$$

So $\mathrm{Aut}(G)$ contains exactly the $4! = 24$ permutations of the four branches.



Any permutation of the four coloured branches is an automorphism of G.

2. Group Action on the Vertex Set

The group $\mathrm{Aut}(G)$ acts on the vertex set $V(G)$ by evaluation. This means that we have a map

$$\mathrm{Aut}(G) \times V(G) \to V(G), \qquad (\gamma, v) \longmapsto \gamma(v).$$

So the input is a pair: an automorphism γ and a vertex v. The output is the image of v under γ. This action tells us which vertices are considered symmetric by the current group.

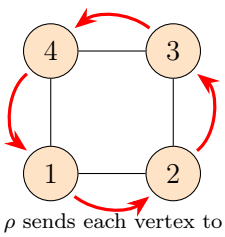
Example. For C_4, the rotation $\rho = (1\ 2\ 3\ 4)$ sends

$$(\rho, 1) \mapsto 2, \qquad (\rho, 2) \mapsto 3, \qquad (\rho, 3) \mapsto 4, \qquad (\rho, 4) \mapsto 1.$$

Equivalently, one may simply write

$$1 \mapsto 2, \qquad 2 \mapsto 3, \qquad 3 \mapsto 4, \qquad 4 \mapsto 1.$$

So the group action describes how a chosen symmetry moves individual vertices.



The action of ρ sends each vertex to the next one.

3. Subgroup

A subgroup of a group Γ is a subset $H \subseteq \Gamma$ that is itself a group under the same composition law. In particular, H must contain the identity and be closed under composition and inversion.

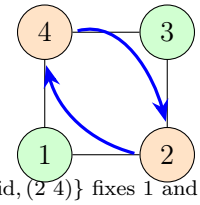
Example. For C_4, the set

$$H = \{\mathrm{id}, (2\ 4)\}$$

is a subgroup of $\mathrm{Aut}(C_4)$. It contains the identity, and

$$(2\ 4)^2 = \mathrm{id},$$

so it is closed under composition and inversion.



The subgroup $H = \{\mathrm{id}, (2\ 4)\}$ fixes 1 and 3 and swaps 2 with 4.

4. Orbit

Let a group Γ act on a set X. The orbit of an element $x \in X$ is

$$\Gamma \cdot x = \{\gamma(x) : \gamma \in \Gamma\}.$$

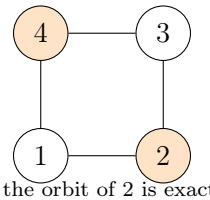
Thus the orbit consists exactly of the elements reachable from x by the group action. If two vertices lie in the same orbit, then the current symmetry group regards them as interchangeable.

Example. For the subgroup $H = \{\mathrm{id}, (2\ 4)\}$ acting on $V(C_4)$, we have

$$H \cdot 2 = \{2, 4\}, \qquad H \cdot 1 = \{1\}, \qquad H \cdot 3 = \{3\}.$$

Under the full automorphism group $\mathrm{Aut}(C_4)$, however, every vertex can be mapped to every other one, so

$$\mathrm{Aut}(C_4) \cdot 1 = \{1, 2, 3, 4\}.$$



Under H, the orbit of 2 is exactly $\{2, 4\}$.

5. Stabilizer

Let Γ act on a set X, and let $S \subseteq X$. The stabilizer of S *in the group* Γ is the subgroup of all elements $\gamma \in \Gamma$ that fix every element of S:

$$\mathrm{Stab}_\Gamma(S) = \{\gamma \in \Gamma : \gamma(s) = s \text{ for all } s \in S\}.$$

So the letter γ is crucial: the stabilizer is not just a property of the set S, but a subset of the group Γ that we are working in. In this note, when that surrounding group is clear from the context, we sometimes write $\mathrm{Stab}(S)$ instead of $\mathrm{Stab}_\Gamma(S)$. For a singleton set $\{v\}$, it is also common to write $\mathrm{Stab}(v)$ as a short form of $\mathrm{Stab}(\{v\})$.

In search trees, the already individualized vertices form an ordered sequence

$$\tau = (v_1, \ldots, v_k).$$

The relevant subgroup is then the pointwise stabilizer of that sequence:

$$\mathrm{Stab}(\tau) = \{\gamma \in \mathrm{Aut}(G) : \gamma(v_i) = v_i \text{ for all } i\}.$$

So after we have committed to some branch, we may only use automorphisms that keep all previous choices fixed.

Example. For C_4, the stabilizer of the singleton set $\{1\}$ in the group $\mathrm{Aut}(C_4)$ is

$$\mathrm{Stab}_{\mathrm{Aut}(C_4)}(\{1\}) = \{\mathrm{id}, (2\ 4)\}.$$

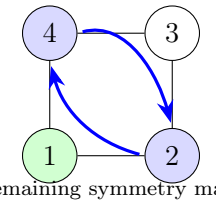
In this document, we may abbreviate this as

$$\mathrm{Stab}_{\mathrm{Aut}(C_4)}(1) = \{\mathrm{id}, (2\ 4)\}.$$

With the same pointwise definition, we also have

$$\mathrm{Stab}_{\mathrm{Aut}(C_4)}(\{1,3\}) = \{\mathrm{id}, (2\ 4)\}.$$

So after fixing 1, or after fixing both 1 and 3, the vertices 2 and 4 are still interchangeable.



After fixing 1, the remaining symmetry may still swap 2 and 4.

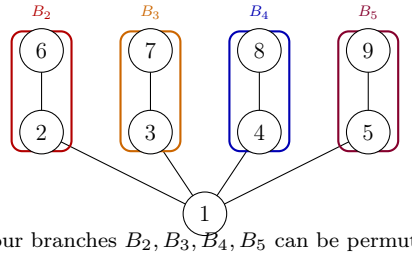
A less trivial example. Now use our main graph G and let

$$\Gamma = \mathrm{Aut}(G),$$

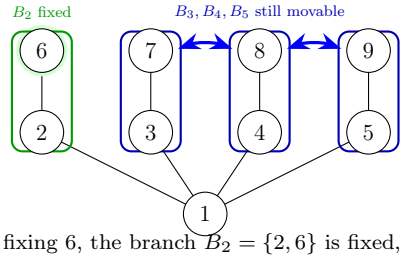
that is, Γ is the automorphism group of G. Then Γ permutes the four branches

$$\{2,6\}, \qquad \{3,7\}, \qquad \{4,8\}, \qquad \{5,9\}.$$

So Γ may rearrange these four branches in all possible ways. The following two pictures show the full branch decomposition of G and the effect of fixing the vertex 6.



The four branches B_2, B_3, B_4, B_5 can be permuted by $\Gamma = \mathrm{Aut}(G)$.



After fixing 6, the branch $B_2 = \{2,6\}$ is fixed, while B_3, B_4, B_5 may still be permuted.

The stabilizer of the vertex 6 in the group Γ is

$$\mathrm{Stab}_\Gamma(6) = \{\gamma \in \Gamma : \gamma(6) = 6\}.$$

This means that we keep only those automorphisms that fix the vertex 6. Equivalently, the branch $\{2, 6\}$ is fixed, while the three other branches may still be permuted among themselves. So this stabilizer is neither the whole group Γ nor the trivial group. For instance,

$$\beta = (3\ 4)(7\ 8) \in \mathrm{Stab}_\Gamma(6), \qquad \gamma = (4\ 5)(8\ 9) \in \mathrm{Stab}_\Gamma(6),$$

but

$$\alpha = (2\ 3)(6\ 7) \notin \mathrm{Stab}_\Gamma(6),$$

because α moves 6 to 7. Thus, after fixing 6, the remaining three branches may still be permuted among themselves.

This shows a genuine proper stabilizer: some nontrivial symmetries remain, but not all of Γ.

6. Orbits of the Target Cell Under a Stabilizer

The combination of a stabilizer and its orbits on the current target cell is the exact pruning mechanism used in McKay-type search trees. At a node with current individualization sequence

$$\tau = (v_1, \ldots, v_k),$$

we want to compare the current children of the search tree and decide which of them are equivalent, so that redundant branches can be pruned. For this purpose, we do not use all of $\mathrm{Aut}(G)$ any more. Instead, we use the pointwise stabilizer

$$\mathrm{Stab}(\tau) = \{\gamma \in \mathrm{Aut}(G) : \gamma(v_i) = v_i \text{ for all } i\},$$

because these are exactly the automorphisms that respect the choices already made on the current branch. We then look at the action of $\mathrm{Stab}(\tau)$ on the current target cell W and split W into stabilizer orbits. Two vertices in W lead to equivalent children precisely when they lie in the same orbit under this action. Therefore, it is enough to explore one representative from each orbit.

Example. Still in C_4, suppose the current branch has already chosen the vertex 1, and the current target cell is

$$W = \{2, 4\}.$$

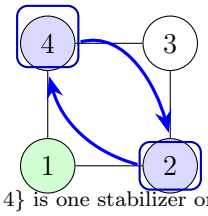
Then

$$\mathrm{Stab}(\{1\}) = \{\mathrm{id}, (2\ 4)\}$$

acts on W with one orbit:

$$\{2, 4\}.$$

So choosing 2 and choosing 4 lead to equivalent children, because some automorphism in the allowed group $\mathrm{Stab}(\{1\})$ maps one choice to the other while keeping the earlier choice fixed. Hence one of them may be pruned.



The target cell $W = \{2, 4\}$ is one stabilizer orbit, so one child suffices.

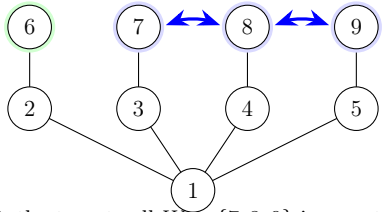
Main running example. Now return to our graph G. Suppose the current branch has already chosen the vertex 6, so the current individualization sequence is $\tau = (6)$, and suppose the current target cell is

$$W = \{7, 8, 9\}.$$

Then Stab(6) acts on W with one orbit:

$$\{7, 8, 9\}.$$

So choosing 7, choosing 8, and choosing 9 lead to equivalent children, because some automorphism in the allowed group Stab(6) maps each of these choices to the others while fixing 6. Hence, at this node, it is enough to explore one representative child from this target cell.



After fixing 6, the target cell $W = \{7, 8, 9\}$ is one stabilizer orbit.

Symmetry in the Running Example

We now translate the group-theoretic notions from the previous section into nodes of the search tree for G. Conceptually, the most important situations are the root, the node (6), and the node (6, 7). The actual `graph-canon` trace later uses (6, 9) instead of (6, 7) at the third level; the stabilizer argument is the same.

1. The Root

At the root, the equitable partition is

$$\pi_{()} = [\,6\,7\,8\,9 \mid 2\,3\,4\,5 \mid 1\,],$$

and the current target cell is

$$W_0 = \{6, 7, 8, 9\}.$$

Here the relevant symmetry group is the full automorphism group $\mathrm{Aut}(G)$, because at the root there are no earlier choices that would have to be fixed. Equivalently, if we write the root sequence as the empty sequence $\tau = ()$, then the relevant stabilizer is

$$\mathrm{Stab}(()) = \mathrm{Aut}(G).$$

The four root children (6), (7), (8), and (9) all lie in one orbit under this group. For example,

$$(2\ 3)(6\ 7)\colon (6) \mapsto (7), \qquad (2\ 4)(6\ 8)\colon (6) \mapsto (8), \qquad (2\ 5)(6\ 9)\colon (6) \mapsto (9).$$

So, at the root level, it is enough to keep one representative child, namely (6), and to explore that subtree first. So yes: the argument here is the same stabilizer-orbit argument as everywhere else, with Stab(()) at the root.

2. After Choosing 6

After individualizing 6 and refining, the equitable partition becomes

$$\pi_{(6)} = [\,6 \mid 7\,8\,9 \mid 3\,4\,5 \mid 2 \mid 1\,],$$

and the new target cell is

$$W_1 = \{7, 8, 9\}.$$

Now the relevant symmetry group is Stab(6), because all later automorphisms must keep the earlier choice 6 fixed. Every element of this subgroup keeps the branch $\{2, 6\}$ fixed and may still permute the three remaining branches

$$\{3, 7\}, \qquad \{4, 8\}, \qquad \{5, 9\}.$$

To show that the three children $(6, 7)$, $(6, 8)$, and $(6, 9)$ lie in one orbit, it is enough to exhibit automorphisms in Stab(6) that send one representative child, say $(6, 7)$, to the other two. For example,

$$(3\ 4)(7\ 8) \in \mathrm{Stab}(6), \qquad (3\ 5)(7\ 9) \in \mathrm{Stab}(6),$$

and these send

$$(6, 7) \mapsto (6, 8), \qquad (6, 7) \mapsto (6, 9).$$

These are not meant to be an exhaustive list of all automorphisms in Stab(6); they are witness automorphisms showing that all three children belong to one orbit. So the target cell $\{7, 8, 9\}$ is one orbit under Stab(6).

3. After Choosing 6 and 7

After individualizing 7 as well, the equitable partition becomes

$$\pi_{(6,7)} = [\,6 \mid 7 \mid 8\,9 \mid 4\,5 \mid 3 \mid 2 \mid 1\,],$$

and the current target cell is

$$W_2 = \{8, 9\}.$$

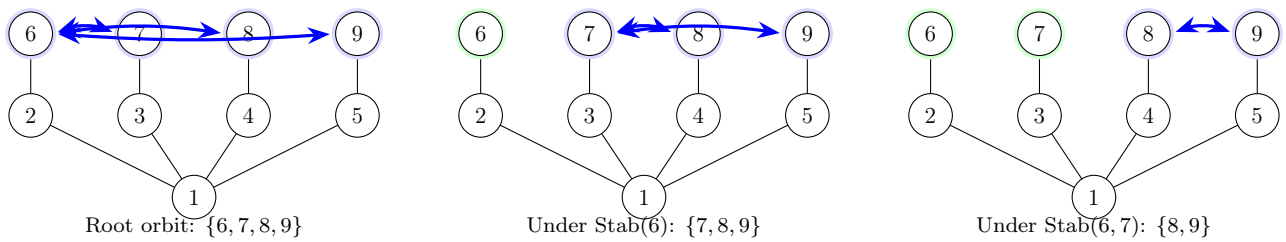
Now we may only use automorphisms in $\mathrm{Stab}(6, 7)$. The last remaining nontrivial symmetry is

$$(4\ 5)(8\ 9) \in \mathrm{Stab}(6, 7),$$

and it sends

$$(6, 7, 8) \mapsto (6, 7, 9).$$

So the target cell $\{8, 9\}$ is one orbit under $\mathrm{Stab}(6, 7)$. In the small picture below we highlight only the action on the current target cell $\{8, 9\}$, not the full action of this automorphism on all vertices.



The same graph, but with a smaller usable symmetry group at each deeper node of the tree.

How the Actual Method Trace Differs

The actual GraphCanon trace is very close in structure to the conceptual discussion from the previous section, but it is not identical in notation.

1. The same symmetry information is used

The graph is still the same graph G, so the global automorphism group is still

$$\mathrm{Aut}(G) = \langle (2\ 3)(6\ 7),\ (3\ 4)(7\ 8),\ (4\ 5)(8\ 9) \rangle.$$

The conceptual discussion used the nodes (), (6), and (6, 7). Their relevant groups are

$$\mathrm{Aut}(G), \qquad \mathrm{Stab}(6) = \langle (3\ 4)(7\ 8),\ (4\ 5)(8\ 9) \rangle,$$

$$\mathrm{Stab}(6,7) = \langle (4\ 5)(8\ 9) \rangle.$$

In the actual method trace, the first two levels are still the same in spirit: the root uses $\mathrm{Aut}(G)$ and the next level uses $\mathrm{Stab}(6)$. However, the DFS branch chosen by the method is $(6, 9)$ rather than $(6, 7)$, so the third relevant stabilizer becomes

$$\mathrm{Stab}(6,9) = \langle (3\ 4)(7\ 8) \rangle.$$

This is completely analogous to the conceptual $\mathrm{Stab}(6, 7)$ case: in both cases, once two branches have been fixed, only one swap of the two remaining symmetric branches survives.

The three explicit automorphisms that are actually discovered in the trace are

$$\alpha = (3\ 4)(7\ 8), \qquad \beta = (3\ 4\ 5)(7\ 8\ 9), \qquad \gamma = (2\ 3\ 5)(6\ 7\ 9).$$

They are used at three different heights of the tree:

$$\alpha \in \mathrm{Stab}(6,9), \qquad \beta \in \mathrm{Stab}(6), \qquad \gamma \in \mathrm{Aut}(G).$$

So the pruning logic is the same as before: a deeper leaf reveals an automorphism, and that automorphism is then applied at the highest node whose earlier choices it still fixes.

2. The main presentational difference

The main difference is the ordering convention inside the method's ordered partitions.

- In the conceptual discussion, we wrote partitions in the more natural pedagogical style

  $$[\,6\,7\,8\,9 \mid 2\,3\,4\,5 \mid 1\,], \qquad [\,6 \mid 7\,8\,9 \mid 3\,4\,5 \mid 2 \mid 1\,],$$

  and so on.

- In the actual method trace, the internal cell order is implementation dependent, so already at the root we see

  $$(\,6\,7\,8\,9 \mid 5\,4\,3\,2 \mid 1\,)$$

  instead of

  $$[\,6\,7\,8\,9 \mid 2\,3\,4\,5 \mid 1\,].$$

- More importantly, when the method individualizes the first element of the current target cell, it places the new singleton *after the remaining part of that cell*. Thus

$$(6\,7\,8\,9 \mid 5\,4\,3\,2 \mid 1) \longrightarrow (9\,7\,8 \mid 6 \mid 5\,4\,3 \mid 2 \mid 1),$$

  and then

$$(9\,7\,8 \mid 6 \mid 5\,4\,3 \mid 2 \mid 1) \longrightarrow (8\,7 \mid 9 \mid 6 \mid 3\,4 \mid 5 \mid 2 \mid 1).$$

  So the individualized element is not shown at the front, but as a singleton directly after the remainder of the old target cell.

This is why the actual DFS trace continues with $(6, 9)$ instead of $(6, 7)$: after the first individualization, the first block of the ordered partition has underlying set $\{7, 8, 9\}$ but is stored internally in the order $(9\,7\,8)$, and the method chooses the first element of that ordered block. So the notation is different, but the underlying symmetry argument is the same one discussed before.

Step-by-Step Exploration and Pruning

This section follows the actual JSON trace produced by `graph-canon` with DFS, target-cell rule `f`, and without implicit automorphisms. To stay close to the implementation, we translate the internal 0-based labels to the 1-based labels of this sheet, but we keep the ordered partitions exactly as they appear in the trace. In particular, the order inside a cell is the method's internal order, not the geometric left-to-right order from the picture of G. Each step includes a small local search tree in the same notation as the complete tree shown later in *Explored Search Tree from the JSON Trace*. To connect the trace to the conceptual discussion, each step also states the *current relevant stabilizer*. This is the subgroup whose automorphisms are allowed to compare the current children and justify pruning at that node.

1. **Expand the root.** At the root, the equitable partition is

$$\pi_{()} = (6\,7\,8\,9 \mid 5\,4\,3\,2 \mid 1).$$

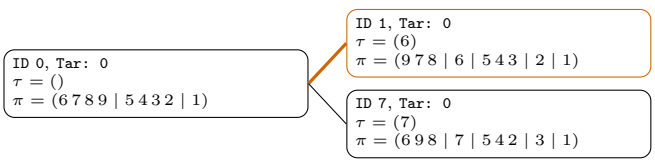
   In the JSON trace this is node `ID 0`, and the target cell index is `Tar: 0`. Its first cell is

$$W_0 = \{6, 7, 8, 9\}.$$

   At the root, the relevant stabilizer is

$$\mathrm{Stab}(()) = \mathrm{Aut}(G),$$

   because there are no earlier choices to preserve yet. Conceptually, the four possible root children all lie in one orbit under this stabilizer. However, at this point the DFS run has not *found* any explicit automorphism yet, so the method still has to explore a first representative branch in order to discover such symmetry information. The first child created by the method is `ID 1`, which corresponds to individualizing the first element of this ordered cell, namely 6. No automorphism has been discovered yet.



2. **Expand the node** (6)**.** After individualizing 6 and refining, the trace creates node `ID 1` with equitable partition

$$\pi_{(6)} = (\,9\,7\,8 \mid 6 \mid 5\,4\,3 \mid 2 \mid 1\,).$$

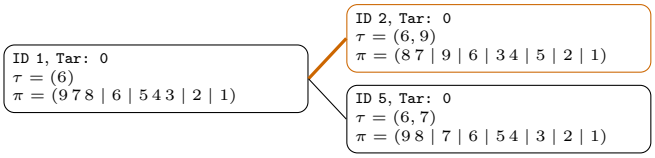
Again the target cell index is `Tar: 0`, so the current target cell is the first block of the ordered partition. Its underlying set is

$$W_1 = \{7, 8, 9\},$$

but the block itself is written in the trace as (9 7 8). The method therefore creates `ID 2` by choosing the first element of this ordered block, namely 9. The sibling `ID 5` will only be created later, after the subtree below `ID 2` has been explored. The relevant stabilizer at this node is

$$\mathrm{Stab}(6).$$

Conceptually, this stabilizer acts on the target set $W_1 = \{7, 8, 9\}$ with one orbit, so $(6, 7)$, $(6, 8)$, and $(6, 9)$ are equivalent children. But again, the implementation does not know this *yet*: it first has to reach leaves and compare them in order to discover automorphisms that witness this orbit structure.



3. **Expand the node** $(6, 9)$**.** The next created node is `ID 2`, with

$$\pi_{(6,9)} = (\,8\,7 \mid 9 \mid 6 \mid 3\,4 \mid 5 \mid 2 \mid 1\,).$$

Its target cell is again the first block of the ordered partition. Its underlying set is

$$W_2 = \{7, 8\},$$

while the block is written as (8 7) in the trace. The method first creates the leaf `ID 3`, corresponding to $\tau = (6, 9, 8)$. This is the first leaf in the run, so it becomes the current reference leaf. The sibling leaf `ID 4`, corresponding to $\tau = (6, 9, 7)$, is then compared against it and yields the automorphism

$$\alpha = (3\ 4)(7\ 8).$$

The two leaf partitions are

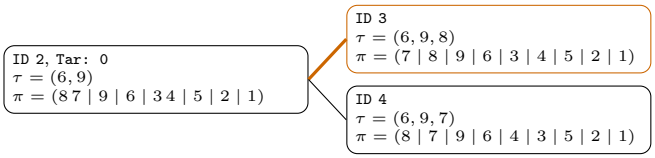
$$\pi_{(6,9,8)} = (\,7 \mid 8 \mid 9 \mid 6 \mid 3 \mid 4 \mid 5 \mid 2 \mid 1\,),$$

$$\pi_{(6,9,7)} = (\,8 \mid 7 \mid 9 \mid 6 \mid 4 \mid 3 \mid 5 \mid 2 \mid 1\,).$$

So `ID 4` is the moment where α is *found*. Here the current relevant stabilizer is

$$\mathrm{Stab}(6, 9),$$

and indeed α lies in this stabilizer: it fixes both earlier choices 6 and 9 and only swaps the two remaining options in the current target cell $W_2 = \{7, 8\}$. Therefore α can already be used at the parent node `ID 2` to mark the branch through `ID 4` as redundant. Since `ID 4` is already a leaf, this coincides here with pruning the node `ID 4` itself.



4. **Backtrack to `ID 1` and explore the sibling `ID 5`.** After finishing the subtree below `ID 2`, the method returns to `ID 1` and creates the sibling `ID 5`, corresponding to $\tau = (6, 7)$. Its partition is

$$\pi_{(6,7)} = (\,9\,8 \mid 7 \mid 6 \mid 5\,4 \mid 3 \mid 2 \mid 1\,).$$

The method then continues to the leaf `ID 6`, corresponding to $\tau = (6, 7, 9)$, with leaf partition

$$\pi_{(6,7,9)} = (\,8 \mid 9 \mid 7 \mid 6 \mid 4 \mid 5 \mid 3 \mid 2 \mid 1\,).$$

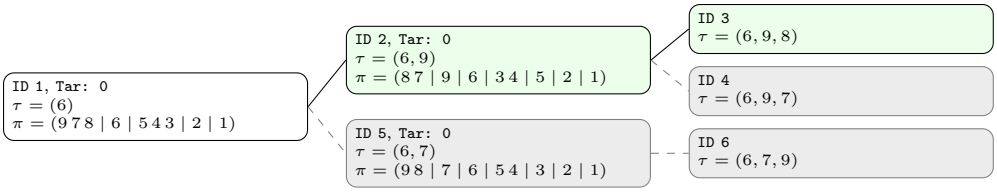
Comparing this leaf with the reference leaf `ID 3` produces the automorphism

$$\beta = (3\ 4\ 5)(7\ 8\ 9).$$

This automorphism does *not* fix the longer sequence $(6, 9)$, so it is not usable at the lower node `ID 2`. However, it *does* fix the shorter earlier choice 6, so

$$\beta \in \mathrm{Stab}(6).$$

Hence the highest node where it becomes usable is `ID 1`. There it shows that the sibling child `ID 5` belongs to the same Stab(6)-orbit as the already explored child `ID 2`. Therefore the whole subtree rooted at `ID 5` is pruned. In the actual event order of the JSON trace, the leaf `ID 6` is created first, and then both `ID 5` and `ID 6` are marked as pruned. So `ID 6` is the moment where β is *found*, and `ID 1` is the higher node where it is then *used*.



5. **Backtrack to the root and explore the sibling `ID 7`.** The method now returns to the root and creates the sibling `ID 7`, corresponding to $\tau = (7)$, with partition

$$\pi_{(7)} = (\,6\,9\,8 \mid 7 \mid 5\,4\,2 \mid 3 \mid 1\,).$$

It then creates `ID 8` and the leaf `ID 9`, corresponding to $\tau = (7, 6, 8)$, with leaf partition

$$\pi_{(7,6,8)} = (\,9 \mid 8 \mid 6 \mid 7 \mid 5 \mid 4 \mid 2 \mid 3 \mid 1\,).$$

Comparing `ID 9` with the reference leaf `ID 3` produces the automorphism

$$\gamma = (2\ 3\ 5)(6\ 7\ 9).$$

This automorphism does not fix 6, so it is not in Stab(6) and cannot be used at node `ID 1`. But at the root the relevant stabilizer is again

$$\mathrm{Stab}(()) = \mathrm{Aut}(G),$$

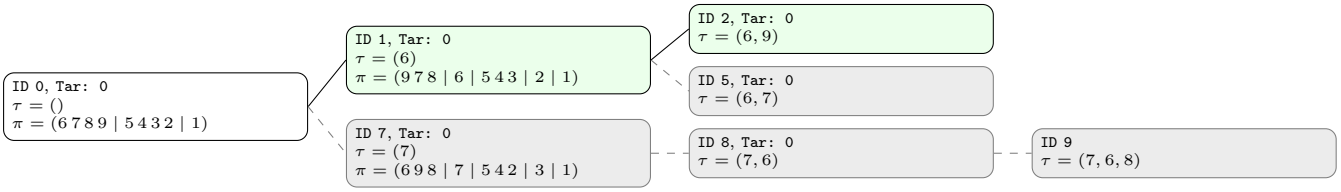
and of course γ lies in that full group. So the highest node where γ becomes usable is now the root `ID 0`. There it shows that the root child `ID 7` belongs to the same root orbit as the already explored child `ID 1` under the current known subgroup of $\mathrm{Aut}(G)$. Hence the whole subtree rooted at `ID 7` is redundant and is pruned. Again the JSON trace first creates `ID 8` and `ID 9`, and only then marks `ID 7`, `ID 8`, and `ID 9` as pruned. So `ID 9` is the moment where γ is *found*, and the root `ID 0` is the higher node where it is then *used*.

This also explains why the root has only two created children, even though the root target cell has four elements. The method creates children lazily in DFS order; it does *not* generate all four root children immediately.

- First it creates the child `ID 1`, corresponding to 6, and explores that subtree.
- The automorphisms found inside that subtree, namely α and β, still fix 6. So at the root they show only that $7, 8, 9$ are mutually equivalent, not yet that they are equivalent to 6.
- Therefore, after the subtree below `ID 1` is finished, one more root child is still needed as a representative of the remaining root orbit, and the method creates `ID 7`, corresponding to 7.
- Only when the leaf `ID 9` is compared with the reference leaf `ID 3`, the automorphism $\gamma = (2\ 3\ 5)(6\ 7\ 9)$ is found. At that moment, the root-level symmetry information becomes strong enough to identify the branch through `ID 7` with the already explored branch through `ID 1`.
- From then on, no further root children need to be created. The potential root children corresponding to 8 and 9 are never created at all, because their entire subtrees are already known to be redundant.

For example, the root child corresponding to 8 is never individualized at all. After γ has been found, the currently known root-level group generated by the discovered automorphisms already contains $\beta\gamma$, and this permutation sends 6 to 8. Hence the root child for 8 lies in the same root orbit as the already explored root child for 6, so a separate individualization of 8 at the root would be redundant. The same idea then applies to 9 as well.

Technical remark. In the JSON trace, pruning is recorded on nodes that have already been created. This is a technical consequence of the DFS run: the automorphism is only discovered once a new leaf has actually been reached and compared with the reference leaf. The mathematically more important effect is that, from that moment on, the corresponding subtree is known to be redundant and no further descendants of it need to be explored. These never-created descendants do not appear in the JSON at all.



6. **What remains.** At this point the search stops. The representative leaf is `ID 3`, corresponding to $\tau = (6, 9, 8)$, and all later branches have been identified with it by the three explicit leaf automorphisms

$$\alpha = (3\ 4)(7\ 8), \qquad \beta = (3\ 4\ 5)(7\ 8\ 9), \qquad \gamma = (2\ 3\ 5)(6\ 7\ 9).$$

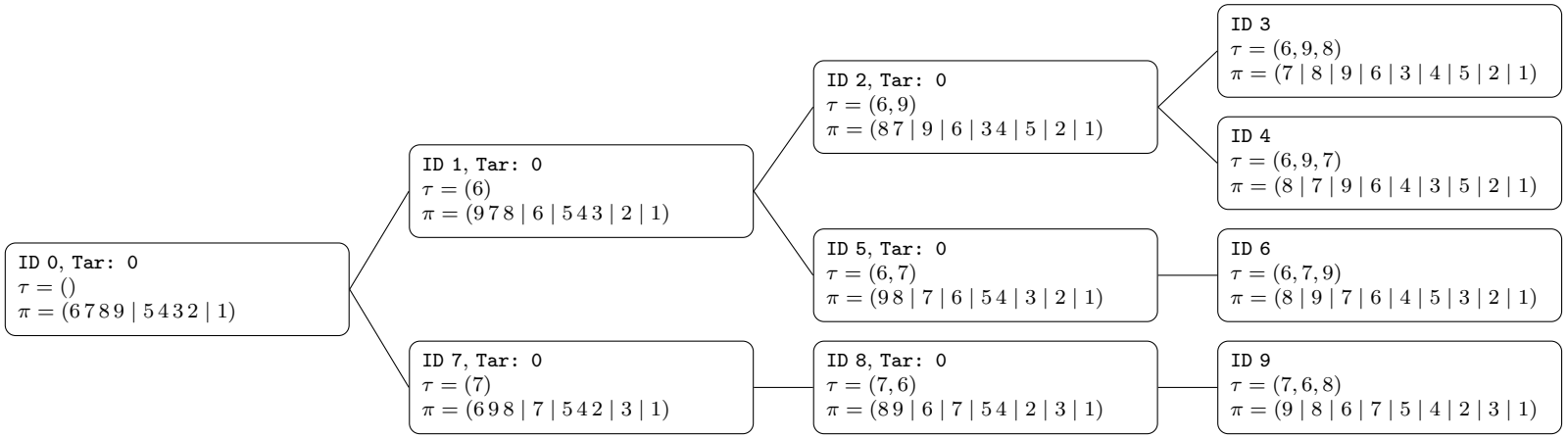
So the key pattern is always the same:

- first a representative branch is explored,
- then a leaf in a sibling branch reveals an automorphism,
- and that automorphism is used at the highest node whose already chosen vertices are fixed by it.

This is exactly why stabilizers appear: the higher the node, the fewer vertices must be fixed, so the more branches may still be identified as equivalent. In the actual trace above, α acts at the level of `ID 2`, β at the level of `ID 1`, and γ already at the root.

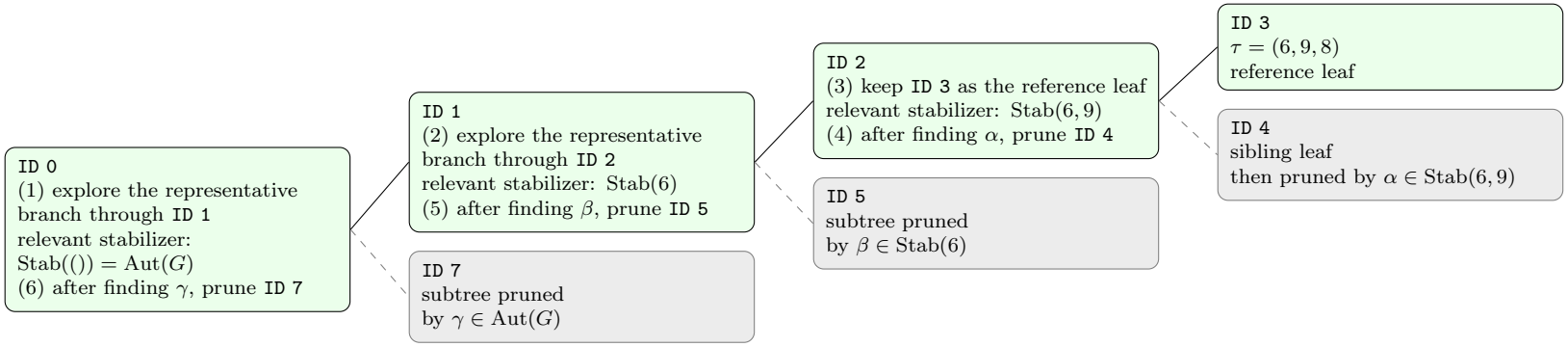
Explored Search Tree from the JSON Trace

Every node below is a node that is actually created in the JSON trace. We show the method node id, the individualization sequence τ, the target cell index `Tar`, and the resulting ordered partition.



Pruned Search Tree Using Stabilizers

Now we compress the explored tree by pruning at the highest node justified by the automorphism discovered later in the run.



The numbers in the boxes above indicate the order in which the corresponding actions become relevant in the DFS run. The same chronology is summarized again in the table below:

found at leaf	automorphism	usable in stabilizer	used for pruning at node
ID 4	$\alpha = (3\ 4)(7\ 8)$	$\mathrm{Stab}(6, 9)$	ID 2
ID 6	$\beta = (3\ 4\ 5)(7\ 8\ 9)$	$\mathrm{Stab}(6)$	ID 1
ID 9	$\gamma = (2\ 3\ 5)(6\ 7\ 9)$	$\mathrm{Stab}(()) = \mathrm{Aut}(G)$	ID 0

The deeper nodes ID 6, ID 8, and ID 9 do appear in the JSON trace, but only as descendants of subtrees that are immediately pruned again. So the compact picture above keeps the pruning locations ID 4, ID 5, and ID 7, which are the nodes where the discovered automorphisms actually take effect.

Summary of Answers

The main calculations from the sheet are collected here as a short answer key. The detailed arguments are in the worked sections before this summary, and the reproducibility commands are given in the final section.

1. The root partition is

$$\pi_{()} = [\,6\,7\,8\,9 \mid 2\,3\,4\,5 \mid 1\,], \qquad W_0 = \{6,7,8,9\}.$$

   See *Example Graph* and *Symmetry in the Running Example, 1. The Root.*

2. The automorphism group consists exactly of the permutations of the four branches $\{2,6\}$, $\{3,7\}$, $\{4,8\}$, and $\{5,9\}$. Hence

$$\mathrm{Aut}(G) = \{\gamma_\pi : \pi \text{ is a permutation of } \{2,3,4,5\}\}.$$

   The root children (6), (7), (8), and (9) form one orbit under this group. See *Automorphism and Automorphism Group* and *Symmetry in the Running Example, 1. The Root.*

3. The full explored trace is given in *Step-by-Step Exploration and Pruning* and summarized again in *Explored Search Tree from the JSON Trace*. The created nodes are `ID 0` to `ID 9`.

4. The three relevant orbit decompositions are:

$$W_0 = \{6,7,8,9\} \quad \text{under } \mathrm{Aut}(G),$$
$$W_1 = \{7,8,9\} \quad \text{under } \mathrm{Stab}(6),$$
$$W_2 = \{7,8\} \quad \text{under } \mathrm{Stab}(6,9).$$

   In each case the current target cell is one orbit. See *Orbits of the Target Cell Under a Stabilizer*, *Symmetry in the Running Example*, and *How the Actual Method Trace Differs.*

5. The chronology is:

$$\texttt{ID 4} \text{ finds } \alpha, \qquad \texttt{ID 6} \text{ finds } \beta, \qquad \texttt{ID 9} \text{ finds } \gamma.$$

   They are then used for pruning at

$$\texttt{ID 2}, \qquad \texttt{ID 1}, \qquad \texttt{ID 0},$$

   respectively. See *Step-by-Step Exploration and Pruning* and *Pruned Search Tree Using Stabilizers.*

6. The root has only two created children because the method creates children lazily in DFS order. First the subtree through `ID 1` is explored. The automorphisms α and β found there still fix 6, so at the root they show only that $7, 8, 9$ are mutually equivalent. Therefore one further representative root child, namely `ID 7`, is still needed. Only after γ is found at `ID 9` does the known root-level group become large enough to identify the branch through `ID 7` with the already explored branch through `ID 1`. In particular, the root child for 8 is never individualized, because the currently known root-level group already contains an automorphism sending 6 to 8. See the root discussion at the end of *Step-by-Step Exploration and Pruning.*

7. The JSON trace can be reproduced with the commands in *Reproducing the GraphCanon Trace*. The input graph is available as `https://ac2026.algochem.techfak.de/Material/Other/stabilizer-example.dimacs`. The important command is the `graph-canon` run that writes `/tmp/stabilizer-example.json`. That JSON file contains the same node ids, target-cell indices, partitions, and automorphism events used throughout this sheet.

Reproducing the GraphCanon Trace

The graph used in this sheet can be written in DIMACS format as follows.

```
p edge 9 8
e 1 2
e 1 3
e 1 4
e 1 5
e 2 6
e 3 7
e 4 8
e 5 9
```

Save this text as `/tmp/stabilizer-example.dimacs`, or download the same file from the course material page. In the course Docker image, `graph-canon` is already available on the `PATH`. The JSON trace can then be generated by the command

```
curl -L -o /tmp/stabilizer-example.dimacs \
  https://ac2026.algochem.techfak.de/Material/Other/stabilizer-example.dimacs

graph-canon \
  --fno-aut-implicit \
  --ftarget-cell f \
  --ftree-traversal dfs \
  -f /tmp/stabilizer-example.dimacs \
  --json /tmp/stabilizer-example.json \
  --tree-dot /tmp/stabilizer-example.dot
```

The command produces a short summary on standard output.

```
seed = 1777821293
/tmp/stabilizer-example.dimacs  target-cell  tree-traversal  edge-labels
/tmp/stabilizer-example.dimacs  f            dfs             no-el
                                refine  aut-pruner  aut-implicit
                                WL-1    basic       False
                                partial-leaf  trace  quotient  degree-1
                                True          True   True      True
Time: 0 ms (0 ms, 6 rounds)
```

The more interesting information is in the JSON file itself. For this example, the trace creates the nodes `ID 0` to `ID 9`, and the three explicit leaf automorphisms are

$$\texttt{ID 4} \to \texttt{ID 3}\colon (3\ 4)(7\ 8),$$

$$\texttt{ID 6} \to \texttt{ID 3}\colon (3\ 4\ 5)(7\ 8\ 9),$$

$$\texttt{ID 9} \to \texttt{ID 3}\colon (2\ 3\ 5)(6\ 7\ 9).$$

These are exactly the three automorphisms used in the worked trace to explain the pruning at `ID 2`, `ID 1`, and the root.