Generating All k-Subsets of $\{1 \ldots n\}$ with Minimal Changes

T. A. Jenkyns, Department of Mathematics
D. McCarthy, Dept. of Computer Science

Brock University
St. Catharines, Ontario
Canada L2S 3A1

Abstract. An algorithm is given to generate all k-subsets of $\{1 \ldots n\}$ as increasing sequences, in an order so that going from one sequence to the next, exactly one entry is changed by at most 2.

1. Introduction

Throughout this paper, n and k are assumed to be integers for which $0 < k \leq n$.

Every k-subset of $\{1 \ldots n\}$ can be written in natural order as an increasing k-sequence. These sequences can be generated in lexicographic order very easily [10,13] and efficiently [1]. However for certain applications it is particularly convenient to generate the subsets in an order so that the next one differs from the previous one by a minor change.

A variety of algorithms have been developed to generate the subsets in lists satisfying a variety of refinements [3] of the notion of a "minor change":

WMCP	(Weak Minimal Change Property) The next subset is obtained from the previous one by removing a single element, x, and replacing it by a new one, y [2,9,11,12].
SMCP	(Strong Minimal Change Property) The next increasing sequence is obtained from the previous one by changing exactly one entry [4,6]. SMCP implies WMCP.
VSMCP	(Very Strong Minimal Change Property) WMCP where $\lvert x - y\rvert = 1$ [7]. VSMCP implies SMCP.

In the lexicographic list of 3-subsets, $\{1, n-1, n\}$ is followed by $\{2, 3, 4\}$ so unless $n = 4$ this would constitute a major change. In [7] it is shown that lists with VSMCP only exist in special cases: k may be $1, n-1$, or n, but if $1 < k < n-1$ then n must be even and k must be odd.

The recent algorithm by Philip Chase [5] generates all k-subsets in a list satisfying what might be called:

AVSMCP	(Almost Very Strong Minimal Change Property) WMCP where $\lvert x - y\rvert \leq 2$.

But the algorithm very frequently alters two entries in the increasing sequences so does not satisfy SMCP.

The purpose of this paper is to demonstrate the existence of a list satisfying both SMCP and AVSMCP and to give a new and efficient algorithm for generating such a list for any n and k. A "strong and almost very strong" minimal change will be referred to simply as a minimal change.

2. Existence of a Minimal Change List

We shall say two distinct integer sequences differ by a *minimal change* if they are identical except in one entry and the difference in that entry is at most 2. Our objective is to construct a list of all $M = \binom{n}{k}$ increasing k-sequences on $\{1 \ldots n\}$ as $S = (S_1, S_2, \ldots, S_M)$ where, for $1 \le i < M$, S_{i+1} differs from S_i by a minimal change; we will call S a *minimal change list.*

Let $S = (S_1, S_2, \ldots, S_M)$ be any list of the increasing k-sequences on $\{1 \ldots n\}$. If $T = (T_1, T_2, \ldots, T_m)$ is any permutation of $\{k, k+1, \ldots, n\}$ then each entry in T except the last has a unique successor in T, and we shall say S *follows* T if and only if whenever S_{i+1}, differs from S_i in position k then the last entry in S_{i+1} is the successor in T of the last entry in S_i.

We shall say a permutation $T = (T_1, T_2, \ldots, T_p)$ of $\{j, j+1, \ldots, m\}$ is *suitable* if:

(1) $|T_i - T_{i+1}| \le 2$ for $i = 1, 2, \ldots, p-1$; and
(2) if $1 \le q < r < s \le p$ then either $T_q > T_r$ or $T_r < T_s$.

Condition (1) asserts that successive entries in T differ by at most two, and condition (2) implies that T is "concave up" so there are never three consecutive entries where $T_i < T_{i+1} > T_{i+2}$.

For example, if $m = 9$ and $j = 4$ then $p = m - j + 1 = 6$ and the following permutations of $\{4, 5, \ldots, 9\}$ are suitable:

4	5	6	7	8	9
5	4	6	7	8	9
6	4	5	7	8	9
9	7	5	4	6	8

It is easy to see that:

(3) T^R, T written in reverse order, is also a suitable permutation;
(4) the extreme value, m, must occur at one end of T or the other;
(5) if the end of T which equals m is removed, the remaining sequence T' is a suitable permutation of $\{j, j+1, \ldots, m-1\}$;
(6) T is completely determined by its two end values and j, its minimum value.

Theorem. *Given any suitable permutation T of $\{k, k+1, \ldots, n\}$ of length $m = n - k + 1$, there is a minimal change list $S = (S_1, S_2, \ldots, S_M)$ which:*

(a) *follows T*

(b) *starts with* $S_1 = (T_1 - k + 1, \ldots, T_1 - 2, T_1 - 1, T_1)$ *and*
(c) *ends with* $S_M = (T_m - k + 1, \ldots, T_m - 2, T_m - 1, T_m)$

For example, if $n = 6$, $k = 3$, and $T = (4,3,5,6)$ then the following is a minimal change list satisfying (a), (b), and (c).

i	S_i		
1	2	3	4
2	1	3	4
3	1	2	4
4	1	2	3
5	1	2	5
6	1	3	5
7	2	3	5
8	2	4	5
9	1	4	5
10	3	4	5
11	3	4	6
12	2	4	6
13	1	4	6
14	1	2	6
15	1	3	6
16	2	3	6
17	2	5	6
18	1	5	6
19	3	5	6
M=20	4	5	6

Proof: We will assume $0 < k \leq n$ and proceed by induction on n. If $n = 1$ then $k = n$ and, in general, when $k = n$ the suitable sequence T must be (n). There is then one k-subset of $\{1,2,\ldots,n\}$ corresponding to the increasing sequence $S_1 = (1,2,\ldots,n)$. But then $S = (S_1)$ satisfies the conditions of the theorem.

The case that $k = 1$ may also be dealt with in general because if S_j is set equal to (T_j) for $j = 1,2,\ldots,n$ the list $S = (S_1, S_2, \ldots, S_n)$ satisfies the conditions of the theorem.

Assume the theorem for all values of n from 1 to $p-1$ and suppose T is a given, suitable sequence on $\{k, k+1, \ldots, n\}$ where $n = p > k > 1$.

Either $T_1 = n$ or $T_m = n$ and we deal with these cases separately.

Suppose $T = (T_1, T_2, \ldots, T_m = n)$. The minimal change list we want will decompose into two "blocks" because all S_j which end in n occur after those which do not end in n.

$$\begin{array}{ll}
\text{j} & S_j \\
1 & (T_1 - k + 1, \ldots, T_1 - 2, T_1 - 1, T_1 \quad) \\
\vdots & \qquad\qquad\qquad \vdots \\
M' & (\qquad\qquad\qquad \vdots \qquad\qquad , T_{m-1}) \\
M' + 1 & (\qquad\qquad\qquad \vdots \qquad\qquad , n \quad) \\
\vdots & \qquad\qquad\qquad \vdots \\
M & (n - k + 1, \ldots, \; n - 2, \; n - 1, n \quad)
\end{array}$$

The first block consists of the S_j which do not end in n, so they do not contain n and correspond to the $M' = \binom{n-1}{k}$ k-subsets of $\{1, 2, \ldots, n-1\}$. Since $T' = (T_1, T_2, \ldots, T_{m-1})$ is a suitable permutation of $\{k, k+1, \ldots, n-1\}$ by our inductive assumption, there is a minimal change list $S' = (S_1, S_2, \ldots, S_{M'})$ which follows T', starts with

$$S_1 = (T_1 - k + 1, \ldots, T_1 - 2, T_1 - 1, T_1)$$

and ends with

$$S_{M'} = (T_{m-1} - k + 1, \ldots, T_{m-1} - 2, T_{m-1} - 1, T_{m-1}).$$

Now what remains is construction of the second block. To begin let

$$S_{M'+1} = (T_{m-1} - k + 1, \ldots, T_{m-1} - 2, T_{m_1} - 1, T_m = n).$$

The change from $S_{M'}$ to $S_{M'+1}$ is then a minimal change.

When $T_m = n$, $n-2 \leq T_{m-1} < n$ so either $T_{m-1} - 1 = n-3$ or $n-2$. In either case there is a unique, suitable permutation T'' of $\{k-1, k, \ldots, n-1\}$ which ends in $(n-1)$ and starts with $T_1'' = T_{m-1} - 1$. By the inductive assumption, there is a minimal change list S'' of the $M'' = \binom{n-1}{k-1}$ increasing $(k-1)$-sequences on $\{1, 2, \ldots, n-1\}$ starting with

$$S_1'' = (T_1'' - k + 2, \ldots, T_1'' - 1, T_1'' = T_{m-1} - 1)$$

and ending with

$$S_{M''}'' = (n - k + 1, \ldots, n - 2, n - 1).$$

To complete the construction of the second block, for $j = 1, 2, \ldots, M''$ let

$$S_{M'+j} = S_j'' \text{ followed by } n \text{ as a last entry.}$$

Then the first block, S', followed by this second block is a minimal change list satisfying all the conditions of the theorem.

Finally we deal with the case when T begins with n. In this case T^R is a suitable sequence which ends with n and by the above argument there is a minimal change list S which

(a) follows T^R

(b) starts with

$$S_1 = (T_1^R - k + 1, \ldots, T_1^R - 1, T_1^R)$$

and (c) ends with

$$S_M = (T_m^R - k + 1, \ldots, T_m^R - 1, T_m^R).$$

But then S^R, S written in reverse order, satisfies the conditions of the theorem. ∎

Conditions (a), (b), and (c) do not determine S uniquely unless they are applied recursively to construct S in "blocks" as described in the proof. The algorithm in the next section does precisely that.

Many variations of the construction are possible, such as making the first entry (rather than the last) of the increasing sequences "follow" some suitable (but concave down) permutation $T = (T_1, T_2, \ldots, T_m)$ of $\{1, 2, \ldots, n-k+1\}$ and specifying that the list (and subsequent blocks) begin with $(T_1, T_1 + 1, T_1 + 2, \ldots, T_1 + k - 1)$ and end with $(T_m, T_m + 1, T_m + 2, \ldots, T_m + k - 1)$. For example when $n = 6$, $k = 3$ and $T = (3, 4, 2, 1)$ we obtain the following minimal change list:

```
345  346  356  456  256  246  245  235  236  234
134  135  136  156  146  145  125  126  124  123
```

Evidently there are many minimal change lists of all k-subsets of $\{1, 2, \ldots, n\}$.

3. An Implementation Algorithm

In the construction, values in $S[k]$ "follow" some predetermined, suitable permutation T of $\{k, k+1, \ldots, n\}$ and are constant in "blocks" until the last increasing k-sequence in the block is encountered. At the end of a block the current value of $S[k]$ is changed to its successor in T, nv.

We remarked earlier that a suitable permutation is completely determined by its first, last, and minimum entries. The function **nextvalue** appearing below has parameters x, *first*, *min*, and *last* as inputs and returns either the successor of x in T or, when $x =$ *last*, returns $x + 1$ or $x + 2$ by default. A default value larger than last is important to the algorithm.

While $S[k]$ is fixed at some value x, all $(k-1)$-subsets of $\{1, \ldots, x-1\}$ are generated in $S[1], \ldots, S[k-1]$ to follow the unique suitable permutation of $\{k-$

$1,\ldots,x-1\}$ with *first* $= S[k-1]$, *min* $= k-1$ and, *last* $=$ *minimum*$\{x, nv\}-1$. These are generated recursively until $x =$ *last*.

The algorithm is given below (without types and declarations) as a pseudo code procedure where n, k, and S are global variables and Process is a procedure which receives the subsets as they are generated for some unspecified purpose.

```
FUNCTION nextvalue ($x$, $first$, $min$, $last$);
BEGIN
    IF $x > first$ THEN                                  RETURN $x + 1$
    ELSIF $x > last + 1$ THEN                            RETURN $x - 1$
    ELSIF  (($first < last$) AND ODD($first - x$) OR
           (($first > last$) AND ODD($last + 1 - x$)) THEN
                                                         RETURN $x + 2$
    ELSIF $x > min + 1$ THEN                             RETURN $x - 2$
    ELSIF $x = min + 1$ THEN                             RETURN $x - 1$
    ELSE                                                 RETURN $x + 1$
    END
END nextvalue;

PROCEDURE GenSubsets($first$, $index$, $last$);
BEGIN
    IF $index = k$ THEN
        FOR $i$ := 1 TO $k$ DO
            $S[i] := first - k + i$
        END;
        Process
    END;
    $nv := first$;
    REPEAT
        $x := nv$; $S[index] := x$;
        IF $x <> first$ THEN Process END;
        $nv$ := nextvalue($x$, $first$, $index$, $last$);
        IF ($index > 1$) AND ($x > index$) THEN
            IF $x < nv$ THEN $newlast := x - 1$
            ELSE $newlast := nv - 1$
            END;
            GenSubsets($S[index - 1]$, $index - 1$, $newlast$)
        END
    UNTIL $x = last$
END GenSubsets;
```

4. Complexity of the Algorithm

Since each complete execution of the procedure results in several subsets being generated and sent to Process, the number of calls of GenSubsets is less than the number of k-subsets. Therefore the amortized running time per subset is $O(1)$ independent of n and k.

The recursion may be removed by "stacking" values of *first*, *last*, and nv in arrays indexed from 1 to k. Also a "next index to alter" array may be constructed for returns from the recursion to produce a non-recursive version. However formulating a version which is "loopless" in the sense of [8] remains an open problem.

References

[1] Akl, S., *A Comparison of Combination Generation Methods*, ACM Trans. Math. Software 7 (1981), 42–45.

[2] Bitner, J.R., Ehrlich, G., and Reingold, E.M., *Efficient Generation of the Binary Reflected Gray Code and Its Applications*, Comm. ACM **19** (Sept. 1976), 517–521.

[3] Carkeet, M. and Eades, P.,, *Performance of Subset Generating Algorithms*, Annals of Discrete Math. 26 (1985), 49–58.

[4] Chase, P.J., *Algorithm 382. Combinations of M out of N Objects*, Comm. ACM **13** (June 1970), 368.

[5] Chase, P.J., *Combination Generation and Graylex Ordering*, Congressus Numerantium 69 (1989), 215–242.

[6] Eades, P. and McKay, B.D., *An Algorithm for Generating Subsets of a Fixed Size with a Strong Minimal Change Property*, Information Processing Letters 19 (1984), 131–133.

[7] Eades, P., Hickey, M., and Read, R.,\ *Some Hamilton Paths and a Minimal Change Algorithm*, J. ACM **31** (1984), 19–29.

[8] Ehrlich, G., *Loopless Algorithms for Generating Permutations Combinations, and Other Combinatorial Configurations*, J. ACM **20, 3** (July 1973), 500–513.

[9] Ehrlich, G., *Algorithm 466. Four Combinatorial Algorithms*, Comm. ACM 16 (Nov. 1973), 690–691.

[10] Kurtzberg, J., *Algorithm 94. Combination*, Conun. ACM 5 (June 1962), 344.

[11] Lam, C.W. and Soicher, L.H., *Three Combination Algorithms with the Minimal Change Property*, Comm. ACM 25 (1982), 555–559.

[12] Liu, C N. and Tang, D.T., *Algorithm 452. Enumerating Combinations of m out of n Objects*, Comm. ACM **16, 8** (Aug. 1973), 485.

[13] Mifsud, C.J., *Algorithm 154. Combination in Lexicographical Order*, Comm. ACM 6 (Mar. 1963), 103.