COMBINATORIAL GAMES: TOWERS FOR THE K-PEG GAME

Samee Ullah Khan
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX-76019, USA
sakhan@cse.uta.edu

Abstract. The "Towers of Hanoi" puzzle invented by Edouard Lucas in 1883 is one of the most widely studied problems in Computer Science. In this paper we will talk about a combinatorial game played by two players which has similar characteristics to that of the Towers of Hanoi problem, the *k-peg game*. We will introduce some of the known results, and show that this game is has a first player forced win.

Introduction. Most of the combinatorial games have high computational costs, majority either belong to the class NP or the class PSPACE. On such family of games called the "pebble game" first introduced in [Kasai et al. 1979] are also mostly NP-complete. The pebble game is a conceptual game played on graphs with vertices which form the pits and pebbles as marbles. The basic moves involve displacing the pebbles according to certain predefined rules. The goal of the game is to put a pebble onto a particular place. An interesting variant is called the cat and k-mouse game. The cat and k-mouse game is played on a directed graph; the cat tries to catch a mouse, and all the mice try to escape the cat's claws and reach a predefined goal node [Adachi et al. 1984].

What is the k-peg game? The k-peg game is similar to the Towers of Hanoi problem. Played by two players, each player alternately moves some donut shaped disks from peg to peg according to certain predefined rules. Initially all the k-disks are threaded onto the first peg. The winner of the game is the one who first places all the k disks onto the last peg, or who forces the other player into a position from which he is unable to move any of the disks. Formally, we can define the k-peg game G as a triplet (V, k, l). Thus, there are initially l pegs fixed in upright position on a board,

and k disks. A rule vector $v = (v_1, ..., v_l) \in V$ means that for each i, if $v_i \geq 0$ then we put v_i disks on the i-th peg, if $v_i < 0$ then we remove v_i disks from the i-th peg.

The two main problems: The two very natural questions asked about the k-peg game would be: 1. Does the first player have a forced win over the second player on arbitrary configuration of the k-peg game? 2. If there is an optimal solution to the game, what would be its computational cost? The game seems fairly easy to analyze, but the complexity comes from the exponential number of choices available to both the players, with the increase in the number of pegs.

Known Results. With the number of disks fixed, the k-peg game was show to have a forced first player win in $\Omega(n^{(k-2)/4-\epsilon})$ for any $\epsilon > 0$ and $k > 6$ [Adachi et al. 1984]. The basic assumption of the fixed number of disks is important. If assumed otherwise, the k-peg game will take at least exponential time to complete. The result of Adachi et al. is obtained by making the k-pebble game problem polynomial time reducible to the (k+1)-peg game. The k-pebble is an immediate variant of the family of pebble games. Here the player who moves k number of pebbles to a designated location wins. The problem of having a forced first player win in k-pebble game was shown impossible to be computed within $O(n^{(k-1)/4-\epsilon})$ for any $\epsilon > 0$.

Their result of $\Omega(n^{(k-2)/4-\epsilon})$ win for the first player has a very unique peg construction. The total number of l pegs is divided into three parts. The first m pegs are used for nodes in G to represent the placement of the disks in G. The next $m + 4n$ pegs were used to simulate the optimal moves for the forced win, and the last 3 pegs are used as control pegs (similar to the finite control in a Turing machine). Thus, the final goal peg where k disks have to be displaced would be the $(2m + 4n + 3)$rd peg. Since the last 3 pegs are not useful initially in the displacement of the disks, the moves are only defined over $p = 2m + 4n$ number of pegs. The optimal moves are defined as a vector set $V = V_1 \cup V_2 \cup V_3$. We will now describe each one of the vector subsets individually.

For V_1 (first m pegs) the set consists of two vectors u and v as:

$$u(i)=\begin{cases}-(k+1) & if \quad i=1,\\ k+1 & if \quad i=p+2,\\ 0 & otherwise.\end{cases}$$

$$v(i)=\begin{cases}1 & if \quad x_i \in S, \quad 1\le i\le m,\\ 1 & if \quad i=p+1,\\ -(k+1) & if \quad i=p+2,\\ 0 & otherwise.\end{cases}$$

At the initial phase of the game, the first player can only apply $u(i)$, since there is no other rule which can be used (initially disks are all stacked on the first peg). The second player immediately replies with $v(i)$. Once the first player has move disk(s) with $u(i)$, there has to be at least one disk on $(p+1)$ st peg.

For V_2 (simulation pegs) the set consists of six vectors. The rules are interdependent of a set of first 3 consecutive pegs in m, i.e. (l_1,l_2,l_3). This is done in order to maintain the overall structure of the rules and the number of pegs as defined previously (three parts of pegs). If $l_3\neq m$, then there are six vectors that are to be simulated ($v_{r_i1},v_{r_i2},v_{r_i3},v_{r_i4},v_{r_i5},v_{r_i6}$). If $l_3=m$, then there are only five vectors $v_{r_i1},v_{r_i3},v_{r_i4},v_{r_i5},v_{r_i6}$ that are used for simulation. For both $l_3\neq m$ and $l_3=m$ separate v_{r_i6} is defined. The rules are as follows:

$$v_{r_i1}(i)=\begin{cases}+1 & if \quad i=2m+4(l-1)+1,\\ -1 & if \quad i=p+1,\\ 0 & otherwise.\end{cases}$$

$$v_{r_i2}(i)=\begin{cases}+1 & if \quad i=m, \quad i=p+3,\\ -1 & if \quad i=l_3, \quad i=2m+4(l-1)+1,\\ 0 & otherwise.\end{cases}$$

$$v_{r_i 3}(i) = \begin{cases} +1 & if \quad i = 2m + 4(l-1) + 2, \\ -1 & if \quad i = 2m + 4(l-1) + 1, \\ 0 & otherwise. \end{cases}$$

$$v_{r_i 4}(i) = \begin{cases} +1 & if \quad i = m + l_1, \quad i = m + l_2, \quad i = 2m + 4(l-1) + 3, \\ -1 & if \quad i = l_1, \quad i = l_2, \quad i = 2m + 4(l-1) + 2, \\ 0 & otherwise. \end{cases}$$

$$v_{r_i 5}(i) = \begin{cases} +1 & if \quad i = 2m + 4(l-1) + 4, \\ -1 & if \quad i = 2m + 4(l-1) + 3, \\ 0 & otherwise. \end{cases}$$

If $l_3 \neq m$, then

$$v_{r_i 6}(i) = \begin{cases} +1 & if \quad i = l_2, \quad i = l_3, \quad i = p+1, \\ -1 & if \quad i = m + l_1, \quad i = m + l_2, \quad i = 2m + 4(l-1) + 4, \\ 0 & otherwise. \end{cases}$$

If $l_3 = m$, then

$$v_{r_i 6}(i) = \begin{cases} +1 & if \quad i = l_2, \quad i = l_3, \quad i = p+3, \\ -1 & if \quad i = m + l_1, \quad i = m + l_2, \quad i = 2m + 4(l-1) + 4, \\ 0 & otherwise. \end{cases}$$

Once the middle part of the pegs are simulated with moves, the final step would be simply to collect k disks onto the $(p+3)$rd peg. The basic idea is to enable the player who first moves a peg to the $(p+3)$rd peg to move all the k disks to the $(p+3)$rd peg.

For V_3(collection of pegs) the set consists of $m+1$ vectors, w and $v_1, v_2, ..., v_m$. The move w is written in such a way that whosoever executes it first, can now collect the desired k number of disks onto the last peg.

$$w(i) = \begin{cases} +1 & if \quad i = p+2, \\ -1 & if \quad i = p+3, \\ 0 & otherwise. \end{cases}$$

For each j $(1 \le j \le m)$,

$$v_j(i) = \begin{cases} +2 & if \quad i = p+3, \\ -1 & if \quad i = p+2, \quad i = j, \\ 0 & otherwise. \end{cases}$$

Once w is executed, it is easy to see that the only other rule(s) applicable are the $v_j(i)$. Since the first player will have the first change to execute w, it can therefore force a win over the second player. Moreover, since the k-pebble game can be reduced to the k-peg game, it will at least take $\Omega(n^{(k-2)/4-\epsilon})$ to finish all the moves.

Open Problems.

i. As mentioned earlier, the assumption that there are fixed number of disks is important to derive the result. It is easy to see that with the number of disks not fixed, the k-peg game should take exponential amount of time since with the increase of a single peg the number of choices for the players grows much faster. It would be interesting to see an exact mathematical derivation of the proof.

ii. In [Adachi et al 1984], the authors suspect that the k-peg game can be solved within a deterministic polynomial time bound. Can one verify this and give an exact algorithm to solve the k-peg game?

References

T. Kasai, A. Adachi and S. Iwata. Classes of pebble games and complete problems. SIAM Journal on Computing, vol. 8, no. 4, pages 576-586, 1979.

A. Adachi, S. Iwata and T. Kasai. Some combinatorial game problems require Ω(nk) time. Journal of Association of Computing Machine, vol. 31, no. 2, pages 361-376, 1984.