BACKTRACK PROGRAMMING AND ISOMORPH REJECTION ON ORDERED SUBSETS

Brendan D. McKay

ABSTRACT: Let V be a finite set, where $|V| = n > 0$ and let Q_n denote the set of sequences of n distinct elements of V. Suppose X is a subset of Q_n which is invariant under the action of a permutation group Γ acting on V. We consider the application of backtrack programming to the problem of finding a transversal for the action of Γ on X. Effective methods are devised which require no information about Γ other than a procedure for recognising whether or not a given permutation is in Γ. We show that these algorithms find a small set of generators for Γ. An application to graph isomorphism is mentioned.

1. Introduction. Let V be a finite non-empty totally ordered set. Without loss of generality, we will assume that $V = \{1, 2, \dots, n\}$ with the usual ordering. For each k $(0 \le k \le n)$ define $Q_k = Q_k(V)$ to be the set of all sequences $\nu = [v_1, \dots, v_k]$ of distinct elements of V. The *length* of $\nu \in Q_k$ is $|\nu| = k$. If $k = 0$, then Q_k contains only the null sequence [], sometimes written as $[v_1, \dots, v_k]$ where $k = 0$.

The set of all sequences of distinct elements of V will be denoted $Q = \bigcup_{k=0}^{n} Q_k$. The ordering on V can be extended to an ordering on Q in a natural way. If $\mu = [v_1, \dots, v_k] \in Q$ and $\nu = [w_1, \dots, w_\ell] \in Q$ then μ is *earlier* than ν, written $\mu < \nu$, if either

(i) $v_i = w_i$ $(1 \le i < t \le \min(k, \ell))$, and $v_t < w_t$, or

(ii) $v_i = w_i$ $(1 \le i \le k)$ and $k < \ell$.

Let S_n be the symmetric group on V. For $\gamma \in S_n$ and $\nu = [v_1, \dots, v_k] \in Q$ we can define $\nu^\gamma = [v_1^\gamma, \dots, v_k^\gamma]$. Obviously, $\nu^\gamma \in Q$ and $|\nu^\gamma| = |\nu|$. More generally, if $U \subseteq Q$ and $\Psi \subseteq S_n$ we can define $U^\Psi = \{u^\psi \mid u \in U, \psi \in \Psi\}$.

Let $\Gamma \leq S_n$. If $\mu, \nu \in Q$ and $\nu = \mu^\gamma$ for some $\gamma \in \Gamma$ we say that μ and ν are *equivalent under* Γ, denoted $\mu \underset{\Gamma}{\sim} \nu$ or simply $\mu \sim \nu$.

2. *The Problem.* Suppose X is a subset of Q_n such that $X^\Gamma = X$. Clearly X is a union of equivalence classes under Γ. Let R be the set containing the earliest element of each equivalence class. The problem is to find R either

(P1) if Γ is specified explicitly, or

(P2) if elements of Γ can be recognised, i.e. a procedure is given which can decide whether or not a given permutation is in Γ.

The case P1 has been treated in somewhat greater generality by Fillmore and Williamson [4]. The different case where $Q(V)$ is replaced by the power set 2^V of V has been treated by Perlman [11, 12]. Our main contributions here will be towards the solution of case P2 when the set X occurs as the terminal nodes of a backtrack program. The proofs of all results not proved here can be found in [6].

3. We begin with a formal description of a large class of backtrack programs.

Let $W : Q \to 2^V$ be a function such that for any $\nu = [v_1, \ldots, v_k] \in Q$ we have $W(\nu) \subseteq V \backslash \{v_1, \ldots, v_k\}$. The *search tree* with *defining function* W is the set

$$T = \{\nu = [v_1, \ldots, v_k] \in Q \mid v_k \in W([v_1, \ldots, v_{k-1}]), \ k \geq 1\} \cup Q_0 .$$

To avoid trivial cases, we will assume that T has at least two elements. The following algorithm, known as a *backtrack program* or *depth-first search* generates T from W in the order induced from Q. The symbol $\leftarrow$ denotes an assignment of value. For example, "$k \leftarrow k - 1$" decrements k by 1.

4. **ALGORITHM.** Find T given W.

```
(1) $k \leftarrow 0$
    $T \leftarrow Q_0$

(2) $U_k \leftarrow W([v_1, \ldots, v_k])$

(3) If $U_k = \phi$ go to (6).

(4) $v_{k+1} \leftarrow \min U_k$
      $U_k \leftarrow U_k \setminus \{v_{k+1}\}$
      $k \leftarrow k + 1$
      $T \leftarrow T \cup \{[v_1, \ldots, v_k]\}$

(5) If $k < n$ go to (2).

(6) $k \leftarrow k - 1$
    If $k \geq 0$ go to (3); otherwise stop.
```

. *Example.* Let G be the labelled graph of figure 1.

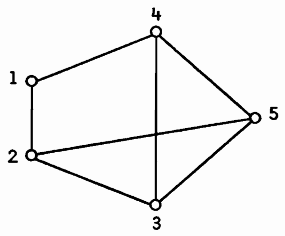


Figure 1

Let $\Gamma = \mathrm{Aut}(G) = \{(1), (2\ 4), (3\ 5), (2\ 4)(3\ 5)\}$.

Suppose X is the set of sequences $[v_1, \ldots, v_5] \in Q_5$ such that v_i and v_{i+1} are adjacent in G for i = 1, 2, 3, 4. Thus X is the set of directed Hamiltonian paths in G.

Algorithm 4 can be used to find X by defining the function W as follows.

(1) $W([\,]) = V$

(2) $W([v_1, \ldots, v_k]) = \{v \in V \backslash \{v_1, \ldots, v_k\} \mid v \text{ is adjacent to } v_k\}$, $k \geq 1$.

The search tree T for this example can be drawn as in figure 2.

The elements of T will be called *nodes* to avoid confusion with the *points* of the graph G. A node of the form $[v_1, \ldots, v_k, v_{k+1}]$ is called a *successor* of the node $[v_1, \ldots, v_k]$.

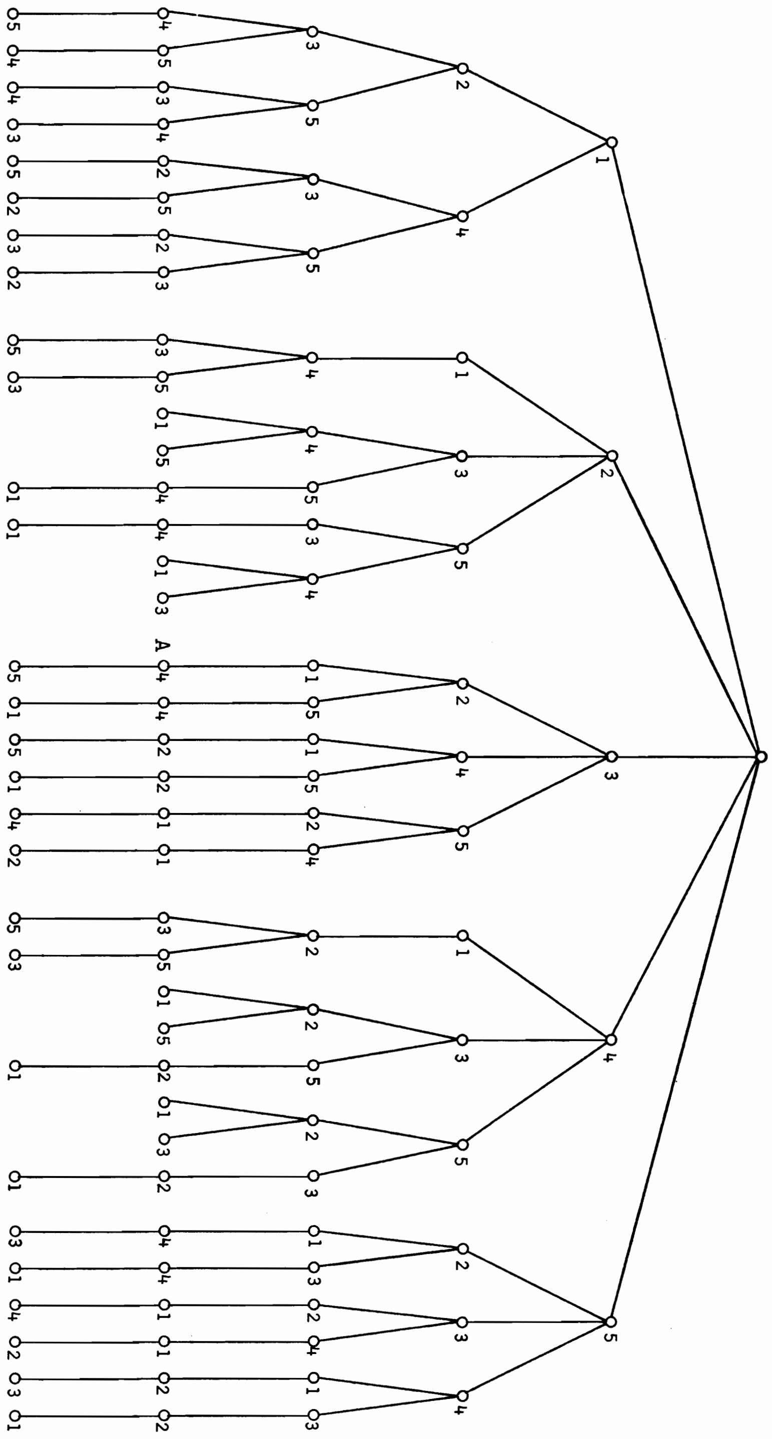


Figure 2

The lines drawn in figure 2 join each node to its successors. For clarity, the nodes are labelled only with their last entries, but the full sequence can be obtained by tracing a path downwards from the top (*root*) of the tree. Thus the node marked A is [3, 2, 1, 4]. Algorithm 4 begins at the root of the tree and works downwards where possible, taking the left-most branches on the way down. If it reaches a dead-end it "backtracks" to find another path downwards, and thus continues until it has traversed the entire tree.

The successor relationship can be extended as follows. A node ν_1 of the form $[v_1, \ldots, v_k, \ldots, v_r]$ $(0 \le k < r)$ is called a *descendant* of the node $\nu_2 = [v_1, \ldots, v_k]$. Conversely, ν_2 is an *ancestor* of ν_1. If a node has no successors (and hence no descendants) it is called an *endnode* of T. An endnode of length n is a *terminal node* of T.

We are going to examine solutions to problems P1 and P2 in the case where X is the set of terminal nodes of a search tree T for which $T^{\Gamma} = T$. Although this latter condition is clearly stronger than the condition $X^{\Gamma} = X$, it seems that comparitively few problems of practical interest have $X^{\Gamma} = X$ but $T^{\Gamma} \neq T$. In any case, the discussions to follow can be generalised to the case $T^{\Gamma} \neq T$ by restricting attention to the subtree of T formed by X and its ancestors.

6. A convenient tool for the study of search trees is the *successor function* $F : Q \to 2^Q$ defined by

$$F(\nu) = \begin{cases} \{\mu \in Q \mid \mu \text{ is a successor of } \nu\}, & \nu \in T \\ \phi & , \nu \notin T . \end{cases}$$

The condition $T^\Gamma = T$ can often be verified by application of the following theorem.

7. THEOREM. *Let* T *be the search tree with successor function* F. *Let* $\Gamma \leq S_n$. *Then the following are equivalent.*

(1) $T^\Gamma = T$.

(2) $E^\Gamma = E$, *where* E *is the set of endnodes of* T.

(3) *For each* $\nu \in Q$, $\gamma \in \Gamma$,

$$F(\nu^\gamma) = F(\nu)^\gamma .$$

From now on we will assume that $T^\Gamma = T$ and $X \neq \phi$.

As described earlier, the terminal nodes X of T can be divided into equivalence classes according to the action of Γ. The earliest terminal nodes in each class will be called the *identity nodes* of T and denoted by the set $R = \{e_1, e_2, \ldots, e_r\}$ in the order induced from Q.

The basis of our solutions to problems P1 and P2 lies in the following simple ideas.

Let ν be a node of T. Then the *subtree of* T *rooted at* ν is the set $T(\nu)$ consisting of ν and all its descendants in T.

8. LEMMA. *Let* $\nu \in T$, $\gamma \in \Gamma$. *Then*

$$T(\nu^\gamma) = T(\nu)^\gamma .$$

9. LEMMA. *Let* $\nu_1, \nu_2 \in T$ *where* $|\nu_1| = |\nu_2|$, *and suppose* $\mu_1 \in T(\nu_1)$ *and* $\mu_2 \in T(\nu_2)$. *Then* $\nu_1 < \nu_2$ *if and only if* $\mu_1 < \mu_2$

10. LEMMA. *Let* $\nu_1, \nu_2 \in T$ *where* $\nu_1 \sim \nu_2$ *and* $\nu_1 < \nu_2$. *Then* $T(\nu_2)$ *contains no identity nodes of* T.

An immediate consequence of lemma 10 is that, in our search for identity nodes, having generated the subtree $T(\nu_1)$, any later subtree of the form $T(\nu_1^\gamma)$ for $\gamma \in \Gamma$ can be ignored. This idea gives us a simple solution to problem P1.

For any $\nu \in Q$, we define Γ_ν to be the subgroup of Γ which individually fixes each element of ν. Define a function $W/\Gamma : Q \rightarrow 2^V$ as follows. For any $\nu \in Q$, consider the orbits of Γ_ν which intersect $W(\nu)$. Define $(W/\Gamma)(\nu)$ to be the set of the smallest elements from each of these orbits.

11. THEOREM. *Let* T/Γ *be the search tree with defining function* W/Γ. *Then the terminal nodes of* T/Γ *are the identity nodes of* T.

For the example of §5 we find T/Γ to be the search tree drawn in figure 3. The labelling is the same as for figure 2.

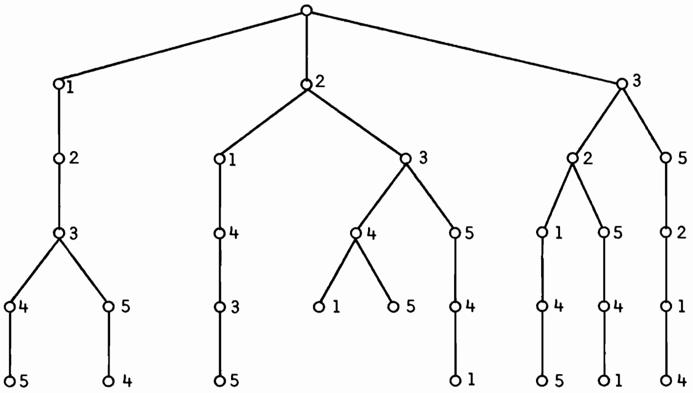


Figure 3

12. We now turn to problem P2. Suppose that $\tau_1 = [v_1, \ldots, v_n]$ and $\tau_2 = [w_1, \ldots, w_n]$ are elements of X, where $v_i = w_i$ for $0 \leq i \leq k < n$ but $v_{k+1} \neq w_{k+1}$. Then define

$$\tau_1 - \tau_2 = [v_1, \ldots, v_k, v_{k+1}]$$

and

$$\tau_2 - \tau_1 = [w_1, \ldots, w_k, w_{k+1}] .$$

Clearly $\tau_1 - \tau_2$ and $\tau_2 - \tau_1$ are both in T. Suppose $\tau \in X\backslash R$. Then for some $e_i \in R$, $\gamma \in \Gamma$ we have $\tau = e_i^{\gamma}$. Consequently, if $e_i - \tau = [v_1, \ldots, v_k, v_{k+1}]$ then $\tau - e_i = [v_1, \ldots, v_k, v_{k+1}^{\gamma}]$.

Hence $$\tau - e_i = (e_i - \tau)^{\gamma}$$

and so $$T(\tau - e_i) = T(e_i - \tau)^{\gamma} \qquad \text{by lemma 8.}$$

Since e_i is earlier than τ, $e_i - \tau$ is earlier than $\tau - e_i$. Hence $T(\tau - e_i)$ contains no identity nodes, by lemma 10, and so can be ignored. This process of removing subtrees is carried out systematically by the following algorithm.

13. ALGORITHM. Find $R = \{e_1, e_2, \ldots, e_r\}$.

```
(1) $k \leftarrow 0$
    $r \leftarrow 0$

(2) $U_k \leftarrow W([v_1, \ldots, v_k])$

(3) If $U_k = \phi$ go to (9).

(4) $v_{k+1} \leftarrow \min U_k$
    $U_k \leftarrow U_k \setminus \{v_{k+1}\}$
    $k \leftarrow k + 1$

(5) If $k < n$ go to (2).

(6) {We have found a terminal node $\tau = [v_1, \ldots, v_n]$.}
    If $\tau \sim e_j$ for some j $(1 \le j \le r)$ go to (8).

(7) $r \leftarrow r + 1$
    $e_r \leftarrow \tau$
    Go to (9).

(8) $k \leftarrow |\tau - e_j|$

(9) $k \leftarrow k - 1$
    If $k \ge 0$ go to (3); otherwise stop.
```

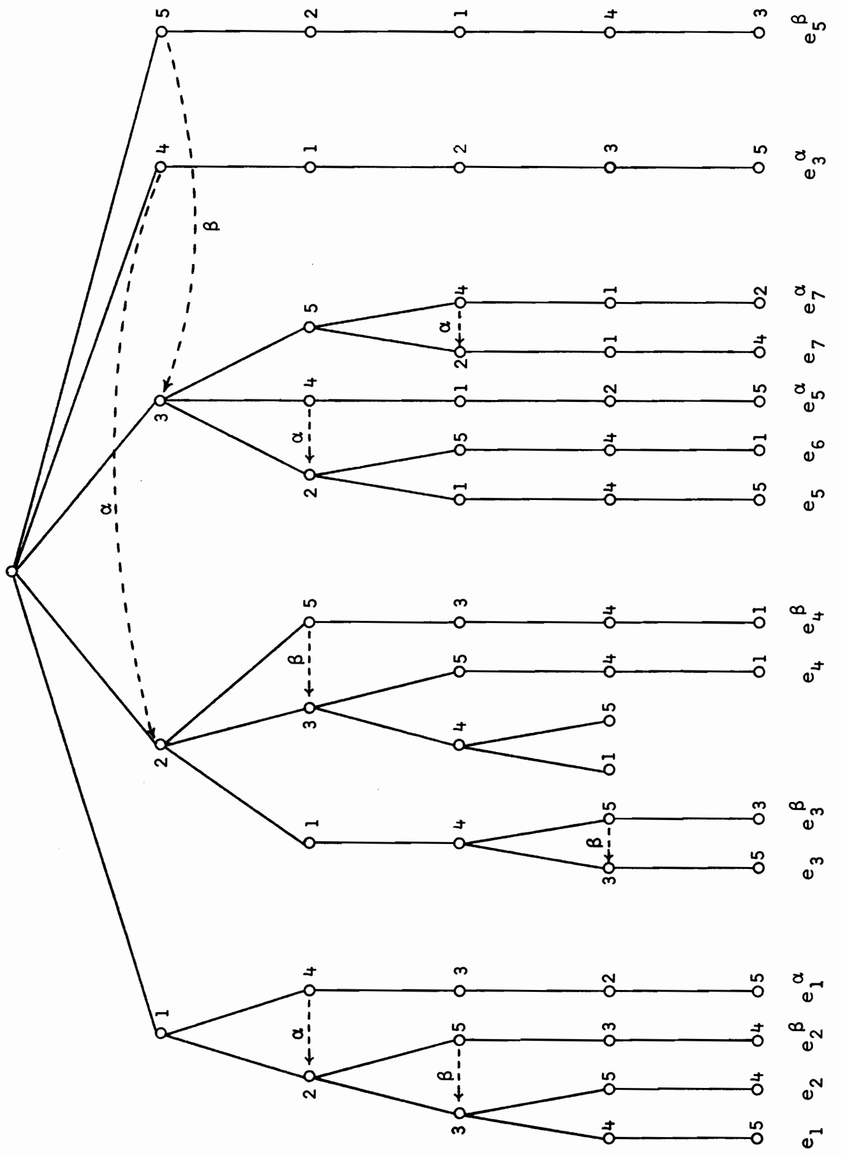


Figure 4

$\alpha = (2\ 4), \quad \beta = (3\ 5)$

The search tree T_1 produced by algorithm 13 for the example of §5 is shown in figure 4.

Suppose that during the execution of algorithm 13 we have found an identity node e_j and a terminal node $\tau \neq e_j$ such that $\tau = e_j^\gamma$ for some $\gamma \in \Gamma$. Then we say that $\tau - e_j$ is *absorbed onto* $e_j - \tau$ *by* γ. In figure 4 such absorptions are indicated by arrows.

14. LEMMA. *Let* $e_j = [v_1, \ldots, v_n] \in R$. *Then any node of* T *of the form* $[v_1, \ldots, v_k, w]$ $(0 \leq k < n)$ *will also be in* T_1. *In particular*, $e_j \in T_1$.

15. COROLLARY. *If* $\tau \in X$, $\tau - e_j \in T_1$.

16. LEMMA. *Let* $e_i = [v_1, \ldots, v_n] \in R$. *Let* $\nu_1 \neq \nu_2 \in T$ *where* $\nu_1 = [v_1, \ldots, v_k, v_{k+1}]$ *and* $\nu_2 = [v_1, \ldots, v_k, w]$. *Then if* $\nu_2 = \nu_1^\gamma$ *for some* $\gamma \in \Gamma$, ν_2 *will be absorbed onto* ν_1 *by some* $\delta \in \Gamma$.

The element $\delta \in \Gamma$ of lemma 16 is the permutation for which $\tau = e_j^\delta$, where τ is the earliest terminal node of $T(\nu_2)$ and $e_j \in R$. It is not hard to show that $\nu_2 = \tau - e_j$.

17. Let $e_j = [v_1, \ldots, v_n] \in R$. For $0 \leq k \leq n$ define $\nu_k = [v_1, \ldots, v_k]$ and $\Gamma^{(k)} = \Gamma_{\nu_k}$.

18. THEOREM. *For each* k $(0 \leq k < n)$ *we have the disjoint union*

$$\Gamma^{(k)} = \bigcup_{i=1}^{s_k} \Gamma^{(k+1)} \gamma_i^{(k)}$$

where $\gamma_1^{(k)} = (1)$ *and* $\{\gamma_2^{(k)}, \ldots, \gamma_{s_k}^{(k)}\}$ *are the elements of* Γ *by which nodes of* T *are absorbed onto* ν_{k+1}.

19. COROLLARY 1. *For* $0 \le h < n$, $\Gamma^{(h)}$ *is generated by the set* $\Omega_h = \{\gamma_i^{(k)} \mid h \le k < n,\ 2 \le i \le s_k\}$. *In particular,* Ω_0 *generates* Γ.

20. COROLLARY 2. T_1 *has* $\le |R| \left[\binom{n}{2} + 1\right]$ *terminal nodes, with equality only if* $\Gamma = S_n$.

For example, in figure 4 with $e_j = e_1$, we have

$$\begin{aligned}
\Gamma^{(0)} &= \Gamma^{(1)} \\
\Gamma^{(1)} &= \Gamma^{(2)} \cup \Gamma^{(2)}(2\ 4) \\
\Gamma^{(2)} &= \Gamma^{(3)} \cup \Gamma^{(3)}(3\ 5) \\
\Gamma^{(3)} &= \Gamma^{(4)} \\
\Gamma^{(4)} &= \Gamma^{(5)} = \{(1)\}.
\end{aligned}$$

Hence $$\Gamma = \langle (2\ 4), (3\ 5) \rangle.$$

21. Let $U \subseteq V$. A *partition* of U is a set π of disjoint non-empty sets whose union is U. The elements of π are called its *cells*. The family of all partitions of U will be denoted $\Pi(U)$.

If π_1 and π_2 are partitions of U, we say that π_1 is *finer* than π_2, denoted $\pi_1 \le \pi_2$, if each cell of π_1 is contained in some cell of π_2. Conversely, π_2 is *coarser* than π_1. It is not hard to show that $(\Pi(U), \le)$ is a lattice. The finest partition of U has only one element of U in each cell, and will be called the *discrete* partition of U.

Suppose $\pi = \{C_1 | C_2 | \cdots | C_k\} \in \Pi(U)$ and $\psi \in S_n$. If $C_i^\psi = C_i$ for $1 \le i \le k$ we say that π is *fixed* by ψ. Notice that each cell must be individually fixed by ψ. If $\Psi \subseteq S_n$ and $U^\Psi = U$, we say that π is *fixed* by Ψ if it is fixed by each element of Ψ. The finest partition which is coarser than π but fixed by Ψ is denoted $\pi \vee \Psi$. For simplicity $\pi \vee \{\psi\}$ will be written $\pi \vee \psi$.

22. LEMMA. (1) $\pi \vee \Psi = \pi \vee \langle\Psi\rangle$, *where* $\langle\Psi\rangle$ *is the group generated by* Ψ.

(2) *If* $\langle\Psi\rangle = \langle\psi_1, \psi_2, \ldots, \psi_\ell\rangle$,

$$\pi \vee \Psi = ((\cdots(\pi \vee \psi_1) \vee \psi_2) \vee \cdots \vee \psi_\ell).$$

A simple algorithm for computing $\pi \vee \Psi$ can be found in [6].

We will find it convenient to assume that the cells of a partition are ordered according to their lowest elements. Thus when we write $\pi = \{C_1 | C_2 | \cdots | C_k\}$ we implicitly assume that $\min C_1 < \min C_2 < \cdots < \min C_k$, where $\min C_i$ is the least element of C_i.

23. We return to our discussion of problem P2. Define e_j, ν_k and $\Gamma^{(k)}$ as in §17, and suppose that $0 \le q < n$.

Let $\{\gamma_1, \ldots, \gamma_m\}$ be a set of elements of Γ by which nodes of T are absorbed onto nodes ν_k where $k > q$. Clearly $\Lambda \le \Gamma^{(q)}$, where $\Lambda = \langle\gamma_1, \ldots, \gamma_m\rangle$. Let $U = W(\nu_q)$. Then, by theorem 7, $U^\Lambda = U$. Suppose $w_1 < w_2$ where $w_1, w_2 \in U$ and $w_2 = w_1^\lambda$ for some $\lambda \in \Lambda$.

Then $\mu_2 = \mu_1^\lambda$ where $\mu_1 = [v_1, \ldots, v_q, w_1]$ and $\mu_2 = [v_1, \ldots, v_q, w_2]$. Hence, by lemma 10, $T(\mu_2)$ contains no identity nodes.

In order to construct a useful algorithm out of these ideas we must introduce some new data items. Upon encountering a node $\nu \in T$ we compute $W(\nu)$ and create a partition π_ν equal to the discrete partition of $W(\nu)$. Thereafter, whenever we discover an element $\gamma \in \Gamma$ by which a node is absorbed onto a descendant of ν we set $\pi_\nu \leftarrow \pi_\nu \vee \gamma$. At any stage we need partitions only for the current node and its ancestors (excepting that we don't need a partition for a terminal node) and so at most n partitions must be stored at one time.

In the following description, a cell of a partition is regarded as having been "chosen" if any element of the cell has been chosen.

24. ALGORITHM: Find $R = \{e_1, e_2, \ldots, e_r\}$.

```
(1)  $k \leftarrow 0$
     $r \leftarrow 0$

(2)  $U \leftarrow W([v_1, \ldots, v_k])$
     If $U = \phi$ go to (9).

(3)  $\pi_k \leftarrow$ discrete partition of $U$

(4)  $C \leftarrow$ first cell of $\pi_k$ not yet chosen
     $k \leftarrow k + 1$
     $v_k \leftarrow \min C$

(5)  If $k < n$ go to (2).

(6)  {We have found a terminal node $\tau = [v_1, \ldots, v_n]$.}
     If $\tau \sim e_j$ for some $j$ $(1 \le j \le r)$ go to (8).

(7)  $r \leftarrow r + 1$
     $e_r \leftarrow \tau$
     Go to (9).

(8)  Compute $\gamma$ such that $\tau = e_j^\gamma$.
     $k \leftarrow |\tau - e_j|$
     For $0 \le i < k$ set $\pi_i \leftarrow \pi_i \vee \gamma$

(9)  If $k = 0$ stop.
     $k \leftarrow k - 1$

(10) If all cells of $\pi_k$ have been chosen go to (9);
     otherwise go to (4).
```

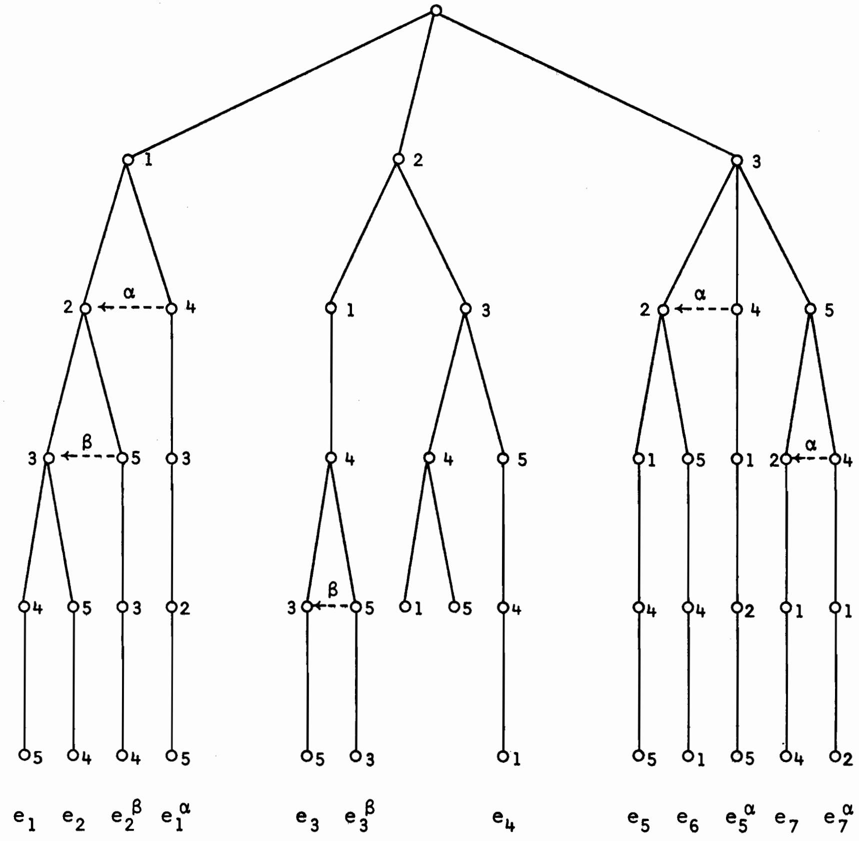


Figure 5

$\alpha = (2\ 4)$, $\beta = (3\ 5)$

The search tree T_2 produced by algorithm 24 for the example of §5 is shown in figure 5.

Suppose that at some execution of step (8) in algorithm 24 we have $[v_1, \ldots, v_n] = \tau = e_j^\gamma$ where $\tau \in X$, $e_j \in R$, $\gamma \in \Gamma$. Let $0 \leq i < |\tau - e_j|$. Suppose that for some node $\nu = [v_1, \ldots, v_i, w]$ of T_2 the operation $\tau_i \longleftarrow \tau_i \vee \gamma$ causes the cell of π_i containing w to be increased in size. Then we can say that γ is *active* at ν.

25. **LEMMA.** γ *is active at* $e_j - \tau$.

Recall the definitions of §17.

26. **THEOREM.** *Let* Y *be the set of elements of* Γ *found by algorithm 24 which are active at ancestors of* e_j. *Then*

(1) *For* $0 \leq k \leq n$, $\Gamma^{(k)} = \langle Y \cap \Gamma^{(k)} \rangle$. *In particular*, Y *generates* Γ.

(2) $|Y| \leq n-p \leq n-1$, *where* Γ *has* p *orbits*.

27. **COROLLARY.** T_2 *has* $\leq |R|(n-p+1)$ *terminal nodes*.

Despite the power of algorithm 24, further improvement is possible. Upon creating a node ν of T_2, algorithm 24 initially sets π_ν to the discrete partition of $W(\nu)$. In this sense it assumes no prior knowledge of Γ_ν. However, if we have a set $\{\gamma_1, \ldots, \gamma_m\}$ of previously discovered elements of Γ then some of them, say $\{\gamma_1, \ldots, \gamma_q\}$ may be in Γ_ν. Clearly we can set $\pi_\nu \longleftarrow \pi_\nu \vee \{\gamma_1, \ldots, \gamma_q\}$ at this stage without losing identity nodes.

In practice it seems best to limit the number of group elements stored, since the time for initialisation of each π_ν may become excessive. In the following algorithm, J is a positive integer specifying the maximum number of elements to be stored. It is actually only necessary to keep a list of the non-trivial cycles of each such element, since this is all that matters in the operation $\pi_\nu \leftarrow \pi_\nu \vee \gamma_i$.

28. ALGORITHM: Find $R = \{e_1, e_2, \ldots, e_r\}$.

```
(1)  $k \leftarrow 0$
     $r \leftarrow 0$
     $m \leftarrow 0$

(2)  $U \leftarrow W([v_1, \ldots, v_k])$
     If $U = \phi$ go to (10).

(3)  $\pi_k \leftarrow$ discrete partition of U
     For $1 \le i \le m$ such that $\gamma_i$ fixes $[v_1, \ldots, v_k]$
         set $\pi_k \leftarrow \pi_k \vee \gamma_i$

(4)  C $\leftarrow$ first cell of $\pi_k$ not yet chosen
     $k \leftarrow k + 1$
     $v_k \leftarrow \min C$

(5)  If $k < n$ go to (2).

(6)  {We have found a terminal node $\tau = [v_1, \ldots, v_n]$.}
     If $\tau \sim e_j$ for some j $(1 \le j \le r)$ go to (8).
```

```
(7)  $r \leftarrow r + 1$
     $e_r \leftarrow \tau$
     Go to (10).

(8)  Compute $\gamma$ such that $\tau = e_j^\gamma$.
     $k \leftarrow |\tau - e_j|$
     For $0 \le i < k$ set $\pi_i \leftarrow \pi_i \vee \gamma$

(9)  If $m = J$ go to (10).
     $m \leftarrow m + 1$
     $\gamma_m \leftarrow \gamma$

(10) If $k = 0$ stop.
     $k \leftarrow k - 1$

(11) If all cells of $\pi_k$ have been chosen go to (10);
     otherwise go to (4).
```

If $J = 0$, algorithm 28 is identical to algorithm 24. If $J \ge 2$, algorithm 28 applied to the example of §5 produces the search tree T_3 shown in figure 6. At the termination of the algorithm we have $\gamma_1 = (3\ 5)$, $\gamma_2 = (2\ 4)$.

29. THEOREM: *For any* $J \ge 0$, *theorem 26 holds for algorithm 28 provided* $j = 1$.

If J is sufficiently large, the number of terminal nodes of T_3 seems to be typically of order $|R| + n$, but no bound better than that for T_2 has been proven. For search trees with large numbers of endnodes that are not terminal nodes, T_3 is often

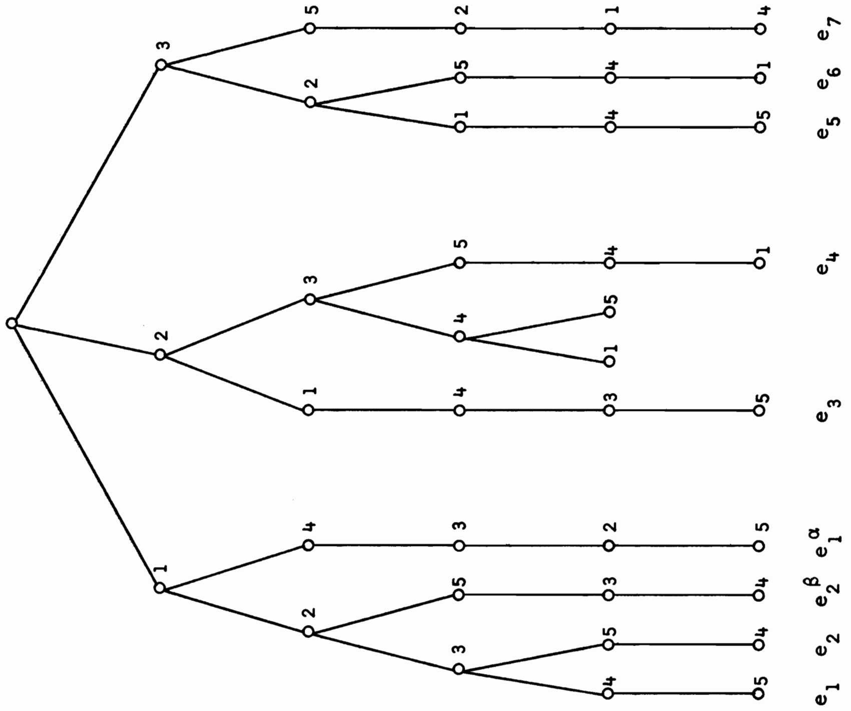


Figure 6

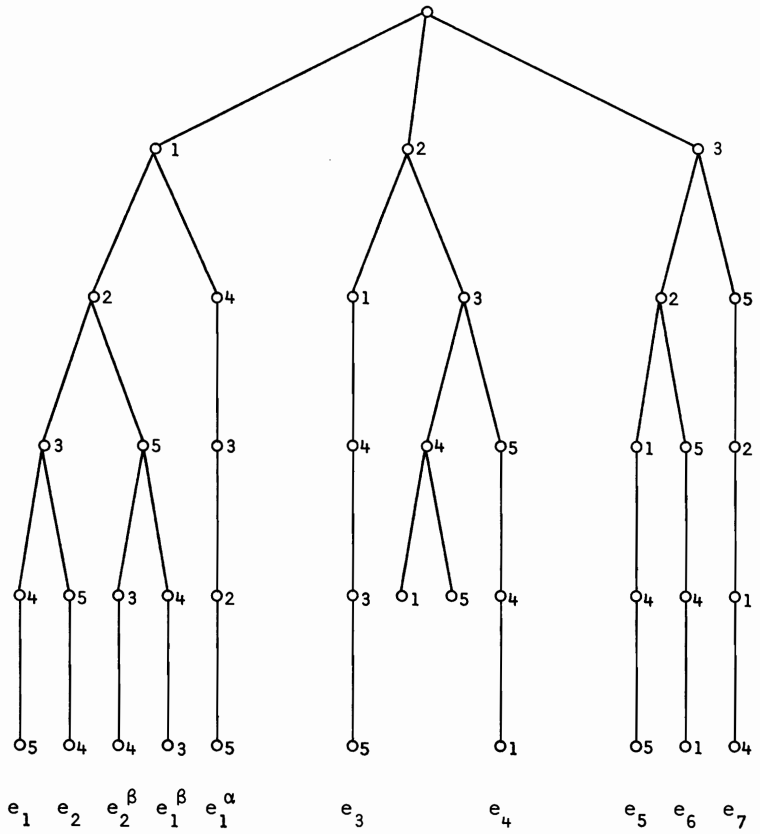
$\alpha = (2\ 4),\ \ \beta = (3\ 5)$

considerably smaller than T_2, since the size of subtrees having no terminal nodes can be reduced.

30. A possibly undesirable feature of these algorithms is that they require the full set of identity nodes to be stored. If the set R must be determined exactly, there seems to be no alternative, since otherwise further identity nodes could not be positively identified. However, in some applications a larger set of terminal nodes, known to contain R will be sufficient. One method which appears to work very well is to choose an integer $L \geq 0$ and to store the first identity node e_1 and the latest L terminal nodes which have not been recognised as *not* identity nodes. Any further terminal node which is not equivalent to one of the stored terminal nodes is regarded as "possibly an identity node". The reason for storing e_1 is that theorems 18, 26 and 29 will still hold for this identity node.

Algorithm 28 often works very well even if $L = 0$, in which case only e_1 is stored. The set Y of theorem 26 will contain all the elements of Γ discovered by the algorithm. The resulting search tree in this case for the example of §5 is shown in figure 7.

31. We have shown that, in addition to finding R, algorithms 24 and 28 can be used to find a set of $\leq n - 1$ elements of Γ which generate Γ. Consequently they can be used for this purpose whether or not R is required, and we will now concentrate on this aspect of algorithm 28. We begin by showing how the full group Γ can be found from its generators.



$\alpha = (2\ 4), \quad \beta = (3\ 5)$

<u>Figure 7</u>

Let $e_1 = [v_1, \ldots, v_n]$.

Let $Y = \{\gamma_1, \ldots, \gamma_m\}$ be the elements of Γ found by algorithm 28 which are active at ancestors of e_1, in the order they are produced.

32. ALGORITHM. Generate Γ, given Y and e_1.

```
(1) $\eta_0 \leftarrow (1)$
    Output $\eta_0$.
    If $m = 0$ stop.

(2) $j \leftarrow 1$
    $\ell \leftarrow 1$
    $G \leftarrow$ digraph with vertices $V$ and no edges

(3) $k \leftarrow \min\{i \mid v_i^{\gamma_j} \neq v_i\} - 1$

(4) For each $v = v_i$ $(k < i \le n)$ moved by $\gamma_j$, add the
        directed edge $[v, v^{\gamma_j}]$, labelled $\gamma_j$, to $G$ —
        unless it is already there.

(5) $j \leftarrow j + 1$
    If $j \le m$ and $v_i^{\gamma_j} = v_i$ for $1 \le i \le k$, go to (4).

(6) {Let $\{w_1, \ldots, w_r\}$ be the component of $G$ containing
        $w_1 = v_{k+1}$}
    $s_\ell \leftarrow r - 1$
    For $2 \le i \le r$ set $\psi(i-1, \ell) \leftarrow \delta_1 \delta_2 \cdots \delta_t$, where
        $\delta_1, \delta_2, \ldots, \delta_t$ are the labels of the edges of
        a directed path in $G$ from $w_1$ to $w_i$.
```

```
(7)  If $j > m$ go to (8).
     $\ell \leftarrow \ell + 1$
     Go to (3).

(8)  $k \leftarrow 1$
     $j_1 \leftarrow 1$
     $i_1 \leftarrow 0$

(9)  $i_k \leftarrow i_k + 1$
     If $i_k \le s_{j_k}$ go to (10).
     $j_k \leftarrow j_k + 1$
     If $j_k > \ell$ go to (11).
     $i_k \leftarrow 1$

(10) $\eta_k \leftarrow \psi(i_k, j_k)\eta_{k-1}$
     Output $\eta_k$.
     If $j_k = \ell$ go to (9).
     $k \leftarrow k + 1$
     $j_k \leftarrow j_{k-1} + 1$
     $i_k \leftarrow 1$
     Go to (10).

(11) $k \leftarrow k - 1$
     If $k = 0$ stop; otherwise go to (9).
```

The ideas behind algorithm 32 can be found in [6], p. 24-25. Note that the storage requirements depend on $|V|$ and not on $|\Gamma|$. If $|\Gamma|$ is very large, the algorithm requires only marginally more than one permutation multiplication (at step (10)) per element of Γ. The group elements $\psi(i, j)$ used in the algorithm are coset representations akin to the elements $\gamma_i^{(k)}$ of theorem 18.

If it is not practical or desirable to generate the whole of Γ there is still a lot of information about Γ which may be obtained more directly. For example, lemma 22 shows that the orbits of Γ can be easily found.

For $0 \leq j \leq n$ define $\nu_j = [v_1, \ldots, v_j]$, and $\Gamma^{(j)} = \Gamma_{\nu_j}$, where $\nu_n = e_1$ as before.

The following result shows that we can easily obtain $|\Gamma^{(j)}|$ for any j.

33. THEOREM. *Consider algorithm 28 immediately it has completed the examination of the subtree* $T(\nu_j)$, *where* $0 \leq j < n$.

(a) *The cells of* π_j *are orbits of* $\Gamma^{(j)}$.

(b) *If* C *is the cell of* π_j *containing* v_{j+1}, *then*
$$|\Gamma^{(j)}| = |\Gamma^{(j+1)}|\,|C|.$$

The following theorem can be used to prove theorem 26 (2).

34. THEOREM. *For* $0 \leq i \leq m$ *define* $\Psi^{(i)} = \langle\gamma_1, \gamma_2, \ldots, \gamma_i\rangle$. *Then if* $0 \leq i < j \leq m$, $\Psi^{(j)}$ *has strictly fewer orbits than does* $\Psi^{(i)}$.

For the following results we will assume, in the notation of §30, that all the terminal nodes stored by algorithm 28 as *possible* identity nodes are *actual* identity nodes. This will be the case, for example, if $L = 0$ or $L \geq |R| - 1$.

35. LEMMA. *Suppose that at step (8) of algorithm 28 we have* $\tau = e_j^\gamma$, *where* $\tau \in X$, $e_j \in R$, $\gamma \in \Gamma$. *Then* τ *is the earliest terminal node of* $T(\tau - e_j)$ *which is equivalent to* e_j.

Let $\Psi \subseteq \Gamma$. The *support* of Ψ, $\text{supp}(\Psi)$ is the set of elements of V moved by some element of Ψ. If $\Psi = \{\psi\}$ we write $\text{supp}(\psi)$ for $\text{supp}(\{\psi\})$. Thus $\text{supp}(\psi) = \{v \in V \mid v^\psi \neq v\}$.

36. THEOREM. *Suppose that for some* $\gamma \in Y$, $\gamma = \alpha\beta$ *where* $\alpha, \beta \in \Gamma$, $\alpha \neq (1)$ *and* $\text{supp}(\alpha) \cap \text{supp}(\beta) = \phi$. *Then* $\beta = (1)$.

Proof. Suppose that during the algorithm we have found $\tau = e_j^\gamma$ where $\tau \in X$, $e_j = [w_1, \ldots, w_n] \in R$. Without losing generality, there is some j $(0 \leq j < n)$ such that $w_i^\gamma = w_i$ for $1 \leq i \leq j$ but $w_{j+1}^\alpha \neq w_{j+1}$. If δ is a non-identity element of Γ such that $\text{supp}(\delta) \subseteq \{w_i \mid j+1 \leq i \leq n, w_i \notin \text{supp}(\alpha)\}$, then $e_j^{\alpha\delta} > e_j^\alpha$ or else $e_j^\delta < e_j$. Hence the earliest terminal node of $T([w_1, \ldots, w_j, w_{j+1}^\alpha])$ equivalent to e_j must be e_j^α. Hence $\beta = (1)$. □

Let $\Psi^{(1)}$ and $\Psi^{(2)}$ be subgroups of Γ such that $\text{supp}(\Psi^{(1)}) \cap \text{supp}(\Psi^{(2)}) = \phi$. The *direct sum* $\Psi^{(1)} \oplus \Psi^{(2)}$ is the group $\langle \Psi^{(1)} \cup \Psi^{(2)} \rangle$.

37. THEOREM. *Suppose that for some subset* $Y^* \subseteq Y$, *we have* $\langle Y^* \rangle = \Psi^{(1)} \oplus \Psi^{(2)}$ *where* $\Psi^{(1)}$ *and* $\Psi^{(2)}$ *are non-trivial subgroups of* Γ. *Then* Y^* *is a disjoint union* $Y^{(1)} \cup Y^{(2)}$, *where* $\langle Y^{(1)} \rangle = \Psi^{(1)}$ *and* $\langle Y^{(2)} \rangle = \Psi^{(2)}$.

Proof. Any element of $\Psi^{(1)} \oplus \Psi^{(2)}$ is of the form $\psi_1\psi_2$ where $\psi_1 \in \Psi^{(1)}$ and $\psi_2 \in \Psi^{(2)}$. By Theorem 36, one of ψ_1 and ψ_2 is trivial. $\square$

38. THEOREM. *Suppose that for some* $\nu \in Q$, *the subgroup* Γ_ν *has exactly one non-trivial orbit* C. *Under any of the following conditions, there is a subset* $Y^* \subseteq Y$ *such that* $\langle Y^* \rangle$ *is conjugate to* Γ_ν *in* Γ.

(1) Γ *is abelian.*

(2) $N(\Gamma_\nu) = \Gamma_\nu$ *where* $N(\Gamma_\nu)$ *is the normaliser of* Γ_ν *in the symmetric group* $S(C)$ *on* C. [*For example, this is true if* $\Gamma_\nu = S(C)$.]

(3) $\Gamma_\nu = \{\gamma \in \mathrm{Aut}(G) \mid \gamma$ *fixes* $\pi\}$ *where* G *is a graph (or digraph) with vertices* V *and* $\pi \in \Pi(V)$.

Proof. For $\Lambda \leq \Gamma$ denote $\ell(\Lambda) = \min\{i \mid v_i \in \mathrm{supp}(\Lambda)\}$. Suppose Ψ is a conjugate of Γ_ν in Γ for which $t + 1 = \ell(\Psi)$ is a maximum. Then $\Psi \leq \Gamma^{(t)}$.

Let $D = \mathrm{supp}(\Psi)$. The maximality of $\ell(\Psi)$ ensures that D is an orbit of $\Gamma^{(t)}$.

(i) If $\Psi = \Gamma^{(t)}$ the theorem follows from theorem 29.

(ii) Suppose $\Psi < \Gamma^{(t)}$ and let T denote the permutation group on D induced by $\Gamma^{(t)}$. Clearly $\Psi \leq T$. We now treat cases (1)-(3) separately.

(1) If Γ is abelian, so are Ψ and T. But then $|\Psi| = |T| = |D|$ since transitive abelian groups are regular. Thus $\Psi = T$.

(2) Since Ψ is normal in $\Gamma^{(t)}$, it is normal in T. But then $N(\Psi) = \Psi$ implies $\Psi = T$.

(3) Suppose $\Gamma = \{\gamma \in \mathrm{Aut}(G) \mid \gamma \text{ fixes } \pi\}$ where G is a graph with vertices V and $\pi \in \Pi(V)$. The case where G is a digraph is similar.

Let ρ be the partition of V whose cells are the orbits of $\Gamma^{(t)}$. Then $\Gamma^{(t)} = \{\gamma \in \mathrm{Aut}(G) \mid \gamma \text{ fixes } \rho\}$.

Suppose D_1 is a cell of ρ other than D and that some vertex v in D_1 is adjacent to a vertex in D. Since Ψ is transitive on D and fixes v, v must be adjacent to every vertex in D. Furthermore, considering the action of $\Gamma^{(t)}$, every vertex in D is adjacent to every vertex in D_1.

Consequently D is either not joined, or completely joined, to every other cell of ρ. It is easy to see that this implies $\Psi = T = \mathrm{Aut}(\langle D \rangle)$, where $\langle D \rangle$ is the subgraph of G induced by D.

We have found in each case that $T = \Psi$. Since any element γ of

$\Gamma^{(t)}$ is of the form $\alpha\beta$ where $\alpha \in T$ and $\text{supp}(\beta) \cap D = \phi$, we see that $\alpha \in \Psi$ and $\beta \in \Gamma^{(t)}$.

Thus $\Gamma^{(t)} = \Psi \oplus \Gamma_D^{(t)}$, and the result follows from theorems 29 and 37. $\square$

A simple implication of theorem 38 is that Y contains at least one transposition from each conjugacy class of transpositions in Γ. This observation can be generalised as follows, but we omit the proof.

39. THEOREM. *Suppose that in theorem 38,* Γ_ν *acts as the symmetric group on* C. *Then* Y* *can be chosen to consist of* $|C| - 1$ *transpositions.*

40. Application to Graph Isomorphism. One of the standard problems in computational graph theory is that of canonically labelling a graph. Intuitively, this involves assigning labels to the vertices of a graph in an order independent of any previous labels. We shall define this idea rigorously and then demonstrate how our techniques can be applied.

Suppose $\underset{\sim}{G} = \underset{\sim}{G}(V)$ is the set of all (labelled) graphs with vertex set $V = \{1, 2, \ldots, n\}$. For $G \in \underset{\sim}{G}$, $\gamma \in S_n$ define $G^\gamma \in \underset{\sim}{G}$ to be the graph in which vertices v^γ and w^γ are adjacent exactly when v and w are adjacent in G.

A *canonical labelling algorithm* is a map

$$c : \underset{\sim}{G} \rightarrow \underset{\sim}{G}$$

such that for each $G \in \underset{\sim}{G}$, $\gamma \in S_n$,

(1) $c(G)$ is isomorphic to G,

(2) $c(G^\gamma) = c(G)$.

Conditions (1) and (2) imply that two graphs $G, H \in \underset{\sim}{G}$ are isomorphic if, and only if, $c(G) = c(H)$.

41. The operation of the known canonical labelling algorithms can be described in terms of a *canonical map*. This is defined to be a map

$$\Delta : \underset{\sim}{G} \rightarrow 2^{\underset{\sim}{G}} \setminus \{\phi\}$$

such that for each $G \in \underset{\sim}{G}$, $\gamma \in S_n$,

(1) every member of $\Delta(G)$ is isomorphic to G.

(2) $\Delta(G^\gamma) = \Delta(G)$.

Canonical maps are also important in algorithms which test two graphs for isomorphism by examining them together.

Given a canonical map, we can define a canonical labelling algorithm by choosing a linear ordering on $\underset{\sim}{G}$. A common method is to apply the usual ordering of the integers by considering the $0-1$ adjacency matrix of a graph as an n^2-bit binary number. Another method uses the incidence matrix similarly [9, 13]. Relative to whatever order on $\underset{\sim}{G}$ we have chosen, a canonical labelling algorithm c can be defined by

$$c(G) = \max \Delta(G) \qquad (G \in \underset{\sim}{G}).$$

42. A great many canonical maps have been used, explicitly or not, in published algorithms. However the majority of them fall into the class we now describe.

For $G \in \underset{\sim}{G}$, $\tau = [v_1, \ldots, v_n] \in Q_n$, define the graph $G^\tau \in \underset{\sim}{G}$ to have vertices i and j adjacent exactly when vertices v_i and v_j are adjacent in G.

Let $\underset{\sim}{W} : \underset{\sim}{G} \times Q \to 2^V$ be a map such that for each $G \in \underset{\sim}{G}$, $\nu \in Q$, $\gamma \in S_n$ we have

(1) $\underset{\sim}{W}(G, \nu) \subseteq V \setminus \nu$

(2) $\underset{\sim}{W}(G^\gamma, \nu^\gamma) = \underset{\sim}{W}(G, \nu)^\gamma$

(3) The program tree T_G with defining function $\underset{\sim}{W}(G, \cdot)$ has at least one terminal node.

43. THEOREM. *For any* $G \in \underset{\sim}{G}$ *define*

$$\Delta(G) = \{G^\tau \mid \tau \textit{ is a terminal node of } T_G\}.$$

Then (1) Δ *is a canonical map*

(2) $T_G^\Gamma = T_G$, *where* $\Gamma = \mathrm{Aut}(G)$.

Explicit uses of this method for finding a canonical map have been given by Berztiss [2], Overton and Proskurowski [9, 13], Ullmann [15] and Arlazarov et al. [1]. However, as demonstrated in [6], most of the so-called "partitioning" procedures [3, 5, 8, 10, 14, 16] also fall into this class.

Obviously, any terminal nodes of T_G which are equivalent under Aut(G) correspond to the same labelled graph. Consequently, the definition of Δ in theorem 43 can be replaced by

$$\Delta(G) = \{G^{\tau} \mid \tau \text{ is an identity node of } T_G\}.$$

Since T_G often has vastly fewer identity nodes than terminal nodes, any of our methods for producing the identity nodes can be used to advantage in computing Δ. The first attempt along these lines was made by Arlazarov et al. [1], who used a method something like algorithm 24. A faster method (probably the fastest available) can be found in [6]. Recent improvements in data structures have enabled the development of an algorithm [7] which can efficiently compute Aut(G) and c(G) for most graphs with up to about 1000 vertices, without use of secondary storage.

REFERENCES

[1] V. ARLAZAROV, I. ZUEV, A. USKOV and I. FARADZEV: An algorithm for the reduction of finite non-oriented graphs to canonical form. *Zh. vȳchisl. Mat. mat. Fiz. 14*, *3* (1974) 737-743.

[2] A. BERZTISS: A backtrack procedure for isomorphism of directed graphs. *JACM 20*, *3* (1973) 365-377.

[3] D.G. CORNEIL and C.C. GOTLIEB: An efficient algorithm for graph isomorphism. *JACM 17*, *1* (1970) 51-64.

[4] J. FILLMORE and S. WILLIAMSON: On backtracking: a combinatorial description of the algorithm. *SIAM J. Comput. 3*, *1* (1974) 41-55.

[5] G. LEVI: Graph isomorphism: a heuristic edge-partitioning-oriented algorithm. *Computing 12* (1974) 291-313.

[6] B.D. McKAY: Backtrack programming and the graph isomorphism problem. *M.Sc. Thesis*, *Dept. of Mathematics*, *Univ. of Melbourne* (1976).

[7] B.D. McKAY: Computing automorphisms and canonical labellings of graphs. International Conf. on Combinatorial Mathematics, Canberra, August 1977, to appear.

[8] H.L. MORGAN: The generation of a unique machine description for chemical structures. *J. Chem. Docum. 5* (1965) 107-113.

[9] M. OVERTON and A. PROSKUROWSKI: *Finding the maximal incidence matrix of a large graph.* Stanford Univ., Dept. of Computer Sci. STAN-CS-509 (1975).

[10] R. PARRIS and R.C. READ: *A coding procedure for graphs.* Scientific Report. UWI/CC 10. Univ. of West Indies Computer Centre (1969).

[11] D.M. PERLMAN: Computational methods for pattern enumeration and isomorph rejection. *Ph.D. Thesis, Univ. of Calif. at San Diego* (1973).

[12] D.M. PERLMAN: Isomorph rejection on power sets. *SIAM J. Comput. 3, 3* (1974) 177-183.

[13] A. PROSKUROWSKI: Search for a unique incidence matrix of a graph. *BIT 14* (1974) 209-226.

[14] G. SAUCIER: Un algorithme efficace recherchant l'isomorphisme de 2 graphes. *RIRO R-3, 5* (1971) 39-51.

[15] J.R. ULLMANN: An algorithm for subgraph isomorphism. *JACM 23, 1* (1976) 31-42.

[16] S.H. UNGER: GIT - A heuristic program for testing pairs of directed line graphs for isomorphism. *CACM 7, 1* (1964) 26-34.

Department of Mathematics,
Melbourne University,
Parkville, 3052,
Australia.