RAIRO-Theor. Inf. Appl. 44 (2010) 525–543
DOI: 10.1051/ita/2011004

Available online at:
www.rairo-ita.org

A CAT ALGORITHM FOR THE EXHAUSTIVE GENERATION OF ICE PILES*

Paolo Massazza[1] and Roberto Radicioni[1]

Abstract. We present a CAT (constant amortized time) algorithm for generating those partitions of n that are in the *ice pile model* $\mathrm{IPM}_k(n)$, a generalization of the *sand pile model* $\mathrm{SPM}(n)$. More precisely, for any fixed integer k, we show that the negative lexicographic ordering naturally identifies a tree structure on the lattice $\mathrm{IPM}_k(n)$: this lets us design an algorithm which generates all the ice piles of $\mathrm{IPM}_k(n)$ in amortized time $O(1)$ and in space $O(\sqrt{n})$.

Mathematics Subject Classification. 05A17, 68R99.

1. Introduction

In this paper, we consider the problem of generating particular integer partitions that are called *ice piles*. Ice piles were introduced by Goles *et al.* in [6], where they defined the ice pile model (IPM) as a generalization of the sand pile model (SPM), a discrete dynamical system which describes the behaviour of a simple game (the sand pile game).

The sand pile game simulates the fall of sand grains organized into adjacent columns of decreasing heights and has a simple evolution rule: if there are two adjacent columns, say i and $i+1$, with heights differing by at least 2, a grain can fall down from column i to column $i+1$ (the game starts with n grains stacked in column 1).

SPM was introduced by Goles and Kiwi [5] as a restriction of the discrete dynamical model proposed in 1973 by Brylawski [2] in order to study the set of

Keywords and phrases. Sand pile model, ice pile model, integer partitions, exhaustive generation, CAT algorithms, discrete dynamical systems.

* *Partially supported by Project M.I.U.R. PRIN 2007–2009: Mathematical aspects and forthcoming applications of automata and formal languages.*

[1] Università degli Studi dell'Insubria, Dipartimento di Informatica e Comunicazione, Via Mazzini 5, 21100 Varese, Italy; {paolo.massazza,roberto.radicioni}@uninsubria.it

Article published by EDP Sciences

the linear partitions of an integer n. SPM has been widely studied in physics and in the theory of cellular automata to represent granular objects, and also analyzed from a combinatorial point of view, see [1,5–7].

IPM generalizes SPM by introducing an additional rule. Let k be a fixed integer, then a grain can *slide* from column i with height p to column $i+k'$ with height $p-2$ if and only if $k' \leq k$ and all the columns from $i+1$ to $i+k'-1$ have height $p-1$. For any k, the set of linear partitions obtained from (n) by applying the previous transition rules is denoted by $\mathrm{IPM}_k(n)$. Moreover, $\mathrm{IPM}_k(n)$ turns out to be a lattice with respect to a suitable ordering (the dominance ordering) induced by these rules [6].

Many combinatorial properties of IPM_k are known. In particular, useful bounds for the number of ice piles in $\mathrm{IPM}_k(n)$ have been obtained in [3], together with area-and-length, area-and-height and length-and-height generating functions.

Here we study the problem of the exhaustive generation of $\mathrm{IPM}_k(n)$ by means of a CAT (constant amortized time) algorithm. We recall that a CAT algorithm for generating sand piles in $\mathrm{SPM}(n)$ using $O(n)$ space has been recently presented in [8]. We follow a similar approach and, by exploiting the negative lexicographic ordering, we show that a tree structure associated with the lattice $\mathrm{IPM}_k(n)$ naturally arises: this lets us design an algorithm that sequentially generates all the elements of $\mathrm{IPM}_k(n)$ in $O(1)$ amortized time using $O(\sqrt{n})$ space.

2. Preliminaries

A linear partition of n is a non-increasing sequence of positive integers $s = (s_1, \ldots, s_l)$ such that $\sum_{i=1}^{l} s_i = n$; its *height* is s_1 and its *length* $l(s)$ is l. The *height difference* of s at i is defined as $\delta_i(s) = s_i - s_{i+1}$ (assume $s_i = 0$ for $i > l(s)$).

Let $s = (s_1, \ldots, s_l)$ and $k > 0$. On the set of linear partitions we consider two (partial) functions defined as follows

$$\mathrm{Fall}(s,i) = \begin{cases} (s_1, \ldots, s_{i-1}, s_i - 1, s_{i+1} + 1, \ldots, s_l) & \text{if} \quad 1 \leq i \leq l, \\ & \qquad \delta_i(s) \geq 2 \\ \qquad \bot & \qquad \text{otherwise} \end{cases}$$

and

$$\mathrm{Slide}_k(s,i) = (s_1, \ldots, s_{i-1}, \underbrace{p, p, \ldots, p}_{k'+2}, s_{i+k'+2}, \ldots, s_l)$$

if there is $k' < k$ such that

$$s = (s_1, \ldots, s_{i-1}, p+1, \underbrace{p, p, \ldots, p}_{k'}, p-1, s_{i+k'+2}, \ldots, s_l)$$

otherwise $\mathrm{Slide}_k(s,i) = \bot$.

For any two integers n, k, the *ice pile model* $\mathrm{IPM}_k(n)$ is defined as the set of linear partitions of n (called ice piles) obtained by closing $\{(n)\}$ with respect to

Fall and Slide$_k$. Given $s \in \mathrm{IPM}_k(n)$ and two integers i, j, with $1 \leq i < j \leq l(s)$, we denote by $s[i, j]$ the subsequence $(s_i, s_{i+1}, \ldots, s_j)$. Moreover, we indicate by $p^{[h]}$ the sequence $(\underbrace{p, p, \ldots, p}_{h})$, that is, a *block* of h adjacent columns of equal height p, and by $\cdot$ the *catenation* product,

$$(a_1, \ldots, a_r) \cdot (b_1, \ldots, b_p) = (a_1, \ldots, a_r, b_1, \ldots, b_p).$$

Ice piles have been characterized in [6], Thm. 3. Here we are interested in a characterization expressed in terms of forbidden patterns, as shown in Figure 1 and in the following:

Theorem 2.1. *A linear partition of n belongs to $IPM_k(n)$ if and only if it does not contain any subsequence of type*

$$p^{[k+2]} \quad or \quad (p+1)^{[k+1]} \cdot p^{[k+1]} \quad or \quad (p+h)^{[k+1]} \cdot \prod_{i=1}^{h-1} (p+h-i)^{[k]} \cdot p^{[k+1]}$$

with $p > 0$ and $h > 1$.

Proof. Directly follows from Conditions II.1 and II.2 in [6], Thm. 3. □

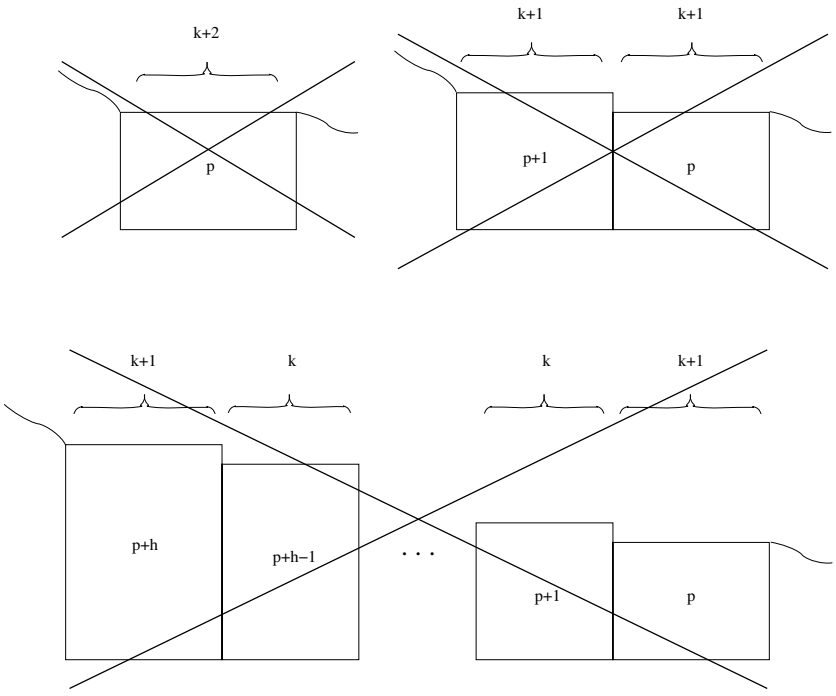

FIGURE 1. $\mathrm{IPM}_k(n)$: forbidden patterns.

The following result concerning subsequences of ice piles can be seen as an immediate consequence of Theorem 2.1, although a direct proof could be easily provided.

Corollary 2.2. *Let $s = (s_1, \dots, s_l) \in IPM_k(n)$. Then, for $1 \le i < j \le l$ we have $s[i,j] \in IPM_k(m)$ where $m = \sum_{e=i}^{j} s_e$.*

$\mathrm{IPM}_k(n)$ has a unique *fixed point i.e.* a configuration where no grain can fall or slide, characterized in [6], Prop. 2 and it turns out to be a lattice with respect to the *dominance ordering* $\ge$, usually defined as

$$s \ge t \quad \Longleftrightarrow \quad \sum_{i=1}^{j} s_i \ge \sum_{i=1}^{j} t_i, \quad j = 1, \dots, \max(l(s), l(t)).$$

In the sequel we are interested in a particular total ordering on $\mathrm{IPM}_k(n)$ known as the negative lexicographic or *neglex* ordering, $<_{\mathrm{nlex}}$, which is defined as follows:

$$(s_1, \dots, s_l) <_{\mathrm{nlex}} (t_1, \dots, t_m)$$

if and only if

$$\exists i,\ 1 \le i \le \min(l, m) :\ \forall j,\ 1 \le j < i,\ s_j = t_j,\ s_i > t_i.$$

Moreover, we write $s \le_{\mathrm{nlex}} t$ if $s <_{\mathrm{nlex}} t$ or $s = t$. It is immediate to note that if $s, t \in \mathrm{IPM}_k(n)$ and $s > t$ (dominance ordering) then $s <_{\mathrm{nlex}} t$.

The *set of moves* of an ice pile s is defined as

$$\mathrm{M}(s) = \{e | 1 \le e \le l(s),\ \mathrm{Fall}(s, e) \ne \perp\ \vee\ \mathrm{Slide}_k(s, e) \ne \perp\}.$$

An ice pile s with $|\mathrm{M}(s)| > 1$ is called *branching*. The index of the rightmost column of s where a move is possible is denoted by $\mathrm{Rmost}(s) = \max(\mathrm{M}(s))$. For any integer i we fix $\mathrm{M}(s)_{>i} = \{e \in \mathrm{M}(s) \mid e > i\}$ and $\mathrm{M}(s)_{<i} = \{e \in \mathrm{M}(s) \mid e < i\}$. In the sequel, we write $s \overset{i}{\Rightarrow} t$ if $t = \mathrm{Slide}_k(s, i)$ or $t = \mathrm{Fall}(s, i)$. As a matter of fact, the effect of a move is local, in the sense that if $s \overset{i}{\Rightarrow} t$ then the symmetric difference between $\mathrm{M}(s)$ and $\mathrm{M}(t)$, $\mathrm{M}(s) \triangle \mathrm{M}(t)$, contains at most six entries. Formally, we have:

Lemma 2.3. *Let $s \overset{i}{\Rightarrow} t$ and $k' = \max\{j < i \ | j = 0 \vee \delta_j(s) > 0\}$. Moreover, if $\delta_i(s) > 1$ set $k'' = 1$ else $k'' = \min\{j > i \ | s_j = s_i - 2)\}$. Then, one has*

$$M(s) \triangle M(t) \subseteq \{i - k', i - 1, i, i + 1, i + k'' - 1, i + k''\}.$$

Proof. Both s and t consist of the same sequence except $s_i = t_i + 1$ and $s_{i+k''} = t_{i+k''} - 1$ ($k'' = 1$ if $t = \mathrm{Fall}(s, i)$ or $1 < k'' \le k$ if $t = \mathrm{Slide}(s, i)$). Thus, it is immediate to see that for all $j \notin \{i - 1, i, i + k''\}$ we have $\mathrm{Fall}(s, j) \ne \perp$ if and only if $\mathrm{Fall}(t, j) \ne \perp$. Similarly, for all $j \notin \{i - k', i - 1, i, i + 1, i + k'' - 1, i + k''\}$ we have $\mathrm{Slide}(s, j) \ne \perp$ if and only if $\mathrm{Slide}(t, j) \ne \perp$. □

From now on, we consider the sequence of ice piles in $\mathrm{IPM}_k(n)$ ordered with respect to $<_{\mathrm{nlex}}$ and denote the hth element by $s^{(h)}$. We conclude this section by recalling the following property (see [5]).

Lemma 2.4. *For any $s \in IPM_k(n)$ we have $l(s) = O(\sqrt{n})$.*

2.1. A spanning tree for $\mathrm{IPM}_k(n)$

In Section 4, we exhibit an algorithm that generates all the ice piles in ascending neglex order, by traversing an implicitly defined spanning tree of the lattice $\mathrm{IPM}_k(n)$. In this subsection, we formally define this spanning tree by means of the notions of "father" and "grand ancestor" of an ice pile.

Definition 2.5. The function $f : \mathrm{IPM}_k(n) \setminus \{(n)\} \mapsto \mathrm{IPM}_k(n)$ given by

$$f(s) = \max{}_{<_{\mathrm{nlex}}}\{t \in \mathrm{IPM}_k(n) \mid \exists i, t \overset{i}{\Rightarrow} s\}$$

is called the *father function*.

Figure 2 illustrates the spanning tree associated with $\mathrm{IPM}_3(10)$, obtained by drawing a thick edge between s and $f(s)$, for all $s \in \mathrm{IPM}_3(10)$. In the sequel, we often consider the set of ice piles belonging to a subtree with a specific root s, hence:

Definition 2.6. Given $s \in \mathrm{IPM}_k(n)$, we denote by $T(s)$ the set

$$T(s) \;=\; \{s\} \cup \bigcup_{s=f(x)} T(x).$$

The natural distinction between internal nodes and leaves is held by the notion of *grand ancestor*.

Definition 2.7. Let $t \in \mathrm{IPM}_k(n)$. The grand ancestor of t is the ice pile

$$a(t) = \min{}_{<_{\mathrm{nlex}}}\{s \in \mathrm{IPM}_k(n) \mid s[1,i] = t[1,i], i = \mathrm{Rmost}(t) \in M(s)\}.$$

It is immediate to note that either $a(t) = t$ or $a(t) <_{\mathrm{nlex}} t$. We say that an ice pile s is a *proper* grand ancestor if and only if there is $t \neq s$ such that $s = a(t)$. We underline the ith column of s if and only if s is the grand ancestor of $t \neq s$ and $i = \mathrm{Rmost}(t)$: in this case we say that s is a grand ancestor at i. Symmetrically, an ice pile s is a grand ancestor at i if and only if

$$s = \min{}_{<_{\mathrm{nlex}}}\{t \in \mathrm{IPM}_k(n) \mid t[1,i] = s[1,i], i \in M(t)\}.$$

Informally, all the ice piles which are internal nodes in the spanning tree turn out to be grand ancestors, while leaves correspond to those ice piles which are not grand ancestors. These facts will be proved in Lemma 2.10 and Corollary 2.14, respectively.

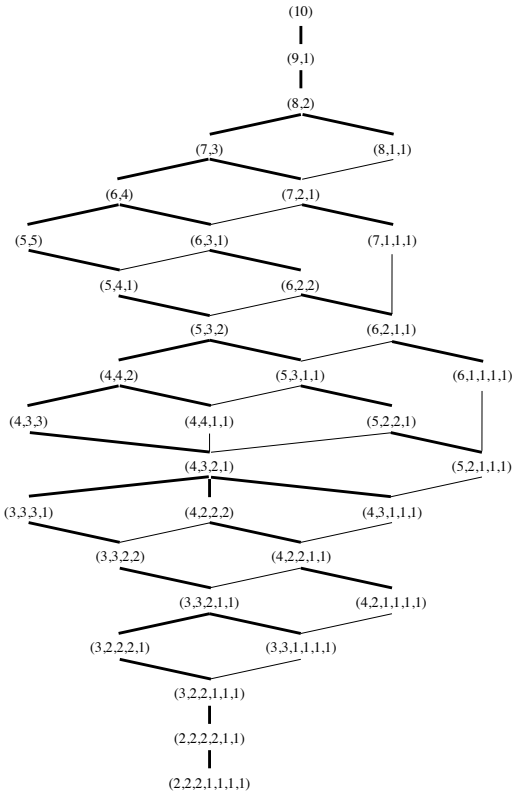

FIGURE 2. The spanning tree of $\mathrm{IPM}_3(10)$.

Example 2.8. Let us consider the ice piles in $\mathrm{IPM}_3(10)$ and list them according to the neglex ordering:

$(10), (9,1), (\underline{8},2), (8,1,1), (\underline{7},3), (7,2,1), (7,1,1,1), (\underline{6},4), (6,3,1)(6,2,2),$
$(6,2,1,1), (6,1,1,1,1), (5,5), (5,4,1), (\underline{5},3,2), (5,3,1,1), (5,2,2,1),$
$(5,2,1,1,1), (4,\underline{4},2), (4,4,1,1), (4,3,3), (\underline{4},\underline{3},2,1), (4,3,1,1,1), (4,2,2,2),$
$(4,2,2,1,1), (4,2,1,1,1,1), (3,3,3,1), (3,3,2,2), (3,\underline{3},2,1,1),$
$(3,3,1,1,1,1), (3,2,2,2,1), (3,2,2,1,1,1), (2,2,2,2,1,1), (2,2,2,1,1,1,1).$

For instance, with respect to Figure 2, we have $a((6,1,1,1,1)) = (\underline{6},4)$, $a((6,2,2)) = (6,2,2)$ and $a((4,2,1,1,1,1)) = a((4,3,1,1,1)) = (\underline{4},\underline{3},2,1)$. Look at the internal nodes of the spanning tree: proper grand ancestors turn out to be branching, while ice piles satisfying the relation $t = a(t)$ are not branching.

The father of an ice pile s is determined by the rightmost grain in s which can be moved to the left by reversing either a Slide or a Fall. Usually, this grain belongs to one of the last two blocks of s, with one exception occurring when s admits a suffix of particular shape. The four possible cases which arise are depicted in Figure 3 and proved in the next lemma.

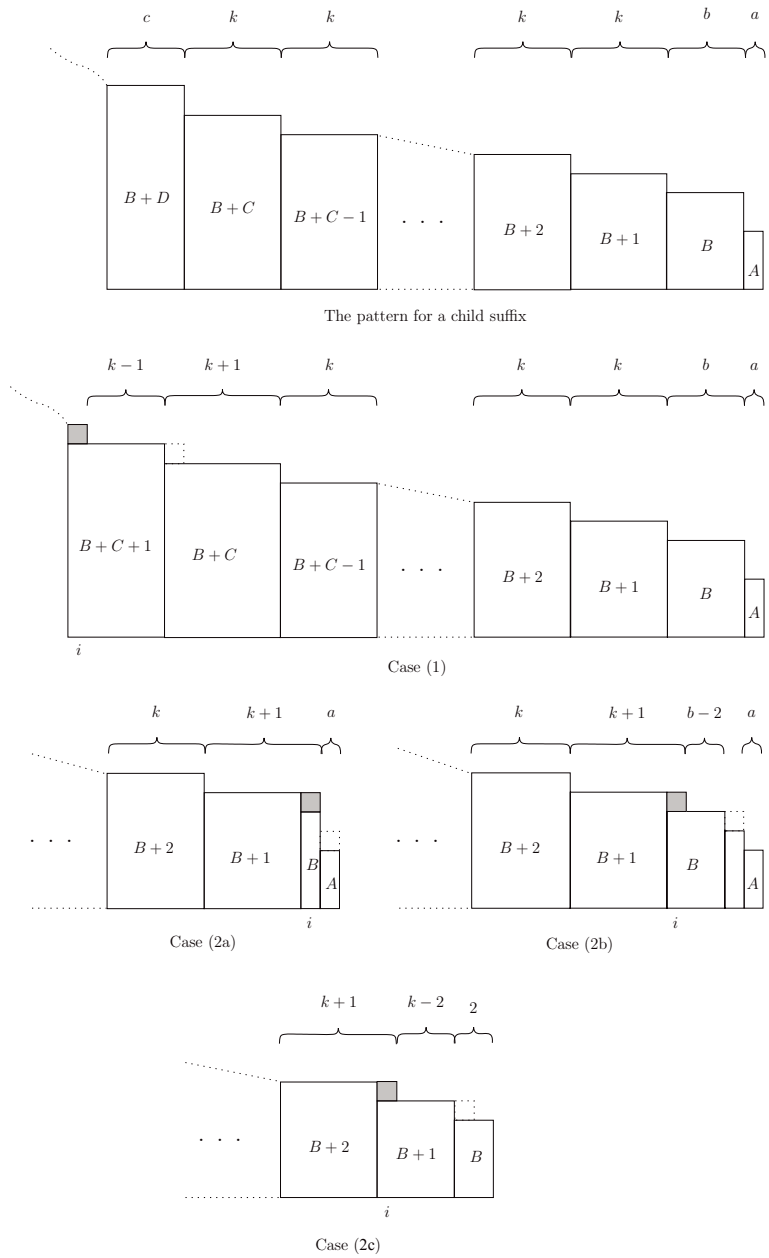

FIGURE 3. Finding the father: suffix shape from Lemma 2.9.

Lemma 2.9. *Let $t \in IPM_k(n)$, $l = l(t)$ and consider the longest suffix w of t such that*

$$w = (B+D)^{[c]} \prod_{1 \le j \le C} (B+j)^{[k]}\ B^{[b]} A^a$$

for suitable integer values A, B, C, D, a, b, c satisfying the constraints

$$0 < A < B, \quad 0 \le C, \quad 0 \le a \le 1, \quad 0 \le b < k, \quad 0 < c \le k+1$$

and either $D = C+1$ and $c \neq k$ or $D > C+1$.

Then, the ice pile $s = f(t)$ and the position i such that $s \overset{i}{\Rightarrow} t$ are determined as follows:

(1) *if $c = k+1$ and $D = C+1$ then $i = l - |w| + 1 = l - (a+b+Ck+c) + 1$ and*

$$s = t[1, i-1](B+C+2)(B+C+1)^{[k-1]}(B+C) \prod_{1 \le j \le C} (B+j)^{[k]}\ B^{[b]} A^a;$$

(2) *if $c < k$ or $D > C+1$, then:*
   (a) *if $a = b = 1$ we have $i = l - 1$ and*

$$s = t[1, i-1](B+1)(A-1);$$

   (b) *if $b > 1$ we have $i = l - a - b + 1$ and*

$$s = t[1, i-1](B+1)B^{[b-2]}(B-1)A^a;$$

   (c) *if $b = 1$ and $a = 0$ we have $i = l - k + 1$ and*

$$s = t[1, i-1](B+2)(B+1)^{[k-2]}B^2.$$

Proof. First, we have $s \in \mathrm{IPM}_k(n)$ since it satisfies Theorem 2.1. In fact, it is immediate to see that, in all cases, if one forbidden pattern occurred in s then it would also occur in t. Then, $s \overset{i}{\Rightarrow} t$ by construction. Moreover, $s = f(t)$. Indeed, as suggested by Figure 3, it is clear that in cases (2a), (2b) and (2c) there is not an ice pile s' such that $s' \overset{j}{\Rightarrow} t$ and $j > i$, while in case (1) s' would contain a forbidden pattern of type $p^{[k+1]}(p-1)^{[k]}(p-2)^{[k]} \dots (p-h+1)^{[k]}(p-h)^{[k+1]}$. □

The father of an ice pile has several interesting properties; in particular, it turns out to be a grand ancestor as stated in the following:

Lemma 2.10. *Let $t \in IPM_k(n)$ and $s = f(t)$ with $s \overset{i}{\Rightarrow} t$. Then s is a grand ancestor at i.*

Proof. We argue by contradiction and suppose that the grand ancestor at i with prefix $s[1,i]$ is $x <_{\mathrm{nlex}} s$. Let $\bar{\imath} > i$ be the smallest index for which $x[\bar{\imath}] > s[\bar{\imath}]$.

Obviously we have $\delta_{\bar{\imath}-1}(s) > 0$ and $\bar{\imath} < l(s)$. We recall Lemma 2.9 and distinguish four cases. In case (1),

$$s = v(B+C+2)(B+C+1)^{[k-1]}(B+C)^{[k+1]}\cdots(B+2)^{[k]}(B+1)^{[k]}B^{[b]}A^{a}$$

and $i = l(v)+1$. Since $i \in M(x)$, one has $\bar{\imath} \neq i+1, i+k$. For all the other positions, $x[\bar{\imath}] > s[\bar{\imath}]$ implies the existence in x of the forbidden pattern $p^{[k+1]}(p-1)^{[k]}(p-2)^{[k]}\ldots(p-h+1)^{[k]}(p-h)^{[k+1]}$. Case (2) is even simpler since $i = l(s)-1$ and then x would not belong to $\mathrm{IPM}_k(n)$. A similar analysis holds in case (3) and in case (4), since $x[\bar{\imath}] > s[\bar{\imath}]$ implies $i \notin M(x)$. □

The following lemma plays a fundamental role for the correctness of the algorithm, since it shows how to visit the spanning tree in order to obtain the neglex ordered sequence of all ice piles.

Lemma 2.11. *Let $t, v \in IPM_k(n)$. Then:*

(1) *$f(t) <_{nlex} t$;*

(2) *if $f(t) = f(v) = s$ and $s \overset{j}{\Rightarrow} t$, $s \overset{i}{\Rightarrow} v$ with $i < j$, then for all $x \in T(t) \setminus \{t\}$ and $y \in T(v) \setminus \{v\}$, we have*

$$t <_{nlex} x <_{nlex} v <_{nlex} y.$$

Proof. Statement (1) immediately follows from the definitions of $<_{\mathrm{nlex}}$ and f, and implies the relations $t <_{\mathrm{nlex}} x$ and $v <_{\mathrm{nlex}} y$ in statement (2). So, we have only to prove $x <_{\mathrm{nlex}} v$. By contradiction, let $\hat{x}$ be the smallest ice pile in $T(t)$ such that $v <_{\mathrm{nlex}} \hat{x}$ and consider its father $\tilde{x} = f(\hat{x})$. Note that along the path from t to $\hat{x}$ the move $\tilde{x} \overset{p}{\Rightarrow} \hat{x}$ is the first which is not to the right of i, that is, $p \leq i$. By recalling Lemma 2.10, $\tilde{x}$ is a grand ancestor at p: this implies $p < i$ since the grand ancestor at i with prefix $\tilde{x}[1,i]$ is $s <_{\mathrm{nlex}} \tilde{x}$.

Finally, consider the ice pile $\bar{x} = f(\tilde{x})$, $\bar{x} \overset{r}{\Rightarrow} \tilde{x} \overset{p}{\Rightarrow} \hat{x}$ with $r > i > p$, and note that $p \in M(\bar{x})$ since $\delta_i(\bar{x}) \geq 1$. Then, we have $\bar{x} \overset{p}{\Rightarrow} z \overset{r}{\Rightarrow} \hat{x}$ with $z <_{\mathrm{nlex}} \tilde{x}$ and this implies $\tilde{x} \neq f(\hat{x})$. □

In particular, from statement (2) of Lemma 2.11 we directly obtain:

Corollary 2.12. *A preorder visit of the spanning tree of $IPM_k(n)$ produces the neglex ordered sequence of all ice piles.*

2.2. Properties of grand ancestors

Lemma 2.10 lets us walk through the branches of the spanning tree of $\mathrm{IPM}_k(n)$ only by moving a grain in column i in an ice pile s which is a grand ancestor at i. If the current ice pile t satisfies $t = a(t)$ we move at column $\mathrm{Rmost}(t)$ in t and, by Lemma 2.3, we update the set of moves in constant time; if t is a leave, in order to continue the visit of the tree we have to reconstruct the grand ancestor of t at $\mathrm{Rmost}(t)$. We will prove that the cost of reconstructing a grand ancestor is bounded by the distance between t and $a(t)$ in the tree.

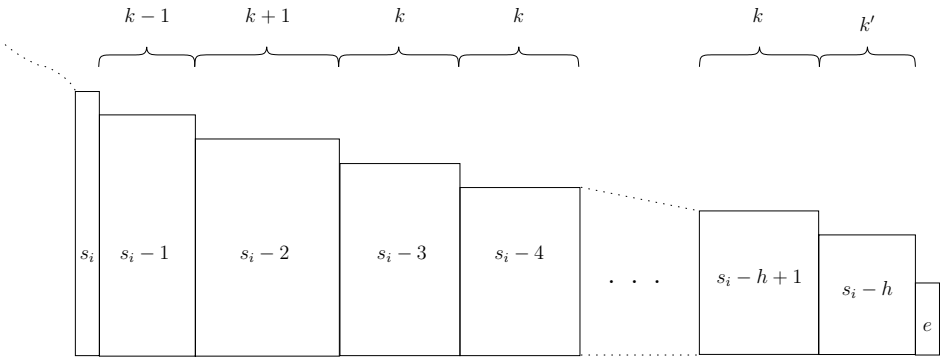

FIGURE 4. $\mathrm{IPM}_k(n)$: the suffix of a proper grand ancestor.

In this subsection we formalize these topics, basically by showing that the set of fathers of ice piles (*i.e.* the internal nodes of the tree) coincides with the set of grand ancestors. We first present a lemma which characterizes the suffixes of grand ancestors: as shown in Figure 4, "long" suffixes look like stairs with a first block of width $k-1$, a second one of width $k+1$, and then a sequence of blocks of decreasing height and width k (with possibly exceptions regarding the last two blocks). "Short" suffixes have a simpler structure consisting of either a single column or a block of width at most $k-1$ followed by at most two columns. Moreover, the lemma also illustrates how the value s_i and the number of grains in the suffix $s[i+1, l(s)]$ of a grand ancestor s at i univocally determine the subsequence $(s_{i+1}, \ldots, s_{l(s)})$.

Lemma 2.13. *Let $s \in IPM_k(n)$ be a grand ancestor at $i = Rmost(t)$, $s = a(t)$, with $d = \sum_{j=i+1}^{l(t)} t_j = \sum_{j=i+1}^{l(s)} s_j$. Then we have:*

- *if $d < s_i - 1$, then $l(s) = i+1$ and $s_{i+1} = d$;*
- *if $s_i - 1 \le d < k(s_i-1) - 1$, then $s[i+1, l(s)] = (s_i-1)^{[k']} \cdot (s_i-2) \cdot (e)$, with*

$$k' = \left\lfloor \frac{d - s_i + 2}{s_i - 1} \right\rfloor, \qquad e = d - k'(s_i - 1) - s_i + 2;$$

- *if $d \ge k(s_i-1) - 1$, then*

$$s[i+1, l(s)] \;=\; (s_i-1)^{[k-1]} \cdot (s_i-2) \cdot \prod_{j=2}^{h-1} (s_i-j)^{[k]} \cdot (s_i-h)^{[k']} \cdot e, \qquad (2.1)$$

with

$$h = \left\lfloor \left(s_i - \frac{1}{2}\right)\left(1 - \sqrt{1 - \frac{d+1}{k(4s_i-2)^2}}\right)\right\rfloor, \tag{2.2}$$

$$k' = \left\lfloor \frac{d - k(h-1)(s_i - h/2) + 1}{s_i - h}\right\rfloor, \tag{2.3}$$

$$e = d - k(h-1)(s_i - h/2) + 1 - k'(s_i - h). \tag{2.4}$$

Proof. By definition, the grand ancestor of t is the smallest ice pile s such that $s[1,i] = t[1,i]$ and $i \in M(s)$. So, the idea is that we start from column $i+1$ and proceed from left to right, setting each column as high as possible, while ensuring $i \in M(s)$ and avoiding forbidden patterns.

Trivially, if $d < s_i - 1$ the grand ancestor at i is $s = t[1,i]\cdot(d)$, that is, $l(s) = i+1$ and $s_{i+1} = d$.

Otherwise, if $s_i - 1 \le d < k(s_i-1)-1$, then we set $s_{i+1} = s_i - 1$ only if $d \ge 2s_i - 3$ (otherwise $i \notin M(s)$). More generally, we set $s_j = s_i - 1$ for $j = i+1, i+2, \ldots i+k'$ only if we can place $s_i - 2$ grains in column $i + k' + 1$ (so that $\mathrm{Slide}(s,i) \neq \perp$). Moreover, observe that $d < k(s_i - 1) - 1$ implies $k' < k$. In other words, k' is the largest integer smaller than $(d - s_i + 2)/(s_i - 1)$. Hence, we place k' columns of height $s_i - 1$, then a column of height $s_i - 2$ and, finally, a column with the remaining $d - k'(s_i - 1) - (s_i - 2)$ grains. The resulting partition satisfies the characterization of Theorem 2.1.

Now, consider the case $d \ge k(s_i - 1) - 1$. First we set $s[i+1, i+k] = (s_i - 1)^{[k-1]} \cdot (s_i - 2)$ (this guarantees $\mathrm{Slide}(s,i) \neq \perp$), then we place $k(s_i - 2)$ grains in $s[i+k+1, i+2k] = (s_i - 2)^{[k]}$, $k(s_i - 3)$ grains in $s[i+2k+1, i+3k] = (s_i - 3)^{[k]}$, and so on as long as there are enough grains to fill each block. Thus, the number of grains in $s[i+1, i+(h-1)k]$, is

$$\begin{aligned} f(h) &= (k-1)(s_i - 1) + (k+1)(s_i - 2) + \sum_{j=3}^{h-1} k(s_i - j) \\ &= \sum_{j=1}^{h-1} k(s_i - j) - 1 = k(h-1)(s_i - h/2) - 1. \end{aligned}$$

Since there are d grains in the suffix, the number $h-1$ of blocks of maximal width satisfies the relation $f(h) \le d < f(h+1)$. By solving the two inequalities we obtain the value in Expression (2.2). The remaining grains lie in a block of height $s_i - h$ or in the last column. The width of this last block is obtained by dividing the number of remaining grains by $s_i - h$, as shown in Expression (2.3). Last column is filled with the rest, as stated in Expression (2.4).

In all cases, minimality is ensured by construction and no forbidden pattern arises. □

The shape of the suffix of a grand ancestor provided by the previous lemma directly leads to the following

Corollary 2.14. *Let s be a grand ancestor at i and $s \overset{i}{\Rightarrow} t$. Then, $s = f(t)$.*

Indeed, the grain involved in the move $s \overset{i}{\Rightarrow} t$ is exactly the rightmost grain in t which can be moved back without introducing a forbidden pattern (see Fig. 4).

Grand ancestors have further interesting properties. First, their suffixes contain at most four positions where a move possibly occurs. Second, if s is a grand ancestor at i, by moving the grain in column i one obtains the smallest ice pile with that prefix of length i. More formally we have:

Lemma 2.15. *Let $s \in IPM_k(n)$ be a grand ancestor at i, $l = l(s)$ and $s \overset{i}{\Rightarrow} t$. Then, one has*

(1) $M(s)_{>i} \subseteq \{l - \langle l - i\rangle_k, l-2, l-1, l\}$;
(2) $v \in IPM_k(n) \wedge v[1,i] = t[1,i] \implies t \leq_{nlex} v$.

Proof.

(1) We follow the three cases of Lemma 2.13. The first two cases are trivial and we have $\mathrm{M}(s)_{>i} \subseteq \{l-2, l-1, l\}$. In the third case, depicted in Figure 4, for each j, with $i < j < l-1$, we have $\delta_j(s) \leq 1$ and then $\mathrm{Fall}(s,j) = \perp$. Moreover, each column j such that $\delta_j(s) = 1$ is always followed by a block of width either $k+1$ or k (and so $\mathrm{Slide}_k(s,j) = \perp$), unless $j = l - \langle l-i\rangle_k$. Therefore, since $\delta_j(s) = 0$ for $l - \langle l-i\rangle_k < j < l-1$, we have $\mathrm{M}(s)_{>i} \subseteq \{l - \langle l-i\rangle_k, l-1, l\}$.
(2) Note that $s = f(t)$ and then, by considering the four cases in Lemma 2.9, the two conditions $v[1,i] = t[1,i]$ and $v <_{\mathrm{nlex}} t$ imply that either v admits a forbidden pattern or $v \in \mathrm{IPM}_k(m)$ with $m > n$.

□

The following lemma lets us easily compute the ordered sequence of ice piles in $\mathrm{IPM}_k(n)$. Such a sequence is obtained by defining, for any $h > 0$, $s^{(h+1)}$ as either $\mathrm{Fall}(a(s^{(h)}), i)$ or $\mathrm{Slide}(a(s^{(h)}), i)$, with $i = \mathrm{Rmost}(s^{(h)})$.

Lemma 2.16. *Let $s^{(h)} \in IPM_k(n)$ and $i = Rmost(s^{(h-1)})$. Then we have*

(1) $a(s^{(h-1)}) \overset{i}{\Rightarrow} s^{(h)}$;
(2) $f(s^{(h)}) = a(s^{(h-1)})$.

Proof.

(1) Let $s^{(h-1)} = (s_1, \ldots, s_l)$, $d = \sum_{j>i} s_j$ and $m = l(s^{(h)})$. By definition, one has $a(s^{(h-1)})[1,i] = s^{(h-1)}[1,i]$ and so the move $a(s^{(h-1)}) \overset{i}{\Rightarrow} t$ implies $t[1,i] = (s_1, \ldots, s_{i-1}, s_i - 1)$. Now, suppose $s^{(h-1)} <_{\mathrm{nlex}} s^{(h)} \leq_{\mathrm{nlex}} t$ and note that this implies $s^{(h)}[1, i-1] = t[1, i-1] = (s_1, \ldots, s_{i-1})$ and $s_i - 1 \leq s_i^{(h)} \leq s_i$. If $s_i^{(h)} = s_i$ then we have $s^{(h-1)}[i+1, l] <_{\mathrm{nlex}} s^{(h)}[i+1, m]$, which is obviously impossible since $s^{(h-1)}[i+1, l], s^{(h)}[i+1, m] \in \mathrm{IPM}_k(d)$ and $s^{(h-1)}[i+1, l]$ is the bottom ($M(s^{(h-1)}[i+1, l]) = \emptyset$). So, we have

$s^{(h)}[1,i] = t[1,i]$ and, by statement (2) in Lemma 2.15, the relation $t \leq_{\text{nlex}} s^{(h)}$ holds, that is, $s^{(h)} = t$.

(2) Suppose $s = f(s^{(h)}) \neq a(s^{(h-1)})$ and $s \overset{j}{\Rightarrow} s^{(h)}$ with $j \neq i$. Statement (1) and $j < i$ imply $s <_{\text{nlex}} a(s^{(h-1)})$ and so s would not be the father of $s^{(h)}$. Thus, we have $j > i$ and $s^{(h)}[1,i] = s[1,i]$. Statement (2) in Lemma 2.15 immediately leads to the contradiction $s^{(h)} \leq_{\text{nlex}} s$.

□

3. Exhaustive generation

Here we come to our goal, that is, the design of a CAT algorithm which, for any fixed integer k, solves the following problem:

Problem: ice piles generation (IPG_k)
Input: an integer n
Output: the ordered sequence of ice piles in $\text{IPM}_k(n)$.

Note that because of the exponential growth of $|\text{IPM}_k(n)|$ (see [3], Thm. 1) a standard approach based on dynamic programming is not feasible due to the huge space requirement. Nevertheless, we show that we can easily define an iterative CAT algorithm that traverses the spanning tree associated with $\text{IPM}_k(n)$ and produces the sequence of ice piles

$$s^{(1)} = (n),\ s^{(2)} = (n-1,1),\ \ldots,\ s^{(m)},$$

where $s^{(m)}$ is the unique fixed point of $\text{IPM}_k(n)$. This result extends the method introduced in [8] for the distributive lattice $\text{SPM}(n)$ to the (non modular) lattice $\text{IPM}_k(n)$. The main difference is that the formal definition of a tree structure associated with $\text{IPM}_k(n)$ (see Def. 2.5) together with the characterization of the suffixes of grand ancestors in Lemma 2.13, let us lower the space requirement from $O(n)$ to $O(\sqrt{n})$, while providing a simpler proof of the CAT property.

The algorithm is based on Lemma 2.16, that is, it starts from $s^{(1)} = (n)$ and, for any $h > 1$, it computes $s^{(h)}$ by moving a grain in column $\text{Rmost}(s^{(h-1)})$ in $a(s^{(h-1)})$. As shown in Corollary 2.12, this corresponds to a preorder visit of the spanning tree of $\text{IPM}_k(n)$. To this aim, a crucial point is that of testing for each ice pile s in the sequence whether $s = a(s)$. This can be done by maintaining, at each step h, an array $GrAn$ of couples of integers that univocally identify the suffixes of all generated ice piles which are proper grand ancestors at $j \leq \text{Rmost}(s^{(h)})$ and share with $s^{(h)}$ a prefix of length j.

More formally, if $s^{(h)}$ is the current ice pile and $i = \text{Rmost}(s^{(h)})$ then the entries of $GrAn$ are defined as

$$GrAn[j] = \begin{cases} (p,q) & \text{if } j \leq i \wedge \exists g \leq h \text{ s.t. } s^{(g)} \text{ is a proper grand ancestor at } j, \\ & s^{(g)}[1,j] = s^{(h)}[1,j], l(s^{(g)}) = p, s^{(g)}_p = q, \\ (0,0) & \text{otherwise.} \end{cases}$$

Thus, a single iteration consists of three steps:

(1) Determine the grand ancestor at i with prefix $s^{(h)}[1,i]$: if $GrAn[i] = (0,0)$ then $a(s^{(h)}) = s^{(h)}$, otherwise construct $a(s^{(h)})$ from $s^{(h)}[1,i]$ and $(p,q) = GrAn[i]$ and then set $GrAn[i] := (0,0)$;
(2) Make the move $a(s^{(h)}) \stackrel{i}{\Rightarrow} s^{(h+1)}$;
(3) Check whether $s^{(h+1)}$ is a proper grand ancestor and update $GrAn$: for $e \in \mathrm{M}(s^{(h+1)})$, $i-k \le e < \mathrm{Rmost}(s^{(h+1)})$, if $GrAn[e] = (0,0)$ set $GrAn[e] := (l(s^{(h+1)}), s^{(h+1)}_{l(s^{(h+1)})})$.

Example 3.1. Let us consider $\mathrm{IPM}_2(8)$. Starting with $s^{(1)} = (8)$, we proceed to $s^{(2)} = (7,1)$ and then to the branching ice pile $s^{(3)} = (\underline{6},2)$ (note that this is the grand ancestor of the ice pile t having (6) as a prefix and with $\mathrm{Rmost}(t) = 1$). Now we have $\mathrm{M}(s^{(3)}) = \{1,2\}$ and we set $GrAn[1] := (2,2)$. Once we have computed $s^{(4)} = \mathrm{Fall}(s^{(3)}, \mathrm{Rmost}(s^{(3)})) = (6,1,1)$, the next ice pile is obtained by a move in the first column. Actually, since $GrAn[1] = (2,2)$, we proceed as follows: first, we construct $a(s^{(3)}) = (6,2)$ and set $GrAn[1] := (0,0)$, then we compute $s^{(5)} = \mathrm{Fall}((6,2),1) = (\underline{5},3)$. Now, we have $\mathrm{M}(s^{(5)}) = \{1,2\}$ and set $GrAn[1] := (2,3)$. The process continues by computing the ice piles

$$
\begin{array}{lcll}
s^{(6)} &=& \mathrm{Fall}(s^{(5)}, \mathrm{Rmost}(s^{(5)})) &= (5,2,1) \quad (GrAn = [(2,3)]),\\
s^{(7)} &=& \mathrm{Slide}_2(s^{(6)}, \mathrm{Rmost}(s^{(6)}) &= (5,1,1,1) \quad (GrAn = [(2,3)]),\\
s^{(8)} &=& \mathrm{Fall}(s^{(5)}, \mathrm{Rmost}(s^{(7)})) &= (4,4) \quad (GrAn = [(0,0)]),\\
s^{(9)} &=& \mathrm{Fall}(s^{(8)}, \mathrm{Rmost}(s^{(8)})) &= (4,3,1) \quad (GrAn = [(0,0)]),\\
s^{(10)} &=& \mathrm{Fall}(s^{(9)}, \mathrm{Rmost}(s^{(9)})) &= (\underline{4},2,2) \quad (GrAn = [(3,2)]),\\
s^{(11)} &=& \mathrm{Fall}(s^{(10)}, \mathrm{Rmost}(s^{(10)})) &= (4,2,1,1) \quad (GrAn = [(3,2)]),\\
s^{(12)} &=& \mathrm{Fall}(s^{(10)}, \mathrm{Rmost}(s^{(11)})) &= (3,3,2) \quad (GrAn = [(0,0)]),\\
s^{(13)} &=& \mathrm{Fall}(s^{(12)}, \mathrm{Rmost}(s^{(12)})) &= (3,3,1,1) \quad (GrAn = [(0,0)]),\\
s^{(14)} &=& \mathrm{Fall}(s^{(13)}, \mathrm{Rmost}(s^{(13)})) &= (3,2,2,1) \quad (GrAn = [(0,0)]),\\
s^{(15)} &=& \mathrm{Slide}_2(s^{(14)}, \mathrm{Rmost}(s^{(14)})) &= (3,2,1,1,1) \quad (GrAn = [(0,0)]),\\
s^{(16)} &=& \mathrm{Slide}_2(s^{(15)}, \mathrm{Rmost}(s^{(15)})) &= (2,2,2,1,1) \quad (GrAn = [(0,0)]).
\end{array}
$$

4. The algorithm

The idea presented in the previous section leads immediately to Algorithm 1. In the code we use two arrays, *partition* and *GrAn*, a stack *St*, two procedures, GrAncestor and Check, and two functions, Move and MSet.

At each step the array of integers *partition* represents an ice pile which has a set of moves represented by an increasing sequence of integers contained in *St*. The elements of *GrAn* are couples of integers with the meaning given in the previous section.

GrAncestor(*partition*,i,l,m,q) sets the ice pile *partition* with rightmost move in i to its grand ancestor: it modifies the suffix *partition*$[i+1,l]$ according to Lemma 2.13 (knowing the length m of the proper grand ancestor and the height q of its last column). Move(*partition*,i,l) executes either Fall(*partition*,i) or

Algorithm 1 Exhaustive generation of $\mathrm{IPM}_k(n)$.

```
1: Procedure IcePileGeneration_k(n)
2: partition ← [n, 0, ..., 0]; GrAn ← [(0,0), ..., (0,0)];
3: St ← EmptyStack(); Push(St,1); l ← 1;
4: while IsNotEmpty(St) do
5:    i ← Top(St); Pop(St); (m, q) ← GrAn[i];
6:    if (m, q) ≠ (0, 0) then
7:       GrAncestor(partition,i, l, m, q);
8:       GrAn[i] ← (0, 0); l ← m;
9:    end if
10:   l ← Move(partition,i,l); Check(St,i);
11:   (l_1, ..., l_p) ← MSet(partition,i);
12:   for e = 1 to p − 1 do
13:      if GrAn[l_e] = (0, 0) then
14:         GrAn[l_e] ←(1,partition[l]);
15:         Push(St,l_e);
16:      end if
17:   end for
18:   Push(St,l_p);
19: end while
```

Slide(*partition*,i) and returns the length of the new generated ice pile. Check(St,i) is used to check the entries of St after the move. More precisely, it possibly eliminates the two top elements of St if they are no more valid after the move in column i (see Lem. 2.3). If $partition = s^{(h)}$ is an ice pile of length l obtained by a move in column $i = \mathrm{Rmost}(s^{(h-1)})$ in $a(s^{(h-1)})$, then MSet(*partition*,i, l) returns an ordered list of integers corresponding to $\mathrm{M}(s^{(h)}) \setminus \mathrm{M}(s^{(h-1)})$. By Lemma 2.3 this list contains at most six integers in the range $[i-k, l]$.

Given an integer n (k is fixed), we start with *partition*$=[n, 0, \ldots, 0]$ (the first ice pile in $\mathrm{IPM}_k(n)$) and by pushing 1 onto St (lines 2-3, we suppose $n > 1$). We also set the length l of the ice pile to 1. Then, as long as St is not empty, an integer i is popped: this indicates the rightmost move in the latest generated ice pile (line 5). If *partition* has a proper grand ancestor then $GrAn[i] \neq (0, 0)$ and GrAncestor (line 7) restores it. Hence, we generate the next ice pile (line 10) and update the set of moves. Note that all the integers in St denote valid moves, with possibly two exceptions regarding the two elements at the top (the call to Check in line 10 fix them). So, The other valid moves are found in the suffix $partition[i-k, l]$ (the list returned by MSet, line 11). Finally, St and $GrAn$ are updated (lines 12-17).

The correctness of the algorithm is proved in the following theorem.

Theorem 4.1. *IcePileGeneration$_k(n)$ solves IPG_k.*

Proof. We prove, by induction on h, that at the end of the hth iteration we have

(1) $partition = s^{(h+1)}$;

(2) $St \equiv \mathrm{M}(s^{(h+1)})$, that is, St contains the ordered sequence of integers in $\mathrm{M}(s^{(h+1)})$, with $\mathrm{Top}(St) = \mathrm{Rmost}(s^{(h+1)})$;
(3) $GrAn[j] = (l, p) \neq (0, 0)$ if and only if there is $e \leq h + 1$ such that $s^{(e)}$ is a proper grand ancestor at j of length l with $s^{(e)}[1, j] = s^{(h+1)}[1, j]$, $j \leq \mathrm{Rmost}(s^{(h+1)})$ and $s^{(e)}_l = p$.

Basis ($h = 1$). At the end of the first iteration we have

(1) *partition* $= (n - 1, 1)$;
(2) St is either empty (if $n = 2$ or $n = 3$ and $k = 1$) or contains the integer 1;
(3) $GrAn[j] = (0, 0)$ for all j (no proper grand ancestor has been generated).

Induction ($h > 1$). By induction hypothesis, at the beginning of the hth iteration we have *partition* $= s^{(h)}$ and $St \equiv \mathrm{M}(s^{(h)})$ (with $\mathrm{Top}(St) = \mathrm{Rmost}(s^{(h)})$). Moreover, $GrAn$ contains all the couples of integers that univocally identify all the proper grand ancestors at $j \leq \mathrm{Rmost}(s^{(h)})$ with prefix $s^{(h)}[1, j]$ ($GrAn[j] = (0, 0)$ for $j > \mathrm{Rmost}(s^{(h)})$). Then,

(1) i is assigned the value $\mathrm{Top}(St) = \mathrm{Rmost}(s^{(h)})$ and the algorithm moves a grain in column i in $a(s^{(h)})$. In fact, if $GrAn[i] = (0, 0)$ then $s^{(h)} = a(s^{(h)})$ and the move executed in line 10 is $s^{(h)} \stackrel{i}{\Rightarrow} s$. Otherwise, $GrAn[i] = (l, p)$ and (line 7) the grand ancestor at i is restored, say $s^{(e)} = a(s^{(h)})$, and then the move $s^{(e)} \stackrel{i}{\Rightarrow} s$ is executed. Lemma 2.16 states that $s = s^{(h+1)}$ and so *partition* $= s^{(h+1)}$ (line 10);
(2) after line 10, St contains only the values of $\mathrm{M}_{<i-k}(s^{(h+1)})$ together with the values of $\mathrm{M}(s^{(h)}) \cap \mathrm{M}(s^{(h+1)})$ in the range $[i - k, i]$. By definition, the call to MSET in line 11 returns the list $(l_1, \ldots, l_p)$ of integers in $\mathrm{M}(s^{(h+1)}) \setminus \mathrm{M}(s^{(h)})$ (in the range $[i-k, l]$). These values are then pushed in order onto St, yielding $St \equiv \mathrm{M}(s^{(h+1)})$;
(3) since $\mathrm{M}_{<i-k}(s^{(h+1)}) = \mathrm{M}_{<i-k}(s^{(h)})$ and $s^{(h+1)}[1, i - 1] = s^{(h)}[1, i - 1]$, the first $i - k - 1$ entries of $GrAn$ keep holding the right value. This is also true for the nonzero entries in the range $[i - k, i - 1]$. Finally, $s^{(h+1)}$ turns out to be a proper grand ancestor at e for all e belonging to the list returned by MSET (integers in the range $[i - k, l(s^{(h+1)})]$, see Lem. 2.3), with the only exception given by $e = l_p = \mathrm{Rmost}(s^{(h+1)})$. It is immediate to see that after the for loop (line 17) also the entries $GrAn[e]$, $i - k \leq e \leq \mathrm{Rmost}(s^{(h+1)})$, have assigned the right value.

□

4.1. Complexity

In order to determine the complexity of IcePileGeneration$_k$, we independently analyze MOVE, CHECK, MSET and GRANCESTOR, respectively. Clearly, MOVE admits an implementation running in time $O(1)$. The same remark holds for CHECK since it accesses at most two integers in St (at the top) and, for any integer i, the cost of testing whether $\mathrm{Fall}(\mathit{partition}, i) \neq \perp$ or $\mathrm{Slide}(\mathit{partition}, i) \neq \perp$ is $O(1)$. Then, Lemmas 2.3 and 2.15 lead to an implementation of MSET which

also runs in $O(1)$. Finally, the complexity of GrAncestor is not $O(1)$. Nevertheless, an amortized analysis shows that the cost of generating the sequence of all ice piles in $\mathrm{IPM}_k(n)$ by calling $\textsc{IcePileGeneration}_k(n)$ is $O(|\mathrm{IPM}_k(n)|)$. The proof is based on the following two lemmas.

Lemma 4.2. *For each leaf t in the spanning tree of $IPM_k(n)$, we can compute $a(t)$ in time $O(d)$ where d is the length of the path from t to $a(t)$.*

Proof. We assume that t is represented by an array of integers and we show how to modify it in order to obtain the representation of $s = a(t)$. So, we have as input t, $i = \mathrm{Rmost}(t)$ and $GrAn[i] = (p, q)$, where $p = l(s)$ and $q = s_{l(s)}$. Observe that the pair (p, q) determines the exact number of grains in t that lie in a different column in s: this is $N = |t_p - q| + \sum_{j=p+1}^{l(t)} t_j$ and is bounded by d, since each of these grains has been eventually moved during the derivation of t from s.

By Theorem 2.13, in nontrivial cases $s[i+1, p]$ is the sequence of blocks illustrated in Figure 4. So, we fix $x = p - \langle p - i \rangle_k + 1$ and $y = t_i - \lfloor (p-i)/k \rfloor$, and construct the sequence from right to left, starting from column p which has q grains and is preceded by $p-x$ columns of height $y-1$. Then, columns in positions from $x - jk - 1$ to $x - (j+1)k$ have height $y + j$, with $j = 0, \ldots, \lfloor (p-i)/k \rfloor$, except column $i + k$ that has height $t_i - 2$, see Figure 4.

So, we start from the rightmost column of t and proceed to construct s by moving all the grains we encounter in positions $l(t), l(t) - 1, \ldots, p + 1$ of t, while setting $l(t) - p$ entries of t to 0.

More precisely, s can be constructed by an iterative process that, at each iteration, moves a grain of t to its final destination in s. The number of iterations is N and at each step we have two indices, j_1 and j_2, denoting a column to fill (in s) and a column to empty (in t), respectively. As soon as a column gets the right height, the corresponding index takes a new value which identifies the first column (to the left) with a wrong height. Note that updating j_1 has cost $O(1)$. In fact, as soon as we get $t_{j_1} = s_{j_1} = a$, if $t_{j_1 - 1} = s_{j_1 - 1}$ we have only to check whether $t_h = s_h$ for the largest r such that $h = i + rk < j_1$. Therefore, it is immediate to see that the overall cost is $O(N) = O(d)$. □

Lemma 4.3. *Let $s^{(d)}, s^{(e)}$ be two leaves in the spanning tree of $IPM_k(n)$ and let P_d, P_e be the two paths joining $s^{(d)}$ to $a(s^{(d)})$ and $s^{(e)}$ to $a(s^{(e)})$, respectively. Then, P_d and P_e do not have a common edge.*

Proof. Lemma 2.16 states that $s^{(d+1)}$ is obtained by a move in column $i = \mathrm{Rmost}(s^{(d)})$ in $a(s^{(d)})$, and then we have $a(s^{(d)}) = f(s^{(d+1)})$. Thus, let

$$a(s^{(d)}) <_{\mathrm{nlex}} \cdots <_{\mathrm{nlex}} s^{(c)} <_{\mathrm{nlex}} \cdots <_{\mathrm{nlex}} s^{(d)} <_{\mathrm{nlex}} s^{(d+1)} <_{\mathrm{nlex}} \cdots <_{\mathrm{nlex}} s^{(e)}$$

where $s^{(c)}$ is the node preceding $a(s^{(d)})$ in the path from $s^{(d)}$ to $a(s^{(d)})$, $f(s^{(c)}) = a(s^{(d)})$. By Lemma 2.11, if $s^{(e)} \in T(s^{(c)})$ then $s^{(e)} <_{\mathrm{nlex}} s^{(d+1)}$. Therefore, $s^{(e)}$ does not belong to $T(s^{(c)})$ and this means that P_d and P_e do not have a common edge. □

We can now prove our main result.

Theorem 4.4. *For any fixed integer k,* IcePileGeneration$_k(n)$ *runs in time* $O(|IPM_k(n)|)$.

Proof. The number of iterations of IcePileGeneration$_k$ is exactly $|\mathrm{IPM}_k(n)|$. The only instruction which has not cost $O(1)$ is the call to GrAncestor (line 7). GrAncestor is called only on leaves and if it is called at the ith iteration on the leaf $s^{(i)}$ then, by Lemma 4.2, its cost is $O(|P_{s^{(i)}}|)$, where $|P_{s^{(i)}}|$ is the length of the path from $s^{(i)}$ to $a(s^{(i)})$.

Thus, if $T(n)$ denotes the running time of IcePileGeneration$_k(n)$ we have

$$T(n) \leq C_1|\mathrm{IPM}_k(n)| + \sum_{s \text{ is a leaf}} C_2|P_s|$$

where C_1, C_2 are suitable constants. Finally, by Lemma 4.3 each edge of the spanning tree belongs to at most one path P_s. Therefore, we obtain

$$T(n) \leq C_1|\mathrm{IPM}_k(n)| + C_2(|\mathrm{IPM}_k(n)| - 1) = O(|\mathrm{IPM}_k(n)|).$$

□

Lastly, the only data structures used in IcePileGeneration$_k$ are the arrays *partition*, *GrAn* and the stack *St*, all having size $O(\sqrt{n})$ (see Lem. 2.4). Thus, from the previous theorem we immediately get:

Corollary 4.5. *Procedure* IcePileGeneration$_k(n)$ *runs in* $O(1)$ *amortized time and uses* $O(\sqrt{n})$ *space.*

5. Conclusions

We considered in this paper the problem of generating all the reachable configurations of the discrete dynamical system $\mathrm{IPM}_k(n)$, showing a CAT algorithm working on a tree structure which naturally arises when considering the negative lexicographic ordering. Hence, it is quite natural to ask whether a similar approach can be applied to deal with the exhaustive generating problem for other (similar) discrete models.

In particular, the discrete models BSPM (bidimensional sand pile model) and BIPM (bidimensional ice pile model) have been introduced in [5] by adding a further dimension to $\mathrm{SPM}(n)$ and $\mathrm{IPM}_k(n)$, respectively. Thus, the elements of BSPM and BIPM are plane partitions, that is, matrices of non-negative integers that are non-increasing from top to bottom and from left to right. These models exhibit some important differences with respect to the unidimensional case. First, they are not lattices and several fixed points possibly exist; second, no characterization is known for reachable states and for fixed points. As a consequence, the presented algorithm can not be easily extended to these models. For instance, a first problem is that of finding a suitable ordering which lets us construct a tree structure associated with $\mathrm{BSPM}(n)$ (or $\mathrm{BIPM}_k(n)$).

An intermediate step towards the design of a CAT generation algorithm for $\mathrm{BSPM}(n)$ or $\mathrm{BIPM}_k(n)$ could be the study of a simpler problem, that is, the exhaustive generation of the plane partitions of an integer n in constant amortized time.

Acknowledgements. The authors would like to thank Roberto Mantaci of LIAFA for stimulating discussions.

References

[1] P. Bak, C. Tang and K. Wiesenfeld, Self-organized criticality. *Phys. Rev. A* **38** (1988) 364–374.
[2] T. Brylawski, The lattice of integer partitions. *Discrete Math.* **6** (1973) 201–219.
[3] S. Corteel and D. Gouyou-Beauchamps, Enumeration of sand piles. *Discrete Math.* **256** (2002) 625–643.
[4] E. Duchi, R. Mantaci, H.D. Phan and D. Rossin, Bidimensional sand pile and ice pile models. *PU.M.A.* **17** (2007) 71–96.
[5] E. Goles and M.A. Kiwi, Games on line graphs and sand piles. *Theoret. Comput. Sci.* **115** (1993) 321–349.
[6] E. Goles, M. Morvan and H.D. Phan, Sandpiles and order structure of integer partitions. *Discrete Appl. Math.* **117** (2002) 51–64.
[7] M. Latapy, R. Mantaci, M. Morvan and H.D. Phan, Structure of same sand piles model. *Theoret. Comput. Sci.* **262** (2001) 525–556.
[8] P. Massazza, A CAT algorithm for sand piles. *PU.M.A.* **19** (2008) 147–158.

Communicated by Ch. Choffrut.
Received February 7, 2009. Accepted January 13, 2011.