

Optimality of an Algorithm Solving the Bottleneck Tower of Hanoi Problem

Yefim Dinitz * Shay Solomon *

February 1, 2006

Abstract

We study the bottleneck Tower of Hanoi puzzle, which was posed by D. Wood in 1981. There, a relaxed placement rule, depending on a parameter k , allows a bigger disk to be placed *higher than* a smaller one if their size difference is less than k . A shortest sequence of moves (or “optimal algorithm”) transferring all the disks placed on some peg, in the decreasing order, to another peg, in the same order, is in question. In 1992, D. Poole named the problem, computed the length of such a sequence, and (indirectly) described the optimal algorithm. However, his proof of its optimality is insufficient. In 1998, Beneditkis, Berend, and Safro suggested the same algorithm (independently) and proved its optimality for the case $k \leq 2$. We prove it to be optimal in the general case.

1 Introduction

The first version of the puzzle that we consider was marketed by Edouard Lucas in 1883 under the name “Tower of Hanoi”. It consisted of three pegs (or “towers”) on a wooden base, and eight disks of different sizes, arranged on one of the pegs in decreasing size, from the largest at the bottom to the smallest at the top. The goal of the puzzle is to transfer all disks to another

*Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel. E-mail: {dinitz, shayso}@cs.bgu.ac.il .

peg, placed in the same order. The rules are to move a single disk from (the top of) one peg to (the top of) another one, at each step, subject to the divine rule: to never have a larger disk above a smaller one. It was introduced as a variant of the mythical "Towers of Brahma", which had real towers and sixty four golden rings. According to a Buddhist legend, Judgment Day will arrive, when the monks from the monastery in the Low Himalayas finish to move all 64 disks to the target peg.

Let us describe the well known algorithmic solution to the classic Tower of Hanoi (ToH) problem. There are n disks $n, n - 1, \dots, 2, 1$, $n \geq 1$, at one peg (*source*), which are due to be moved to another peg (*target*), using the third peg (*auxiliary*); the size of a disk is referred as its name, for simplicity. We will refer this problem as $HT = HT_n$. Let us define the algorithm $\gamma_n = \gamma_n(\textit{source}, \textit{target})$, solving HT_n :

- If n is 1, move disk n from *source* to *target*.
- Otherwise:
 - recursively perform $\gamma_{n-1}(\textit{source}, \textit{auxiliary})$;
 - move disk n from *source* to *target*;
 - recursively perform $\gamma_{n-1}(\textit{auxiliary}, \textit{target})$.

It is taught in most introductory CS courses as a nice example of a recursive algorithm.

Let us call the shortest sequence of moves solving a ToH problem *optimal*; in what follows, we do not distinguish between a sequence of moves and an algorithm generating it, if this does not lead to a misunderstanding. It is known and easy to prove that γ_n is optimal for HT_n :

Theorem 1.1 *Algorithm γ_n solves HT_n in $2^n - 1$ disk moves. Any sequence of moves solving HT_n requires at least this number of moves.*

In the recent two decades, there is an increasing interest on the Tower of Hanoi problem and its generalizations. A wide bibliography on the ToH research (more than 300 references) is maintained by Paul Stockmayer [5].

In 1981, D. Wood [6] suggested the following generalization of HT , depending on a parameter, k , $k \geq 1$. Following D. Poole [3], we call it the *bottleneck Tower of Hanoi problem*, and denote it by $BTH = BTH_{n,k}$. It

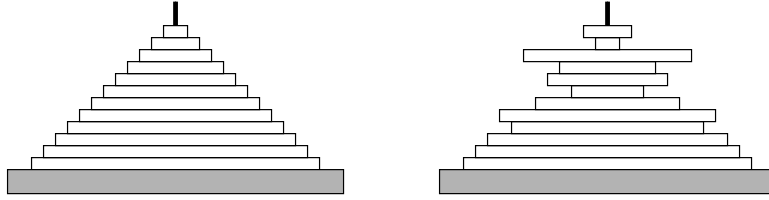


Figure 1: On the left, the perfect configuration of the disk set $[1..12]$ is shown. On the right, another configuration of these disks: 12, 11, 10, 8, 9, 6, 3, 5, 4, 7, 1, 2, is presented; it is legal for $BTH_{12,k}$, if k is at least 5, but is not legal if k is at most 4, since disk 7 is placed higher than disk 3.

differs from the classic HT_n by the following *k-relaxed placement rule*: *If disk j is placed higher than disk i on the same peg (not necessarily neighboring it), then their size difference $j - i$ is less than k .* Now, there are more than one legal way to place a given set of disks on the same peg, in general. We refer the decreasing bottom-to-top placement of all disks on the same peg as the *perfect* disk configuration. For illustration, see Figure 1. If k is 1, we arrive at the classic problem. In this paper, we consider k be fixed, and study $BTH = BTH_n$ usually without mentioning k as a parameter.

D. Poole [3] was the first to find the minimal number of moves in any solution to BTH_n . He had suggested a proof of its minimality, which turned out to be insufficient (see details in Section 5.2).

S. Beneditkis, D. Berend, and I. Safro [1] reinvented the optimal algorithm for BTH_n (which is implied indirectly in [3]); it is described below by name α_n . They proved its optimality for the case $k = 2$. Similarly to [3], they began from another related problem of "moving somehow", under the *k-relaxed placement rule*: *To move m disks $[1..m]$, placed entirely on one peg, to another peg, in any order.* They used for it the following algorithm $\beta_m = \beta_m(\text{source}, \text{target})$ (for illustration see Figure 2a):

- If m is at most k , move all disks from *source* to *target* one by one.
- Otherwise:
 - recursively perform $\beta_{m-k}(\text{source}, \text{auxiliary})$;
 - move disks $[(m - k + 1)..m]$ from *source* to *target* one by one;
 - recursively perform $\beta_{m-k}(\text{auxiliary}, \text{target})$.

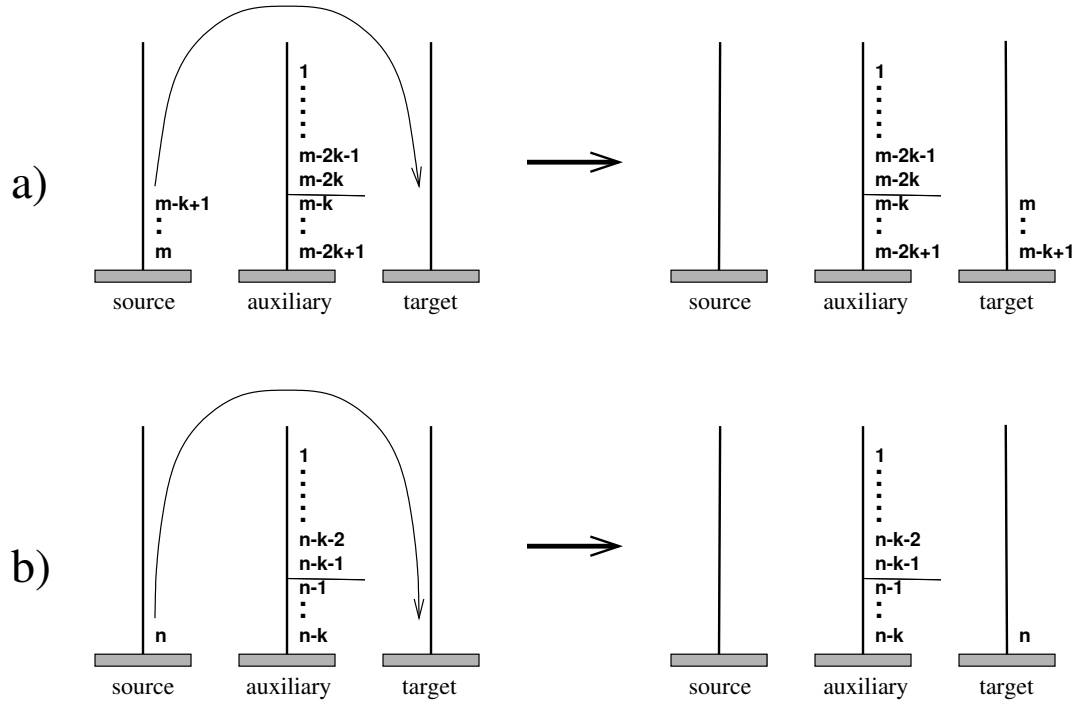


Figure 2: (a) Performing the second item of β_m . (b) Performing the second item of α_n .

For better mnemonics (following [2]), let us divide the entire set of disks $[1..m]$ into $\lceil \frac{m}{k} \rceil$ blocks $B_i = B_i(m)$: $B_1 = [(m - k + 1)..m]$, $B_2 = [(m - 2k + 1)..(m - k)]$, \dots , $B_{\lceil \frac{m}{k} \rceil} = [1..(m - (\lceil \frac{m}{k} \rceil - 1) \cdot k)]$. Note that all blocks, except for the last one, are of length k , while the last one is of length k as well, if $n = 0 \pmod k$, and of length $n \pmod k$, otherwise. Note that the set of disks in any block is allowed to be placed on the same peg in an arbitrary order (the same holds for any set of disks $[r..q]$, $q - r + 1 \leq k$). Notice that the sequence β_n is similar to γ_n , but deals with blocks, instead of single disks.

It is easy to show that β_m might be not legal, depending on the initial placement of the m disks. Notice that β_m applied to the perfect configuration of m disks is legal, and results in the configuration, where the first block of disks is at the bottom of *target* in the reverse (i.e. increasing) order, while the rest of the disks are above it in the decreasing order. As well, β_m applied to the latter configuration is legal and results in the perfect disk configuration on *target*.

Beneditkis, Berend, and Safro proved, for the case $k = 2$ [1], that no sequence of moves shorter than β_m can transfer m disks from one peg to another. We generalize their result to the case of an arbitrary k .

Now, we describe the *algorithm* α_n (for illustration see Figure 2b):

- perform $\beta_{n-1}(source, auxiliary)$;
- move disk n from *source* to *target*;
- perform $\beta_{n-1}(auxiliary, target)$.

The above observation concerning the effect of two composed β 's implies that α_n solves BTH_n . Beneditkis, Berend, and Safro proved, for the case $k = 2$, that the sequence α_n is an optimal solution to BTH_n [1]. In this paper, we prove optimality of α_n for BTH_n in the general case.

Interestingly, another, different proof of optimality of α_n for BTH_n is suggested independently by X. Chen et al. [2].

The preliminary version of this paper was presented at the Workshop on the Tower of Hanoi and Related Problems, held in September 2005 at Maribor, Slovenia.

2 Definitions and Notation

A *configuration* is a specification of ordered sets of disks on the three pegs. We consider only legal configurations, i.e. those obeying the the k -relaxed placement rule of *BTH*. A configuration of a disk set D is called *gathered*, if all disks in D are on the same peg, while the two other pegs are empty. A configuration of the set of disks D is called *perfect*, if it is gathered, D is an initial interval of naturals, and the order of disks (on a single peg) is decreasing. For any configuration C of D and a subset $D' \subseteq D$, let us define the *restriction of C to D'* , denoted by $C|_{D'}$, as the configuration C with all disks not in D' removed.

A move of disk m from peg X to peg Y is denoted by the triplet (m, X, Y) . For a disk set D , obviously, the configuration of $D \setminus \{m\}$ is the same before and after such a move; we refer it as the *configuration of $D \setminus \{m\}$ during (m, X, Y)* (though a move is considered as an atomic operation, we chose the word “during” as most intuitive).

Let us introduce the notion of a *packet-move* of a disk set D as a sequence of disk moves from an (initial) gathered configuration of D on one peg, X , to a (final) gathered configuration of D on another peg, Y ; the generic notion is $P = P(D, X, Y)$. In this notation, we call X *source* = $source(P)$, Y *target* = $target(P)$, and the third peg *auxiliary* = $auxiliary(P)$, or the *source, target, and auxiliary pegs*, respectively; we do not mention P if it is clear from the context. The *length* of a packet-move P , denoted by $|P|$, is the number of moves in it. We use the notion of $\mathcal{P}(D, X, Y)$ for the family of all packet-moves of D from X to Y . If both initial and final configurations of P are perfect, we call P a *perfect-to-perfect* packet-move of D . For any $m \geq 1$, let D_m denote $[1..m]$. In the above notion, BTH_n concerns finding the shortest perfect-to-perfect packet-move of D_n .

For any set of disks placed continuously on the top of some peg, we call *simple* their packet-move to another peg one by one (for example, see the description of β_m). Note that a simple packet-move reverses the order of disks, so it may be legal only if their number is at most k (a necessary but not sufficient condition).

The *composition* of several packet-moves of the same disk set, D , is the packet-move of D defined as the concatenation of their related sequences of moves. Such a composition is legal if and only if the final configuration of any non-last packet-move coincides with the initial configuration of the

next packet-move, in the composition. Note that any (legal) composition of packet-moves of D is a new packet-move of D , from the initial configuration of the first packet-move to the final configuration of the last one. We use the notion of $\mathcal{P}^t(D, X, Y)$ for the family of all packet-moves in $\mathcal{P}(D, X, Y)$, which are compositions of t packet-moves of D .

In our notion, we relate to a packet-move as to a sequence of moves. In particular, we say that a packet-move, P , *contains* another packet move if the latter one is a subsequence of P , we say that a few packet-moves, contained in some sequence of moves, are *disjoint* if the last move in any non-last one of them precedes the first move in the next one, we say that some move in P *divides* P into its parts before and after that move, etc.

Lemma 2.1 *If a packet-move P of D contains t disjoint packet-moves of D then P is in $\mathcal{P}^{t'}(D, X, Y)$, $t' \geq t$.*

Proof: Let us consider the non-empty intervals of P (if any) between the i th and the $(i+1)$ th out of the t packet-moves, as in Lemma, $1 \leq i \leq t-1$, before the first one, and after the last one of them. Clearly, each such sub-sequence of moves begins and ends at gathered configurations of D . For each such sub-sequence, S , if those configurations are at the same peg, we extend one of the neighboring packet-moves with the moves in S ; otherwise, we define S as an additional packet-move of D . As a result, P is divided into at least t packet-moves of D , as required. \square

For any sequence of moves S of D and any disk subset $D' \subseteq D$, we define the *restriction of S to D'* , denoted by $S|_{D'}$, as the result of omission of all moves of disks not in D' , from S . Clearly, if C_1 is the initial configuration and C_2 the final one, w.r.t. S , then $S|_{D'}$ transforms $C_1|_{D'}$ to $C_2|_{D'}$. Note that, by the rules of *BTH*, the restriction of any legal sequence of moves to any disk subset is legal as well. In particular, for any two disk sets $D' \subset D$ and any packet-move P of D , its restriction $P|_{D'}$ is a legal packet-move of D' , with $|P|_{D'}| < |P|$. Moreover, if D is partitioned into two non-intersecting subsets D' and D'' , then $|P| = |P|_{D'}| + |P|_{D''}|$.

Recall that the first block of the set D_m , $B_1(m)$, is defined as the set of the $\min\{k, m\}$ biggest disks in D_m . For $m > k$, we denote its complement set $D_{m-k} = \bigcup_{i>1} B_i(m)$ by *Small*(m).

Consider a sequence of moves, S , containing two consequent moves of the same disk: (i, X, Y) and (i, Y, Z) . Their replacement by the single move

(i, X, Z) , if $X \neq Z$, or the deletion of both, if $X = Z$, is called a *pruning* of S . Notice that if S is legal, its pruned variant is legal as well and has the same initial and final states. We denote by $Prune(S)$ the result of all possible prunings, at S ; it is easy to see that such a result does not depend on the order of particular prunings, so $Prune(S)$ is well defined.

3 The Shortest "Somehow" Packet-Move

In this section, we consider general, not perfect-to-perfect packet-moves, and show that β_m (defined in Section 1) is a shortest possible packet-move of D_m . Let b_m denote the length of β_m . By definition of β_m :

$$b_m = \begin{cases} m & \text{if } m \leq k \\ 2b_{m-k} + k & \text{if } m > k \end{cases} \quad (1)$$

In [3], the recurrence relation (1) is shown to imply the explicit formula:

$$b_n = k \cdot (2^{\lfloor \frac{n}{k} \rfloor} - 1) + r \cdot 2^{\lfloor \frac{n}{k} \rfloor} = (k + r) \cdot 2^{\lfloor \frac{n}{k} \rfloor} - k, \quad r = n \bmod k \quad (2)$$

As a consequence, $b_n - b_{n-1} = 2^{\lceil n/k \rceil - 1}$; in particular, the sequence $\{b_i\}$ is *strictly* monotonous.

Let us begin with some general observations.

Fact 3.1 1. *During a move (m, X, Y) , all disks in $Small(m)$ are on the spare peg $Z \neq X, Y$.*

2. *If some sequence of moves S begins from a configuration, where disk m and $Small(m)$ are gathered on X , and finishes at a configuration, where disk m and $Small(m)$ are gathered on Y , $X \neq Y$, then it contains two disjoint packet-moves of $Small(m)$: one (from X) before the first move of disk m and another (to Y) after its last move.*

Theorem 3.2 *Under the rules of BTH, the length of any packet-move of D_m is at least b_m .*

Proof: By a complete induction on m . *Basis:* The case $m \leq k$ is trivial. *Inductive step:* For any $m > k$, we consider an arbitrary packet-move P of

D_m , assuming the statement valid for all lesser values of m . By Fact 3.1(2), $P|_{Small(m)}$ contains two disjoint packet-moves of $Small(m)$; by the induction hypothesis, their total length is at least $2 \cdot b_{m-k}$. Besides, every disk in $B_1(m)$ must move at least once, which sums to at least k moves. Hence,

$$|P| = |P|_{Small(m)} + |P|_{B_1(m)} \geq 2 \cdot b_{m-k} + k = b_m,$$

as required. \square

We will use the following corollary:

Corollary 3.3 *For any packet-move P of D_m , $P|_{Small(m)}$ contains two disjoint packet-moves of $Small(m)$.*

4 Optimality of α_n for BTH_n

Let a_n denote the length of α_n . The explicit formula for a_n (established in [3]) is implied straight-forwardly by that for b_{n-1} (2):

$$a_n = 2(r + k) \cdot 2^{\lfloor \frac{n-1}{k} \rfloor} - 2k + 1, \quad r = (n - 1) \bmod k. \quad (3)$$

The following theorem, which is the main goal of this paper, implies that α_n is an optimal algorithm that solves the puzzle.

Theorem 4.1 *Under the rules of BTH_n , any perfect-to-perfect packet-move of D_n is of length at least a_n .*

The rest of this section is devoted to the proof of this Theorem. For any perfect-to-perfect packet-move P of D_m , we distinguish the following two cases:

Case 1: *During P , disk m never moves to the auxiliary peg.*

Case 2: *P contains a move of disk m to the auxiliary peg.*

We prove that, for a packet-move P of D_n as in Theorem 4.1, in Case 1, the length of P is *at least* a_n , while in Case 2, the length of P *exceeds* a_n .

Recall that $Small(m) = [1..(m - k)]$, $B_1(m) = [(m - k + 1)..m]$. In our study of a packet-move P of D_m , $m > k$, we usually consider separately

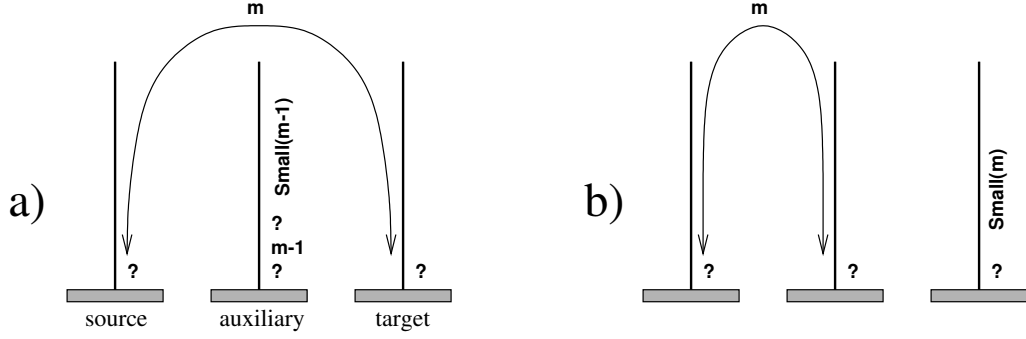


Figure 3: (a) A distinguished move of disk m . Disks in $[(m-k+1)..(m-2)]$ can be placed at any peg; at *auxiliary* they, as well as disk $m-k$, may be below or above disk $m-1$ and may be interleaved with disks in $Small(m-1)$. (b) A move of disk m . Disks in $[(m-k+1)..(m-1)]$ can be placed at any peg; they may be interleaved with disks in $Small(m)$.

$P|_{Small(m)}$ and $P|_{B_1(m)}$. The reason is that any move (m, X, Y) defines completely the placement of $Small(m)$ —on the spare peg Z ,—while the placement of disks in $B_1(m) \setminus \{m\}$ may be arbitrary, during such a move. For the analysis of $P|_{B_1(m)}$, we use the following statement:

Fact 4.2 *Any perfect-to-perfect packet-move P of D_m contains at least two moves of any disk i , $i \neq m$: at least one before the first move of disk m and at least one after its last move. Hence, for any $D \subseteq D_{m-1}$, holds $|P|_D \geq 2|D|$.*

4.1 Case 1: Disk n never moves to the auxiliary peg

Let us call a move of disk m , in a packet-move of D_m , *distinguished* if it is from *source* to *target* and disk $m-1$ is at *auxiliary* during that move (for illustration see Figure 3a).

Lemma 4.3 *Any packet-move P of D_m , which preserves the initial order between disks m and $m-1$, and such that disk m moves only between the source and target pegs, contains a distinguished move of disk m .*

Proof: Define $P' := Prune(P|_{\{m-1, m\}})$. Note that P' preserves the initial order between disks m and $m-1$, since P preserves it. As well, a move of m is distinguished in P' if and only if it is distinguished in P . By the definition of $Prune$, the moves of disks m and $m-1$ must interchange, in P' .

Observation 4.4 *Under the conditions of the Lemma, the first move of disk $m - 1$ should be $(m - 1, source, auxiliary)$.*

Proof: Indeed, otherwise it is $(m - 1, source, target)$. Then, the first two moves transfer both disks to *target*, since the first move of disk m should be $(m, source, target)$, by the condition of the Lemma. This should be the end of P' , since otherwise, the disk on the top of *target*, which made the last move, should make the next move, which contradicts the definition of *Prune*. However, after the two first moves as above, the final order of the disks m and $m - 1$ is inverse w.r.t. the initial one,—a contradiction. \square

Based on this Observation, let us consider the possible scenarios, from the two possible initial configurations of P' . Assume, first, that disk $m - 1$ is initially above disk m . Then, necessarily, the first two moves are: $(m - 1, source, auxiliary)$ and $(m, source, target)$; the latter one is distinguished. Assume, now, that disk $m - 1$ is initially below disk m . Then, necessarily, the first three moves are: $(m, source, target)$, $(m - 1, source, auxiliary)$, $(m, target, source)$; the latter one is distinguished. \square

Proposition 4.5 *For any $m > k + 1$ and any packet-move P of D_m preserving the initial order between disks m and $m - 1$, in the case when disk m moves only between the source and target pegs, P contains four disjoint packet-moves of $Small(m - 1)$.*

Proof: Let us consider the first distinguished move of disk m during P ; it exists by Lemma 4.3. Clearly, during such a move, all disks in $Small(m - 1)$ are on *auxiliary*, together with disk $m - 1$; by the k -relaxed placement rule, they are placed above disk $m - 1$. By Fact 3.1(2), the parts of P before and after that move contain two disjoint packet-moves of $Small(m - 1)$ each. Hence altogether, there are four disjoint packet-moves of $Small(m - 1)$, in P . \square

Corollary 4.6 *The length of any perfect-to-perfect packet-move of D_n , such that disk n moves only between the source and target pegs, is at least a_n .*

Proof: Let P be a packet-move as in the Corollary. By Fact 4.2, $|P|_{B_1(n-1)} \geq 2 \cdot |B_1(n-1)|$. If $n \leq k + 1$, $|P| = |P|_{B_1(n-1)} + |P|_{\{n\}} \geq 2(n - 1) + 1 = 2 \cdot b_{n-1} + 1 = a_n$. Otherwise, by Proposition 4.5 and Theorem 3.2, $|P|_{Small(n-1)} \geq 4 \cdot b_{n-k-1}$. Hence, $|P| = |P|_{Small(n-1)} + |P|_{B_1(n-1)} + |P|_{\{n\}} \geq 4 \cdot b_{n-k-1} + 2k + 1 = 2 \cdot b_{n-1} + 1 = a_n$. \square

4.2 Case 2: Disk n moves to the auxiliary peg

Lemma 4.7 *If $m > k$ and a packet-move P of disks D_m contains a move of disk m to the auxiliary peg, then $P|_{Small(m)}$ contains three disjoint packet-moves of $Small(m)$.¹*

Proof: In this proof, we refer the pegs and configurations w.r.t. P . Assume, first, that P contains a move $(m, source, auxiliary)$. At the initial configuration, $Small(m)$ is gathered at the source peg. By Fact 3.1(1), during that move, $Small(m)$ is gathered at the target peg, and during the last move of disk m (to the target peg), $Small(m)$ is gathered at another peg, X (for illustration see Figure 3b). At the final configuration, $Small(m)$ is gathered at the target peg once again. Thus, P contains three disjoint packet-moves of $Small(m)$: $source \rightarrow target$, $target \rightarrow X$, and $X \rightarrow target$.

The remaining case is that disk m never moves from the source peg to the auxiliary peg during P . Then, the first move of m is $(m, source, target)$, and at some point after it, m moves from $target$ to $auxiliary$. During the former move, $Small(m)$ is gathered at the auxiliary peg, and during the latter one, it is gathered at the source peg. Therefore, P contains three disjoint packet-moves of $Small(m)$: $source \rightarrow auxiliary$, $auxiliary \rightarrow source$, and $source \rightarrow target$. \square

Following is the central statement. Its generality w.r.t. l is required because of the inductive nature of the given proof (see a discussion in Section 5).

Proposition 4.8 *For any $m, l \geq 0$, let P be a perfect-to-perfect packet-move of D_m containing $2l + 1$ disjoint packet-moves of D_m . Then, $|P| \geq 2l \cdot b_m + 2 \cdot b_{m-1} + 1 = 2l \cdot b_m + a_m$, and this bound is tight.*

Proof: In this proof, we refer the pegs and configurations w.r.t. P . By Lemma 2.1, P belongs to $\mathcal{P}^t(D_m, source, target)$, where $t \geq 2l + 1$. First of all, if $t \geq 2l + 2$, then, by Theorem 3.2 and the strict monotonicity of (b_i) , $|P| \geq (2l + 2) \cdot b_m \geq 2l \cdot b_m + 2 \cdot b_{m-1} + 2$. Thus, we may henceforth assume that $P \in \mathcal{P}^{2l+1}(D_m, source, target)$.

The remaining part of the lower bound proof is by a complete induction on m , for all l .

¹The algorithm β_n^1 , described in Section 5, shows that the bound of this Lemma is tight.

Basis: $m \leq k$. (Note that in this basic case, the disks may be placed at pegs in an arbitrary order).

Lemma 4.9 *In the case $m \leq k$, the length of any perfect-to-perfect packet-move P of D_m belonging to $\mathcal{P}^{2l+1}(D_m, \text{source}, \text{target})$ is at least $(2l+2)m-1$.*

Proof: We call a disk in D_m *expensive* w.r.t. some packet-move P' , if it moves to the auxiliary peg during P' . Let us prove that there could be at most one disk, which is not expensive w.r.t. any one out of the $2l+1$ packet-moves as in the Lemma. Assume to the contrary that there are two such disks, i and j . Then, after each packet-move, their order is reversed. Since there is an odd number of packet-moves, the order of i and j at the final configuration is inverse to that in the initial configuration,—a contradiction.

It follows that either all m disks or some $m-1$ disks make at least $2l+2$ moves each, while the remained disk, if any, makes at least $2l+1$ moves. Altogether, at least $(2l+2)m-1$ moves are made. \square

As a corollary, since for any r , $1 \leq r \leq k$, holds $b_r = r$, we conclude that $|P| \geq (2l+2)m-1 = 2lm + 2(m-1) + 1 = 2l \cdot b_m + 2 \cdot b_{m-1} + 1$.

Induction step: $m > k$. We suppose that the claim holds for all lesser values of m and for all l , and prove it for m and all l .

Note that at the initial configuration of P , as well as at its final configuration, disk m is placed below disk $m-1$. Since P is a composition of an odd number of disjoint packet-moves of D_m , there exists a packet-move among them, henceforth denoted by \tilde{P} , which preserves the order between disks m and $m-1$. We bound $|P|_{\text{Small}(m)}$ separately, for two complementary types of \tilde{P} .

Case 1: During \tilde{P} , disk m never moves to the auxiliary peg.

By Proposition 4.5, \tilde{P} contains four disjoint packet-moves of $\text{Small}(m-1)$. By Fact 4.2, \tilde{P} contains at least two moves of disk $m-k$. By Theorem 3.2, $|\tilde{P}|_{\text{Small}(m)} \geq 4 \cdot b_{m-k-1} + 2 = 2 \cdot b_{m-1} - 2k + 2$. By Corollary 3.3, the other $2l$ packet-moves of D_m contain two disjoint packet-moves of $\text{Small}(m)$ each. Hence, their total length is at least $4l \cdot b_{m-k} = 2l(b_m - k)$. Therefore, $|P|_{\text{Small}(m)} \geq 2l \cdot b_m - 2lk + 2 \cdot b_{m-1} - 2k + 2 = 2l \cdot b_m + 2 \cdot b_{m-1} - (2l+2)k + 2$. We denote this value by N .

Case 2: \tilde{P} contains a move of disk m to the auxiliary peg.

By Lemma 4.7, $\tilde{P}|_{\text{Small}(m)}$ contains three disjoint packet-moves of $\text{Small}(m)$. By Corollary 3.3, the other $2l$ packet-moves of D_m contain two disjoint

packet-moves of $Small(m)$ each. Thus, $P|_{Small(m)}$ contains $4l + 3$ disjoint packet-moves of $Small(m)$. By the induction hypothesis, $|P|_{Small(m)}| \geq (4l + 2) \cdot b_{m-k} + 2 \cdot b_{m-k-1} + 1 \geq 4l \cdot b_{m-k} + 4 \cdot b_{m-k-1} + 3 = 2l \cdot b_m + 2 \cdot b_{m-1} - (2l + 2)k + 3 = N + 1$ (the second inequality holds since the sequence (b_i) is strictly monotonous).

By Lemma 4.9, $|P|_{B_1(m)}$ is at least $(2l + 2)k - 1$. So, $|P| = |P|_{Small(m)}| + |P|_{B_1(m)}| \geq N + |P|_{B_1(m)}| \geq 2l \cdot b_m + 2 \cdot b_{m-1} - (2l + 2)k + 2 + (2l + 2)k - 1 = 2l \cdot b_m + 2 \cdot b_{m-1} + 1$, as required.

The bound is tight, since the sequence composed from $2l \beta_m$ and one α_m , in this order, is a perfect-to-perfect packet-move of D_m of length equal to this bound. \square

Now, Theorem 4.1 follows as the particular case of Proposition 4.8, where $l = 0$ and $m = n$.

Remark: Note that Corollary 4.6 is not necessary, since Proposition 4.8 covers it. Corollary 4.6 is included into Section 4.1 for the completeness of the autonomic consideration of Case 1 of Theorem 4.1.

Let us make refining observations.

Corollary 4.10 1. *No optimal algorithm for BTH_n contains a move of disk n to the auxiliary peg.*

2. *Any optimal algorithm for BTH_n contains just a single move of disk n , from the source peg to the target peg.*
3. *The only difference of an arbitrary optimal algorithm for BTH_n from α_n could be in choosing another optimal algorithms, for the two included optimal "somehow" packet-moves of D_{n-1} , instead of β_{n-1} .*

Proof: 1. Note that in the proof of Proposition 4.8, the bound for Case 2 exceeds strictly that for Case 1; hence, the minimum length of P can be achieved in Case 1 only. Observe that in the case $l = 0$ of Proposition 4.8, holds $2l + 1 = 1$, so there is just a single packet-move $\tilde{P} = P$; in other words, Cases 1 and 2 of Theorem 4.1 coincide with those in the proof of Proposition 4.8, for the case $l = 0$. Hence, the item follows.

2. The statement holds, since in the bound for Case 1, in the proof of Proposition 4.8, only one move of disk n is counted.

3. Obviously, during the single move of disk n , the entire D_{n-1} is gathered at *auxiliary*. So, the entire packet-move of D_n consists of a packet-move of

D_{n-1} from *source* to *auxiliary*, that move of disk n , and a packet-move of D_{n-1} from *auxiliary* to *target*, as in α_n . In the proof of Proposition 4.8, the lengths of the two packet-moves of D_{n-1} are counted as the minimum possible (b_{n-1}) each, so both of them should be optimal. \square

5 Discussion

5.1 Our Solution

Let us take a bird's-eye view on various possibilities to solve BTH_n . Algorithm α_n moves disk n just once, from *source* to *target*, while the set of all other $n - 1$ disks makes *two* packet-moves. This seems wasteful. Another idea of a perfect-to-perfect packet-move of D_n is to move *all* disks block by block. The simplest algorithm of this kind is β_n ; it is not correct since as a result, the order of the disks in $B_1(n)$ is inverse w.r.t. its initial order.

Let us try to correct it, suggesting the following algorithm β_n^1 (for illustration see Figure 4):

- Perform $\beta_{n-k}(\textit{source}, \textit{target})$.
- Perform the simple packet-move of $B_1(n)$ from *source* to *auxiliary*.
- Perform $\beta_{n-k}(\textit{target}, \textit{source})$.
- Perform the simple packet-move of $B_1(n)$ from *auxiliary* to *target*.
- Perform $\beta_{n-k}(\textit{source}, \textit{target})$.

Now, the order of disks in $B_1(n)$ is kept. In β_n^1 , only three packet-moves of $Small(n)$ are made, which is much shorter than the four packet-moves of $Small(n - 1)$ made in α_n , since the lengths of shortest packet-moves of $Small(n)$ and $Small(n - 1)$ are approximately the same. Hence, the latter approach seems promising.

On the other hand, the lack of β_n^1 is in that, when applying β_{n-k} three times, the order of the disks in $B_1(n - k) = B_2(n)$ is inversed. For correcting their order, *at the worst*, we can replace one of the three packet-moves of D_{n-k} , in β_n^1 , say, from peg X to peg Y , by two such packet-moves: from X to the spare peg Z and from Z to Y , thus resulting in four packet-moves of

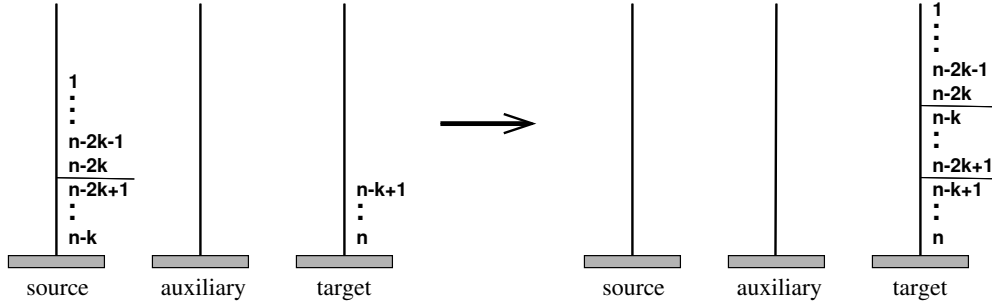


Figure 4: Performing the last packet-move of β_n^1 .

Small(n). A bit more economic approach to keep the initial order of disks in $B_2(n)$ is to replace, in β_n^1 , the first or the last β_{n-k} by α_{n-k} . Unfortunately, both suggestions almost *double* the length of the replaced packet-move, arriving at sequences longer than α_n .

Seemingly, it is worth to try correcting the wrong order of the disks in $B_2(n)$ by a more economical method. Notice that we have the freedom to place disks in $B_1(n) \setminus \{n\}$ independently of the placement of disk n . Moreover, these disks may be placed interleaved with the disks in $B_2(n)$. Thus, a less rigid framework than that of β_n^1 is considered, where the entire packet-move is not necessarily sub-divided into disjoint packet-moves of *Small(n)* and $B_1(n)$. May be, this would help? At least, the situation seems to be unclear.

It turns out that such attempts do not succeed. Proposition 4.8 shows that there exists no modification of β_n^1 , as needed, cheaper than that with two executions of β_{n-k} and one of α_{n-k} . An analytic method for proving this is to consider separately *restrictions* of the entire packet-move to $B_1(n)$ and *Small(n)* or to $B_1(n)$, $\{n-k\}$, and *Small(n-1)*, thus abstracting from a possible interleaved placement of disks in those sets.

The proof of Proposition 4.8 reveals the following interesting effect. An attempt to keep the right order of disks in $B_2(n)$ during *three* packet-moves of *Small(m)* results in an algorithm (β_n^2), generating the wrong order of disks in $B_3(n)$ during *seven* packet-moves of D_{n-2k} ; a further attempt to correct such seven packet-moves results in an algorithm (β_n^3), generating *fifteen* packet-moves of D_{n-3k} , and so on. In fact, such a sequence of attempts starts from the *single* β_n , with the wrong order at $B_1(n)$.

The resulting “tower” of possibilities is resolved by the inductive proof of Proposition 4.8. Interestingly, in the analysis of an arbitrary sequence of an

odd number of packet-moves, we are able to *detect* a “guilty” packet-move—that keeping the order of the two biggest disks,—which is almost twice as long as the shortest possible packet-move of the disk set of the same size.

5.2 Other Attempts to Solve the Problem

In 1992, Poole suggested (indirectly) both algorithms β and α for the bottleneck ToH problem, presented proofs of their optimality, and computed the lengths of the resulting sequences of moves [3]. However, the optimality of both algorithms was proved insufficiently, ignoring some important cases. In particular, when proving optimality of α_n , all the three statements of our Corollary 4.10 were assumed for granted. Hence, the entire Case 2 of our proof—when disk n is moved to the auxiliary peg,—which required our greatest effort, and some substantial considerations made in our Case 1 were excluded.

More concretely, Lemma 4 of [3] asserts optimality of α_n . In its proof, it is stated that before the last move of disk n to the (empty) target peg, *all* other $n - 1$ disks must be gathered on the spare peg. This is not general, since before the last move of disk n , from some peg X to the target peg, any of the disks $n - 1, n - 2, \dots, n - k + 1$ may be placed below disk n on peg X . Thus, in particular, the entire competing approach sampled by algorithms $\beta_n^i, i = 1, 2, \dots$, (see Section 5.1) is ignored.

In 1998, students S. Beneditkis and I. Safro, supervised by D. Berend, made a serious work on *BTH* (unaware of the paper of Poole) [1]. They developed a system of notions, described (the same) algorithms, and for the case $k = 2$, proved the optimality of β_n and α_n . Besides, they presented the competing approach (described in Section 5.1). They suggested the recursive algorithm β_n^∞ (our notion), differing from β_n^1 in that the last packet-move of D_{n-k} is replaced by β_{n-k}^∞ ; they proved that β_n^∞ is longer than α_n .

In 2005, X. Chen et al. considered independently a few ToH problems, including the bottleneck ToH problem (named differently) [2]. The considered algorithms were provided with optimality proofs, based on another technical approach. In our work, we based on [1] (being unaware of [3, 2]).

We believe that the recent advancement in solving the bottleneck ToH problem is done thanks to the algorithmic framework, in particular, a thorough case analysis, mostly novel in the ToH area. Hopefully, joint efforts of Mathematics and Computer Science would bring new fruits soon.

We would like to mention that the main challenge, in the ToH area, is the notorious Frame-Stewart conjecture on the optimality of some beautiful algorithm for the four (or more) pegs ToH problem (see e.g. [4]), which was suggested in 1941. It is believed to be valid, but very hard to prove. The problem is very popular in the ToH literature (see [5]), but the main issues of the related research are in refining bounds and finding equivalent and similar problems.

6 Directions of Further Research

The following questions show directions of possible further research.

1. Is α_n a unique optimal solution to BTH_n ? If not, what is the family of all optimal solutions? What is their number?
2. In the classic Tower of Hanoi problem, there are n disks of different sizes, so that the exact size of each disk is not significant. However, in BTH_n , the exact size of a disk plays an important role and influences the behavior of the problem. It is interesting to generalize BTH_n by allowing the sizes of the n disks be an arbitrary set of distinct integers.
3. While the shortest perfect-to-perfect sequence of moves had been already found, we do not know what is such a sequence for transforming one given (legal) configuration to another given (legal) one, and what is its length. In particular, what is the length of the longest one among all shortest sequences of moves, over all pairs of initial and final configurations? In other words, what is the diameter of the directed graph of all the configurations of the disks in $[1..n]$, under the k -relaxed placement rule?
4. How many configurations, under the k -relaxed placement rule, exist? How many gathered ones exist? (a closed formula depending on k and n is expected).
5. How to generalize the well known algorithm of Frame and Stewart for the four (or more) pegs ToH problem (see e.g. [4]), to the case when the placement rule is k -relaxed?

In general, every Tower of Hanoi problem can be generalized using the k -relaxed placement rule.

Acknowledgment

Authors thank Daniel Berend for his suggestion to consider the question on the optimal algorithm for the bottleneck Tower of Hanoi problem.

References

- [1] S. Beneditkis and I. Safro. Generalizations of the Tower of Hanoi Problem. Final Project Report, supervised by D. Berend, Dept. of Math. and Comp. Sci., Ben-Gurion University, 1998.
- [2] X. Chen, B. Tian, and L. Wang. Santa Claus' Towers of Hanoi. A manuscript, 2005.
- [3] D. Poole. The Bottleneck Towers of Hanoi Problem. *J. of Recreational Math.* **24** (1992), no. 3, 203-207.
- [4] P.K. Stockmayer. Variations on the Four-Post Tower of Hanoi Puzzle. *CONGRESSUS NUMERANTIUM 102 (Proc. of the 25th Southeastern Int. Conf. on Combinatorics, Graph Theory and Computing)* (1994), 3-12. Also available via http://www.cs.wm.edu/~pkstoc/h_Papers.html .
- [5] P.K. Stockmayer. The Tower of Hanoi: A Bibliography. (1998). Available via http://www.cs.wm.edu/~pkstoc/h_Papers.html .
- [6] D. Wood. The Towers of Brahma and Hanoi revisited. *J. of Recreational Math.* **14** (1981-1982), no. 1, 17-24.