ELSEVIER

Theoretical Computer Science 292 (2003) 667–677

Theoretical Computer Science

www.elsevier.com/locate/tcs

Divide-and-conquer recurrences associated with generalized heaps, optimal merge, and related structures

Wei-Mei Chen[a], Gen-Huey Chen[b,*]

[a] *Department of Applied Mathematics, Tatung University, Taipei 104, Taiwan*
[b] *Department of Computer Science & Information Engineering, National Taiwan University, # 1 Roosevelt Rd. Sec. 4, Taipei 10764, Taiwan*

Received May 2000; received in revised form June 2001; accepted July 2001
Communicated by H. Prodinger

Abstract

An elementary approach is given to studying the recurrence relations associated with generalized heaps (or d-heaps), cost of optimal merge, and generalized divide-and-conquer minimization problems. We derive exact formulae for the solutions of all such recurrences and give some applications. In particular, we present a precise probabilistic analysis of Floyd's algorithm for constructing d-heaps when the input is randomly given. A variant of d-heap having some interesting combinatorial properties is also introduced.

Keywords: Divide-and-conquer; Generalized heap; Optimal merge; Huffman coding

1. Introduction

Let $\{g_n\}_{n\geqslant 1}$ be a given sequence. Consider the *heap recurrence* (cf. [14, 20])

$$f_n = f_{2^{\lfloor \log_2(2n/3)\rfloor}-1} + f_{n-2^{\lfloor \log_2(2n/3)\rfloor}} + g_n \quad (n \geqslant 2)$$

with f_1 given. Besides its close relationship with cost of algorithms on heaps, such a recurrence with minor modification (shifting n by 1) describes the cost of queue-mergesort proposed by Golin and Sedgewick [11] and the cost of the optimal merge predicted by Huffman coding as discussed by Glassey and Karp [8]. It also appeared naturally as the solution of the recurrence based on minimization

$$\phi_n = \min_{1\leqslant j\leqslant n/2}(\phi_j + \phi_{n-j}) + g_n \quad (n \geqslant 2)$$

* Corresponding author. Fax: +886-2-362-8167.
E-mail addresses: wmchen@ttu.edu.tw (W.-M. Chen), ghchen@csie.ntu.edu.tw (G.-H. Chen).

0304-3975/03/$ - see front matter
PII: S0304-3975(01)00336-X

with ϕ_1 given, when the sequence g_n is increasing and concave; see [2, 8, 13] for details. This paper is mainly concerned with generalizations of these problems to higher orders (or dimensions). We will give exact solutions for each of these recurrences and indicate some applications.

Consider first heaps. The usual heap structure, when viewed as binary trees, admits natural d-ary ($d \geqslant 3$) tree generalizations, which we will refer to as d-heaps in this paper; cf. [15, 17]. The d-heap is a d-ary tree with the *heap property*: the value of each node in the tree is not greater than that of its parent. Also keys in nodes are usually stored levelwise in a left-to-right order in contiguous locations, introducing another feature of heaps. In addition to their priority queue character, d-heaps are also suitable for sorting; it was observed that such sorting algorithms are more efficient than the usual heapsort ($d=2$) if the branching factor d is properly chosen (usually $d=3$ or 4); see [4, 10, 17–19]. Write $n=v_h+m$, where $v_h=(d^h-1)/(d-1)$ with $h=\lfloor \log_d((n-1)(d-1)+1)\rfloor$ and $1 \leqslant m \leqslant d^h$. Inheriting the structural characteristics of heap, d-heap has at most one incomplete subtree of the root and whose leaves all locate at the last two levels, and the sizes of the remaining complete subtrees of root are v_h or v_{h-1}. The combinatorial structure of d-heaps naturally leads to the following recurrence:

$$A_n = \left\lfloor \frac{m}{d^{h-1}} \right\rfloor A_{v_h} + \left(d - \left\lfloor \frac{m}{d^{h-1}} \right\rfloor - 1\right) A_{v_{h-1}} + A_\lambda + g_n \quad (n \geqslant 2),$$

where $\lambda = \lambda(n) = n - \lfloor m/d^{h-1} \rfloor v_h - (d - \lfloor m/d^{h-1} \rfloor - 1)v_{h-1} - 1$. Note that when $d = 2$, A_n reduces to f_n.

Consider now optimal merge. The optimum strategy for merging sorted files into a longer ordered file is a classical subject in computer algorithms; see [16] for more information. In particular, Knuth (cf. [16, Section 5.4.9]) used the usual Huffman coding algorithm to construct an optimal tree associated with the optimal merge pattern. Such an algorithm can be implemented with a queue, which in the case $d=2$ essentially reduces to the merge patterns in [8] for Huffman trees, in [21] for a variant of mergesort and in [11] for queue-mergesort. If the input subfiles are of unit length, then the optimal merge pattern naturally introduces the following recurrence: $B_1 = g_1$, $B_n = ng_1 + g_n$ for $1 < n < d$, and

$$B_n = \left\lfloor \frac{n-d^L}{d^{L-1}(d-1)} \right\rfloor B_{d^L} + \left(d - \left\lfloor \frac{n-d^L}{d^{L-1}(d-1)} \right\rfloor - 1\right) B_{d^{L-1}} + B_\gamma + g_n \qquad (n \geqslant d), \tag{1}$$

where $L = \lfloor \log_d n \rfloor$ and $\gamma = \gamma(n) = n - \lfloor (n-d^L)/(d^{L-1}(d-1)) \rfloor d^L - (d - \lfloor (n-d^L)/(d^{L-1}(d-1)) \rfloor - 1)d^{L-1}$, see Section 2.2 for details. Note that the recurrence is not the same as that for A_n. Also for $d=2$, B_n reduces to the cost recurrence of queue-mergesort [3].

The recurrences satisfied by A_n and B_n can also be viewed from the usual divide-and-conquer perspective: divide the given problem of size n into d subproblems with $g(n)$ the "merge" cost. Note that the associated d-ary trees of B_n and $A_{n+\lceil (n-1)/(d-1) \rceil}$

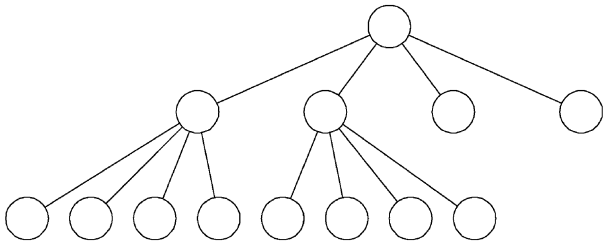

Fig. 1. The quaternary tree with 10 leaves represents the tree structure of 4-heap with $n = 13$ as well as the optimal merge tree for $n = 10$.

have the same shape (see Fig. 1), the difference being the number of external nodes (n) and all nodes ($n + \lceil (n-1)/(d-1) \rceil$). In particular, this relation holds for $d=2$.

In general, the cost of the optimal merge strategy can be formulated as the solution to the following divide-and-conquer recurrence with minimization

$$\mathscr{F}_n = \min_{n_1+\cdots+n_d=n} (\mathscr{F}_{n_1} + \mathscr{F}_{n_2} + \cdots + \mathscr{F}_{n_d}) + g_n, \tag{2}$$

where $1 \leqslant n_j < n$ for $1 \leqslant j \leqslant d$ and g_n is the merge cost. Batty et al. [1] showed that if g_n is increasing and convex then the minimum value of $\mathscr{F}_n$ is attained for $n_j = \lfloor (n + j - 1)/d \rfloor$, namely, n is divided as evenly as possible into d parts; on the other hand, if g_n is increasing and concave, then the best dividing rule is the underlying one of B_n, which might be termed as the "balanced power-of-d" rule. This generalizes the results in [13, 8]. In particular, for the optimal d-way merge pattern when given n files of the same length, we have $g_1 = 0$ and $g_n = n$; so that g_n is increasing and concave, implying that the cost of the best strategy is given by (1). The case $d=2$ is discussed in details in [3].

This paper is organized as follows. In Section 2, we derive exact solutions for the recurrences associated with d-heaps, the optimal merge strategy and the generalized divide-and-conquer minimization problem. We show that the structures of the optimal merge trees and d-heaps are essentially equivalent (in terms of their shapes). As applications of the exact solutions, a probabilistic analysis of an algorithm extending that of Floyd [6] for constructing heaps is given in Section 3 under the uniform permutation model. Finally, we study a variant of d-heaps in Section 4, introducing yet another type of recurrence.

2. d-heaps, optimum merging and recurrence with minimization

In this section, we study first the d-heap recurrence; then we discuss optimal merge patterns and the relationship between d-heap recurrence and optimal merge cost. We also solve the generalized divide-and-conquer minimization problems.

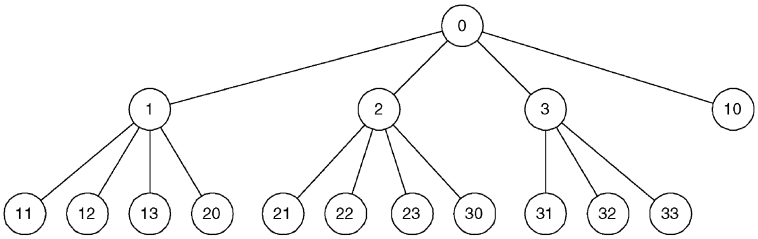

Fig. 2. The 4-heap with size $16=(100)_4$. All numbers are written in their quaternary representation.

2.1. Recurrence on d-heaps

As already described in the Introduction, a d-heap is an array with the heap property and all nodes except at most one have either 0 or d children (when viewed as a d-ary tree). Fig. 2 gives the structure of a 4-heap.

Such d-heaps differ from the usual binary heaps (2-heaps) in the following way: the children of the node with address p have addresses $dp+j$ for $j=1,\ldots,d$ instead of for $j=0,\ldots,d-1$ as in the binary case. Thus, we suggest a new generalized heap structure in the last section sharing the same addressing scheme with binary heaps.

A d-heap has at most one incomplete subtree at each level and its leaves appear only at the last two levels. Let $v_k=(d^k-1)/(d-1)$ and $n=v_h+m$ where $1\leqslant m\leqslant d^h$ and $h=\lfloor \log_d((n-1)(d-1)+1)\rfloor$. For a given sequence $\{g_n\}_{n\geqslant 1}$, the *d-heap recurrence* is of the form $A_0=0$, $A_1=g_1$, and

$$A_n=\left\lfloor\frac{m}{d^{h-1}}\right\rfloor A_{v_h}+\left(d-\left\lfloor\frac{m}{d^{h-1}}\right\rfloor-1\right)A_{v_{h-1}}+A_\lambda+g_n \quad (n\geqslant 2), \tag{3}$$

where $\lambda=\lambda(n)=v_{h-1}+m-\lfloor m/d^{h-1}\rfloor d^{h-1}$. To the best of our knowledge, this recurrence first appeared implicitly in [7] for optimal merge patterns and its optimality was shown in [9].

Lemma 1. *For $n\geqslant 2$, if $n=v_h=(d^h-1)/(d-1)$, the solution A_n of the d-heap recurrence* (3) *is given by*

$$A_{v_h}=\sum_{1\leqslant k\leqslant h} d^{h-k}g_{v_k}. \tag{4}$$

Proof. By (3), we have

$$\begin{aligned} A_{v_h} &= dA_{v_{h-1}}-A_{v_{h-1}}+A_{v_{h-1}}+g_{v_h}\\ &= dA_{v_{h-1}}+g_{v_h}. \end{aligned}$$

Iterating and simplifying yield formula (4). □

Let $\{x\}$ denote the fraction part of x.

Theorem 1. *For $n \geqslant 2$, the solution A_n of the d-heap recurrence* (3) *is given by*

$$A_n = \sum_{1 \leqslant k \leqslant h} \left(\left\lfloor \frac{n}{d^{k-1}} \right\rfloor - \left\lfloor \frac{n}{d^k} \right\rfloor - 1 + \Delta_k(n) \right) g_{v_k} + \sum_{1 \leqslant k \leqslant h} g_{\tau_k} + \left\lfloor \frac{n - v_h}{d^h} \right\rfloor g_{d^h}, \tag{5}$$

where $\tau_k = v_k + m - \lfloor m/d^k \rfloor d^k$ for $1 \leqslant k \leqslant h$ and

$$\Delta_k(n) = \left\lfloor \left\{ \frac{v_h}{d^k} \right\} + \left\{ \frac{m}{d^k} \right\} \right\rfloor - \left\lfloor \left\{ \frac{v_h}{d^{k-1}} \right\} + \left\{ \frac{m}{d^{k-1}} \right\} \right\rfloor. \tag{6}$$

Proof. Rewrite (3) as

$$A_n = \left\lfloor \frac{m}{d^{h-1}} \right\rfloor A_{v_h} + \left(d - \left\lfloor \frac{m}{d^{h-1}} \right\rfloor - 1 \right) A_{v_{h-1}} + A_{\tau_{h-1}} + g_{\tau_h}.$$

When $n = \tau_{h-1}$, we have

$$A_{\tau_{h-1}} = \left(\left\lfloor \frac{m}{d^{h-2}} \right\rfloor - d \left\lfloor \frac{m}{d^{h-1}} \right\rfloor \right) A_{v_{h-1}} + \left(d + d \left\lfloor \frac{m}{d^{h-1}} \right\rfloor - \left\lfloor \frac{m}{d^{h-2}} \right\rfloor - 1 \right) A_{v_{h-2}} + A_{\tau_{h-2}} + g_{\tau_{h-1}}. \tag{7}$$

Applying (4) and (7) iteratively, we obtain $A_n = \sum_{0 \leqslant j \leqslant h} G_{\tau_j}$, where

$$G_{\tau_j} = \sum_{1 \leqslant k \leqslant (h-j-1)} d^{h-j-k-1}(d-1) \left(\left\lfloor \frac{\tau_j - v_j}{d^{h-j-1}} \right\rfloor + 1 \right) g_{v_k} + \left\lfloor \frac{\tau_j - v_j}{d^{h-j-1}} \right\rfloor g_{v_j} + g_{\tau_j}.$$

Interchanging the order of summation and calculating further yield

$$A_n = \sum_{1 \leqslant k \leqslant h} \left(\left\lfloor \frac{m}{d^{k-1}} \right\rfloor - \left\lfloor \frac{m}{d^k} \right\rfloor + d^{h-k} - 1 \right) g_{v_k} + \sum_{1 \leqslant k \leqslant h} g_{\tau_k} + \left\lfloor \frac{m}{d^h} \right\rfloor g_{d^h}.$$

Formula (5) follows from replacing m by $n - v_h$. □

Corollary 1. *Assume that A_n satisfies the d-heap recurrence* (3) *and $g_n = O(1)$, then $A_n = cn + O(\log n)$ where $c = (d-1) \sum_{k \geqslant 1} g_{v_k}/d^k$.*

In particular, when $d=2$, A_n reduces to the heap recurrence in [14]. Observe that $\Delta_k(n)$ is a function assuming only values 0, 1, or 2.

2.2. Optimal merge trees and d-heaps

We briefly describe the optimal algorithm given in [16] for merging files with unit length. Start with n files of the same unit length and arrange these items in a queue. Then insert n_0 virtual files of zero length in the head of the queue, where $0 \leqslant n_0 \leqslant d-1$ and $(d-1)$ divides $(n + n_0 - 1)$. Then remove and merge the first d lists into one and put the new merged file at the tail of the queue. Repeat the above steps until only a

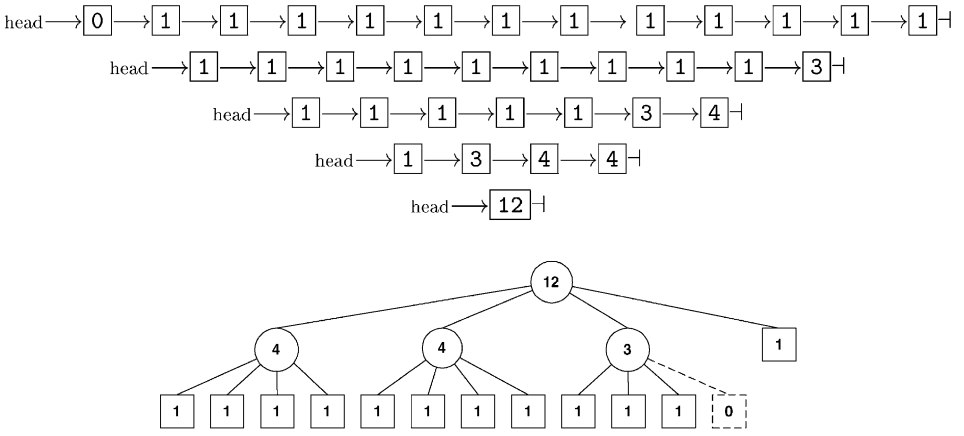

Fig. 3. The optimal merge pattern for $n = 12$ and $d=4$.

single item remains in the queue. An example with $n=12$ and $d=4$ is given in Fig. 3. This d-way merge algorithm is essentially the queue-mergesort generalized to higher order.

Let $n=(\delta_L\delta_{L-1}\dots\delta_0)_d$, where $L=\lfloor\log_d n\rfloor$. The above process introduces the following recurrence for optimum merge cost B_n: $B_0=0$, $B_1=g_1$, $B_n=ng_1+g_n$ if $1<n<d$, and

$$B_n=\left\lfloor\frac{n-d^L}{d^{L-1}(d-1)}\right\rfloor B_{d^L}+\left(d-\left\lfloor\frac{n-d^L}{d^{L-1}(d-1)}\right\rfloor-1\right)B_{d^{L-1}} +B_\gamma+g_n \quad (n\geqslant d), \tag{8}$$

where $\gamma=\gamma(n)=n-\lfloor(n-d^L)/d^{L-1}(d-1)\rfloor d^L-(d-\lfloor(n-d^L)/d^{L-1}(d-1)\rfloor-1)d^{L-1}$.

The method of proof of Theorem 1 applies *mutatis mutandis* to (8); the result is given as follows:

Theorem 2. *For $n\geqslant d$, the solution B_n of the optimum merge cost* (8) *is given by*

$$B_n=\sum_{0\leqslant k\leqslant L}\left(\left\lfloor\frac{n}{d^k}\right\rfloor-1+\Lambda_k(n)\right)g_{d^k}+\sum_{0\leqslant k\leqslant L}g_{s_k},$$

where

$$\Lambda_k(n)=\begin{cases}\lceil\{\frac{n-1}{d-1}\}\rceil & if\ k=0;\\ \lfloor\frac{\sum_{k-1\leqslant j\leqslant L}\delta_j-1}{d-1}\rfloor-\lfloor\frac{\sum_{k\leqslant j\leqslant L}\delta_j-1}{d-1}\rfloor & if\ 1\leqslant k\leqslant L\end{cases}$$

and $s_L=n$, $s_k=(D_k\delta_{k-1}\dots\delta_0)_d$ with $D_k=1+\{(\sum_{k\leqslant j\leqslant L}\delta_j-1)/(d-1)\}(d-1)$ for $0\leqslant k\leqslant L-1$.

Theorem 3. *The tree created according to the d-ary optimal merge algorithm for n unit-length items is homeomorphic to a d-heap with $n+\lceil(n-1)/(d-1)\rceil$ elements.*

The proof follows the same Huffman-coding argument used in [3, 8] and is omitted here.

2.3. Divide-and-conquer recurrences with minimization

Batty et al. [1] studied the divide-and-conquer recurrence with minimization (2); they identified the best dividing rules when g is increasing and is either concave or convex. We give here more explicit solutions to these best rules.

Theorem 4. *For $n \geqslant d$, the solution $\mathscr{F}_n$ of the divide-and-conquer recurrence with minimization* (2) *is given by*

(i) if g_n is increasing and convex, then

$$\mathscr{F}_n = \sum_{1 \leqslant k \leqslant L} d^k \left[\left(1 - \left\{ \frac{n}{d^k} \right\}\right) g_{\lfloor n/d^k \rfloor} + \left\{ \frac{n}{d^k} \right\} g_{\lfloor n/d^k \rfloor + 1} \right] + \chi_{\{d_L = 1\}} 2(n - d^L) g_1 + \chi_{\{d_L \neq 1\}} n g_1, \tag{9}$$

where $\chi_{\mathscr{P}} = 1$ if property $\mathscr{P}$ holds and $\chi_{\mathscr{P}} = 0$ otherwise;

(ii) if g_n is increasing and concave, then $\mathscr{F}_n = B_n$ (in Theorem 2).

Proof (Sketch). Define $\mathscr{F}_0 = 0$ and $\mathscr{F}_1 = g_1$. In case (i), we are solving the recurrence (cf. [1])

$$\mathscr{F}_n = \sum_{1 \leqslant j \leqslant d} \mathscr{F}_{\lfloor (n+j-1)/d \rfloor} + g(n). \tag{10}$$

We apply the argument used in [12]. Rewrite (10) as follows:

$$\mathscr{F}_n = d \left(1 - \left\{ \frac{n}{d} \right\}\right) \mathscr{F}_{\lfloor n/d \rfloor} + d \left\{ \frac{n}{d} \right\} \mathscr{F}_{\lfloor n/d \rfloor + 1} + g_n.$$

Iterating $L - 1$ times yields

$$\mathscr{F}_n = \sum_{1 \leqslant k \leqslant L} d^k \left[\left(1 - \left\{ \frac{n}{d^k} \right\}\right) g_{\lfloor n/d^k \rfloor} + \left\{ \frac{n}{d^k} \right\} g_{\lfloor n/d^k \rfloor + 1} \right] + \begin{cases} 2(n - d^L) g_1 & \text{if } d_L = 1, \\ n g_1 & \text{if } d_L \neq 1, \end{cases}$$

from which (9) follows. □

A well-studied example of the above type is the partial sum of the sum-of-digits functions. Let $S_d(n)$ denote the sum-of-digits function of n when written in d-ary representation. Then the partial sum $T(n) := \sum_{k<n} S_d(k)$ satisfies

$$\begin{aligned} T(n) &= \sum_{1 \leqslant j \leqslant d} \left(\sum_{k < \lfloor (n+j-1)/d \rfloor} S_d(k) \right) + \sum_{1 \leqslant j \leqslant d} (d - j) \lfloor (n + j - 1)/d \rfloor \\ &= \sum_{1 \leqslant j \leqslant d} T(\lfloor (n + j - 1)/d \rfloor) + g_n, \end{aligned}$$

where $g_1 = 0$ and for $n \geqslant 2$

$$g_n = \sum_{1 \leqslant j \leqslant d} (d-j)\lfloor (n+j-1)/d \rfloor = \frac{d(d-1)}{2} \left\lfloor \frac{n}{d} \right\rfloor + \frac{d}{2} \left\{ \frac{n}{d} \right\} \left(d \left\{ \frac{n}{d} \right\} - 1 \right).$$

Thus by (9), we obtain

$$\begin{aligned} T(n) &= \sum_{0 \leqslant j \leqslant L} d^j g_{\lfloor n/d^j \rfloor} + \sum_{0 \leqslant j \leqslant L} d^j \left\{ \frac{n}{d^j} \right\} (g_{\lfloor n/d^j \rfloor + 1} - g_{\lfloor n/d^j \rfloor}) \\ &= \frac{d-1}{2} n \log_d n + \mathrm{O}(n). \end{aligned}$$

The O-term can, of course, be further expanded in more explicit form; see [5] and the references therein for more information. □

3. The cost of constructing d-heaps

We apply in this section Theorem 1 to the analysis of algorithms for constructing d-heaps.

Floyd's algorithm for constructing binary heaps extends in a straightforward way to d-heaps. The worst-case cost is linear. Assume that all the $n!$ permutations of n elements are equally likely. Then this algorithm also preserves randomness (cf. [16, p. 155]); that is, all d-heaps of size n are equally likely to be generated by Floyd's algorithm when given a random permutation.

Let $n = v_h + m$ where $v_h = (d^h - 1)/(d-1)$ and $h = \lfloor \log_d((n-1)(d-1)+1) \rfloor$. We denote by X_n the number of exchanges used by Floyd's algorithm to construct a d-heap from a random permutation.

Theorem 5. *Given a random permutation of n elements, the number X_n of exchanges used by Floyd's algorithm to construct a d-heap satisfies*

$$E(X_n) = c_1 n + \mathrm{O}(\log n), \tag{11}$$

$$\mathrm{Var}(X_n) = c_2 n + \mathrm{O}(\log n), \tag{12}$$

where

$$c_1 = -\frac{1}{d-1} + (d-1) \sum_{k \geqslant 2} \frac{k}{d^k - 1},$$

$$\begin{aligned} c_2 = {} & \frac{d}{(d-1)^2} + \sum_{k \geqslant 1} \frac{(k-1)(kd^2 + d^2 - 3kd + 3d + 2k)}{d^k - 1} \\ & - (d-1) \sum_{k \geqslant 1} \frac{k^2}{(d^k - 1)^2}. \end{aligned}$$

Proof. The mean $E(X_n)$ satisfies the d-heap recurrence (3) with g_n the average height of a node in a random d-heap of size n; that is

$$g_n = \frac{1}{n} \sum_{1 \leqslant k \leqslant n} \lfloor \log_d((k-1)(d-1)+1) \rfloor = h + \frac{h}{(d-1)n} - \frac{d^{h+1}-d}{(d-1)^2 n}.$$

Applying Corollary 1, we obtain (11).

Similarly, the variance $\mathrm{Var}(X_n)$ of X_n satisfies the d-heap recurrence (3) with

$$\begin{aligned} g_n &= \frac{1}{n} \sum_{1 \leqslant k \leqslant n} \lfloor \log_d((k-1)(d-1)+1) \rfloor^2 \\ &\quad - \left(\frac{1}{n} \sum_{1 \leqslant k \leqslant n} \lfloor \log_d((k-1)(d-1)+1) \rfloor \right)^2 \\ &= \frac{h^2}{n} + \frac{(1-2h)d^{h+2} + (2h+1)d^{h+1} - d^3h^2 + 4d^2h^2 - 5dh^2 + 2h^2 - d^2 - d}{(d-1)^3 n} \\ &\quad + \frac{2d^{h+1}h - 2dh}{(d-1)^2 n} - \frac{2h^2}{(d-1)n} + \frac{-d^{2h+2} + 2d^{h+2} - d^2}{(d-1)^4 n^2} \\ &\quad + \frac{2d^{h+1}h - 2dh}{(d-1)^3 n^2} - \frac{h^2}{(d-1)^2 n^2}, \end{aligned}$$

by partial summation. Applying Corollary 1 again, we obtain (12). □

Note that the error term $O(\log n)$ can be further refined but the result is not interesting. Also by a result of Hwang and Steyaert [14], we can show that the distribution of X_n is asymptotically normal.

4. Another generalized heap recurrence

We present in this section another way of generalizing the binary heap in which the addresses of the children of a node p are given by $dp + j$, for $0 \leqslant j \leqslant d - 1$; we will call such structures d^*-heaps. The corresponding recurrence is also studied.

A d^*-heap is almost a d-heap with the sole exception that the root has branch factor $d-1$ instead of d. Fig. 4 shows the structure of a 4^*-heap. In this way, the addressing scheme of d^*-heap is consistent with the binary heap. Conversely, the parent of an internal node at p is at $\lfloor p/d \rfloor$. Assume that $n-1 = (\delta'_\ell \delta'_{\ell-1} \ldots \delta'_0)_d$ and $\ell = \lfloor \log_d(n-1) \rfloor$. Then such a structure introduces the following recurrence: $C_0 = 0$, $C_n = (n-1)g_1 + g_n$

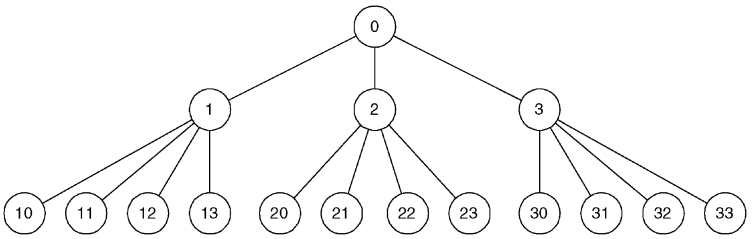

Fig. 4. The 4*-heap with size $16=(100)_4$. Numbers in all nodes are expressed in terms of their quaternary representations.

for $1\leqslant n\leqslant d$, and

$$C_n=\left(\left\lfloor\frac{n-1}{d^\ell}\right\rfloor-1\right)A_{v_{\ell+1}}+\left(d-\left\lfloor\frac{n-1}{d^\ell}\right\rfloor-1\right)A_{v_\ell}+A_\rho+g_n\quad(n>d),\tag{13}$$

where $\rho=\rho(n)=n-(\lfloor(n-1)/d^\ell\rfloor-1)v_{\ell+1}-(d-\lfloor(n-1)/d^\ell\rfloor-1)v_\ell-1$ and A_n satisfies the d-heap recurrence. We can solve the above recurrence as for (3).

Theorem 6. *For $n>d$, the solution C_n of the d^*-heap recurrence* (13) *is given by*

$$C_n=\sum_{1\leqslant k\leqslant\ell}\left(\left\lfloor\frac{n-1}{d^{k-1}}\right\rfloor-\left\lfloor\frac{n-1}{d^k}\right\rfloor-1\right)g_{v_k}+\sum_{0\leqslant k\leqslant\ell+1}g_{t_k},\tag{14}$$

where $t_{\ell+1}=n$, $t_k=1+\sum_{0\leqslant j\leqslant k-1}(\delta'_j+1)d^j$ for $1\leqslant k\leqslant\ell$, and $t_0=0$.

Note that the explicit expression of (14) does not contain the factor $\Delta_k(n)$ of Theorem 1.

Theorem 7. *Given a random permutation of n elements, the number Y_n of exchanges used by Floyd's algorithm to construct a d^*-heap satisfies*

$$E(Y_n)=c_3n+\mathrm{O}(\log n),$$

$$\mathrm{Var}(Y_n)=c_4n+\mathrm{O}(\log n),$$

where

$$c_3=\frac{1}{d-1},$$

$$c_4=\frac{4}{d-1}-2(d-1)\sum_{k\geqslant2}\frac{k}{d^k-1}.$$

Proof (*Sketch*). Apply Theorem 6 with

$$g_n=\frac{1}{n}\left(\sum_{2\leqslant k\leqslant n}(\lfloor\log_d(k-1)\rfloor+1)\right)=\ell-\frac{d^{\ell+1}-1}{(d-1)n}+1$$

for the mean and

$$g_n = \frac{1}{n}\sum_{2\leqslant k\leqslant n}(\lfloor \log_d(k-1)\rfloor + 1)^2 - \left(\frac{1}{n}\sum_{2\leqslant k\leqslant n}(\lfloor \log_d(k-1)\rfloor + 1)\right)^2$$
$$= \frac{-d^{2\ell+2} + 2d^{\ell+1} - 1}{(d-1)^2 n^2} + \frac{(1-2l)d^{\ell+2} + (1+2\ell)d^{\ell+1} + d^2 - d}{(d-1)^2 n}$$
$$+ \frac{2d^{\ell+1}\ell + 2d - 2\ell - 2}{(d-1)n} + \frac{1}{n}$$

for the variance. □

We can also prove the asymptotic normality (in the sense of convergence in distribution) of Y_n.

Note that the two constants c_1 and c_3 both decrease with d.

References

[1] C.J.K. Batty, M.J. Pelling, D.G. Rogers, Some recurrence relations of recursive minimization, SIAM J. Algebraic Discrete Methods 3 (1982) 13–29.

[2] C.J.K. Batty, D.G. Rogers, Some maximal solutions of the generalized subadditive inequality, SIAM J. Algebraic Discrete Methods 3 (1982) 369–378.

[3] W.-M. Chen, H.-K. Hwang, G.-H. Chen, The cost distribution of queue-mergesort, optimal mergesorts, and power-of-2 rules, J. Algorithms 30 (1999) 423–448.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.

[5] P. Flajolet, P. Grabner, P. Kirschenhofer, H. Prodinger, R.F. Tichy, Mellin transforms and asymptotics: digital sums, Theoret. Comput. Sci. 123 (1994) 291–314.

[6] R.W. Floyd, Algorithm 245: treesort 3, Comm. ACM 7 (1964) 701.

[7] E.H. Friend, Sorting on electronic computer systems, J. ACM 3 (1956) 134–168.

[8] C.R. Glassey, R.M. Karp, On the optimality of Huffman trees, SIAM J. Appl. Math. 31 (1976) 368–378.

[9] S. Glicksman, Concerning the merging of equal length tape files, J. ACM 12 (1965) 254–258.

[10] L.M. Goldschlager, P. Eades, Cheapsort, Congr. Numer. 31 (1981) 45–62.

[11] M. Golin, R. Sedgewick, Queue-mergesort, Inform. Process. Lett. 48 (1993) 253–259.

[12] J.M. Hammersley, G.R. Grimmett, Maximal solutions of the generalized subadditive inequality, in: E.F. Harding, D.G. Kendall (Eds.), Stochastic Geometry, Chap. 4, Wiley, New York, 1974.

[13] H.-K. Hwang, Limit theorems for mergesort, Random Struct. Algorithms 8 (1996) 319–336.

[14] H.K. Hwang, J.-M. Steyaert, On the number of heaps and the cost of heap construction, Colloquim on Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities, Versailles, September 16–19, 2002, in Mathematics and Computer Science II, Birkhauser, Basel, 2002, pp. 294–310.

[15] D.B. Johnson, Priority queues with update and finding minimum spanning trees, Inform. Process. Lett. 4 (1975) 53–57.

[16] D.E. Knuth, The Art of Computer Programming, Sorting and Searching, Vol. 3, Second Ed., Addison-Wesley, Reading, MA, 1973.

[17] R.W.P. Luk, Near optimal β heap, Comput. J. 52 (1999) 391–399.

[18] S. Okoma, Generalized Heapsort, Lecture Notes in Computer Science, Vol. 88, Springer, Berlin, 1980, pp. 439–451.

[19] A. Paulik, Worst-case analysis of a generalized heapsort algorithm, Inform. Process. Lett. 36 (1990) 159–165.

[20] R. Sprugnoli, Recurrence relations on heaps, Algorithmica 15 (1996) 467–480.

[21] T. Walsh, How evenly should one divide to conquer quickly? Inform. Process. Lett. 19 (1984) 203–208.