Enumerating submultisets of multisets

Jurriaan Hage*

Abstract

In this paper we consider the problem of enumerating the submultisets of a multiset, in which each element has equal multiplicity. The crucial property is that consecutive submultisets in this listing differ one in the cardinality of only one of the elements. This is a generalization to k-ary numbers of the so-called Gray Codes for binary numbers. In the special case of binary numbers there is also a link to the Game Of Hanoi.

We give an example where these results were put to good use, and indicate when in general the method described can be applied.

1 Introduction

Given the fact that similar bitstrings yield widely different decimal numbers led Frank Gray to define and patent the Gray Code [2]: he devised a coding of decimal numbers into bitstrings such that two decimal numbers k and $k+1$ are coded by bitstrings that differ in exactly one position. This code was used to reduce the importance of transmission errors.

In this paper we generalize this method to codings of decimal numbers by k-ary numbers for arbitrary k. We start with this general result, specializing later to Gray Codes. The result for Gray Codes (and through this result, the link to the Game of Hanoi), is well-known [5], but the proof given here for the more general case, is quite different from known proofs. Maybe the proof method gives some inspiration for problems still remaining for the Game of Hanoi [4].

As an application of our result, we continue then with a method for generating all the graphs in a switching class in an efficient way. Abstractly, a switching class can be generated from an undirected graph, by performing the switching operation to the graph G for all subsets σ of the vertices $V(G)$ of G. The general result proved in this paper also allows for graphs labelled with elements from a group (see the books of Ehrenfeucht, Harju and Rozenberg [1] and Hage [3]). The complexity of doing the switching operation (for a fixed graph) grows proportional with the size of selector. The theory developed here allows for an algorithm in which all possible switchings of a graph are generated in an incremental fashion, using switches of constant size (being, in this case, the constant one).

*Inst. of Information and Computing Sci., Univ. Utrecht, P.O.Box 80.089, 3508 TB Utrecht, Netherlands

We conclude with some experimental timing results for the improvements obtained by exploiting our method and give a simple rule which can be used to detect opportunities where this technique can also be applied.

A more detailed exposition including examples is given in Hage [3].

2 Enumerating submultisets

In this section we devise a way to sequence all submultisets of a certain type of multiset. In this multiset all elements have the same cardinality. This restriction is not necessary, but makes the exposition clearer and the algorithms more consise. We conclude with a few remarks about enumerating submultisets of any multiset.

A *multiset* S over V is a function $S : V \to \mathbf{N}_0$. The value $S(v)$ for $v \in V$ is the *multiplicity* of v in S. A multiset S' over V is a *submultiset* of S if for all $v \in V$, $S'(v) \leq S(v)$. The *cardinality* or *size* of a multiset includes multiplicity: $|S| = \sum_{v \in V} S(v)$.

For $m > 1$, let $S(m, n)$ be the multiset over $V = \{v_0, \ldots, v_{n-1}\}$ so that $S(v) = m - 1$ for all $v \in V$. If we linearly order the elements of V from v_{n-1} to v_0, then we can code any submultiset by the multiplicities, $S(v_{n-1}) \ldots S(v_0)$, in other words, a number with base m. We can also consider these numbers to be strings of length n over $M = \{0, \ldots, m-1\}$. Because of the obvious bijections between submultisets, numbers with base m and these strings, we shall use them interchangeably. We note that there are exactly m^n submultisets of $S(m, n)$.

Example 2.1
For $m = 3$ and $n = 4$, $S = S(m, n) = \{v_0, v_0, v_1, v_1, v_2, v_2, v_3, v_3\}$. The submultiset $\{v_0, v_0, v_1, v_2\}$ can be written as number with base 3 as 0112 or as its decimal equivalent $1 \cdot 9 + 1 \cdot 3 + 2 \cdot 1 = 14$. It is important to remember that our strings are zero-indexed and position zero is at the extreme right. Hence 2 has position zero in 0112. ◇

First we introduce some notation for rooted edge-labelled trees T where the root of T is designated by root(T). The trees we consider have arity m and are complete. In other words, every internal vertex has exactly m children and the leaves are all at the same level. The children are ordered from left to right; we shall refer to them as child i for $i \in M$. The labels we shall use for the edges are the elements of M and we demand that for each internal vertex, the edge to every child is labelled with a unique element of M. Hence the labels on the edges to the children of a certain internal vertex are a permutation of M.

Recall that the *level* of a vertex v in a tree is the number of vertices on the path from the root to v. Hence the level of the root is 1. The *height* of a tree is the level of the lowest leaf in the tree minus one. Hence the height of the trivial tree, consisting of a single node, is 0.

In the following, $a : (a_1, \ldots, a_c)$ denotes the sequence $(a, a_1, \ldots, a_c)$. Let T be a tree of height n and let $\pi = (a_1, \ldots, a_{n'})$ be a sequence over M with $n' \leq n$. Then π determines a vertex $C(\pi, T)$, as follows: $C(\lambda, T) = \text{root}(T)$, and $C(a : \pi', T) = C(\pi', T')$ where T' is the subtree rooted at child a. Similarly we define $L(\pi, T)$

as follows: $L(\lambda, T) = \text{root}(T)$ and $L(a:\pi', T) = L(\pi', T')$ where T' is the subtree reachable from root(T) by an edge labelled with a. As a mnemonic, C stands for Child directed and L for Label directed.

Because the labels on the edges form a permutation of M, it should be clear that in both cases there is a bijection between sequences $\pi = a_1, \ldots, a_{n'}$ over M for $n' \leq n$ and vertices in the tree. Hence we may define for a vertex v in T, $L(v, T) = \pi$ if $L(\pi, T) = v$ and $C(v, T) = \pi$ if $C(\pi, T) = v$. Remember that we can interpret the value of $C(v, T)$ and $L(v, T)$ as a string, as a number with base k, and as the corresponding natural number.

The *least common ancestor* of two different vertices $v_1, v_2 \in V(T)$ is the unique vertex $v = \text{lca}(v_1, v_2)$, so that the paths from the root to v_1 and v_2 split up in v.

If v_1 and v_2 are on the same level of the tree, we say that v_2 *follows* v_1 in T if $C(v_2, T) = C(v_1, T) + 1$. This means that v_2 is the first vertex to be encountered starting at v_1 and going to the right on the same level. Note that in this case, the labels on the edges to the subtrees of $\text{lca}(v_1, v_2)$ containing v_1 and v_2 respectively differ at most one in their labels.

For natural numbers m and k, define $\rho(m, k) = \max_i(m^i | k)$ where $q|k$ means that q divides k. In words, $\rho(m, k)$ yields the number of powers of m dividing k. A basic property of this function is the following:

Lemma 2.2
For $p \geq 0$ and $m^p < k < m^{p+1}$, $\rho(m, k) = \rho(m, k - m^p)$.
Proof:
Let $j = \rho(m, k)$, that is, $k = m^j q$, where $m \mid q$. Then $j \leq p$ and $k - m^p = m^j(q - m^{p-j})$. In particular, $\rho(m, k - m^p) \geq \rho(m, k)$. For inequality we should have $m|(q - m^{p-j})$, that is, $p = j$ and $m|q - 1$, because $m \mid q$. But now, remember in this case $p = j$, $m < q$ contradicts the assumption $k < m^{p+1}$. □

Lemma 2.3
Let v_1 and v_2 be leaves so that v_2 follows v_1. Then the leftmost position in which $c_1 = C(v_1, T)$ and $c_2 = C(v_2, T)$ differ is $\rho(m, C(v_2, T))$.
Proof:
Let v_1 and v_2 be leaves so that v_2 follows v_1. Let $c_1 = C(v_1, T)$ and $c_2 = C(v_2, T)$ and let $v = \text{lca}(v_1, v_2)$ and $w = C(v, T)$. Now

$$c_1 = wi\overbrace{(m-1)\ldots(m-1)}^{p} \text{ and } c_2 = w(i+1)\overbrace{0\ldots0}^{p}$$

for some p and i. This follows because for v_2 to follow v_1, v_2 is the leftmost child in its subtree (of r) and v_1 is the rightmost child in its subtree (of r).

Note that the first position in which c_1 and c_2 differ is the pth. Because c_2 ends in p zeroes, $m^p|c_2$ and, because $i + 1 > 0$, also $m^{p+1} \not| c_2$. Hence $p = \rho(m, C(v_2, T))$. □

Let T be a tree of the kind described above. The *mirror* of T, denoted by $\overleftarrow{T}$, is the tree where, for each internal vertex, the label of child i is exchanged with the label of child $m - 1 - i$, for $i = 0, \ldots, \lfloor m/2 \rfloor$, where $\lfloor a \rfloor$ is the largest integer less

than or equal to a. Note that only the labels are changed and for the rest the tree stays intact. Also note that $\overleftarrow{\overleftarrow{T}} = T$.

We shall now define recursively the type of trees we are interested in. The tree T_0^m is equal to the trivial tree; the tree T_n^m is the tree consisting of a vertex, say v, with m subtrees $T_{n-1}^m, \overleftarrow{T_{n-1}^m}, T_{n-1}^m, \overleftarrow{T_{n-1}^m}, \cdots$ ordered from left to right. The edge to child i of v, for $i \in M$, is labelled with i. In Figure 1 this construction is pictorially represented, where $T = T_{n-1}^m$ if m is odd and $T = \overleftarrow{T_{n-1}^m}$ if m is even.

Given a complete m-ary tree of height n there is also a non-recursive way to obtain the tree T_n^m making it easy to recognize whether the tree is correctly constructed: between each pair of levels the edges are labelled from left to right

$$0, 1, \ldots, m-1, m-1, m-2, \ldots, 0, 0, 1, \ldots$$

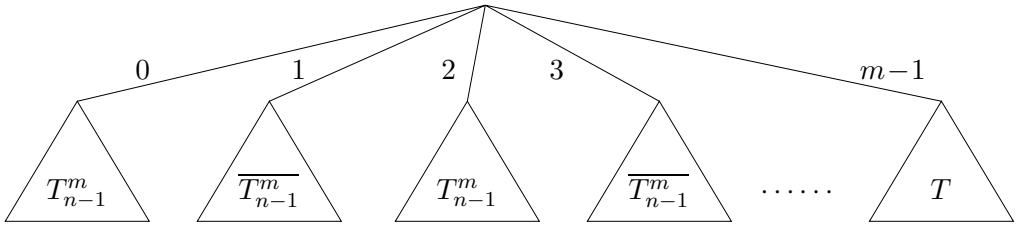

Figure 1: The tree T_n^m schematically

Lemma 2.4
Let $T = T_n^m$ for natural numbers m, n and let $v_1, v_2 \in V(T)$ be such that v_2 follows v_1. Then $\ell_1 = L(v_1, T)$ and $\ell_2 = L(v_2, T)$ differ only in position $p = \rho(m, C(v_2, T))$. Moreover, $\ell_2(p) = \ell_1(p) + 1$ if the number of odd numbers occuring in ℓ_1 to the left of p is even, and $\ell_2(p) = \ell_1(p) - 1$ otherwise.

Proof:
Let $v = \mathrm{lca}(v_1, v_2)$. From the root to v there are obviously no differences between $L(v_1, T)$ and $L(v_2, T)$. Because the path splits up at v and the edges to its children are labelled with different elements of M, the sequences differ in this position. From then on the paths are the same: because v_2 follows v_1 the subtrees of v to which they belong are mirrors of each other and in these subtrees, v_1 and v_2 are rightmost and leftmost vertex of their respective subtrees.

The first position at which $C(v_1, T)$ and $C(v_2, T)$, and $L(v_1, T)$ and $L(v_2, T)$ differ obviously coincide and so $p = \rho(m, C(v_2, T))$ follows from Lemma 2.3.

The last claim follows from the fact that every odd number on the path to the least common ancestor implies a mirroring of the subtree. If this number is even, then an even number of mirror operations yields a tree where the labels to the children are in the original ascending order; otherwise they are in descending order. □

The result of this section can be summed up as follows

Corollary 2.5
Let m, n be integers. A list of all submultisets of $S(m,n)$, say $S_0 = \emptyset, S_1, \ldots, S_{m^n-1}$, can be constructed so that S_i and S_{i+1} differ only in the multiplicity of v_p where $p = \rho(m, i+1)$. Also, $S_{i+1}(v_p) = S_i(v_p) + 1$ if $\sum_{k=p+1}^{n-1} S_i(k)$ is even and $S_{i+1}(v_p) = S_i(v_p) - 1$ otherwise.

Specializing for $m = 2$ we obtain

Corollary 2.6
For an integer n, a list of all subsets of $\{v_0, \ldots, v_{n-1}\}$, say $S_0 = \emptyset, S_1, \ldots, S_{2^n-1}$, can be constructed so that $S_i \ominus S_{i+1} = \{v_p\}$ for $0 \leq i \leq 2^n - 2$ where $p = \rho(2, i+1) = \max_j(2^j | (i+1))$.

The corollaries also imply an algorithm which in the case of Corollary 2.6 is an algorithm to construct the Gray Code for n bits.

3 The computation of the switches of a graph

For a graph $G = (V, E)$ and a function $\sigma : V \to \mathbf{Z}_2$ (called a *selector*) the *switch* of G by σ is defined as the graph $G^\sigma = (V, E')$, where for each $uv \in E(V)$ with exactly one of u and v in σ, we add uv to E if $uv \notin E$, and we remove uv from E if $uv \in E$. The set

$$[G] = \{G^\sigma \mid \sigma \subseteq V\}$$

is called the *switching class* of G.

It is easy to see that switching is a reflexive, symmetric and transitive operation. Hence a switching class is an equivalence class of graphs.

The obvious way to generate all graphs in the switching class of a graph G on $V = \{v_0, \ldots, v_{n-1}\}$ is to switch with respect to all selectors $\sigma \subseteq V$ that do not contain a fixed vertex, say v_{n-1}. The need for omitting v_{n-1} comes from the fact that $G^\sigma = G^{V-\sigma}$. We get an improvement if we apply $V - \sigma$ if $|\sigma| > n/2$. Notwithstanding this improvement, on average $\mathcal{O}(n^2)$ edges must change. To generate the entire switching class we need time $\mathcal{O}(n^2 2^{n-1})$.

The results of the previous section allow us to apply a singleton selector every time and still obtain all possible switches of G exactly once. This method is graphically depicted by the solid lines in Figure 2. The original method of switching is also graphically present in this picture: take the dotted edges, and the solid edge from G to G_1.

The index of the vertex to be switched can be determined using Corollary 2.6 for the set $\{v_0, \ldots, v_{n-2}\}$. Notice that we can in fact return to the original graph by switching with respect to v_{n-2} at the end. This implies that we do not need to make a copy of G before starting to switch. This holds in general if m (in T_n^m) is even: the leftmost and rightmost path in the tree T_n^m for any $n > 0$ differ only in the edges from the root.

Note that we can formulate our result as follows: define a graph where the vertices are the graphs in a certain switching class, where two graphs are adjacent if they can be switched to each other by a singleton selector. The results obtained

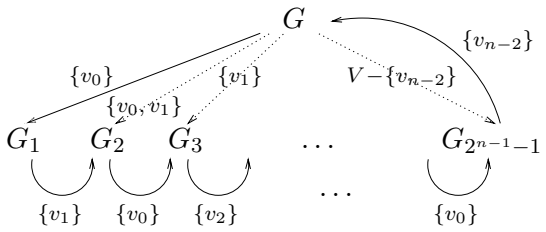

Figure 2: Cumulative switching

so far imply that this graph has a hamiltonian path – if m is even, a hamiltonian cycle.

The overall effect of the improvement of switching singleton selectors instead of arbitrary ones is to reduce the number of edges to be changed in each switch to $n-1$, yielding an average and worst-case complexity of $\mathcal{O}(n2^{n-1})$ instead of $\mathcal{O}(n^2 2^{n-1})$. It should be clear that this is optimal: every switch modifies at least $n-1$ edges. However, we still need an efficient way to compute $\rho(2,k)$. This can be done by looking for the first bit (from the right) that is set in the binary representation of k. The loop that does this is bounded by $\mathcal{O}(\log(k))$.

Some experimental results were obtained running on a computer with an Intel Pentium I 120Mhz running under Linux. The measurements were obtained using the program `gprof 2.9.1`. It excludes the time spent in `mcount`, which is time spent on profiling. In the row "simple" we give timings for a switching algorithm that simply applies half the selectors to a fixed graph in the switching class, while the optimized version uses the just described algorithm for sequentially generating all switches by applying singleton selectors consecutively. The timings for the optimized version are given in the row "cumulative".

Times spent in the actual function that does the switching (in seconds):

n→	8	9	10	11
simple	0.16	3.54	136.21	9609.93
cumulative	0.05	0.75	24.41	6739.49

4 Various generalizations

Although we will not give an example here, the above can also be used when listing selectors mapping the vertices into some carrier set Γ (assume for simplicity that $\Gamma = \{0,\ldots,m-1\}$). Essentially what is needed is to use T_n^m to compute the sequence of selectors from left to right in the tree. The algorithm for computing the vertex whose value is to change is computationally more involved, because instead of finding the number of 2-divisors we have to obtain the number of m divisors. (See Hage [3] for an extensive example.)

Another main difference is that instead of alternating between 0 and 1 as the selected values, we increment a certain element from 0 to $m-1$ and then go back to 0 and so on. The only extra information to keep for each level of the tree is whether

we are currently ascending to $m-1$ or descending to 0.

Some remarks on the general problem of sequentializing every possible multiset are now in order.

For a multiset S over $V = \{v_0, \ldots, v_n\}$ we have for each element v_i a cardinality $k_i = S(v_i)$. For the multisets $S(m,n)$ we have $k_i = k_j$ for all $0 \leq i, j \leq n$.

The generalization to arbitrary arities for different elements, is that nodes on the same level of the tree still have the same arity, but now nodes on different levels may have different arities. The difference between the two algorithms is not large: instead of incrementing on each level to $m-1$ we increment on level j to k_{n+1-j} before going back to zero.

Example 4.1 The situation in illustrated in Figure 3. Here $n = 2$ and we find that on level 1 we increment from 0 to $k_{n+1-1} = k_2 = 2$. Also $k_1 = k_0 = 1$, and we can clearly see the alternation of ascending and descending on those levels. For instance, the path 110 is followed by 111 (both made bold in the picture), because, by Corollary 2.5, the path leading up to their common ancestor contains an even number of odd numbers. ◇

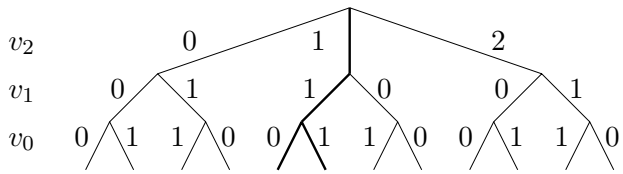

Figure 3: The general picture for the multiset $\{v_0, v_1, v_2, v_2\}$.

Although the method has only been applied to generating switching classes by the author, it is obvious that the method can be applied wherever one has to do something "for all submultisets".

Let $\mathcal{P}(S)$ be the set of submultisets of S. Let $f : \mathcal{G} \times \mathcal{P}(V) \to \mathcal{G}$ be a function that is *transitive*, i.e., $f(f(G,T), U-T)) = f(G,U)$. Depending on the dependance of the complexity of f on the size of the subset we can decide to use either of the two. The method described above was employed in the example of switching classes, because the complexity of switching is quadratic in the size of the subset. In case f is linear it does not really matter, if f is sublinear, then using U directly is advised. Also take into account that especially in the case of multisets (that are not sets), computing the element of which the cardinality should be changed, may take more time than is warranted.

Acknowledgements

We thank Tero Harju and A.M. Hinz for suggestions and improvements in writing this paper. The seed for the paper was sown during my visit to TUCS, Finland in 1998.

References

[1] A. Ehrenfeucht, T. Harju, and G. Rozenberg. *The Theory of 2-Structures*. World Scientific, 1999.

[2] F. Gray. Pulse code communication, Mar. 17 1953. U.S. patent no. 2,632,058.

[3] J. Hage. *Structural Aspects Of Switching Classes*. PhD thesis, LIACS, 2001.

[4] A.M. Hinz. The Towers of Hanoi. *Enseign. Math.*, 2:289–321, 1989.

[5] D. Wood. The Towers of Brahma and Hanoi revisited. *J. Recreational Math.*, 14:17–24, 1981-82.