A Language-Theoretic Approach to the Heapability of Signed Permutations

Gabriel Istrate

Faculty of Mathematics and Computer Science, University of Bucharest
Str. Academiei 14, 011014, Sector 1, Bucharest, Romania
gabriel.istrate@unibuc.ro

Abstract. We investigate a signed version of the Hammersley process, a discrete process on words related to a property of integer sequences called heapability (Byers et al., ANALCO 2011). The specific version that we investigate corresponds to a version of this property for signed sequences.
We give a characterization of the words that can appear as images of the signed Hammersley process. In particular we show that the language of such words is the intersection of two deterministic one-counter languages.

Keywords: signed Hammersley process, formal languages, heapability

1 Introduction

Consider the following process H_k that generates strings over the alphabet $\Sigma_k = \{0, 1, \ldots, k\}$, $k \geq 1$: start with the empty word $w_0 = \lambda$. Given word w_n, to obtain w_{n+1} insert a k at an arbitrary position of w_n (we will regard the new k as a "particle with k lives"). If w_n contained at least one nonzero letter to the right of the newly inserted k then decrease by 1 the leftmost such letter (that is the new particle takes one life from the leftmost live particle to its right). The process has been introduced in [14], related to a special property of integer sequences called *heapability* (see [9] for a definition, and [14,16,15,4,5,3,7] for subsequent work), and has been called (by analogy to the classical case $k = 1$, which it generalized) *the Hammersley process*. This process has proved essential [1] in the scaling analysis of the longest-increasing subsequence of a random permutation, one of the celebrated problems in theoretical probability [17], and was subsequently rigorously analyzed in [4,5].

In [7] we have undertaken a language-theoretic approach to the study of process H_k by characterizing the language $L(H_k)$ of possible words in the process H_k. It was shown that the language $L(H_k)$ is regular for $k = 1$ and context-free but nonregular for $k \geq 2$. We also gave an algorithm to compute the formal power series associated to the language $L(H_k)$ (where the coefficient of each word w is equal to its multiplicity in the process H_k). We attempted to use this algorithm to study the typical large-scale behavior of process H_k (while this study provided valuable information, the convergence to the limit behavior turned out to be fairly slow).

The purpose of this paper is to study (with language-theoretic tools similar to those in [7]) a variant of the Hammersley process that we will call *the signed Hammersley process*, and we will denote by H_k^{sign}. This is a process over the alphabet $\Gamma_k := \{0^+, 0^-, 1^+, 1^-, 2^+, 2^-, \ldots, k^+, k^-\}$ and differs from process H_k in the following manner:

– The newly inserted letter will always be a *signed* version of k, that is, it will either be a k^+ or a k^-.

Gabriel Istrate: A Language-Theoretic Approach to the Heapability of Signed Permutations, pp. 71–85.
Proceedings of PSC 2024, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-07328-5 © Czech Technical University in Prague, Czech Republic

– When a k^- is inserted it subtracts 1 from the closest (if any) nonzero letter to its right having positive polarity. That is, inserting k^- turns a k^+ into a $(k-1)^+$, a $(k-1)^+$ into a $(k-2)^+$, ..., a 1^+ into a 0^+, but has no effect on letters of type 0^+ or letters with negative polarity.
– On the other hand, inserting a k^+ subtracts a 1 from the closest digit with negative polarity and nonzero value (i.e., one in $\{1^-, \ldots, k^-\}$), if any, to its right.

Example 1. A depiction of the first few possible steps in the evolution of the ordinary Hammersley tree process is presented in Figure 1(a). Similarly for the signed Hammersley tree process, see Figure 1(b).

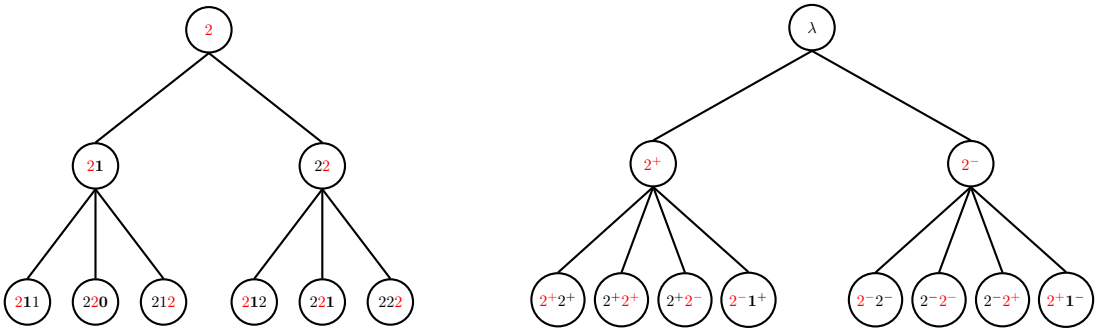

Figure 1. (a). Words in the binary Hammersley process ($k = 2$). The first node λ is omitted. (b). Words in the binary signed Hammersley tree process. Both figures: Insertions are in red. Positions that lost a life at the current stage are bolded.

In this paper we provide (Theorem 7) a complete characterization of the language of words in the signed Hammersley process, they are the words such that all of their prefixes satisfy a certain property called *k-dominance* (see Definition 2 for the precise definition of k-dominant strings). As a consequence, the language of words in the signed Hammersley process is the intersection of two deterministic context-free languages.

Perhaps not surprisingly, the motivation behind our study is a adaptation of the concept of heapability to signed sequences (permutations) $(\sigma(1), \sigma(2), \ldots, \sigma(n))$. Briefly, in the setting we consider, every integer $\sigma(i)$ in the sequence comes with a sign $\tau(i) \in \{\pm 1\}$. An integer $\sigma(i)$ can only become the child of an integer $\sigma(j)$ in a heap-ordered tree when $\sigma(j) < \sigma(i)$ and $\sigma(i)$ has the opposite sign (i.e., $\tau(i) = -\tau(j)$). When the signed permutation cannot be inserted into a single heap-ordered tree we are, instead, concerned with inserting it into the smallest possible number of heap-ordered trees.

The outline of the paper is as follows: In Section 2 we review some preliminary notions we will need in the sequel. Our main result is presented in Section 3. We then give (Section 4) an algorithm for computing the formal power series associated to the signed Hammersley process. Later sections are no longer primarily string-theoretic, and instead attempt to explain the problem on signed integer sequences that motivated our results: in Section 5 we define the heapability of signed permutations and show that a greedy algorithm partitions a signed permutation into a minimal number of heap-ordered trees. This motivates and explains the definition of the signed Hammersley process, that describes the dynamics of "slots" generated by this algorithm.

We refrain in this paper, though, from investigating the scaling in the Ulam-Hammersley problem for signed permutations, as it is not directly connected to the main theme of the conference, and instead leave this topic for future work.

2 Preliminaries

We will assume general acquaintance with formal language theory, as presented in, say, [13], and its extension to (noncommutative) formal power series. For this last topic useful (but by no means complete) references are [18,6]. Denote by Γ_k the alphabet $\{0^+, 0^-, \ldots, k^+, k^-\}$. As usual, for $z \in \Gamma_k^*$ and $p \in \Gamma_k$, denote by $|z|$ the length of z and by $|z|_p$ the number of appearances of letter p in z. Also, for $j \in \{0, \ldots k\}$, define $|z|_j = |z|_{j^+} + |z|_{j^-}$.

Definition 2. *String $z \in \Gamma_k^*$ is called k-dominant iff it starts with a letter from the set $\{k^+, k^-\}$, and the following two conditions are satisfied:*

$$|z|_{k^+} - \sum_{i=1}^{k} i \cdot |z|_{(k-i)^-} + \sum_{i=0}^{k-1} |z|_{i^+} \geq 0 \tag{1}$$

and

$$|z|_{k^-} - \sum_{i=1}^{k} i \cdot |z|_{(k-i)^+} + \sum_{i=0}^{k-1} |z|_{i^-} \geq 0 \tag{2}$$

at least one of the inequalities being strict, namely the one that corresponds to the first letter of z.

Example 3. The only words $z \in \Gamma_k^1$ that are k-dominant are $z = k^+$ and $z = k^-$.

Example 4. For $z \in \{0^+, 0^-, k^+, k^-\}^*$ (or $k = 1$) the two conditions in Definition 2 become $|z|_{k^+} - k \cdot |z|_{0^-} + |z|_{0^+} \geq 0$ and $|z|_{k^-} - k \cdot |z|_{0^+} + |z|_{0^-} \geq 0$.

An example of a k-dominant word is (for $k = 2$) word $z = 2^+2^-$. Indeed, $|z|_{2^+} = 1, |z|_{2^-} = 1, |z|_a = 0$ for $a \notin \{2^-, 2^+\}$. So z satisfies conditions (1) and (2). So does its prefix 2^+.

Definition 5. *A formal power series with nonnegative integer coefficients is a function $f : \Gamma_k^* \to \mathbf{N}$. We will denote by $\mathbf{N}(< \Gamma_k >)$ the set of such formal power series.*

Definition 6. *Given $k \geq 1$,* the signed Hammersley power series of order k *is the formal power series $F_k \in \mathbf{N}(< \Gamma_k >)$ that counts the multiplicity of a given word $w \in \Gamma_k^*$ in the signed Hammersley process of order k.*

The signed Hammersley language of order k, $L(H_k^{sign})$, *is defined to be the support of F_k, i.e., the set of words $w \in \Gamma_k^*$ that are outputs of the signed Hammesley process of order k.*

3 Main Result: The language of the process H_k^{sign}

Our main result is:

Theorem 7. *A word $z \in \Gamma_k^*$ is generated by the signed Hammersley process if and only if z and all its nonempty prefixes are k-dominant.*

Corollary 8. *For every $k \geq 1$ if $L(H_k^{sign})$ is the language of words that appear in the signed Hammersley process there exist two deterministic context-free languages (in fact L_1, L_2 are even deterministic one-counter languages, see [20]) s.t. $L(H_k^{sign}) = L_1 \cap L_2$.*

The proof of Theorem 7 proceeds by first showing that every word generated by the signed Hammersley process satisfies the condition in the theorem, and conversely, every string that satisfies the conditions can be generated by the process. For the first claim we need a couple of simple lemmas:

Lemma 9. *Every word in $L(H_k^{sign})$ starts with a k^+ or a k^-.*

Proof. Digits are only modified by insertions to their left. So the leftmost k^+ or k^- remains unchanged.

□

Lemma 10. *$L(H_k^{sign})$ is closed under prefix.*

Proof. Given $w \in L(H_k^{sign})$ and a position p in w, just ignoring all insertions to the right of p yields a construction for the prefix of w ending at p.

□

Lemma 11. *Every word in $L(H_k^{sign})$ is k-dominant.*

Proof. Let us count the number of particles of type $(k-i)^+$ in z, $i \geq 1$. Such a particle arises from a single k^+ particle through a chain of insertions

$$k^+ \xrightarrow{k^-} (k-1)^+ \xrightarrow{k^-} \ldots \xrightarrow{k^-} (k-i)^+$$

requiring i particles of type k^- and killing the original particle of type k^+. Similarly, a particle of type $(k-i)^-$, $i \geq 1$, arises from a k^- particle through a chain

$$k^- \xrightarrow{k^+} (k-1)^- \xrightarrow{k^+} \ldots \xrightarrow{k^+} (k-i)^-$$

requiring i particles of type k^+ and killing one particle of type k^-.

In addition to the insertions that kill particles, denote by λ^+ the number of particles of type k^+ that, when inserted, don't have any $1^-, \ldots, k^-$ to their right, (hence they don't kill any particle and aren't counted by the above chains) and, similarly, by λ^- particles of type k^+ that don't kill any particle.

Thus the number of particles of type k^+ in word z is

$$|z|_{k^+} = \lambda^+ + \sum_{i \geq 1} i \cdot |z|_{(k-i)^-} - \sum_{i \geq 1} |z|_{(k-i)^+} \tag{3}$$

In the previous equation we have simply tallied up the number of particles k^+ originally inserted, subtracting those that eventually end up as a $(k-i)^+$ for some $i \geq 1$. Similarly

$$|z|_{k^-} = \lambda^- + \sum_{i \geq 1} i \cdot |z|_{(k-i)^+} - \sum_{i \geq 1} |z|_{(k-i)^-} \tag{4}$$

Putting the condition $\lambda^+, \lambda^- \geq 0$ (the appropriate one being strictly > 0) we infer that z is k-dominant. □

From Lemmas 9, 10 and 11 it follows that any word $z \in L(H_k^{sign})$ satisfies the conditions (1) and (2) in the theorem.

We show the converse as follows:

Lemma 12. *Every word z satisfying conditions in Theorem 7 is an output of the signed Hammersley process.*

Proof. Assume otherwise. Consider a z of smallest length that is not the output of the signed Hammersley process. Clearly $|z| > 1$, since $z = k^+$ and $z = k^-$ are outputs.

Consider the last occurrence l of one of the letters k^+, k^- in z. Without loss of generality assume that $l = k^+$ (the other case is similar).

If to the right of l one had only positive letters (if any) then consider the word w_1 obtained by pruning the letter l from z. Its prefixes p are either prefixes of z (hence k-dominant) or are composed of the prefix w_0 of z cropped just before l plus some more positive letters. For such a prefix we have to verify the two conditions (1) and (2) are satisfied: The second one follows directly from condition (2) for the word z, since all letters of z with negative polarity are present in p. As for the first one, it follows from the fact that condition (1) is valid for w_0, since some more positive letters are added. In conclusion, all prefixes of w_1 are k-dominant. As $|w_1| = |z| - 1$, by the minimality of z, it follows that w_1 is an output of the signed Hammersley process. But then the process can output z by simply simulating the construction for w_1 and then inserting the last k^+ of z into w_1 in its proper position.

The remaining case is when $l = k^+$ has some negative letters to its right. Assume that s is the first letter of negative type to the right of l. s cannot be k^-, since we already saw the last of k^+, k^- at position l. Let w_2 be the word obtained by removing l from z and increasing the value of s by 1. We claim that w_2 and all its prefixes are k-dominant.

$$|w_2|_{k^+} - \sum_{i=1}^{k} i \cdot |w_2|_{(k-i)^-} + \sum_{i=0}^{k-1} |w_2|_{i^+} = |z|_{k^+} - \sum_{i=1}^{k} i \cdot |z|_{(k-i)^-} + \sum_{i=0}^{k-1} |z|_{i^+} \quad (5)$$

In fact this is true for every prefix of w_2, since losing a k^- is not counted, and adding one doesn't change the sum.

$$|w_2|_{k^-} - \sum_{i=1}^{k} i \cdot |w_2|_{(k-i)^+} + \sum_{i=0}^{k-1} |w_2|_{i^-} = |z|_{k^-} - \sum_{i=1}^{k} i \cdot |z|_{(k-i)^+} + \sum_{i=0}^{k-1} |z|_{i^-} \geq 0 \quad (6)$$

Again, this is true for every prefix of z containing s, since losing k^- is compensated by increasing s, so the sum doesn't change.

Because of the minimality of z, w_2 is an output of the signed Hammersley process. But then we can obtain z as an output of this process by first simulating the construction of w_2 and then inserting l (which corrects the value of s as well).

□

3.1 Proof of Corollary 8

Proof. The claim that $L(H_k^{sign})$ is a the intersection of two deterministic one-counter languages follows from Theorem 7 as follows: We construct two one-counter PDA's, A_1

and A_2, both having input alphabet Γ_k. Each of them enforces one of the conditions (1) and (2), respectively, including the fact that the appropriate inequality is strict. We describe A_1 in the sequel, the functioning of A_2 is completely analogous.

The stack alphabet of A_1 comprizes two stack symbols, an effective symbol $*$ and the bottom symbol Z. Transitions are informally specified in the following manner:

- A_1 starts with the stack consisting of the symbol Z. If the first symbol it reads is neither k^+ nor k^-, A_1 immediately rejects.
- A_1 has parallel (but disjoint) sets of states corresponding to the situations that the first letter is a k^+, respectively a k^-.
- The first alternative requires that the difference in (1) is strictly positive. A_1 enforces this by first ignoring the first letter and then enforcing the fact that the difference in (1) *corresponding to the suffix obtained by dropping the first letter* is always ≥ 0. This is similar to the behavior in the case when the first letter is a k^-, which we describe below.
- when reading any subsequent positive symbol, A_1 pushes a $*$ on stack.
- if the next symbol of the input is $(k-i)^-$, $i \in 1 \ldots k$, A_1 tries to pop i star symbols from the stack. If this ever becomes impossible (by reaching Z), A_1 immediately rejects.
- When reaching the end of the word A_1 accepts.

□

Observation 1 *A nagging question that would seem to be a simple exercise in formal language theory (but which so far has escaped us) is proving that $L(H_k^{sign})$ is not a context free language.*

This is, we believe, plausible since the words in $L(H_k^{sign})$ have to satisfy not one but two inequalities, (1) and (2) that constrain their Parikh distributions. Verifying such inequalities would seemingly require two stacks.

We have attempted (but failed) to prove this by applying Ogden's lemma. So we leave this as an open question.

4 Computing the formal power series of the signed Hammersley process

In this section we study the following problem: given a word $z \in \Gamma_k^*$, compute the number of copies of z generated by the signed Hammersley process. The problem is, of course, a generalization of the one in the previous section: we are interested not only if a word can/cannot be generated, but in how many ways.

The real motivation for developing such an algorithm is its intended use (similar to the use of the analog algorithm from the unsigned case in [9]) to attempt to analyze the scaling in the Ulam-Hammersley problem for heap decompositions, and is beyond the scope of the current paper. Roughly speaking, by studying the multiplicity of words in the signed Hammersley process we hope to be able to exactly sample from the distribution of words for large n and estimate, using such a sampling process the scaling constant for the decomposition of a random signed permutation into a minimal number of heaps.

The corresponding problem for the unsigned case has been considered in several papers [14,15,4,5], and the application of formal power series techniques to the scaling problem is described in [7].

Multiplicity(k, w) :

Input: $k \geq 1, w \in \Gamma_k^*$
Output: $F_k(w)$
$S = 0.w = w_1 w_2 \dots w_n$

if $w \notin L(H_k^{sign})$ **then**
  return 0
if $w == k^+$ or $w == k^-$: **then**
  return 1
for $i = 1$ to n **do**
  if $w_i == k^+$ and $\exists l : i+1 \leq l \leq n : w_l \in \{0^-, \dots, k^-\}$: **then**
    let $r = min(\{n+1\} \cup \{l \geq i+1 : w_l \in \{1^-, \dots, k^-\})$
    if $r == n+1$ or $w_r \neq k^-$ **then**
      for $j = 1$ to $r-1$ **do**
        if $w_j == 0^-$ **then**
          let $z = w_1 \dots w_{i-1} w_{i+1} \dots w_{j-1} 1^- w_{j+1} \dots w_n$
          let $S := S + Multiplicity(k, z)$
    if $r \leq n$ **then**
      let $z := w_1 \dots w_{i-1} w_{i+1} \dots (w_r + 1)^- w_{r+1} \dots w_n$
      let $S := S + Multiplicity(k, z)$

  if $w_i == k^-$ and $\exists l : i+1 \leq l \leq n : w_l \in \{0^+, \dots, k^+\}$: **then**
    let $r = min(\{n+1\} \cup \{l \geq i+1 : w_l \in \{1^+, \dots, k^+\})$
    if $r == n+1$ or $w_r \neq k^+$ **then**
      for $j = 1$ to $r-1$ **do**
        if $w_j == 0^+$ **then**
          let $z := w_1 \dots w_{i-1} w_{i+1} \dots w_{j-1} 1^+ w_{j+1} \dots w_n$
          let $S := S + Multiplicity(k, z)$
    if $r \leq n$ **then**
      let $z := w_1 \dots w_{i-1} w_{i+1} \dots (w_r + 1)^+ w_{r+1} \dots w_n$
      let $S := S + Multiplicity(k, z)$

  if $w_i == k^+$ or $w_i == k^-$ **then**
    let $Z := w_1 \dots\dots w_{i-1} w_{i+1} \dots w_n$
    let $S := S + Multiplicity(k, z)$

Figure 2. Algorithm Multiplicity(k,w)

Theorem 13. *Algorithm Multiplicity in Figure 2 correctly computes series F_k.*

Proof. The idea of the algorithm is simple, in principle: we simply attempt to "reverse time" and try to identify strings z that can yield the string w in one step of the process. We then add the corresponding multiplicities of all preimages z.

We go from z to w by inserting a k^+ or a k^- and deleting one life from the closest non-zero letter of z that has the correct polarity and is to the right of the newly inserted letter.

Not all letters of k^+, k^- in w can be candidates for the inserted letter, though: a candidate k^+, for instance, cannot have a k^+ as the first letter with positive polarity to its right. That is, if we group the k^+'s in w in blocks of consecutive occurrences, then the candidate k^+'s can only be the right endpoints of a block.

As for the letter l the new k^+ acted upon in z, in w it cannot be a k; also, there cannot be any letters with positive polarity, other than zero, between the new k^+ and l.

So l is either one of the 0^+'s at the right of the new k^+ or the first nonzero positive letter (if not k^+). It is also possible that l does not exist.

Analogous considerations apply if the newly inserted letter is a k^-. □

5 Motivation: The Ulam-Hammersley problem for the heap decomposition of signed permutations

The *Ulam-Hammesley problem* [19,12] can be described as follows: estimate the asymptotic behavior of the expected length of the longest increasing subsequence of a random permutation $\sigma \in S_n$. The correct scaling is $E_{\sigma \in S_n}[LIS[\sigma]] = 2\sqrt{n}(1+o(1))$, however substantially more is known, and the problem has deep connection with several mathematical concepts and areas, including Young tableaux (see, e.g., [17]), random matrix theory [2] and interacting particle systems [1].

The following concept has been defined (for $k = 2$, and can be easily generalized as presented) in [9]: a sequence of integers is *(k)-heapable* if the elements of a sequences can be inserted successively in a k-ary tree, min-heap ordered (i.e., the label of the child is at least as large as that of the parent), not-necessarily complete, so that insertions are always made as a leaf.

Heapable sequences[1] can be viewed as a (loose) generalization of increasing sequences: instead of inserting sequences into an increasing array, so that each node (except the last one) has exactly one successor, we allow a "more relaxed version" of this data structure, in the form of a k-ary tree with a min-heap ordering on the nodes. In other words a node has multiple choices for the insertion (instead of a single one), and using all these leaves can accommodate some limited form of disorder between consecutive elements, although values in such a sequence "tend to increase", simply because the available positions will appear lower and lower in the tree.

It is natural, therefore, to attempt to generalize the Ulam-Hammersley problem to heapable sequences. The direct extension is problematic, though: as discussed in [9], the problem of computing the longest heapable subsequence has unknown complexity (see [10,11] for some related results). A more promissing alternative is the following: by Dilworth's theorem the longest increasing subsequence is equal to the minimal number of *decreasing sequences* in which we can partition the sequence. So the "correct" extension of the longest increasing subsequence is (see [14,3]): **decompose a random permutation $\sigma \in S_n$ into the minimal number of heap-ordered k-ary trees**, and study the scaling of the expected number of trees in such an optimal decomposition.

The longest increasing subsequence problem has also been studied for *signed* or even *colored* permutations [8]. It is reasonable to consider a similar problem for heapability. This is what we do in this paper. Note, though, that the variant we consider here is not the same as the one in [8]. Specifically, in [8] the author requires that the "colors" (i.e., signs) of two adjacent nodes that are in a parent-child relationship are the same. By contrast we require that *the colors of any two adjacent nodes are different.*

Definition 14. *A signed permutation of order n is a pair (σ, τ), with σ being a permutation of size n and $\tau : [n] \to \{\pm 1\}$ being a sign function.*

[1] in this motivating discussion we refer to the original concept, omitting the setting in this paper where letters have polarities

Definition 15. *Given integer $k \geq 1$, signed permutation (σ, τ) is called $\leq k$-heapable if one can successively construct k (min) heap-ordered binary trees (not necessarily complete) $H_0, H_1, \ldots, H_{k-1}$ such that*

- *at time $i = 0$ $H_0, H_1, \ldots H_{k-1}$ are all empty.*
- *for every $1 \leq i \leq n$ element $\sigma(i)$ can be inserted as a new leaf in one of $H_0, H_1, \ldots, H_{k-1}$, such that if $\sigma(j)$, the parent of $\sigma(i)$ exists, $j < i$, then ($\sigma(j) < \sigma(i)$ and) $\tau[i] = -\tau[j]$.*

(σ, τ) will be called k-heapable iff k is the smallest parameter such that (σ, τ) is $\leq k$-heapable. We will write heapable instead of 1-heapable.

First we note that heapability of signed permutations is not a simple extension of heapability of ordinary permutations:

Observation 2 *Heapability of permutation σ is not equivalent to heapability of any fixed signed-version of σ. In particular if $\tau_0[i] = 1$ for all $i = 1, \ldots, n$ then every signed permutation (σ, τ_0) is n-heapable (even though σ might be heapable as a permutation). This is because the sign condition forces all elements of σ to start new heaps, as all values, having the same sign, cannot be inserted as children of any other nodes.*

On the other hand a connection between ordinary heapability and that of signed permutations does exist after all:

Theorem 16. *The following are true:*

- *If (σ, τ) is heapable (as a signed permutation) then σ is heapable (as an ordinary permutation).*
- *There is a polynomial time algorithm that takes as input a permutation $\sigma \in S_n$ and produces a sign $\tau : [n] \to \{\pm 1\}$ such that if σ is heapable (as an ordinary permutation) then (σ, τ) is heapable (as a signed permutation).*

Proof. – Trivial, since ordinary heapability does not care about sign restrictions.

– A simple consequence of the greedy algorithm for (ordinary) heapability [9]: roughly speaking, given a permutation σ there is a canonical way of attempting to construct a min-heap for σ: each element $\sigma(i)$ is added as a child of the largest element $\sigma(j) < \sigma(i)$, $j < i$ that still has a free slot. If no such $\sigma(j)$ is available then σ is not heapable. Further, denote by $p[i]$ the index of $\sigma(j)$ in the permutation σ (i.e., $p[i] = j$).
We use this algorithm to construct signing τ inductively. Specifically, set $\tau(1) = 1$. Given that we have constructed $\tau(j)$ for all $j < i$, define $\tau(i) = -\tau(p[i])$. Then (σ, τ) is heapable as a signed permutation, since the greedy solution for σ is legal for (σ, τ).

□

5.1 A greedy algorithm for the optimal heap decomposition of signed permutations

We now give a greedy algorithm for decomposing signed permutations into a minimal number of heap-ordered k-ary binary trees. The algorithm is presented in Figure 4 and is based on the concept of *slots*. These are positions where a new number can be inserted. So adding one number to a heap removes one slot and creates k new ones.

Each x_i will either be added to an existing heap or will start a new heap. We search for the highest value less than x_i having opposite sign to $\sigma(i)$, and we will create k *slots of value* $\sigma(i)$, allowing the subsequent insertion of values in the interval $[\sigma(i), \infty)$.

Example 17. Consider the sequence of insertions in a heap-ordered tree for a permutation (σ, τ) with initial prefix $\sigma(1) = 1, \tau(1) = -1$, $\sigma(2) = 8, \tau(2) = +1$, $\sigma(3) = 15, \tau(3) = -1$, shortly $\sigma = \{1, 8, 15\}$ and $\tau = \{-, +, -\}$ and $k = 2$. There is essentially a unique way (displayed in Figure 3) to insert these elements successively into a single heap: $\sigma(1)$ at the root, $\sigma(2)$ as a child of $\sigma(1)$, $\sigma(3)$ as a child of $\sigma(2)$. Note that $\sigma(3)$ needs to be a child of $\sigma(2)$ since the condition $\tau(3) = -1$ forbids, e.g., placing $\sigma(3)$ as a child of $\sigma(1)$.

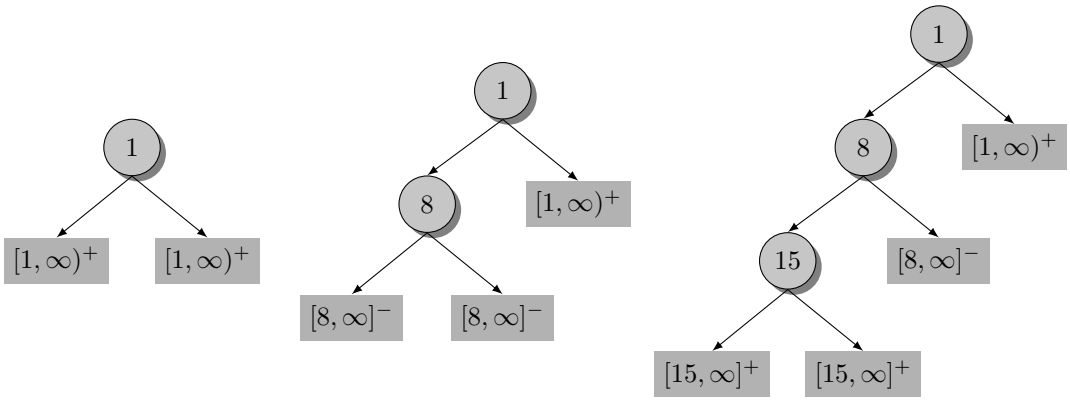

Figure 3. Nodes and slots.

Intuitively the interval of a slot describes the constraints imposed on an integer to be inserted in that position in order to satisfy the heap constraint. For example, after inserting $\sigma(2)$ the unique free slot of the root has value 1, since all children of the root must have values bigger than $\sigma(1) = 1$. After inserting $\sigma(3)$ the unique free slot of node hosting $\sigma(2)$ has value 8, since any descendant of this node must have value at least 8.

Our main result of this section is:

Theorem 18. *The algorithm GREEDY, presented in Fig. 4, decides the heapability of an arbitrary signed permutation and, more generally, constructs an optimal heap decomposition of* (σ, τ).

Proof. The intuition of the proof is that inserting integers into a heap the greedy way makes the collection of available slots grow "as slowly as possible" compared to any other insertion method. We will formalize this by introducing a concept of dominance between multisets and show that dominance is preserved by one insertion step. Since our slots have polarities we have to actually argue separately for the multisets of positive and negative multisets. The importance of dominance is the following: consider the scenario where a newly inserted element (by the greedy algorithm) creates a new heap because no compatible slots are available. Then by dominance no compatible slots are available even in the optimal insertion schedule. Hence greedy creates no more heaps than the optimal solution.

Proving correctness of the algorithm requires the following

$GREEDY(\sigma, \tau)$:

INPUT: $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ a permutation in S_n and
$\tau = (\tau(1), \tau(2), \ldots, \tau(n))$ a list of $\{+, -\}$ signs

start with empty heap forest $T = \emptyset$
for $i = 1, \ldots, n$ **do**
    if there exists a slot where $\sigma(i)$ can be inserted, according to the sign $\tau(i)$ **then**
        insert $\sigma(i)$ in the slot with the largest compatible value
    else
        start a new heap consisting of $\sigma(i)$ only.

Figure 4. The greedy algorithm for decomposing a signed permutation into a forest of heaps.

Definition 19. *Given a heap forest T, define the positive signature of T denoted $sig^+(T)$, to be the vector containing the (values of) free slots with positive polarity in T, sorted in non-decreasing order. The negative signature of T, denoted by $sig^-(T)$, is defined analogously.*

Given two binary heap forests T_1, T_2, T_1 dominates T_2 if

- *$|sig^+_{T_1}| \leq |sig^+_{T_2}|$ and inequality $sig^+_{T_1}[i] \leq sig^+_{T_2}[i]$ holds for all $1 \leq i \leq |sig^+_{T_1}|$.*
- *Similarly for negative signatures.*

Lemma 20. *Let T_1, T_2 be two heap-order forests such that T_1 dominates T_2. Insert a new element x greedily in T_1 (i.e., at the largest slot with value less or equal to x, or as the root of a new tree, if no such slot exists). Also insert x into an arbitrary compatible slot in T_2. These two insertions yield heap-ordered forests T'_1, T'_2, respectively. Then T'_1 dominates T'_2.*

Proof. We need the following definition

Definition 21. *Given integer $k \geq 1$, a k-multiset is a multiset A such that each element of A has multiplicitly at most k, that is a function $f : A \to \{0, 1, \ldots, k\}$.*

Definition 22. *Given multiset A, a Hammersley insertion of an element x into A is the following process:*

- *x is given multiplicity k in A.*
- *Some element $y \in A$, $y > x$ (if any such y exists), has its multiplicity reduced by 1.*

It is a greedy Hammersley insertion if (when it exists) y is the smallest element of A larger than x having positive multiplicity.

Definition 23. *Given two k-multisets A_1 and A_2 of elements of Γ_k, we define A_i^+, A_i^- the submultisets of A_i, $i = 1, 2$ consisting of elements of A_i with positive (negative) polarity only. We say that A_1 dominates A_2 iff:*

- *$|A_1^+| \leq |A_2^+|$ and inequality $A_1^+[i] \leq A_2^+[i]$ holds for all $1 \leq i \leq |A_1^+|$. In order for this equation to make sense, we see the multisets A_1^+, A_2^+ as sorted vectors.*
- *Similarly for negative polarities.*

The following result was not explicitly stated in [14] but was implicitly proved, as the basis of the proof of Lemma 1 from [14] (the analog of Lemma 20 for the unsigned case):

Lemma 24. *If A_1 and A_2 are multisets of integers such that A_1 dominates A_2. Consider x an element not present in either set and let A'_1, A'_2 be the result of greedy Hammersley insertion into A_1 and an arbitrary Hammersley insertion into A_2.*

Then A'_1 dominates A'_2.

Rather than repeating the proof, we refer the reader to [14]. To be able to prove Lemma 20 we also need the following:

Lemma 25. *Let A_1 and A_2 be multisets of integers such that A_1 dominates A_2. The following are true:*

- *Let A_1 and A_2 be k-multisets of integers and let x be an integer that does not appear in A_1, A_2. Then, if A'_1, A'_2 represent the result of inserting x **with multiplicity** k in A_1, A_2, respectively, then A'_1 dominates A'_2.*
- *Let A_1 and A_2 be k-multisets of integers and let x be an integer that appears in both A_1, A_2 with the same multiplicity. Then, if A'_1, A'_2 represent the result of deleting x from A_1, A_2, respectively, then A'_1 dominates A'_2.*

Proof. We prove the two statements as follows:

- It is clear that inserting x adds k to the cardinality of both A_1, A_2. Since $|A_1| \leq |A_2|$, we have $|A'_1| \leq |A'_2|$
  Let i_1 be the smallest index such that $A_1(i_1) > x$ and let i_2 be the smallest index such that $A_2(i_2) > x$. Because A_1 dominates A_2, $i_1 \geq i_2$.
  Recall that we regard multisets as sorted vectors. To prove that domination holds after the insertion of x as well, we have to prove, therefore, that $A'_1[i] \leq A'_2[i]$ for all indices $1 \leq i \leq |A'_1|$. Indeed, let i be such an index:
  - Case 1: $i < i_2$. Then $A'_1[i] = A_1[i]$ and $A'_2[i] = A_2[i]$. Indeed, all these elements are smaller than x. The desired inequality follows from the fact that A_1 dominates A_2.
  - Case 2: $i \geq i_1 + k$. Then $A'_1[i] = A_1[i-k]$ and $A'_2[i] = A_2[i-k]$. Inequality follows from the fact that A_1 dominates A_2.
  - Case 3: $i_2 \leq i < i_1$: Then $A'_1[i] = A_1[i] < x$ and $A'_2[i] = x$. So $A'_1[i] \leq A'_2[i]$.
  - Case 4: $i_1 \leq i < i_1 + k$: Then $A'_1[i] = x$ and $A'_2[i] \geq A'_2[i_1] \geq A'_2[i_2] = x$. So $A'_1[i] \leq A'_2[i]$.
- It is clear that x subtracts k to the cardinality of both A_1, A_2. To prove domination we have to, therefore, prove that $A'_1[i] \leq A'_2[i]$ for all $1 \leq i \leq |A'_1|$. Let i_1 be the smallest index such that $A_1(i_1 + k) > x$ and i_2 be the smallest index such that $A_2(i_2 + k) > x$. i_1, i_2 are well-defined since A_1, A_2 contain k copies of x. Because A_1 dominates A_2, $i_1 \geq i_2$. The first position of x in A_1 is i_1, the last is $i_1 + k - 1$; the first position of x in A_2 is i_2, the last is $i_2 + k - 1$.
  - Case 1: $i < i_2$. Then $A'_1[i] = A_1[i]$ and $A'_2[i] = A_2[i]$. Inequality follows from the fact that A_1 dominates A_2.
  - Case 2: $i \geq i_1$. Then $A'_1[i] = A_1[i-k]$ and $A'_2[i] = A_2[i-k]$. Inequality follows from the fact that A_1 dominates A_2.
  - Case 3: $i_2 \leq i < i_1$: Then $A'_1[i] = A_1[i] < x$ and $A'_2[i] = A_2[i+k] > x$. So $A'_1[i] \leq A'_2[i]$.
  - Case 4: $i_1 \leq i < i_1 + k$: Then $A'_1[i] < x$ and $A'_2[i] \geq A'_2[i_1] \geq A'_2[i_2] = A_2[i_2 + k] > x$. So $A'_1[i] \leq A'_2[i]$.

Now the proof of Lemma 20 follows: the effect of inserting an element with positive polarity x^+ greedily into T_1 can be simulated as follows:

– perform a greedy Hammersley insertion of x^- into $sig^-(T_1)$.
– remove x^- from $sig^-(T_1)$, and insert x^+ into $sig^+(T_1)$.

On the other hand we can insert x^+ into T_2 as follows:

– perform a Hammersley insertion of x^- into $sig^-(T_2)$.
– remove x^- from $sig^-(T_2)$, and insert x^+ into $sig^+(T_2)$.

By applying Lemma 24 and 25 we infer that after the insertion of x^+ T_1' dominates T_2'. The insertion of an element with negative polarity is similar.

□

Using Lemma 20 we can complete the proof of Theorem 18 as follows: by domination, whenever no slot of T_1 can accommodate x (which, thus, starts a new tree) then the same thing happens in T_2 (and thus x starts a new tree in T_2 as well). So the greedy algorithm is at least as good as any sequence of insertions, which means it is optimal.

□

5.2 Connection with the Signed Hammersley process

We are now in a position to explain what role does the signed Hammersley process play in the Ulam-Hammersley problem for the heap decomposition of signed permutations: to each heap-ordered forest F associate a word w_F over Γ_k^* which describes the relative positions of slots in F. Specifically, sort the leaves of F according to their value. If leaf f has at a certain moment $p \leq k$ slots of, say, negative polarity, then encode this into w_F by letter p^-.

Example 26. Let us consider the trees of Example 17. For the first tree the associated word is 2^+. For the second tree the word is 1^+2^-. For the third it is $1^+1^-2^+$.

Observation 3 *Note that when inserting a new element into the heap forest using the greedy algorithm the associated word changes exactly according to the signed Hammersley process*[2]*. In fact we can say more: starting a new heap-ordered tree corresponds to moments when we insert a new k^+ or k^- (whichever is appropriate at the given moment) without subtracting any 1 from the current word. So, if z is the outcome of a sequence of greedy insertions then $trees(z)$, the number of heap-ordered k-ary trees created in the process is equal to $\lambda^+ + \lambda^-$ (in the notation of equations 3 and 4), and is equal to*

$$trees_k(z) = |z|_k - \sum_{i=1}^{k} i \cdot |z|_{k-i} + \sum_{i=1}^{k} |z|_{k-i} = |z|_k - \sum_{i=1}^{k} (i-1)|z|_{k-i}$$

Of course, each word z may arise with a different multiplicity in the signed Hammersley process. So to compute the expected number of heap-ordered k-ary trees generated

[2] or, rather, the signed Hammersley process that removes a one to the left, rather than to the right. We would have to use min-heaps to obtain the signed Hammersley process. But this change is inconsequential.

by using the greedy algorithm with a random signed permutation of length n as input we have to compute amount

$$Z_n^k := \frac{1}{2^n \cdot n!} \sum_{z \in (\Gamma^k)^n} F_k(z) \cdot trees_k(z). \tag{7}$$

The last equation in Observation 3 is what motivated us to give Algorithm 2 for computing the formal power series F_k. We defer, however, the problem of experimentally investigating the scaling behavior of Z_n^k as $n \to \infty$ using Algorithm 2 to subsequent work.

6 Conclusions

The main contribution of this paper is to show that a significant number of analytical tools developed for the analysis of the Ulam-Hammersley problem for heapable sequences [14,7] extend to the case of signed permutations. While going along natural lines, the extension has some moderately interesting features: for instance the languages in the sign case appear to have slightly higher grammatical complexity than those in [7] for the ordinary process.

The obvious continuation of our work is the investigation of the Ulam-Hammer -sley problem for signed (and, more generally, colored) permutations. We do not mean only the kind of experiments alluded to in Observation 3. For instance it is known that the scaling in the case of ordinary permutations is logarithmic [4,5], even though the proportionality constant is not rigorously known. Obtaining similar results for the signed Hammersley (tree) process would be, we believe, interesting.

On the other hand, as we noted, our extension to signed permutations is not a direct version of the one in [8]. Studying a variant of our problem consistent with the model in [8] (or studying the longest increasing subsequence in the model we consider) is equally interesting.

References

1. D. Aldous and P. Diaconis: *Hammersley's interacting particle process and longest increasing subsequences*. Probability theory and related fields, 103(2) 1995, pp. 199–213.
2. J. Baik, P. Deift, and T. Suidan: *Combinatorics and random matrix theory*, vol. 172, American Mathematical Soc., 2016.
3. J. Balogh, C. Bonchiş, D. Diniş, G. Istrate, and I. Todinca: *On the heapability of finite partial orders*. Discrete Mathematics and Theoretical Computer Science, 22(1) 2020, paper # 17.
4. A.-L. Basdevant, L. Gerin, J.-B. Gouéré, and A. Singh: *From Hammersley's lines to Hammersley's trees*. Probability Theory and Related Fields, 2016, pp. 1–51.
5. A.-L. Basdevant and A. Singh: *Almost-sure asymptotic for the number of heaps inside a random sequence*. Electronic Communications in Probability, 23(17) 2018.
6. J. Berstel and C. Reutenauer: *Noncommutative rational series with applications*, vol. 137, Cambridge University Press, 2011.
7. C. Bonchiş, G. Istrate, and V. Rochian: *The language (and series) of Hammersley-type processes*, in Proceedings of the 8th Conference on Machines, Computation and Universality (MCU'18), vol. 10881 of Lecture Notes in Computer Science, 2018.
8. A. Borodin: *Longest increasing subsequences of random colored permutations*. The Electronic Journal of Combinatorics, 6(1) 1999, p. R13.

9. J. Byers, B. Heeringa, M. Mitzenmacher, and G. Zervas: *Heapable sequences and subseqeuences*, in Proceedings of the Eighth Workshop on Analytic Algorithmics and Combinatorics (ANALCO'2011), SIAM Press, 2011, pp. 33–44.
10. K. Chandrasekaran, E. Grigorescu, G. Istrate, S. Kulkarni, Y.-S. Lin, and M. Zhu: *Fixed-parameter algorithms for longest heapable subsequence and maximum binary tree*, in 15th International Symposium on Parameterized and Exact Computation, 2020.
11. K. Chandrasekaran, E. Grigorescu, G. Istrate, S. Kulkarni, Y.-S. Lin, and M. Zhu: *The maximum binary tree problem.* Algorithmica, 83(8) 2021, pp. 2427–2468.
12. J. M. Hammersley et al.: *A few seedlings of research*, in Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics, 1972.
13. M. A. Harrison: *Introduction to formal language theory*, Addison-Wesley Longman Publishing Co., Inc., 1978.
14. G. Istrate and C. Bonchiş: *Partition into heapable sequences, heap tableaux and a multiset extension of Hammersley's process*, in Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching (CPM'15), Ischia, Italy, vol. 9133 of Lecture Notes in Computer Science, Springer, 2015, pp. 261–271.
15. G. Istrate and C. Bonchiş: *Heapability, interactive particle systems, partial orders: Results and open problems*, in Proceedings of the 18th International Conference on Descriptional Complexity of Formal Systems (DCFS'2016), Bucharest, Romania, vol. 9777 of Lecture Notes in Computer Science, Springer, 2016, pp. 18–28.
16. J. Porfilio: *A combinatorial characterization of heapability*, Master's thesis, Williams College, May 2015, available from `https://librarysearch.williams.edu/permalink/01WIL_INST/1htsahc/alma991013795933102786`.
Accessed: July 2024.
17. D. Romik: *The surprising mathematics of longest increasing subsequences*, Cambridge University Press, 2015.
18. A. Salomaa and M. Soittola: *Automata-theoretic aspects of formal power series*, Springer Science & Business Media, 2012.
19. S. M. Ulam: *Monte carlo calculations in problems of mathematical physics.* Modern Mathematics for the Engineers, 261 1961, p. 281.
20. L. G. Valiant and M. S. Paterson: *Deterministic one-counter automata.* Journal of Computer and System Sciences, 10(3) 1975, pp. 340–350.