Chapter 7. Orbits and Schreier Vectors

This chapter defines the concepts of orbit and Schreier vector and presents algorithms for computing them. The main question concerning the permutation group that they allow us to answer is: Given two points δ and γ, is there an element mapping δ to γ, and , if so, determine such an element.

Orbits

Let G be a permutation group acting on the set $\Omega = \{1, 2, ..., n\}$ of points. Let G be generated by the set $S = \{s_1, s_2, ..., s_m\}$.

The orbits of G on Ω indicate how G acts on Ω by telling us whether there is an element mapping one point to another. There are three (equivalent) views of orbits. We will present all three.

The *orbit* of G containing the point δ is the set

$$\delta^G = \{ \delta^g \mid g \in G \}$$

of images of δ under elements of the group G.

Define the relation $\sim$ on the set of points by : $\delta \sim \gamma$ if and only if there is an element g of G such that $\delta^g = \gamma$.

This defines an equivalence relation since it is

i. reflexive (for all $\delta \in \Omega$, $\delta^{id} = \delta$),

ii. symmetrical (if $\delta^g = \lambda$ *then* $\lambda^{g^{-1}} = \delta$), and

iii. transitive (if $\delta^g = \lambda$ *and* $\lambda^h = \alpha$, then $\delta^{g \times h} = \alpha$).

The equivalence classes partition the set Ω of points. An *orbit* is an equivalence class of the relation $\sim$.

The generators S determine a (labelled, directed) graph (possibly with loops) as follows. The vertices are the points of Ω. There is an edge

$$\delta \longrightarrow \gamma$$

in the graph if a generator s maps δ to γ. An *orbit* of G on Ω is the set of vertices of a connected component of the graph. (This definition is independent of the generating set of G.)

It is this last view of an orbit that suggests an algorithm for computing an orbit. Algorithm 1 that determines the orbit δ^G is a breadth-first traversal of the connected component containing δ. This is the same as closing the set $\{ \delta \}$ under the operation of taking images.

Algorithm 1 : Determining one orbit

Input : a set $S = \{s_1, s_2, ..., s_m\}$ of generators of a group G acting on Ω;
a point δ;

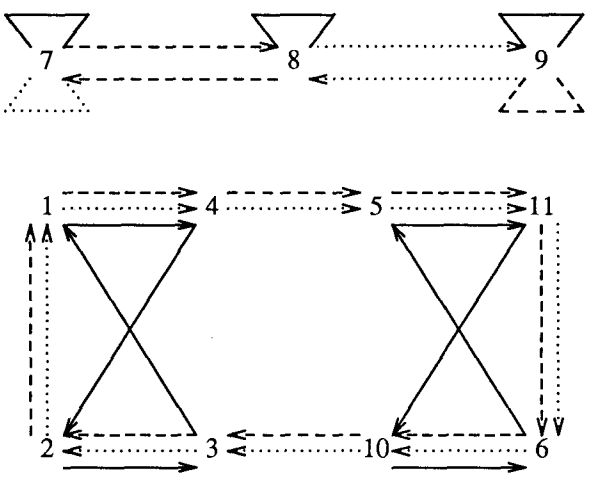
Output : the orbit δ^G;
begin
orbit := { δ };
for each point γ in *orbit* **do**
for each generator s in S **do**
if γ^s not in *orbit* **then** add γ^s to *orbit*; **end if**;
end for;
end for;
end;

As an example, consider the group G acting on $\Omega = \{1,2,...,11\}$ that is generated by

a = (1,4,5,11,6,10,3,2)(7,8)
b = (1,4,5,11,6,10,3,2)(8,9)
c = (1,4,2,3)(5,11,10,6)

The graph we obtain is shown in Figure 1.

Figure 1 : Graph Depicting Action of Generators



The generator *a* is represented by dashed edges.
The generator *b* is represented by dotted edges.
The generator *c* is represented by solid edges.

Algorithm 1 calculates the orbit of $\delta = 7$ as follows: Initially *orbit* = {7}. With γ=7, γ^a=8, so 8 is added to *orbit* giving {7,8}; γ^b=7; γ^c=7. With γ=8, γ^a=7; γ^b=9, so 9 is added to *orbit* giving {7,8,9}; γ^c=8. With γ=9, γ^a=9; γ^b=8, γ^c=9. Hence, the result is {7,8,9}.

Calculating the orbit of $\delta = 1$ adds the points to the orbit in the sequence 1,4,5,2,11,3,6,10.

The analysis of the algorithm shows that

- $|orbit| \times |S|$ images γ^s are formed,
- $|orbit| \times |S|$ tests of membership in the orbit are performed, and
- $|orbit|$ points are added to the orbit.

If the orbit is represented as a characteristic function on the set Ω, then both the membership test and the addition of a point to the orbit require one operation. The cost of initializing the orbit is $|\Omega|$ operations. If permutations are represented in image form then the calculation of an image requires one operation. Under these assumptions the total cost is

$$|orbit| \times \left[2 \times |S| + 1 \right] + |\Omega| \quad \text{operations.}$$

Algorithm 2, which computes all the orbits of a group, is a simple extension of Algorithm 1.

Algorithm 2 : All orbits

```
Input : a set S = {s_1, s_2, ..., s_m} of generators for a group G acting on Ω;
Output : the orbits orbit[1], orbit[2],... of G on Ω;
begin
  orbits := empty;  i := 0;
  for δ := 1 to |Ω| do
    if not ( δ in orbits ) then
      i := i + 1;       orbit[i] := {δ};
      for each point γ in orbit[i] do
        for each generator s in S do
          if γ^s not in orbit[i] then add γ^s to orbit[i]; end if;
        end for;
      end for;
      orbits := orbits ∪ orbit[i];
    end if;
  end for;
end;
```

The sum of the orbit sizes is $|\Omega|$. So the total cost of forming the orbits *orbit*[1], *orbit*[2],...,*orbit*[N] in the inner two loops is

$$|\Omega| \times \left[2 \times |S| + 1 \right] + N \times |\Omega| \quad \text{operations,}$$

where N is the number of orbits. To this we must add the cost of the tests on δ, the cost of initializing *orbits*, and the cost of forming the union of the orbits in *orbits*. If points are added to *orbits* as they are added to *orbit*[i] then the total cost of the these tasks is $3 \times |\Omega|$ operations. Hence, the total cost of Algorithm 2 is

$$|\Omega| \times \left[2 \times |S| + N + 4 \right] \text{ operations.}$$

Schreier Vectors

While the orbits tell us whether there is an element mapping one point to another, it is the Schreier vectors that determine such an element.

Consider the graph constructed from a set of generators for a permutation group acting on Ω. We may consider the edges of the graph to be bi-directional because the edge

$$\alpha \longleftarrow s \longrightarrow \beta$$

is equivalent to the edge

$$\alpha \longrightarrow s^{-1} \longrightarrow \beta$$

Hence, we may assume the edge is in the desired direction. If the points δ and γ are in the same connected component then there is a path

$$\delta = \alpha_0 \longrightarrow s_{i_1} \longrightarrow \alpha_1 \longrightarrow s_{i_2} \longrightarrow \alpha_2 \longrightarrow s_{i_3} \longrightarrow \cdots \longrightarrow s_{i_r} \longrightarrow \alpha_r = \gamma$$

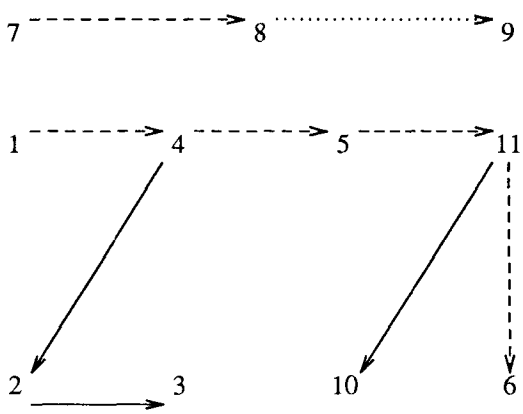
from δ to γ. Then the element

$$s_{i_1} \times s_{i_2} \times \cdots \times s_{i_r}$$

is in the group G and maps δ to γ.

A spanning forest of the graph provides a unique path from the root of a tree to any point in the connected component of the root. A spanning forest of the previous example is shown in Figure 2.

Figure 2 : Spanning Forest



The generator *a* is represented by dashed edges.
The generator *b* is represented by dotted edges.
The generator *c* is represented by solid edges.

A *Schreier vector* of an orbit (or orbits) relative to a set S of generators is a means of representing a spanning tree (forest) of the connected component(s) that corresponds to the orbit(s). The main information we need is the label of the edge leading to a point γ. A Schreier vector stores the edge labels in a vector indexed by the points of Ω. The entry for a

point γ is the generator (or its inverse) that labels the edge leading into γ. For example,

	1	2	3	4	5	6	7	8	9	10	11
v		c	c	a	a	a		a	b	c	a

where the orbit representatives (corresponding to the roots of the trees) have no entry. The father of a point γ in the spanning tree is the image of γ under the inverse of $v[\gamma]$. These are often stored in another vector, called the *backward pointers*. For example,

	1	2	3	4	5	6	7	8	9	10	11
w		4	2	1	4	11		7	8	11	5

To determine an element mapping the orbit representative to a point γ in the connected component, we use Algorithm 3.

Algorithm 3 : Tracing a Schreier vector

Input : a Schreier vector v and backward pointers w of the orbits of a group G;
a point γ;

Output : an element mapping the orbit representative to γ;

```
function trace(γ : point;
               v : Schreier vector;
               w : backward pointers ) : element;
begin
  if γ is orbit representative then
    result is identity;
  else
    result is trace( w[γ], v, w ) × v[γ];
  end if;
end;
```

If γ_1 and γ_2 are two points in the same orbit then

$$trace(\gamma_1, v, w)^{-1} \times trace(\gamma_2, v, w)$$

is an element mapping γ_1 to γ_2.

If the backward pointers are not known explicitly then they can be calculated from the relationship

$$w[\gamma] = \gamma^{v[\gamma]^{-1}}.$$

Of course, this requires us to know (or calculate) the inverses of the generators. We will often abuse the function *trace* by using only the first one or two arguments. In these cases we will be unconcerned whether the backward pointers are known explicitly, or we will assume the the existence of the Schreier vector.

The Schreier vector (and backward pointers) of *one* orbit can be determined by suitably modifying Algorithm 1. We present the modified algorithm as Algorithm 4.

Algorithm 4 : Determining an orbit and Schreier vector

Input : a set $S = \{s_1, s_2, ..., s_m\}$ of generators of a group G acting on Ω;
a point δ;

Output : the orbit δ^G;
a Schreier vector v and backward pointers w of the orbit δ^G
relative to the set S of generators;

```
begin
  orbit := { δ };
  for γ := 1 to |Ω| do v[γ] := 0;     w[γ] := 0; end for;

  for each point γ in orbit do
    for each generator s in S do
      if γ^s not in orbit then
        add γ^s to orbit;  v[γ^s] := s;  w[γ^s] := γ;
      end if;
    end for;
  end for;
end;
```

Similarly to the analysis of Algorithm 1, we see that the total cost of Algorithm 4 is

$$|orbit| \times \left\lceil 2 \times |S| + 3 \right\rceil + 3 \times |\Omega| \quad \text{operations.}$$

Executing Algorithm 4 with our previous group $G = \langle a, b, c \rangle$ on $\Omega = \{1,2,...,11\}$ with $\delta = 1$ gives

	1	2	3	4	5	6	7	8	9	10	11
v		c	c	a	a	a				c	a
w		4	2	1	4	11				11	5

This corresponds to the spanning tree illustrated earlier.

Note that, by virtue of the breadth-first traversal, Algorithm 4 constructs the Schreier vector corresponding to a *minimal* spanning tree of the connected component.

All the orbits and a Schreier vector for a spanning forest can be determined by suitably modifying Algorithm 2.

Analysis of Algorithm 3

The cost of determining an element mapping γ_1 to γ_2 is dependent upon the depth of the spanning tree. Let $d(\gamma)$ be the depth of γ in the spanning tree, then the call *trace*(γ, v, w) requires $d(\gamma)$ multiplications and $2 \times d(\gamma)$ operations. Each multiplication requires $2 \times |\Omega|$ operations. Thus the total cost is

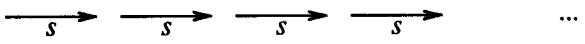
$$2 \times d(\gamma) \times \left\lceil |\Omega| + 1 \right\rceil \text{ operations.}$$

The cost of determining an element mapping γ_1 to γ_2 is

$$2 \times \left\lceil d(\gamma_1) + d(\gamma_2) \right\rceil \times \left\lceil |\Omega| + 1 \right\rceil \text{ operations.}$$

The greatest value for the depth of a point in a spanning tree is the size of the orbit. This occurs for the cyclic group $G = \langle s \rangle$, where the spanning tree is linear, as in Figure 3.

Figure 3 : Spanning Tree of Cyclic Group



The *best* worst case we can expect is a spanning tree of depth $\log_{|S|}(|orbit|)$. In general, any tree can be a spanning tree. More accurately,

Proposition

Let T be a tree with n nodes, where each node has at most m children. Then there are permutations $s_1, s_2, \ldots, s_m$ of $\Omega = \{1, 2, \ldots, n\}$ and a labelling of the nodes of T and the edges of T such that Algorithm 4 using $S = \{s_1, s_2, \ldots, s_m\}$ as generators and $\delta = 1$ produces the labelled tree T as the spanning tree.

Alas, we have no idea how many choices of the permutations and labellings there are for a given tree. So we do not know the distribution of spanning trees, and hence do not know the average depth of a point in the spanning tree.

Summary

Orbits and Schreier vectors allow us to decide whether there is an element mapping one point to another, and to determine such an element, if it exists. The cost of determining all the orbits and Schreier vectors is $O(|\Omega| \times |S|)$. The cost of determining an element mapping one point to another is $O(|\Omega|^2)$.

Exercises

(1/Easy) For the groups given in chapter 2, calculate their orbits and Schreier vectors.

(2/Easy) Using the Schreier vector calculated in the example of Algorithm 4, determine an element mapping the point 2 to the point 11.

(3/Moderate) A group G acts on its elements by conjugation. The orbits in this case are called *conjugacy classes*. The elements in a conjugacy class have the same order. This information is useful in finding the zuppos - the initial step of the subgroup lattice algorithm. Use Algorithm 2 to determine the conjugacy classes of the symmetric group of degree 4. Take the generators to be {(1,2,3,4), (1,2)}.

Bibliographical Remarks

The algorithms of this chapter are adaptations of graph theoretic algorithms for determining closure and spanning trees.

The concept of an orbit is a very old one. The computational significance of them was recognized by Charles Sims, who also introduced the concept of a Schreier vector. The algorithms to compute them and to trace them occur in lectures that Sims gave in Oxford in January and February 1973. The algorithms appear in J. S. Leon, "*On an algorithm for finding a base and strong generating set for a group given by generating permutations*", Mathematics of Computation **35**, 151 (1980) 941-974.

Orbits can also be computed as partition joins. One partition join algorithm for computing orbits is presented in J. S. Leon, "*An algorithm for computing the automorphism group of a Hadamard matrix*", Journal of Combinatorial Theory, series A **27**, 3 (1979) 289-306. Asymptotically these algorithms are $O(|\Omega| \times log(|\Omega|) + |\Omega| \times |S|)$. One can also adapt union-find algorithms to compute orbits and achieve a complexity which is essentially linear in $|\Omega|$ and $|\Omega| \times |S|$.

The view of a Schreier vector as a tree is first presented in C. M. Hoffman, **Group-theoretic Algorithms and Graph Isomorphism**, Springer-Verlag, Berlin, 1982, which also presents algorithms for constructing orbits and Schreier vectors.