Abstract—A \( k \)-partial permutation out of \( n \) elements is a \( k \)-tuple \((p_1, p_2, \ldots, p_k)\) with \( k \) distinct elements and \( p_i \in [n] = \{1, 2, \ldots, n\}, i = 1, 2, \ldots, k \). Let \((p_1, p_2, \ldots, p_n)\) be a full permutation of size \( n \), where the elements are distinct and \( p_i \in [n], i = 1, 2, \ldots, n. \) Then we say the \( k \)-partial permutation \((p_1, p_2, \ldots, p_k)\) is induced from the full permutation. A universal cycle on \( k \)-partial permutations is a cycle \((a_1, a_2, \ldots, a_N)\) with \( N = (\binom{n}{k})!\), \( a_i \in [n], i = 1, 2, \ldots, N, \) where each \( k \)-partial permutation appears as a subsequence in this cycle exactly once. The main contribution of the paper is the first explicit construction of universal cycles on \( k \)-partial permutations for arbitrary \( 1 \le k < n \).

I. INTRODUCTION

For a set \( \Sigma \) with \( n \) elements, a permutation of size \( n \) is an \( n \)-tuple \( P = (p_0, \ldots, p_{n-1}) \), where \( p_i \in \Sigma \), and \( p_i \) are all distinct, \( i = 1, 2, \ldots, n \). For simplicity, \( \Sigma \) is taken as the set \([n] = \{1, 2, \ldots, n\}\) in the paper. The indices are ordered decreasingly for convenience of description. Every permutation defines a mapping from \( \Sigma \) to itself, i.e., the \( i \)-th element of \( \Sigma \) is mapped to \( p_i \). All of the permutations of size \( n \) form the symmetric group \( S_n \) with function compositions as operations. In this paper, we are going to use \( S_n \) to represent the set of all permutations of size \( n \). A \( k \)-partial permutation, or \( k \)-permutation, out of \( n \) elements is a \( k \)-tuple \( Q = (p_0, \ldots, p_{k-1}) \) with \( k \) distinct elements in \( \Sigma \), for some \( k \le n - 1 \). We say \( Q \) is induced from \( P \) if the first \( k \) elements of \( P \) are the same as \( Q \).

A de Bruijn sequence is a cyclic sequence of some alphabet \( \Sigma \) where each possible \( k \)-tuple of this alphabet appears as a consecutive subsequence exactly once [1]. The length of a de Bruijn sequence is \(|\Sigma|^k\). For example, if \(|\Sigma| = \{0, 1\}\) and \( k = 3 \), then 00010111 is a de Bruijn sequence. More generally, let \( C \) be a set whose elements are represented by \( k \)-tuples of the alphabet \( \Sigma \). Then a universal cycle on the set \( C \) is a cyclic sequence of length \(|C|\) for which every element in \( C \) corresponds to exactly one subsequence [2].

For universal cycles on \( k \)-partial permutations, \( \Sigma \) is a set of size \( n \), and \( C \) contains all the \( k \)-partial permutations, \( k < n \). For example, \((4, 3, 4, 2, 3, 2, 4, 1, 3, 1, 2, 1)\) is a universal cycle for \( 2 \)-partial permutations out of \( n = 4 \). It contains all the \( 2 \)-partial permutations in its subsequences: \((4, 3), (3, 4), (4, 2), \ldots, (2, 1), (1, 4)\).

A natural question to ask is: is there a universal cycle on partial permutations for any parameters \( k, n \)? We will show in this paper that the answer is yes, and give explicit constructions of such cycles.

Universal cycles were first proposed in [2], where the set \( C \) was considered to be all possible tuples, order-isomorphic permutations, partitions, or subsets. The existence of universal cycles on \( k \)-partial permutations was proved in [3], however, the proof was nonconstructive. An explicit construction of a cycle for \( k = n - 1 \) was given in [4]. But for other values of \( k \), the construction is still an open problem. In this paper, we will provide an algorithm to construct the cycle for all \( k \le n - 1 \). The algorithm is dependent only on the top \( k \) elements of each permutation, and has complexity \( O(k) \).

The problem of order-isomorphic permutations is similar to \( k \)-partial permutations because its elements in \( C \) is also \( k \)-partial permutations. However, two permutations are isomorphic (and therefore considered duplicated) if their relative ordering are the same. Namely, suppose \( Q = (p_k, \ldots, p_2, p_1) \) and \( Q' = (p'_k, \ldots, p'_2, p'_1) \) are isomorphic, then \( p_i < p_j \) if and only if \( p'_i < p'_j \), for \( 1 \le i \neq j \le k \). For example, \((1, 3, 2)\) and \((2, 5, 3)\) are isomorphic permutations. In [5], constructions of universal cycles on order-isomorphic permutations are given for \( n = k + 1 \), and it proves that \( n = k + 1 \) symbols are sufficient for such universal cycles.

Another related problem is the Hamiltonicity of graphs of \( k \)-permutations with \( n - k \) transitions (\( k < n \)), where each \( k \)-permutation corresponds to a vertex and each \( k + 1 \)-permutation corresponds to an directed edge. Namely, it is a universal cycle with distinctive elements in any consecutive \( k + 1 \) subsequences. In [6], Hamiltonicity was shown for \( k = 2 \). And in [7], Hamilton cycles are proved for all \( n \) and \( k \le n - 3 \), and for \( n \ge 4, k = n - 2 \), the graph is proved to be not Hamiltonian.

A universal cycle on partial permutations defines a Gray code, namely, a sequence of partial permutations, such that the transition between two adjacent permutations belongs to a set of valid transitions. Suppose there is a set of transitions (or functions) defined on \( S_n \), then we can draw a directed graph with all the permutations as vertices and all the transitions as edges.
Denote this graph by $\mathcal{G}$. See Figure 1 (a) for an example. We would like to know if there a cycle in $\mathcal{G}$ whose vertices induce all $k$-partial permutations. In this paper, we confine ourselves to the push-to-the-top transitions $t_i$, $i = k, k+1, \ldots, n$: for permutation $P = (p_1, p_2, \ldots, p_n)$, the $i$-th element becomes the first. That is, $t_i(P) = (p_i, p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)$. Through out the paper, the left-hand side is considered as top, and the left side is bottom. Thus, in the graph $\mathcal{G}$, each permutation $P$ has $n-k+1$ outgoing edges, $t_k(P), t_{k+1}(P), \ldots, t_n(P)$. If this cycle exits, we call it a Gray code on $k$-partial permutations. For example, $(1, 2, 3), (3, 1, 2), (2, 3, 1)$ is a Gray code for $k = 1$, $n = 3$. When $k = n-1$, the cycle consists of a Gray code on full permutations.

The choice of push-to-the-top transitions relates the Gray codes to universal cycles on partial permutations. In this paper, we are going to show that the Gray code described above is equivalent to the universal cycle on $k$-partial permutations.

We remark here that the Gray code on $k$-partial permutations can be applied to the recently proposed rank modulation schemes for flash memories [8]. Rank modulation is an information representation scheme in which permutations, instead of the normally used absolute integer values, are used to denote information. In [8], Gray code for the full permutations was constructed ($k = n-1$), with push-to-the-top transitions $t_i$, $i = 2, 3, \ldots, n$. The Gray code can be used as a permutation counter that traverses $0, 1, \ldots, n!-1$, and by combining the counter with other coding techniques (e.g., floating code [9][10]), we are able to represent arbitrary information efficiently. In multi-level flash memories, the programming (or writing) process is asymmetric: it is easy to increase the cell levels of flash memory, but hard to lower them. Due to the physical structure of flash devices, one has to erase a block of cells (typically $10^9$ cells) in order to decrease one cell level [11]. Moreover, leakage of a group of cells causes the decrease of a lot of cells at the same time. Therefore, it is hard to use absolute values and require exact cell levels for flash memory. Rank modulation and push-to-the-top transitions solve the above problems because during writing, we can first write the lowest cell of a permutation, and then push the second lowest cell to the top of all current levels, and then write the third lowest cell, and so on. There is no need to lower a cell level unless we reach the maximum possible level. In addition, the leakage among a group of cells will not change the information as long as the relative values stay the same.

We propose using $k$-partial permutations out of $n$ cells to represent information in flash memory. It is a good candidate of coding schemes for flash memory because it preserves the advantages of rank modulation, and the sorting of $k$ cells is less complex than $n$ cells ($k \log k$ complexity for $k$ cells). Also, the unused $n-k$ cells provide redundancy and can be used to correct errors. In this paper, we focus on construction of Gray codes for $k$-partial permutations, for all $k < n$. Recently, Gray code for local rank modulation, another variation of rank modulation, was raised in [12][13]. And in local rank modulation, $k$-permutations with overlaps are extracted from a sequence of $m > k$ cells.

The paper is organized as follows. In Section II, definitions and notations are introduced. Construction of the universal cycles and Gray code on $k$-permutations are discussed in Section V. And finally, in Section VI we make conclusions.

II. DEFINITIONS AND NOTATIONS

A $k$-partial permutation, or $k$-permutation, out of $n$ element is a $k$-tuple $Q = (p_1, p_2, \ldots, p_k)$ with distinct $k$ elements of the set $[n] = \{1, 2, \ldots, n\}$. The number of possible $k$-partial permutations is $\binom{n}{k}k!$. The partial permutation is said to be induced from the permutation $P = (p_1, p_2, \ldots, p_n)$, for any ordering of $p_{k+1}, p_{k+2}, \ldots, p_n$, and the permutation $P$ is said to be a realization of $Q$. Note that there are multiple realizations for each partial permutation when $k < n - 1$. Sometimes we will denote this permutation by $P = (p_1, p_2, \ldots, p_k | p_{k+1}, p_{k+2}, \ldots, p_n)$ to emphasize the first $k$ elements. When $n, k$ are know in the context, we usually refer to the bottom elements in $P$ as the bottom $n-k$ elements, and the top elements as the top $k$ elements.

An $(n, k)$ universal cycle of $k$-partial permutations is a sequence $A = (a_1, a_2, \ldots, a_N)$, $N = \binom{n}{k}k!$, $a_i \in \{1, 2, \ldots, n\}$, such that each $k$-partial permutation...
out of $n$ elements is represented by exactly one subsequence $(a_{i+1}, a_{i+2}, \ldots, a_{i+k})$ for some $1 \leq i \leq N$. In this section, the index additions are computed modulo $N$. For example, if $n = 4$, $k = 2$, the sequence $A = (4, 3, 4, 2, 3, 2, 4, 1, 3, 1, 2, 1)$ has subsequences $(a_1, a_2) = (4, 3), (a_2, a_3) = (3, 4), \ldots, (a_{12}, a_1) = (1, 4)$. It can be seen that every $k$-partial permutation is included in this sequence. Thus, it is a universal cycle for $n = 4$ and $k = 2$. The question is, does such a cycle exist for all $n, k$? If yes, how to find this cycle? We will see in the paper that the answer is yes, and there are indeed a large number of such cycles. Moreover, we will give constructions of universal cycles.

Another way to view a universal cycle is to break it up to a sequence of $k$-permutations, $(a_1, a_2, \ldots, a_k), (a_2, a_3, \ldots, a_{k+1}), \ldots, (a_N, a_1, a_2, \ldots, a_k)$:

For the purpose of description, we write each permutation backwards. So a universal cycle is equivalent to a cycle of $k$-permutations $(a_k, a_1, a_2, a_3), (a_{k+1}, a_2, a_3, a_4), \ldots, (a_{N-1}, a_1, a_2, \ldots, a_k)$ such that each partial permutation appear exactly once. Hence in this paper we focus on constructing such a sequence of partial permutations.

For any $k$-permutation in the universal cycle, we can see that there are only $n - k + 1$ possible permutations following it. We define the transition from one full permutation to the next as follows. For $k \leq i \leq n$, define mapping $t_i: S_n \rightarrow S_n$, where $t_i(p_1, p_2, \ldots, p_n) = (p_i, p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n)$. $t_i$ is called a push-to-the-top transition. By abuse of notation, $t_i$ can be defined on $k$-permutations, where we expand $Q = (p_1, p_2, \ldots, p_k)$ to a full permutation $P = (p_1, p_2, \ldots, p_n)$ with $p_n > p_{n-1} > \cdots > p_{k+1}$. And define $t_i(p_1, p_2, \ldots, p_k) = (p_i, p_1, p_2, \ldots, p_{k-1})$ if $k \leq i \leq n$.

Therefore, a universal cycle is equivalent to the following graph interpretation. Consider a directed graph $G$, with vertices corresponding to $k$-permutations and edges corresponding to $t_i$ defined on $k$-permutations. Again, assume that $k \leq i \leq n$. Figure 1 (b) shows an example of such graphs. Then a Hamiltonian cycle in this graph is a universal cycle.

Typically, we assume $2 \leq k < n$ because when $k = 1$, construction of universal cycles is trivial (we can do it in $n$ times).

**Example 1.** Figure 2(a) shows a $(4, 2)$ Gray code. Only the top two digits in each permutation are of concern. And the push-to-the-top transitions are indicated above each edge. Note that right-hand side is considered top.

We next define a partition on partial permutations, which will help us to construct the universal cycle. Define the partition by the relation $\sim$, where $P_1 \sim P_2$ if the first $k$ elements of $P_2$ is a cyclic shift (to the left) of those of $P_1$. For example, if $k = 3$, $n = 6$, $E = \{(1, 3, 6), (6, 1, 3), (3, 6, 1)\}$ is an equivalence class. $\sim$ is obviously an equivalence relation. Let $E$ be an equivalence class, if $P = (p_1, p_2, \ldots, p_k) \in E$, and $p_k = \max\{p_1, p_2, \ldots, p_k\}$, then we choose $P$ as the representative of the equivalence class $E$ (unless mentioned otherwise). We will denote $E = (p_1, p_2, \ldots, p_k)$. In the previous example, $(1, 3, 6)$ is the representative and we will write $E = (1, 3, 6)$. Each equivalence class has $k$ members and there are $\binom{n}{k}(k-1)!$ equivalence classes.

**III. Construction of Universal Cycles**

As mentioned in Section I, no explicit construction of universal cycles is known for $k < n - 1$, and we will give such a construction for all $k \leq n - 1$.

The universal cycles are constructed in several steps. We first construct a tree that contains all the equivalence classes, and then use this tree to generate all the partial permutations. Also, the construction of this tree is reduced to: (1) generating sub-trees containing all the combinations of $k$ elements out of $n$, and (2) finding a sequence of permutations on $k - 1$ elements to connect the sub-trees.

In the following, we will represent a universal cycle by its subsequences: $Q_1, Q_2, \ldots, Q_N$, where each $Q_i$ is a $k$-permutation, and $N = \binom{n}{k}!$. The transitions between the $k$-permutations are push-to-the-top operations $t_i, k \leq i \leq n$, or equivalently, push the $i$-th element $p_i$.

Notice that starting from any partial permutation, if we do $(k - 1)$ times the transition $t_k$, we can traverse an equivalence class. If we start from some partial permutation in equivalence class $E_1$ and apply operations

$$t_k, t_{k+1}, t_k, \ldots, t_k, t_{k+1}, t_k, \ldots, t_k, t_{k+1}, \ldots \quad (1)$$

for $0 \leq a \leq k - 2$ and $i > k$, we can still traverse all members of $E_1$. And if the partial permutation after $t_i$ belongs to $E_2$, the above operations also traverse $E_2$. In another word, inserting $E_2$ does not affect the completion of $E_1$. We call the above operation insertion. See Figure 3 for an example. In addition, if $a = k - 1$ in the above operation sequence, we omit the last operations $t_{k+1}$ and $(k - 2 - a)$ times of $t_k$. And we traverse $E_1$, followed by traversing $E_2$. We consider this case as insertion, too. The permutations circled by dashed line in Figure 2(a) also illustrates this notion, where $E_1 = (2, 4)$ is inserted by $E_2 = (2, 3)$.

We will draw insertions in a directed tree. The root of the tree is the unique node with in-going degree 0, and an edge starts from the parent and ends at the child. The nodes of the tree are equivalence classes. An edge means the child is inserted in the parent. Since transition $t_i$ in 2 only changes one digit in the partial permutation, the parent and the child differ in only one digit for their representatives (after cyclic shift). We can insert at most $k$ children to a parent, each changing one of its $k$ digits. In a word, define an insertion tree to be a tree with equivalence classes as nodes, with $k$ children at most for each parent, each changing one different digit of their
Let the root be a tree with depth of \( k \). The sub-sequence in the square brackets corresponds to the sub-tree. If \( i = 1 \), the part after the square brackets is omitted, and the last permutation is \((p_2, \ldots, p_n, p_1')\). If \( i \neq 1 \), the last permutation is \((p_2, \ldots, p_k, p_1)\). In both cases, this sequence goes through the equivalence classes in the sub-tree and the root completely and it is indeed a cycle. Similarly, we can insert other sub-trees into this cycle in the same manner. As each child changes a different digit of the root, these sub-trees will not affect each other. So the claim is true.

Now as a generating tree contains each equivalence class exactly once, we have a cycle with every partial permutation exactly once.

The above theorem is similar to the cycle merging technique in [5], which was used to generate order-isomorphic permutations.

By the proof, if we start from some node \( m \) of the insertion tree, we can construct a cycle that traverse the sub-tree rooted at \( m \) in the following manner: (1) pass some of the \( k \)-partial permutations belonging to \( m \) first, and (2) traverse one of the sub-trees that is rooted at a child of \( m \), and repeat (1) and (2) until the \( k \)-partial permutations belonging to \( m \) is traversed.

This procedure works like a stack. That is, when we go to the current node \( m \), put it in the stack. And then pass some of its corresponding partial permutations. If every element in the equivalence class of \( m \) is passed through, remove \( m \) from the stack. Next, go to a child of \( m \), and treat this child as the current node. Now it has been traversed, and has no children, then visit a node from the stack. One cycle ends when the stack is empty, and we can start again by pushing the first element of the first permutation to the top. For example, in Figure 2(a), the stack after each partial permutation is written in the lower dashed box.

Now the problem of universal cycles for partial permutations is reduced to constructing generating trees.

Before going through constructions, let us think about whether such generating trees exist. Consider an insertion tree of \( k \) out of \( n \) in Figure 4. In this figure, the first digit in the representative of an equivalence class does not have to be the largest. Each rectangular represents a sub-tree with all equivalence classes that have one digit fixed.
as some integer, and possibly duplicates occur in each rectangular. An X in the graph represents a non-fixed digit. Given a tree of all equivalence classes of \((k-1)\) out of \(n\), we can add the fixed integer at the fixed position in each node, and connection rules for insertion trees are not violated. Thus we obtain a sub-tree in the rectangular.

By induction on \(k\) for fixed \(n\), these sub-trees exist. The equivalence classes \((1, 2, \ldots, k-1, k+1), (1, 2, \ldots, k-1, k+2), \ldots, (1, 2, \ldots, k-1, n)\) are nodes in the sub-tree \(1X \ldots X\), and each of these nodes leads to a new sub-tree.

Obviously the tree in Figure 4 contains every equivalence class, but there are some duplicates. Starting from bottom to up, if there are two identical class \(E_1 = E_2\) in this tree, we can remove \(E_2\) and move all its descendent under \(E_1\) (treat them as the descendants of \(E_1\)). If thus \(E_1\) has 2 children that change the same digit of \(E_1\), then those two children differ in only one digit. Thus we can move one child and its descendent under the other. Repeat this procedure until all descendants of \(E_1\) have valid children. Afterwards, remove the next duplicate, and so on. Therefore, generating trees exist, and as we can imagine, there are many such trees. A proof of existence of universal cycles using graph theory can be found in [3].

We will give a special class of generating trees based on combinations, where the elements are ordered increasingly. Let \((c_1, c_2, \ldots, c_n)\) and \((d_1, d_2, \ldots, d_n)\) be vectors with distinct values. Then they are order-isomorphic if for all \(1 \leq i < j \leq n\), \(c_i < c_j\) if and only if \(d_i < d_j\). Suppose we have an insertion tree \(T_1\) that contains each combination of \(k\) out of \(n\) once, and the nodes in \(T_1\) are all ordered increasingly. Namely, each node \((p_1, p_2, \ldots, p_k)\) in \(T_1\) satisfies \(p_1 < p_2 < \cdots < p_k\). Suppose the root of the tree is \((n-k+1, n-k+2, \ldots, n)\). Then we will construct trees \(T_2, T_3, \ldots\) such that they have similar structure as \(T_1\), but the nodes do not have increasing order. Moreover, these trees will be connected by permutations \(\sigma_1, \sigma_2, \sigma_3, \ldots\) on \([k-1]\), where \(\sigma_1\) is fixed to be the identity permutation. Obviously, all nodes in \(T_1\) are order-isomorphic to \((\sigma_1, k) = (1, \ldots, k-1, k)\).

For an integer \(x\) and a permutation \(\sigma\) on \([k-1]\), write \(\sigma + x = (\sigma(1) + x, \sigma(2) + x, \ldots, \sigma(k-1) + x)\). If we can transit from a node in \(T_1\) to the node \((\sigma_2 + (n - k), n)\) and does not violate the rule of an insertion tree (a child changes one digit from the parent and each child changes a different digit), then we can use this as the root for a new tree \(T_2\). And by the same structure as \(T_1\), except that we change each \((p_1, p_2, \ldots, p_k)\) in \(T_1\) to \((p_{\sigma_2(1)}, p_{\sigma_2(2)}, \ldots, p_{\sigma_2(k-1)}, p_k)\) in \(T_2\), we get another insertion tree \(T_2\) and each node in \(T_2\) is order-isomorphic to \((\sigma_2, k)\). After that, we transit from \(T_2\) to \(T_3\) with permutation \(\sigma_3\), and each node in \(T_1\) is replaced by \((p_{\sigma_3(1)}, p_{\sigma_3(2)}, \ldots, p_{\sigma_3(k-1)}, p_k)\), and so on.

For example, in Figure 5 \((n = 6, k = 3)\), the sub-tree on the left (in dashed lines) is tree \(T_1\), and we transit from \(T_1\) to node \((5, 4, 6)\). Hence \(\sigma_1 = (1, 2)\) and \(\sigma_2 = (2, 1)\).

The node in \(T_1\), for instance, \((p_1, p_2, p_3) = (2, 4, 6)\), is replaced with \((p_{\sigma_1(1)}, p_{\sigma_1(2)}, p_{\sigma_1(3)}) = (p_2, p_1, p_3) = (4, 2, 6)\). Each node in \(T_2\) is isomorphic to \((\sigma_2, k) = (2, 1, 3)\).

We have the following theorem.

**Theorem 3.** If there is an insertion tree \(T_1\) that contains every combination of \(k\) out of \(n\) once, and a sequence of valid transitions from previous trees (as described above), \(\sigma_1, \sigma_2, \ldots, \sigma(k-1)\), traversing all permutations on \([k-1]\), then \(T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_{(k-1)}\) is a generating tree of \(k\)-partial permutations.

**Proof:** We first show that this three is an insertion tree. Since the transition within \(T_i\) and from one sub-tree to the next is assumed to be valid, we need to show each child in \(T_i\) changes only one digit of the parent and changes a different digit, for \(i \geq 2\). However, the nodes in \(T_i\) and \(T_1\) are identical except the ordering of their elements are permuted by \(\sigma_i\), so \(T_i\) is also a valid insertion tree.

Next we show that the tree contains every equivalence class exactly once. Recall that we represent the equivalence classes with \((p_1, \ldots, p_k)\) where \(p_k = \max\{p_1, \ldots, p_k\}\). For any \(p_1 < p_2 < \cdots < p_k\), and any equivalence class \((p_{\sigma(1)}, p_{\sigma(2)}, \ldots, p_{\sigma(k-1)}, p_k)\), there exists exactly one \(1 \leq i \leq (k-1)!\), such that \(\sigma = \sigma_i\) and this equivalence class appears exactly once in the tree \(T_i\).

Now the problem of finding universal cycles for partial permutations reduces to finding \(T_1\) and \(\sigma_1, \sigma_2, \ldots, \sigma(k-1)\).

**Construction 4.** (Construction of \(T_1\))

The root \((n-k+1, n-k+2, \ldots, n)\) has one child \((n-k, n-k+2, \ldots, n)\), and the node \((1, 2, \ldots, k)\) has a parent \((1, 2, \ldots, k-1, k+1)\) and no children. Starting from the root, and following the connection rules below for all the other nodes, we will get a \(T_1\).

1) A node \((p_1, p_2, \ldots, p_{k-n+t-1}, t, t+1, \ldots, n)\), for \(1 \leq p_{k-n+t-1} < t-1\) and \(n-k+2 \leq t \leq n\), is
connected to the at most 3 nodes:

Parent \((p_1, p_2, \ldots, p_{k+1}−1, t, t + 1, \ldots, n)\)

Child \(_1 (p_1, p_2, \ldots, p_{k+1}−1, 1, t, t + 1, \ldots, n)\) if \(p_{k−n+t−1}−1 > p_{k−n+t−2} \geq 1\)

Child \(_2 (p_1, p_2, \ldots, p_{k+1}−1, t−1, t + 1, \ldots, n)\)

2) Otherwise, a node \((p_1, p_2, \ldots, p_k)\) is connected to at most two nodes:

Parent \((p_1, p_2, \ldots, p_k + 1)\)

Child \((p_1, p_2, \ldots, p_k−1)\) if \(p_k−1 > p_{k−1}\)

Figure 5. A generating tree of \(T\) example of \(6\) out of \(3\) is shown in Figure 5. \(T\) is a connected graph with no cycles. Moreover, as a node is always smaller than its parent in lexicographical order, there is no cycle in \(T\). Hence \(T\) is a tree.

It is easy to observe that \(T\) can be viewed as a sub-tree of \(k−1\) out of \(n−1\) combined with a sub-tree of \(k\) out of \(n−1\). Namely, it is formed by a tree with the digit 1 and a tree without 1. And each sub-tree can be further decomposed into two. Therefore, \(T\) can be constructed recursively.

Another possible \(T\) is a single line, where each parent has only one child. There are several different constructions in the literature. For example, the homogeneous scheme and the near-perfect scheme can be found in [14]. Here we list two examples for \(n = 6, k = 3\) taken from [14]. A homogeneous example: 123, 124, 134, 234, 235, 135, 125, 145, 245, 345, 346, 146, 246, 236, 136, 126, 156, 256, 356, 456. A near-perfect example: 456, 256, 156, 356, 464, 146, 126, 136, 236, 134, 124, 123, 125, 135, 235, 245, 145, 345. Here a node is the parent of its neighbor on the right.

For construction of \(\sigma_1, \ldots, \sigma_{(k−1)!}\), we will assume \(k \geq 3\), because otherwise only \(T\) itself is the generating tree. We first find which transitions are allowed from sub-tree \(T_t\) to \(T_{t+1}\). Consider the node in \(T_t\)

\[
a = (p_1, p_2, \ldots, p_{t−1}, n−k, p_{t+1}, \ldots, p_{k−1}, n)
\]

where \(\{p_1, \ldots, p_{t−1}, p_{t+1}, \ldots, p_{k−1}\}\) = \(\{n−k+1, n−k+2, \ldots, n−1\} \setminus \{y\}\)

(2)

for some \(n−k+2 \leq y \leq n−1\). This node exists since \(n \geq k+1\) and \(k \geq 3\). We can now define a child of \(a\) as \(b = (p_1, p_2, \ldots, p_{t−1}, y, p_{t+1}, \ldots, p_{k−1}, n)\).

Let \(x = y−n+k\) and \(b\) be the root of \(T_{t+1}\). Since \((\sigma_1, k)\) is isomorphic to \(a\) and \((\sigma_{t+1}, k)\) is isomorphic to \(b\), we can see that \(\sigma_{t+1} = \alpha_x \circ \sigma_t\), where \(\alpha_x = (x, 1, 2, \ldots, x−1, x+1, \ldots, k−1)\).

Here \(\circ\) is the composition of permutations and is computed from right to left.

From Construction 4, we know that node \(a\) has only one child \(c\) in \(T_t\) (we need to order elements in \(a\) increasingly first), which changes \(y+1\) of \(a\) to \(y\). And \(b\) changes \(n−k\) of \(a\) to \(y\). Thus, children \(b\) and \(c\) are valid for the parent \(a\), since they change different digits of \(a\). Hence, \(\alpha_x\) is a valid transition from tree \(T_t\) to \(T_{t+1}\).

Permuted by \(\alpha_x\), the lowest position in \(\sigma_t\) becomes the \(x\)-th lowest position in \(\sigma_{t+1}\). In other words, from the dual (inverse permutation) of \(\sigma_{t+1}\) to the dual of \(\sigma_t\), we push the \(x\)-th element to the top.

Construction 6. (Construction of \(\sigma_1, \ldots, \sigma_{(k−1)!}\))

Suppose \(C\) is a cycle of full permutations of \([k−1]\) with push-to-the-top transitions. First take the dual of each permutation in \(C\). Then replace each push-to-the-top operation \(t_x\) with \(\alpha_x\) and reverse the cycle direction. And thus we will get a cycle of \(\sigma_1, \sigma_2, \ldots, \sigma_{(k−1)!}\), generated by operations \(\alpha_x, 1 < x \leq k−1\). Starting from the identity permutation \(\sigma_1\), and ignoring the last transition, we get a path of \(\sigma_1, \sigma_2, \ldots, \sigma_{(k−1)!}\).

\(\sigma_1, \sigma_2, \ldots, \sigma_{(k−1)!}\) can easily generated once we have recursively generated \((k−2)\)-partial permutations of \(k−1\) elements (or full permutations on \(k−1\) elements) using the constructions above. We can also
use the cycle of full permutations in [8][4]. For example, if \( k = 4 \), then Construction 4 will form a cycle \( C \) on 3 elements using the operation sequence \( t_2, t_3, t_2, t_3, t_3 \). And the push-to-the-bottom sequence is \( 231 \ 321 \ 132 \ 213 \ 231 \ 132 \ 321 \ 213 \ 231 \ 132 \ 321 \ 213 \ 231 \ 132 \ 321 \). The corresponding \( \sigma \)'s are \( 312 \ 231 \ 321 \ 213 \ 132 \ 231 \ 123 \ 231 \ 213 \ 132 \ 231 \ 123 \ 231 \ 213 \ 132 \ 231 \ 123 \ 231 \). Deleting the arrow in brackets, we get a path.

Notice that if we construct \( \sigma_i \)'s using a \((k-1,k-2)\) universal cycle, then the push-to-the-top transitions are only \( t_\alpha \), with \( x = k - 1 \) or \( x = k - 2 \). Therefore, \( y = x + n - k \) in (2) is either \( y = n - 1 \) or \( y = n - 2 \). Thus, if we want to recursively construct our \((n,k)\) universal cycle, there are only two possible nodes in each subtree that may lead to a new subtree.

The following example shows how to decide the next sub-tree, given a permutation. Assume \( n = 7, k = 4 \), and the current \( k \)-partial permutation is \((5, 4, 6, 1)\). This node belongs to the equivalence class \((1, 5, 4, 6)\), and we know \( \sigma_1 \) is \((1, 3, 2)\) for the current sub-tree. The dual of \((1, 3, 2)\) is itself. To find the next sub-tree, we need to find the permutation before \((1, 3, 2)\) in a cycle of permutations of size 3. By the previous example, it should be \((3, 2, 1)\), whose dual is itself again. Thus, the next sub-tree should have \( \sigma_{i+1} = (3, 2, 1) \). The root of the next sub-tree is \((6, 5, 4, 7)\).

Having constructed \( T_1 \) and \( \sigma_1, \ldots, \sigma_{(k-1)} \), we are able to draw a generating tree. Figure 5 shows a generating tree of 3 out of 6. The sequence \( \sigma_1, \sigma_2 \) is \( 123 \rightarrow 21 \), and the transition node from \( T_1 \) to \( T_2 \) is \((3, 4, 6)\).

Knowing a generating tree and the current partial permutation, how shall we decide the operation for the next step? Define \( w_{mn} \) to be the position of the newly changed digit from the parent to the node \( m \), and the position is counted from top to bottom. More specifically, let \( m = (p_1, \ldots, p_k) \), then its parent \( p_{wm} = (p_1, \ldots, p_{wm-1}, p_{wm} + 1, \ldots, p_k) \), where \( p_{wm} \neq p_{wm}' \). Its parent defers from \( m \) only at position \( w_{mn} \). For the root, \( w_{root} \) can be any integer between 1 and \( k \). But we will assign \( w_{root} = k \). For example, in Figure 5, \((1, 2, 4)\) is the parent of the node \( m = (1, 2, 3) \), and the 3rd digit is changed from 4 to 3, so \( w_{mn} = 3 \), \( p_{wm} = p_3 = 3 \).

Let the current partial permutation be \((p_{j-1}, p_k, p_1, \ldots, p_{j-1})\) and the current equivalence class be \( m = (p_1, p_2, \ldots, p_k) \). The following algorithm is designed based on Lemma 2 and finds the next partial permutation for an arbitrary generating tree:

**Algorithm 7.**

- If \( m \) has no children,
  - if \( j \equiv w_{mn} + 1 \mod k \), then do step A;
  - else, do \( t_k \).
- Else (if \( m \) has children),
  - if \( j \equiv w_{y} + 1 \mod k \) for some child \( y = (p_1^{(y)}, p_2^{(y)}, \ldots, p_k^{(y)}) \), push \( p_{w_{mn}}^{(y)} \) to the top;
  - else if \( j \equiv w_{mn} + 1 \mod k \), then do step A;
  - else do \( t_k \).

**Step A:** find the parent of \( m \), say \( z \). If \( w_z = w_m \), find the parent of \( z \), repeat until \( w_z \neq w_m \) for some ancestor \( l = (p_1^{(l)}, p_2^{(l)}, \ldots, p_k^{(l)}) \). Then, push \( p_{wm}^{(l)} \) to the top. If no such ancestor exists, we are at the end of a cycle, so push the \( k \)-th element of the root to the top.

Thus at any moment with any partial permutation, we first find out which equivalence class it is in, and then we have a mechanical way to decide the next step. This algorithm does not need the ordering of the entire permutation, but only the first \( k \). At each step we either push the \( k \)-th highest element to the top, or push some particular element (e.g., \( p_{wm}^{(l)} \)) to the top. The ranking of this particular element does not matter.

If we use a recursive construction for \( \alpha_x \) in Construction 6 and for \( T_1 \) in Construction 4, Algorithm 7 can be simplified. In the following, Algorithm 8 or \( G(n, k) \) decides the transition in a universal cycle, and 9 or \( H(n, k) \) decides the transition in a reverse universal cycle. Namely, given a current permutation, \( G(n, k) \) decides the next transition and \( H(n, k) \) decides the previous transition.

Given a permutation, take the top \( k \) elements \((q_1, q_2, \ldots, q_k)\) and suppose \( q_t = \max_{k+1} q_i \). Order the top \( k \) elements increasingly as \( m = (p_1, p_2, \ldots, p_k) \). Let \( Q = (q_{t-1}, q_{t-2}, \ldots, q_1, q_1, \ldots, q_{t-1}) \). Let \( \sigma \) be isomorphic to \( Q \) and \( \sigma \in S_{k-1} \) be a permutation on \([k - 1]\). Let \( \pi \) be the dual of \( \sigma \).

**Algorithm 8.** (Universal cycle \( G(n, k) \))

1. (Transit to the next sub-tree)
   If \( m \) satisfies (2), with \( y = n - 1 \) or \( y = n - 2 \), and \( q_k = n - k \), find \( \pi \) and compute the transition \( t_x \) before \( \pi \) using \( H(k - 1, k - 2) \).
   - if \( x = y - n + k \) and \( (y/q_1, \ldots, q_{k-1}) \neq (y, \ldots, n, n - k + 1, \ldots, y - 1) \), push \( y \) to the top.
   - else, go to step 3.

2. (Transit to the previous sub-tree)
   Else if \( m = (n - k + 1, n - k + 2, \ldots, n) \), \( Q \neq (n - k + 1, \ldots, n - 1) \), and \( q_k = n - 1 \) or \( n - 2 \), find \( \pi \) and compute the transition \( t_x \) after \( \pi \) using \( G(k - 1, k - 2) \). Let \( y = x + n - k \).
   - if \( q_k = y \), push \( n - k \) to the top.
   - else, go to step 3.

3. (Transit within a sub-tree)
   If \( m = (p_1, p_2, \ldots, p_{k-n+1}, t, t + 1, \ldots, n) \), for \( 1 \leq p_{k-n+1} < t - 1 \) and \( n - k + 2 \leq t \leq n \),
   - if \( q_k = t \), push \( t - 1 \) (Child2).
   - else if \( q_k = p_{k-n+1} \) and \( p_{k-n+1} - 1 > p_{k-n+2} \), push \( p_{k-n+1} - 1 \) (Child1).
   - else if \( q_k = p_{k-n+1} \), push \( t - 1 \) (Ancestor).
   - else, do \( t_k \).

Else,
If \( q_k = p_k \) and \( p_k - 1 > p_{k-1} \), push \( p_k - 1 \) (Child).

Else if \( q_k = p_k \), push \( n \) (Ancestor).

Else, do \( t_k \).

Step 3 corresponds to the two cases in Construction 4. \( H(n, k) \) finds the previous permutation and is very similar to \( G(n, k) \), but there is no symmetry as \( G(n, k) \). Let the current permutation be \( (q_1, q_2, \ldots, q_n) \).

The following algorithm realizes \( H(n, n-1) \) and finds its previous transition, which should be either \( t_n \) or \( t_{n-1} \). Define \( Q \) and \( \pi \) in the same way as in Algorithm 8.

Algorithm 9. (Reverse universal cycle \( H(n,n-1) \))

1 (Transit to the previous sub-tree)
   If \( q_1 = 1 \), and \( q_n = n - 1 \) or \( n - 2 \), find \( \pi \) and compute the previous \( t_x \) of \( \pi \) using \( H(n-2, n-3) \).
   - If \( x = q_1 = n - 1 \) and \( (q_2, \ldots, q_n) \neq (q_1 + 1, \ldots, n, 2, \ldots, q_n) \), output \( t_n \).
     - Else, go to step 3.

2 (Transit to the next sub-tree)
   Else if \( q_1 = 1 \), \( Q \neq (2, \ldots, n) \), and \( q_1 = n - 1 \) or \( n - 2 \), find \( \pi \) and compute the next \( t_x \) of \( \pi \) by \( G(n-2, n-3) \).
   - If \( q_1 = x + 1 \), output \( t_n \).
     - Else, go to step 3.

3 (Transit with in a sub-tree)
   - If \( q_1 - q_n = 1 \) or \( -1 \), output \( t_n \).
     - Else, output \( t_{n-1} \).

Figure 2(a) is an example of the above algorithm with \( n = 4, k = 2 \).

IV. COMPLEXITY ANALYSIS

In this section, we analyze the computational complexity of Algorithm 8 and 9 and compare them with some previous constructions of sub-tree \( T_1 \) and cycle of full permutations \( \sigma_1, \sigma_2, \ldots, \sigma_{(k-1)} \).

Even though the algorithm \( G(n, k) \) is recursive, we will show the number of recursions is low on average. To compute \( G(n, k) \), in the worst case, we need to call \( H(k - (2i - 1), k - 2i) \), for all \( 1 \leq i \leq \lfloor k/2 \rfloor \). There are 4 partial permutations in each sub-tree (and in total \( 4(k - 1)! \) partial permutations) that we need to compute \( \alpha_x \) using \( H(k - 1, k - 2) \) or \( G(k - 1, k - 2) \). Thus, the \( (k - 1, k - 2) \) cycle is computed 4 times, leading to a total of \( 4(k - 1)! \) recursive iterations.

To compute the \( (k - 1, k - 2) \) cycle, \( H(k-3, k-4) \) and \( G(k-3, k-4) \) are called \( 4^2(k-3)! \) times. Similarly, the iteration of the \( (k - (2i - 1), k - 2i) \) cycle is computed \( 4^i(k - 2i + 1)! \) times, for \( 1 \leq i \leq \lfloor k/2 \rfloor \). Thus, the total number of calls of \( G \) and \( H \) is

\[
\sum_{i=1}^{\lfloor k/2 \rfloor} 4^i(k - 2i + 1)! < \sum_{i=1}^{\lfloor k/2 \rfloor} (k - i)(k - i)! = k! - 1
\]

And the average number of calls of function \( G \) or \( H \) for all the permutations is less than

\[
1 + \frac{k!}{(k)!} \rightarrow 1
\]

when \( n \to \infty \). Therefore, we only need to run \( G \) or \( H \) once on average.

Hence, the average computational complexity of \( G(n, k) \) is only dependent on the non-recursive operations. In particular, recognizing the form of \( m \) (such as \( m = (p_1, p_2, \ldots, p_{k-n-t-1}, t, t + 1, \ldots, n) \), for some \( 1 \leq p_{k-n-t-1} < t - 1 \) and \( n - k + 2 \leq t \leq \infty \)) does not require the exact ordering of \( m \), if we have \( O(k) \) auxiliary space \( (u_n-k+1, u_{n-k+2}, \ldots, u_n) \). For all \( i = 1, 2, \ldots, k \), if \( q_k \geq n - k + 1 \), set \( u_{q_k} = 1 \). If \( u_n = 1 \), then \( m \) is in the form \( (p_1, p_2, \ldots, p_{k-n-t-1}, t, t + 1, \ldots, n) \). And \( t - 1 \) is the largest index such that \( u_t = 0 \). In this way, \( O(k) \) time and \( O(k) \) space are needed.

The complexity of the rest is \( O(1) \). Therefore, we can conclude that \( G(n, k) \) has computational complexity \( O(k) \) in both time and space.

It should be noted that if \( k = n - 1 \), in the algorithm for \( H(n, n-1) \) or \( G(n, n-1) \), we only need to check if \( q_1 - q_n = \pm 1 \) in step 3. Therefore, the average complexity is \( O(1) \) in time and space.

We do not store the value \( y \) in step 1 and 2 for each sub-tree in Algorithm 8 because in flash memory we prefer no auxiliary storage and use storage only for the data (or permutation in our case). If we were allowed to store \( y \), then the number of recursions can be further reduced.

For comparison, let us consider homogeneous and near-perfect algorithms [14] for generating all \( (n, k) \) combinations and replace Construction 4. Both homogeneous and near-perfect algorithms require sorting the \( k \) elements in a partial permutation first. If we insist on linear computation time, then this will take \( O(n) \) space and \( O(k) \) time. And if we do not allow auxiliary storage as mentioned in the previous paragraph, these two algorithms are both recursive and take \( O(k) \) time.

The overall complexity is \( O(k) \) time and \( O(n) \) space.

In addition, we can compare Algorithm 9 with the algorithms in [8] and [4] for generating the \( \sigma_i \)’s. All algorithms require \( O(1) \) time on average. However, an \( (n, n-1) \) permutation cycle in [8] has transitions \( t_i \), \( i = 2, 3, \ldots, n \). And therefore we will need to check \( 2(k-2) \) partial permutations in step 1 and 2 in each sub-tree in Algorithm 8, instead of 4 as in Algorithm 9.
The comparison is summarized in Table I. One advantage of the algorithms in [14][8][4] is that one can find an explicit mapping between integers and combinations (or permutations) according to the cycle, even though it is still not clear how to map integers to partial permutations.

### V. EQUIVALENCE OF UNIVERSAL CYCLES AND GRAY CODES

As mentioned before, universal cycles are related to Gray codes, which have applications in codes for flash memories, therefore, we will study Gray codes in this section.

Consider a directed graph $G$, whose vertices are permutations of length $n$. There is a directed edge if there is a push-to-the-top transition from one vertex to another, for $k \leq i \leq n$. So each vertex in $G$ is restricted to $n-k+1$ outgoing edges. An $(n,k)$ Gray code on partial permutations is a cycle in $G$ whose vertices induce every $k$-partial permutation once and only once. Note that this cycle contains only a subset of the vertices in $G$. A Gray code contains $N = \binom{n}{k}!$ permutations, and we denote them by $P = (P_1, P_2, \ldots, P_N)$. It should be noted that each $P_i$ is a full permutation.

In Gray code we only allow $t_i$, for $k \leq i \leq n$. The constraint that $i \geq k$ is imposed for two reasons: (a) these are the valid transitions in universal cycles, and (b) in flash memory, to minimize the effect of leakage. The first reason will be revealed in the next section. Notice that if $2 \leq i < k$, then after the operation $t_i$, there will be a charge gap between the $p_{i-1}$-th and the $p_{i+1}$-th cell. As information is stored in the highest $k$ cells, we want to keep these $k$ cells close in charge levels. When large deflation happens, lower charge levels may decrease to none at all, and gap among the first $k$ cells may cause an error.

Recall the graphic description of universal cycles in II as a Hamiltonian cycle in graph $H$. $H$ and in $G$ differ in the fact that in $G$, we require the bottom $n-k$ elements of a permutation are consistent along the cycle, while in $H$ the bottom $n-k$ elements can be in arbitrary order. For example, if $n = 4$, $k = 2$, in $G$, $(1,3|2,4) \rightarrow (4,1|3,2)$ by pushing value 4 to the top. But in $H$, $(1,3) \rightarrow (4,1)$ can be induced by the permutations $(1,3|2,4) \rightarrow (4,1|2,3)$, which is not a valid transition in $G$.

In this section, we will show the universal cycle is equivalent to the Gray code on partial permutations using transitions $t_i$, $k \leq i \leq n$. For example, it is easy to see that the Gray code in Figure 2 induces the universal cycle $A = (4,3,4,2,3,2,4,1,3,1,2,1)$. However, it is not clear whether a universal cycle corresponds to exactly one Gray code, because we may change the bottom $n-k$ elements in each permutation and get another universal cycle, or the bottom $n-k$ elements are not consistent so there is no Gray code. We will show in the section that the Gray code is indeed unique.

Define a path of elements as a sequence of elements $(a_1, a_2, \ldots, a_N)$ where subsequences are take without wrapping around, e.g., $(a_N, a_1, \ldots, a_{k-1})$ is not considered as a subsequence. Similarly, a path of permutations is a sequence of elements $(P_1, P_2, \ldots, P_N)$ where the transition from $P_N$ to $P_1$ is not considered. Moreover, the transitions between permutations are always assumed to be push-to-the-top operation, and only the bottom $n-k$ elements are pushed. On the other hand, a cycle of elements or permutations is a sequence with wrapping around.

**Lemma 10.** Let $(P_1, P_2, \ldots, P_N)$ be an $(n,k)$ Gray code, then there is a unique $(n,k)$ universal cycle corresponding to it.

**Proof:** Given the Gray code, suppose $P_1 = (a_k, a_{k-1}, a_1|b_1, b_2, \ldots, b_{n-k})$ and let $b_0 = a_1$ be the $k$-th element of $P_1$. Assume $P_2 = t_{j+k}(P_1)$ for some $0 \leq j \leq n-k$. Then $P_2 = (b_j, a_k, a_{k-1}, \ldots, a_2|b_0, \ldots, b_{j-1}, b_{j+1}, \ldots, b_{n-k})$ if $j > 0$ and $P_2 = (b_j, a_k, a_{k-1}, \ldots, a_2|b_1, b_2, \ldots, b_{n-k})$ if $j = 0$. Define $a_{k+1} = b_j$ as the top element of $P_2$. Then $(a_1, a_2, \ldots, a_k, a_{k+1})$ contains two $k$-permutations (without wrapping around). In the same manner, let $a_{k+i}$ be the top element of $P_{i+1}$, $i = 1, 2, \ldots, N-k$. Then $(a_1, a_2, \ldots, a_N)$ is a path generating $k$-permutations.

Moreover, $P_1$ is obtained from $P_{N-k+2}$ by applying the push-to-the-top transitions $k-1$ times, and only the bottom $n-k+1$ elements are pushed. Thus $a_1$ must be the top element of $P_{N-k+2}$. Similarly, $a_i$ must be the top element of $P_{N-k+i+1}$, for $i = 1, 2, \ldots, k-1$. Hence $(a_1, a_2, \ldots, a_N)$ is a cycle generates $k$-permutations. It contains all $k$-permutations exactly once because a Gray code contains every $k$-permutation exactly once. Thus we get a universal cycle from the Gray code.

Generalizing the observation in the above proof, we can easily check that $a_{i+k-1}$ in the universal cycle is the $l$-th element in $P_{i+k-l}$.

On the other hand, we show that a universal cycle can be mapped to a unique Gray code. The proof first shows that there are multiple permutation paths mapped to a universal cycle, and then finds the only path that is also a cycle.
Lemma 11 Given an \((n,k)\) universal cycle \(A = (a_1, a_2, \ldots, a_N)\), there is a unique corresponding Gray code \(P = (P_1, P_2, \ldots, P_N)\).

Proof: We first claim that given a universal cycle \(A\), there is a corresponding path of permutations \(P = (P_1, P_2, \ldots, P_N)\). For a given \(1 \leq i \leq N\), let \(b_1, b_2, \ldots, b_{n-k} \in [n]\) be distinct integers different from \(\{a_i, \ldots, a_{i+k-1}\}\) in arbitrary order. Therefore, \(P_i = (a_{i+k-1}, a_{i+k-2}, \ldots, a_i \mid b_1, b_2, \ldots, b_{n-k})\) is a permutation. And let \(b_0 = a_i\). Since \((a_{i+1}, a_{i+2}, \ldots, a_{i+k})\) does not include duplicated elements, \(a_{i+k} = b_j\) for some \(0 \leq j \leq n - k\). If \(j > 0\), let \(P_{i+1} = (a_{i+k}, a_{i+k-1}, \ldots, a_{i+1} \mid b_0, \ldots, b_{j-1}, b_{j-2}, \ldots, b_{n-k})\). If \(j = 0\), let \(P_{i+1} = (a_{i+k}, a_{i+k-1}, \ldots, a_{i+1} \mid b_1, b_2, \ldots, b_{n-k})\). From \(P_i\) to \(P_{i+1}\), we push \(b_i = a_{i+k}\) to the top, which is a push-to-the-top transition and only the lower \(n - k + 1\) elements can be pushed. Hence the claim holds.

From above, we can see: (a) For \(1 \leq i \leq k\), the \(i\)-th element in \(P_i\) is \(a_{i+k-i}\). In particular, the \(k\)-th elements of \(P\) consist of the universal cycle \(A\). (b) Given universal cycle \(A\), the permutation path \(P\) is uniquely defined by the bottom elements of \(P_i\).

Since \(A\) is a cycle, we can assume the top elements of \(P_1\) are \((p_1, p_2, \ldots, p_k)\), and the top elements of \(P_N\) are \((p_2, p_3, \ldots, p_k, p_1)\). If \(P\) is a Gray code, then from \(P_N\) to \(P_1\) we need to push \(p_1\) to the top.

Now we claim that there is a unique \(P_1\) such that \(P\) is a Gray code. For \(1 \leq i \leq N\), define \(q_i\) to be the \((k + 1)\)-th element of \(P_i\). For \(2 \leq i \leq N\), from \(P_{i-1}\) to \(P_i\), if we push the \(k\)-th element to the top (do \(t_k\)), then \(q_i = q_{i-1}\); otherwise, \(q_i\) is the \(k\)-th element of \(P_{i-1}\). Since we cannot do \(t_k\) for all \(1 \leq i \leq N\) and get all the \(k\)-permutations, let \(i_1, i_2, \ldots\) be the indices such that we do \(t_k\) \((j > k)\) on \(P_{i_1-1}, P_{i_2-1}, \ldots\). Therefore, \(q_{i_1}\) is the \(k\)-th element of \(P_{i_1-1}\) and \(q_{i_1} = \cdots = q_{i_1}, q_{i_1+1} = \cdots = q_{i_k}\). This implies that \((q_1, q_2, \ldots, q_N)\) are independent of the bottom elements of \(P\), given the \(k\)-th elements of \(Q\), or equivalently, given the universal cycle \(A\).

Let \(P_N = (p_2, p_3, \ldots, p_k, p_1)\mid r_1, r_2, \ldots, r_{n-k}\) and \(R = (r_1, r_2, \ldots, r_{n-k})\) be its bottom elements. Scanning \(q_N, q_{N-1}, \ldots, q_1\) one by one, if an integer already appears before, or if it belongs to \((p_2, p_3, \ldots, p_k, p_1)\), then delete it. The remaining sequence is exactly \(R\). This is because any \(r_i\) must appear in the top elements in at least one of \(P_1, \ldots, P_{N-1}\), and at the transitions previous to \(P_N\), we introduce \(R\) to the bottom elements. Moreover, given the universal cycle \(A\), \((q_1, q_2, \ldots, q_N)\) and \(R\) are independent of the bottom elements of \(P\). Therefore, no matter what the bottom elements in \(P_1\) are, \(P_N\) will be identical. Pushing \(p_1\) to the top in \(P_N\), we get the unique \(P_1\) such that \(P\) is a Gray code.

Combining Lemma 10 and 11, we have the following theorem.

Theorem 12. There is a bijection between \((n,k)\) universal cycles and Gray codes, where the concatenation of the \(k\)-th element of each permutation in the Gray code consists of the universal cycle.

By the above discussions, we know that traversing the \(k\)-partial permutations using push-to-the-top transitions is identical to generating a universal cycle. So we can apply Algorithms 8 and 9 to construct Gray codes. However, we need to define the initial full permutation in the sequence, so that it actually forms a cycle. For example, we can see that the initial permutation should be \((4,5,6|1,2,3)\) in Figure 5 for the specific ordering of the lower elements. In general, we have:

Theorem 13. The first partial permutation should be \((n-k+1, \ldots, n|1, \ldots, n-k)\) so that Algorithm 8 forms a Gray code.

Proof: Let \(Q_0, Q_1, \ldots, Q_{N-1}\) be the \(k\)-partial permutations defined by Algorithm 8. By proof of Lemma 11, the first permutation should be \((n-k+1, \ldots, n|1, \ldots, n)\), where \(r_1, \ldots, r_k\) are the "newly appeared" elements of \(q_0, q_{N-1}, q_{N-2}, \ldots, q_1\) excluding \(\{n-k+1, \ldots, n\}\). If the transition from \(P_{i-1}\) to \(P_i\) is not \(t_k\), then \(q_i\) is the \(k\)-th element of \(P_{i-1}\).

Suppose \(y\) is the value defined in step 1 in Algorithm 8, \(n-k+1 \leq y \leq n-1\). Consider the node in first sub-tree \(T_1\), \(m = (n-k, \ldots, y-1, y+1, \ldots, n)\). It has parent \((n-k, \ldots, y-2, y, y+1, \ldots, n)\), Child2 \((n-k, \ldots, y-1, y, y+2, \ldots, n)\), and Child3 \((y, n-k+1, \ldots, y-1, y+1, \ldots, n)\). We know that Child1 leads to the other sub-trees \(T_2, \ldots, T_{k-1})\). It can be seen that in the Gray code, Child2 and the other sub-trees appear before Child2 and the rest of \(T_1\).

Similarly, for any node \((p_1, p_2, \ldots, p_{k-n+1}, l, l+1, \ldots, n)\), its Child2 appears before Child1. (In Figure 5, for any node in \(T_1\), the Gray code always visit its right branch before its top branch.)

Combining these two facts, the last \(n-k\) partial permutations in the Gray code with transitions \(t_i, i > k\), are \((n-k+2, \ldots, n), (n-k+3, \ldots, n, 1, 2), \ldots, (1, 2, \ldots, k)\). And \((r_1, r_2, \ldots, r_{n-k}) = (1, 2, \ldots, k)\).

VI. CONCLUSIONS

In this paper, we solved the open problem of explicit constructions for universal cycles on \(k\)-partial permutations, a sequence of length \(\binom{n}{k}\) with each \(k\)-partial permutation as its sub-sequence exactly once. This problem is shown to be equivalent to construction of \((n,k)\) Gray code on partial permutations. And we proposed the application of partial permutations for flash memory, which allows less decoding cost while sacrificing some information capacity.

The construction can be broken down into two parts: a sub-tree with all \(\binom{k}{k}\) combinations, and a \((k,k-1)\)
universal cycle. Each transition can be determined solely by the current partial permutation. The algorithm in this paper uses $O(k)$ time and space on average, and no extra storage space when no transition is made.

There are several open problems in this topic. For instance, is it possible to design recursions of an $(n + 1, k + 1)$ Gray code based on an $(n, k)$ Gray code (or some other form of recursion)? If $n - k = 1$, the answer is yes and [4] provides a nice construction. Besides, we do not have efficient ways to map partial permutations to integers according to the constructed cycle, i.e., we do not know the position of a permutation in the cycle. These will be our future work directions.

REFERENCES