Which Multi-peg Tower of Hanoi Problems Are Exponential?

Daniel Berend[1] and Amir Sapir[2,3]

[1] Departments of Mathematics and Computer Science, Ben-Gurion University, Beer-Sheva, Israel
berend@cs.bgu.ac.il
http://www.cs.bgu.ac.il/~berend
[2] Software Systems Department, Sapir Academic College, Western Negev, Israel*
[3] The Center for Advanced Studies in Mathematics at Ben-Gurion University, Beer-Sheva, Israel
amirsa@cs.bgu.ac.il
http://www.cs.bgu.ac.il/~amirsa

Abstract. Connectivity properties are very important characteristics of a graph. Whereas it is usually referred to as a measure of a graph's vulnerability, a relatively new approach discusses a graph's *average connectivity* as a measure for the graph's performance in some areas, such as communication. This paper deals with Tower of Hanoi variants played on digraphs, and proves they can be grouped into two categories, based on a certain connectivity attribute to be defined in the sequel.

A major source for Tower of Hanoi variants is achieved by adding pegs and/or restricting direct moves between certain pairs of pegs. It is natural to represent a variant of this kind by a directed graph whose vertices are the pegs, and an arc from one vertex to another indicates that it is allowed to move a disk from the former peg to the latter, provided that the usual rules are not violated. We denote the number of pegs by h. For example, the variant with no restrictions on moves is represented by the Complete $\mathtt{K}_h$ graph; the variant in which the pegs constitute a cycle and moves are allowed only in one direction — by the uni-directional graph $\mathtt{Cyclic}_h$.

For all 3-peg variants, the number of moves grows exponentially fast with n. However, for $h \geq 4$ peg variants, this is not the case. Whereas for $\mathtt{Cyclic}_h$ the number of moves is exponential for any h, for most of the other graphs it is sub-exponential. For example, for a path on 4 vertices it is $O(\sqrt{n}3^{\sqrt{2n}})$, for n disks.

This paper presents a necessary and sufficient condition for a graph to be an H-subexp, i.e., a graph for which the transfer of n disks from a peg to another requires sub-exponentially many moves as a function of n.

To this end we introduce the notion of a shed, as a graph property. A vertex v in a strongly-connected directed graph $G = (V, E)$ is a *shed* if the subgraph of G induced by $V - \{v\}$ contains a strongly connected subgraph on 3 or more vertices. Graphs with sheds will be shown to be much more efficient than those without sheds, for the particular domain of the Tower of Hanoi puzzle. Specifically we show how, given a graph

* Research supported in part by the Sapir Academic College, Israel.

M.C. Golumbic et al. (Eds.): WG 2012, LNCS 7551, pp. 81–90, 2012.

with a shed, we can indeed move a tower of n disks from any peg to any other within $O(2^{\varepsilon n})$ moves, where $\varepsilon > 0$ is arbitrarily small.

Keywords: Tower of Hanoi, directed graphs, connectivity, sub-exponential complexity, shed.

1 Introduction

Given are 3 pegs and a certain number n of disks of distinct sizes. Initially, the disks form a tower: the largest at the bottom of one of the pegs (the source), the second largest on top of it, and so on, until the smallest at the top of that peg. The well-known Tower of Hanoi problem asks: how do we optimally move the tower to another peg (the destination peg), bounded by the *Hanoi rules* (henceforth HR): 1) At each step only one disk is moved. 2) The moved disk must be a topmost one. 3) At any moment, no disk may reside on a smaller one.

The game was composed over a hundred years ago by Lucas [18]. Ever since then, it was studied from numerous points of view. For example, in [1] it is shown that, with a direct approach coding, the string representing the optimal solution is square-free. This line of work was extended in [2]. As computer science education evolved, the Tower of Hanoi problem has been used as a common example, demonstrating the elegance of recursive programming. The reader is referred to [29] for a review of the history of the problem, and to [30] for an extensive bibliography of papers on various lines of research in the field.

Many variants of the original puzzle came up, some of which we will describe here, though not chronologically. Without changing the basic peg configuration (3 pegs, each pair being connected bi-directionally), one direction is solving the problem for any initial and final configurations [12]. Other challenging versions have been proposed and solved in [20],[21],[22],[23],[19]. In another direction a disk may reside on top of a smaller one, with various limitations, [16], [9].

Another version of the original problem, which was discussed in several papers, is where we impose restrictions on the movements between pegs. In [26], [29], [14], the "three-in-a-row" ($\mathtt{Path}_3$) arrangement is studied. The uni-directional cycle ($\mathtt{Cyclic}_3$) has been solved in [3],[11]. As was mentioned, it is natural to represent a variant by a directed graph. Surprising in its simplicity, a necessary and sufficient condition for a variant to be solvable, for any source and destination pegs and any number of disks, is that the corresponding graph is strongly connected [17]. For 3 pegs there are 5 (up to isomorphism) strongly connected variants. A single optimal algorithm for all these variants was obtained in [25], accompanied with an explicit formula for the minimal number of moves for each variant. (Note that individual algorithms and explicit formulas were known beforehand for $\mathtt{K}_3$, $\mathtt{Path}_3$, $\mathtt{Cyclic}_3$, as mentioned above.)

Probably the first multi-peg version is "The Reve's Puzzle" [10, pp. 1−2], in which there are 4 pegs and various specific numbers of disks. It has been generalized to any number of pegs and any number of disks in [27], with solutions in [28] and [13], which were (among several other solutions) proved to be identical

in [15]. An analysis of the algorithm reveals, somewhat surprisingly, that the number of moves in the solution grows sub-exponentially as a function of n. In the case of 4 pegs, it grows like $\Theta(\sqrt{n}2^{\sqrt{2n}})$ (cf. [29]). The lower bound issue was considered in [31] and [8], where it has been shown to grow at a rate close to that yielded by the algorithm.

Allowing 4 pegs and above, and imposing movement restrictions, we obtain a huge number of graphs (83 non-isomorphic strongly connected digraphs on 4 vertices already), and no algorithm seems a natural candidate to be optimal. The question whether a variant is sub-exponential had been resolved only for particular ones: Star [29], Cyclic [5] and Path [7]. Whereas the complexity of the majority of the multi-peg variants is sub-exponential, Cyclic is among the few which are exponential.

The fact that, even for the original multi-peg variants, on complete graphs, it is not known whether the proposed algorithms are optimal, indicates that the complexity issue here is non-trivial. Facing the wealth of variants, we would like an easy way to determine, given a variant, whether it is exponential or sub-exponential. This paper presents a simple necessary and sufficient condition for a variant to be sub-exponential.

In Section 2 we describe the problem domain. The main results are introduced in Section 3. The proofs of the theorems are presented in Section 4.

2 Problem Domain and Notations

Any arrangement of pegs and their immediate connections, such that each peg is reachable from each other, constitutes a variant. As mentioned above, it is natural to represent a variant by a digraph G, whose vertices are the pegs, and an arc from one vertex to another designates the ability of moving a disk from the former peg to the latter, provided that the HR are obeyed. In the sequel, when we mention a *variant graph*, we mean a strongly connected simple directed graph on $h \geq 3$ vertices.

A *configuration* is a distribution of the disks among the pegs, satisfying HR.3. A configuration is *perfect* if all disks reside on the same peg. Such a configuration will be denoted by $R_{i,n}$, where n is the number of disks (disk 1 being the smallest and disk n the largest) and i the peg containing the disks.

Given a variant graph G and a positive integer n, the corresponding *configuration graph* $G^{(n)}$ is the graph whose vertices are all configurations of n disks over G, where there is an arc from a vertex to another if one can pass from the former to the latter by a single disk move. More generally, for any pair of configurations there is a corresponding *task*, of passing from the first to the second. Thus, an optimal solution of a task corresponds to a shortest path between the vertices of $G^{(n)}$ representing the task's initial and final configurations. The diameter of $G^{(n)}$ is denoted by $D_n(G)$.

A task is *perfect* if both its initial and final configurations are perfect. We use the notation $R_{i,n} \rightarrow R_{j,n}$ both for the task and for a minimal length solution of it. The length of such a minimal solution is expressed by $|R_{i,n} \rightarrow R_{j,n}|$. We

set $d_{i,j,n}(G) = |R_{i,n} \to R_{j,n}|$. An interesting quantity is $d_n(G) = \max_{i,j} d_{i,j,n}(G)$, which we call the *little diameter* of $G^{(n)}$. When the identity of the graph to which we refer is clear, we may omit its indication. For example, we may write D_n instead of $D_n(G)$.

Formally, a *move* is composed of the disk being moved, the peg on which it resides prior to the move, and the peg to which it is transferred. A *solution* to a task is a sequence of moves accomplishing it. The algorithms constructed in the proofs of our results produce solutions to all perfect tasks.

A variant graph G is *H-exp* if $D_n(G)$ grows exponentially fast as a function of the number of disks, namely there exist $C > 0$ and $\lambda > 1$ such that $D_n(G) \geq C\lambda^n$ for all n. G is *H-subexp* if for every $\varepsilon > 0$ there exists a constant $C = C(\varepsilon)$ such that $D_n(G) \leq C(1+\varepsilon)^n$. It will follow, in particular, from Theorem 1 below that each graph is either H-exp or H-subexp.

3 Main Results

The main problem we study is how to identify, given a variant graph, whether it is H-subexp or H-exp. We start by proving that the number of moves behaves regularly as a function of the number of disks.

For a variant graph G, denote $\lambda_G = \inf_{n\geq 1} \sqrt[n]{d_{n+1}(G)}$.

Theorem 1. *For any variant graph G*

$$\lim_{n\to\infty} \sqrt[n]{d_n(G)} = \lambda_G\,.$$

Let us recall Corollary 1 of [6]:

Proposition 1. *For any variant graph and any number of disks,*

$$D_n \leq (2n-1)d_n\,.$$

Combining Theorem 1 and Proposition 1 one can infer

Corollary 1. *For every $\varepsilon > 0$ there exists an n_0 such that*

$$\lambda_G^{n-1} \leq d_n(G) \leq D_n(G) \leq (\lambda_G + \varepsilon)^{n-1}, \qquad n \geq n_0.$$

The main question this paper answers is: what property must G have so that $\lambda_G = 1$? To answer this, we need the following notion.

Definition 1. A *shed* in a strongly connected digraph G (see Fig. 1) is a vertex w with the property that the graph induced by $V(G) - \{w\}$ contains a strongly connected subgraph of size at least 3.

Our main result is

Theorem 2. *A variant graph G with $h \geq 3$ vertices is H-subexp if and only if it contains a shed.*

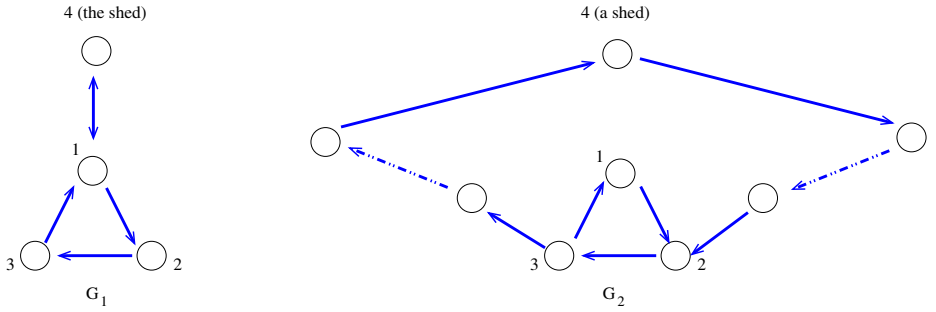

Fig. 1. Two example graphs with sheds

Unlike the proof of sub-exponentiality for $\mathtt{K}_4$ (cf. [29]) or the path $\mathtt{P}_h$ [7], the proof for general graphs containing a shed is quite cumbersome. The reason is, intuitively, that the former graphs have (at least) two sheds. This means that there are a lot more options of keeping a block of small disks on some peg for a while, taking care in the meantime of other (large) disks. On the other hand, if the graph has a single shed, then no obvious sub-exponential algorithm comes to mind, and in fact it may seem surprising at first glance that such an algorithm exists at all.

4 Proofs

To prove Theorem 1, we first need

Lemma 1. *For every strongly connected graph G,*

$$d_{m+n-1} \leq d_m d_n \qquad m, n \geq 1.$$

The proof follows, to a certain extent, the idea of the proof of Theorem 2(a) in [5], and is omitted.

Proof of Theorem 1: Putting $b_n = \ln d_{n+1}$, we obtain by Lemma 1:

$$b_{m+n} \leq b_n + b_m\,, \qquad m, n \geq 0\,.$$

The sequence $(b_n)_{n=0}^{\infty}$ is thus sub-additive, which implies that the sequence $\frac{b_n}{n}$ converges to its greatest lower bound (cf. [24, p. 198]), and hence so does the sequence

$$e^{b_n/n} = \sqrt[n]{d_{n+1}}\,,$$

thus proving the theorem.

Proof of Corollary 1: Since $\sqrt[n]{d_{n+1}(G)}$ converges to λ_G from above, for any $\varepsilon > 0$ and sufficiently large n

$$\lambda_G \leq \sqrt[n]{d_{n+1}(G)} \leq \lambda_G + \varepsilon\,,$$

or, equivalently,

$$\lambda_G^{n-1} \leq d_n(G) \leq (\lambda_G + \varepsilon)^{n-1}\,.$$

In combination with Proposition 1, this proves the corollary.

Remark 1. Corollary 1 states that d_n and D_n are not too far apart, justifying that it suffices to focus on d_n.

To establish the proof of the 'if' part (which is the main part) of Theorem 2, we will need several definitions. Unless explicitly stated otherwise, G will denote a strongly connected graph with $h \geq 4$ vertices containing a shed w, and G' – a strongly connected subgraph of size at least 3 of the graph left after w is removed. (Note that G may contain more than one shed, and for each choice of a shed there may be several appropriate strongly connected subgraphs of G, but for our purposes any choice of a shed vertex and a strongly connected component in accordance with that shed is suitable.) Taking a shortest path from $w \in V(G) - V(G')$ to G', we denote the *entrance vertex* to G' – the last vertex on this path and the only one that belongs to G' – by e. Similarly, an *exit vertex* $x \in V(G')$ is the first vertex on a shortest path leading from G' to w. These paths will be denoted by $p_{w,e}$ and $p_{x,w}$, respectively. The sequence of moves of disk r along, say, the path $p_{w,e}$ is expressed by $t_{w,e,r}$. $S_{(l)}$ is the l-th move in a sequence of moves S.

Example 1. In each of the graphs G_1 and G_2 in Figure 1, the shed is vertex $w = 4$. (In G_2 one could choose any vertex but 2 and 3 as a shed.) For G_1 we have $e = x = 1$, whereas for G_2 we have $e = 2, x = 3$.

As hinted above, most of the work is required for the 'if' part of Theorem 2. The following lemma shows that some of the tasks are of sub-exponential complexity. Let $\texttt{Inner} = V(G') - \{e, x\}$. We first show that tasks, in which the source is the shed and the destination lies in `Inner` (or vice versa), are sub-exponential.

Lemma 2. *Let G, G', w be as above and $v \in$ `Inner`. Then $|R_{w,n} \to R_{v,n}|$ and $|R_{v,n} \to R_{w,n}|$ grow sub-exponentially fast as functions of n.*

Sketch of proof: Fix a number $\lambda > \lambda_{G'}$. It suffices to show that for every $\varepsilon > 0$ there exists a $C = C(\varepsilon)$ such that

$$|R_{w,n} \to R_{v,n}| \leq C\lambda^{\varepsilon n}\,, \qquad n = 1, 2, \ldots\,,$$

and

$$|R_{v,n} \to R_{w,n}| \leq C\lambda^{\varepsilon n}\,, \qquad n = 1, 2, \ldots\,.$$

The proof is based on the procedures `ShedToInner(1...n)` (Algorithm 1) and `InnerToShed(1...n)` (of a similar kind), which perform the tasks $R_{w,n} \to R_{v,n}$ and $R_{v,n} \to R_{w,n}$ in $f_{w,v}(n) \leq C\lambda^{\varepsilon n}$ and $f_{v,w}(n) \leq C\lambda^{\varepsilon n}$ moves, respectively.

The following lemma is required for the proof of the 'only if' part of Theorem 2.

Algorithm 1. `ShedToInner`$(1 \dots n)$

/* This algorithm moves a tower of n disks from the shed w to vertex v, which is neither the entrance nor the exit vertex with respect to w. */
if $n \leq \frac{1}{\varepsilon}$ **then**
  $T \leftarrow$ a sequence of moves that performs the task
else
  $m \leftarrow \lceil n(1-\varepsilon) \rceil$
  $T \leftarrow$ `ShedToInner`$(1 \dots m)$
  for $r \leftarrow m+1$ to n **do**
    $T \leftarrow T * t_{w,e,r}$
    $T \leftarrow T *$ `InnerToShed`$(1 \dots m)$
    $T \leftarrow T *$ `Accumulate`$(e, v, r, m+1 \dots r-1)$
    $T \leftarrow T *$ `ShedToInner`$(1 \dots m)$
  end for
end if
return T

Lemma 3. *There exists a constant C with the following property. For any graph G, not containing a strongly connected component of size 3 or more, and for any initial configuration, the number of different disks which can participate in any legal sequence of moves is bounded above by $Ch^{\frac{1}{2}\lg h+2}$, where $h = |V(G)|$, and $\lg h \equiv \log_2 h$.*

Proof: Let T be any legal move sequence and i any peg.
Set $l = \lfloor (h-2)Ch^{\frac{\lg h}{2}} \rfloor + 2$ (where the constant C is sufficiently large; see below). Consider the first move of the disk residing at the l-th place (counting from the top) of peg i before T is started. Right before this move, all smaller disks (residing on peg i prior to T) have to be spread on $h-2$ of the other pegs, making it necessary for at least one peg to accept more than $Ch^{\frac{1}{2}\lg h}$ disks from peg i, contradicting [4, Theorem 1.3] if C is sufficiently large. It follows that the overall number of disks that can participate in any task is bounded above by $h(h-2)Ch^{\frac{1}{2}\lg h} + 2h$, proving the lemma.

Proof of Theorem 2: (a) We prove first the 'if' part of the theorem. Let w and G' be as before. Take $\lambda > \lambda_{G'}$. We will show that, for every pair of vertices $i, j \in V(G)$ and $\varepsilon > 0$, there exists a constant K such that

$$|R_{i,n} \to R_{j,n}| \leq K\lambda^{\varepsilon n}, \qquad n = 1, 2, \dots . \tag{1}$$

Take $v \in V(G') - \{e, x\}$, where e is the entrance vertex and x the exit vertex in G'. Due to Lemma 2, it remains to consider the following cases:

- Case 1: $i = w$, $j \in V(G') - \{v\}$,
- Case 2: $i \in V(G') - \{v\}$, $j = w$,
- Case 3: $i, j \in V(G')$,
- Case 4: $i, j \in V(G)$, where at least one of i, j does not belong to $V(G') \cup \{w\}$.

(In fact, in Case 1 we need only to take care of the case where $j = e$, and in Case 2 only of $i = x$, but this is of no consequence.)

Case 1: We employ `ShedToAny`$(j, 1 \dots n)$ (Algorithm 2), which moves a tower of n disks from the shed to any vertex $j \neq v \in G'$. It divides the disks into $p+1$ sets, where $p = \left\lfloor \frac{n}{\lfloor \varepsilon n \rfloor} \right\rfloor$ of the sets consist of $l = \lfloor \varepsilon n \rfloor$ consecutive disks each, and the remaining (possibly empty) set consists of up to $l - 1$ disks, which are the smallest disks. Then it moves each set to its destination (vertex j) by:

- Moving all the disks from the shed to vertex v.
- Moving all but the largest $\lfloor \varepsilon n \rfloor$ disks in the set from vertex v back to the shed.
- Moving the largest $\lfloor \varepsilon n \rfloor$ disks from vertex v to the destination.

The procedure `MoveIn`$G'(e, v, m+1 \dots r-1)$ performs the task of moving the tower consisting of disks $m+1, \dots, r-1$ from e to v. By Theorem 1, the length of the solution it produces is bounded above by $C' \lambda^{r-m-1}$, where C' is an appropriate constant, determined by G'.

`Accumulate`$(e, v, r, m+1 \dots r-1)$ adds the 'next' disk to the column already handled. Starting with disks $m+1 \dots r-1$ on peg v and disk r on peg e, utilizing G' only, it unites them so that all disks will reside on peg v. Thus, the number of moves produced by its $(r-m)$-th invocation is bounded above by $2C' \lambda^{r-m}$.

Algorithm 2. `ShedToAny`$(j, 1 \dots n)$

```
/* ShedToAny moves a tower of n disks from the shed to any vertex j ≠ v ∈ G'. */
if n ≤ 1/ε then
  T ← a sequence of moves that performs the task
else
  l ← ⌊nε⌋ ;      p ← ⌊n/l⌋ ;      m ← n − pl
  T ← [ ]                          /* The empty sequence */
  for r ← 1 to p do
    T ← T* ShedToInner(1 ... n)
    T ← T* InnerToShed(1 ... n − l)
    T ← T* MoveIn G'(v, j, n − l + 1 ... n)
    n ← n − l
  end for
  T ← T * ShedToInner(1 ... m) * MoveIn G'(v, j, 1 ... m)
end if
return T
```

`ShedToInner`$(1 \dots n)$ and `InnerToShed`$(1 \dots n)$ require at most $C\lambda^{\varepsilon n}$ moves each, for an appropriate C. We now bound the number of moves $f_{w,j}(n)$ performed by the procedure `ShedToAny`$(j, 1 \dots n)$. For sufficiently large n:

$$f_{w,j}(n) \leq (p+1)C'\lambda^l + 2\sum_{r=1}^{p} C\lambda^{\varepsilon lr} + C\lambda^{\varepsilon n}$$

$$\leq ((p+1)C' + 3C)\lambda^{\varepsilon n} + 2C\sum_{r=1}^{p-1} \lambda^{\varepsilon lr}$$

$$\leq ((\tfrac{1}{\varepsilon}+2)C' + 3C)\lambda^{\varepsilon n} + 2C\frac{\lambda^{\varepsilon l(\frac{1}{\varepsilon}+1)}}{\frac{1}{2}\lambda^{\varepsilon l}}$$

$$\leq ((\tfrac{1}{\varepsilon}+2)C' + 7C)\lambda^{\varepsilon n}\,.$$

Case 2: We employ `AnyToShed`$(i, 1 \ldots n)$, which moves a tower of n disks from any vertex $i \neq v \in G'$ to the shed. We omit the description of this algorithm, which is similar to `ShedToAny` in the way it works, as well as in its analysis.

Case 3 is a consequence of the first two cases, by first moving all disks from i to w, and then moving them from w to j.

Case 4: Observe that, once G' has been set, any vertex in $V(G) - V(G')$ may serve as a shed. Thus, for any $i \in V(G')$ and $j \in V(G) - V(G')$, both inequalities $|R_{i,n} \to R_{j,n}| \leq K\lambda^{\varepsilon n}$ and $|R_{j,n} \to R_{i,n}| \leq K\lambda^{\varepsilon n}$ hold. Now let $i, j \in V(G) - V(G')$. Take a vertex $k \in V(G')$. We move a tower of disks from i to j by first moving it from i to k, and then from k to j. This proves the correctness of (1) in this case.

(b) Sketch of proof of the 'only if' part of Theorem 2.

Take any strongly connected graph G without a shed. Start with any configuration C. Denote by v_0 the vertex on which disk 1 resides in C. Without reaching the same configuration more than once, what is the maximal number of moves which may be done without moving disk 1?

Clearly, as long as we do not move disk 1, all other disks may use only the graph induced by $V - \{v_0\}$. This graph has no strongly connected component of size at least 3. By Lemma 3, since in this graph there are $h-1$ vertices, $m = C(h-1)^{\frac{1}{2}\lg(h-1)+2}$ is an upper bound for the number of disks that may participate in any sequence of moves, yielding exponential growth.

References

1. Allouche, J.-P., Astoorian, D., Randall, J., Shallit, J.: Morphisms, squarefree strings, and the Tower of Hanoi puzzle. Amer. Math. Monthly 101, 651–658 (1994)
2. Allouche, J.-P., Sapir, A.: Restricted Towers of Hanoi and Morphisms. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 1–10. Springer, Heidelberg (2005)
3. Atkinson, M.D.: The cyclic Towers of Hanoi. Inform. Process. Lett. 13, 118–119 (1981)
4. Azriel, D., Berend, D.: On a question of Leiss regarding the Hanoi Tower problem. Theoretical Computer Science 369, 377–383 (2006)
5. Berend, D., Sapir, A.: The Cyclic multi-peg Tower of Hanoi. Trans. on Algorithms 2(3), 297–317 (2006)
6. Berend, D., Sapir, A.: The diameter of Hanoi graphs. Inform. Process. Lett. 98, 79–85 (2006)

7. Berend, D., Sapir, A., Solomon, S.: Subexponential upper bound for the Path multi-peg Tower of Hanoi. Disc. Appl. Math (2012)
8. Chen, X., Shen, J.: On the Frame-Stewart conjecture about the Towers of Hanoi. SIAM J. on Computing 33(3), 584–589 (2004)
9. Dinitz, Y., Solomon, S.: Optimality of an algorithm solving the Bottleneck Tower of Hanoi problem. Trans. on Algorithms 4(3), 1–9 (2008)
10. Dudeney, H.E.: The Canterbury Puzzles (and Other Curious Problems). E. P. Dutton, New York (1908)
11. Er, M.C.: The Cyclic Towers of Hanoi: a representation approach. Comput. J. 27(2), 171–175 (1984)
12. Er, M.C.: A general algorithm for finding a shortest path between two n-configurations. Inform. Sci. 42, 137–141 (1987)
13. Frame, J.S.: Solution to advanced problem 3918. Amer. Math. Monthly 48, 216–217 (1941)
14. Guan, D.-J.: Generalized Gray codes with applications. Proc. Natl. Sci. Counc. ROC(A) 22(6), 841–848 (1998)
15. Klavžar, S., Milutinović, U., Petr, C.: On the Frame-Stewart algorithm for the multi-peg Tower of Hanoi problem. Disc. Appl. Math. 120(1-3), 141–157 (2002)
16. Klein, C.S., Minsker, S.: The super Towers of Hanoi problem: large rings on small rings. Disc. Math. 114, 283–295 (1993)
17. Leiss, E.L.: Solving the "Towers of Hanoi" on graphs. J. Combin. Inform. System Sci. 8(1), 81–89 (1983)
18. Lucas, É.: Récréations Mathématiques, vol. III. Gauthier-Villars, Paris (1893)
19. Lunnon, W.F., Stockmeyer, P.K.: New Variations on the Tower of Hanoi. In: 13th Intern. Conf. on Fibonacci Numbers and Their Applications (2008)
20. Minsker, S.: The Towers of Antwerpen problem. Inform. Process. Lett. 38(2), 107–111 (1991)
21. Minsker, S.: The Linear Twin Towers of Hanoi problem. Bulletin of ACM SIG on Comp. Sci. Education 39(4), 37–40 (2007)
22. Minsker, S.: Another brief recursion excursion to Hanoi. Bulletin of ACM SIG on Comp. Sci. Education 40(4), 35–37 (2008)
23. Minsker, S.: The classical/linear Hanoi hybrid problem: regular configurations. Bulletin of ACM SIG on Comp. Sci. Education 41(4), 57–61 (2009)
24. Pólya, G., Szegő, G.: Problems and Theorems in Analysis, vol. I. Springer (1972)
25. Sapir, A.: The Tower of Hanoi with forbidden moves. Comput. J. 47(1), 20–24 (2004)
26. Scorer, R.S., Grundy, P.M., Smith, C.A.B.: Some binary games. Math. Gazette 280, 96–103 (1944)
27. Stewart, B.M.: Advanced problem 3918. Amer. Math. Monthly 46, 363 (1939)
28. Stewart, B.M.: Solution to advanced problem 3918. Amer. Math. Monthly 48, 217–219 (1941)
29. Stockmeyer, P.K.: Variations on the Four-Post Tower of Hanoi puzzle. Congr. Numer. 102, 3–12 (1994)
30. Stockmeyer, P.K.: Tower of hanoi bibliography (2005), `http://www.cs.wm.edu/~pkstoc/biblio2.pdf`
31. Szegedy, M.: In How Many Steps the k Peg Version of the Towers of Hanoi Game Can Be Solved? In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 356–361. Springer, Heidelberg (1999)