Jan Długosz University in Częstochowa
Scientific Issues, Mathematics XVI, Częstochowa 2011

FINDING THE SHORTEST PATH BETWEEN VERTICES IN A GRAPH HANOI

Sergey Novikov

Institute of Computer Science,
Siedlce University of Natural Sciences and Humanities
ul. 3 Maja 54, 08-110 Siedlce, Poland
e-mail: novikov@uph.edu.pl

Abstract. Three algorithms for finding the shortest path between two vertices with arbitrary labels of any fractal graph Hanoi $S(k, n)$ and the exact estimation of the minimal distance between these vertices for the case $k \geq 3$ and $n < k$ are proposed.

1. Introduction

The problem "Multi-peg Tower of Hanoi" has many variations and generalizations in different directions [1].

In this paper we consider the following generalization: initial and terminal configurations are any arbitrary (no regular) legal distributions of n disks among k pegs; our goal is to get from a given arbitrary initial state to one of the different states by the shortest path between legal configurations [2]. The shortest sequences of moves leading from a given initial configuration to a given terminal configuration is equal to the shortest path between two vertices V_i , V_j of the special graph (graph Hanoi) with special labels [3].

This problem for $k = 3$ was investigated by Andreas Hinz in [3]. If the initial configuration is any arbitrary (no regular) node and terminal configuration is perfect (regular) configuration, the shortest sequences of moves leading from a given initial configuration to a given configuration is equal, on the average, to $(2/3) * (2^n - 1)$.

We consider these problem for the case $k \geq 3$.

The main result of the proposed article is an algorithm for finding the shortest path (A, B) between arbitrary vertices with labels A and B of a fractal graph Hanoi $H(k, n)$ for the case $n < k$ and $k \geq 3$.

Some components in the labels A and B may coincide. This means that the disks in the appropriate configuration A are installed in the same manner as required in a given configuration B. Such components are painted (colored). As building a path (A, B), a growing number of components will be colored.

Bearing in mind the need to minimize the number of moves, note that when $k > n$, each disk can be moved to its desired location at most in two moves. Therefore, each component of the label A in the path (A, B) can be changed at most two times.

2. The exact estimation of the minimal distance between vertices of $S(k, n)$ for $k \geq 3$ and $n < k$

Theorem. For a graph $H(k, n)$ with $k \geq 3$ and $n < k$, the minimal distance between the vertices with arbitrary labels A and B in the path (A, B) is equal to

$$d(A, B) = 2(n - n_0) - n_1, \quad (1)$$

where n is the number of components in A, n_0 the number of invariable components, n_1 the number of components which are variable one-off.

Proof. The proof is carried out by induction with respect to n.

1. For $n = 1$ we have one variable component and two cases:

1a) Our component is variable one-off.

It is clear that $d(A, B) = 1$. On the other hand, $n_1 = 1$, $n_0 = 0$ and from our formula we have $d(A, B) = 2 * 1 - 1 = 1$.

1b) Our component does not change.

Then $d(A, B) = 0$. On the other hand, $n_1 = 0$, $n_0 = 1$ and from our formula we have $d(A, B) = 2 * (1 - 1) - 0 = 0$.

2. Our induction hypothesis for the parameter $n = m$ is

$$d(A^m, B^m) = 2(m - m_0^m) - m_1^m$$

3. For $n = m + 1$ we have three cases:

3a) The new component does not change.

Then $m_0^{m+1} = m_0^m + 1$, $m_1^{m+1} = m_1^m$, and $d(A^{m+1}, B^{m+1}) = d(A^m, B^m) + 0$ $= d(A^m, B^m) = 2(m - m_0^m) - m_1^m = 2(m - m_0^m + 1 - 1) - m_1^m =$ $= 2((m + 1) - (m_0^m + 1)) - m_1^m = 2((m + 1) - m_0^{m+1}) - m_1^{m+1}$.

3b) The new component is variable one-off.

Then $m_1^{m+1} = m_1^m + 1$, $m_0^{m+1} = m_0^m$, and $d(A^{m+1}, B^{m+1}) = d(A^m, B^m) + 1$ $= 2(m - m_0^m) - m_1^m + 1 = 2(m - m_0^m) - m_1^m + 1 + 2 - 2 = 2((m + 1) - m_0^m)$ $-(m_1^m + 1) = 2((m + 1) - m_0^{m+1}) - m_1^{m+1}$.

3c) The new component is variable twice.

Then $m_1^{m+1} = m_1^m$, $m_0^{m+1} = m_0^m$, and $d(A^{m+1}, B^{m+1}) = d(A^m, B^m) + 2 =$ $= 2(m - m_0^m) - m_1^m + 2 = 2((m+1) - m_0^m) - m_1^m = 2((m+1) - m_0^{m+1}) - m_1^{m+1}$.

Our statement is true for all possible cases.

Remark. This result allows us to prove the minimal feature of the path (A, B) built by our algorithm.

3. Sets of invariable components of the label A being variable one-off or invariable

The first step in the process of finding the shortest path (A, B) is to build the set of invariable components in the label A.

Algorithm 1.

Source data: $A = a_1 a_2 \dots a_n$, $B = b_1 b_2 \dots b_n$, where $[a_i], [b_i] \in \{1, \dots, k\}$, $k > n$.

Output data: Z – the set of invariable components in the label A.

1. The set of colored components $Z = \emptyset$, the set of analyzed components $M = \emptyset$. $i := 1$.

2. If $i = n + 1$, then go to step 5. Otherwise, we go (from left to right) and compare a_i with b_j (only no painted).

If $a_i \in M$ or $a_i \in Z$, then $i := i + 1$ and go to step 2.

If $[a_i] \neq [b_i]$, then $M := \{a_i\} \cup M$, $i := i + 1$ and go to step 2.

If $[a_i] = [b_i]$, then $j = i$, and go to step 3.

3. We compare b_i with b_j (only no painted), where $j < i$. If $[b_i] = [b_j]$, then $i := i + 1$ and go to step 2.

If $[b_j] \neq [b_{j-1}]$ and $[b_j] \neq [b_{j-2}]$ and $\dots$ and $[b_j] \neq [b_1]$, we compare a_i with a_j (only no painted), where $j < i$.

If $[a_j] \neq [a_{j-1}]$ and $[a_j] \neq [a_{j-2}]$ and $\dots$ and $[a_j] \neq [a_1]$, then $Z := \{a_j\} \cup Z$, $M := \{a_j\} \cup M$ and go to step 4.

If $[b_j] = [b_{j-1}]$ or $[b_j] = [b_{j-2}]$ or $\dots$ or $[b_j] = [b_1]$, or $[a_j] = [a_{j-1}]$ or $[a_j] = [a_{j-2}]$ or $\dots$ or $[a_j] = [a_1]$, then $M := \{a_j\} \cup M$, $i := i + 1$ and go to step 2.

4. We have s:=j+1. If $s > n$, then $i := i + 1$ and go to step 2.
Otherwise, we compare $[a_j]$ with $[a_s]$, where $s = j + 1, j + 2, \dots, n$.
If $[a_j] \neq [a_s]$, go to step 4.
If s is such that $[a_j] = [a_s]$, where $j < s \leq n$, we compare $[a_s]$ with $[b_s]$.
If $[a_s] \neq [b_s]$, then $M := \{a_s\} \cup M$, $i := i + 1$ and go to step 2.
If $[a_s] = [b_s]$, then $j = s$ and go to step 3.

5. Stop. All the elements of the set Z are painted components.

Each component of the label A in the path (A, B) can be changed at most two times. Elements of Z are invariable components in the label A.

The following algorithm (A2) creates the set $V1$ of components of A which are variable one-off. The algorithm A2 is based on verification of properties to be satisfied by elements from $V1$.

Properties of components to create the set $V1$:

1) $\exists a_i([a_i] = p) \wedge \forall j(j < i)([a_j] \neq p) \Rightarrow (a_i \in V1)$

2) $\exists b_i([b_i] = q) \wedge \forall j(j < i)([b_j] \neq q) \Rightarrow (a_i \in V1)$

3) $\exists a_i([a_i] \neq [b_i]) \wedge (b_{i-1} \in Z) \Rightarrow (a_i \in V1)$

4) $\exists a_i([a_i] = p = [b_i]) \wedge \exists j(j < i)([a_j] = p)(b_{j-1} \in V1) \Rightarrow (a_i \in V1)$

5) $\exists a_i([a_i] = p) \wedge \exists j(j < i)([a_j] = p) \wedge ([b_i] = q \neq p)([b_j] = q) \Rightarrow (a_i \notin V1)$

6) $\exists a_i([a_i] = p) \wedge \exists j(j < i)([a_j] = p) \wedge ([b_i] = p)([b_j] \neq p) \Rightarrow (a_i \notin V1)$

7) $\exists a_i([a_i] = p = [b_i]) \wedge \exists j(j < i)([a_j] \neq p)(b_j] = p) \Rightarrow (a_i \notin V1)$

In these propetries we assume the values $p, q \in \{1, 2, \ldots, k\}$.

We use the algorithm A2 for the proof of the minimal feature of the path which ia created by the following algorithm A3.

4. An algorithm for finding of the shortest path between the labels A and B

The next step in our process of finding the shortest path between vertices with labels A and B is to construct a sequence of vertices labeled $A_0 = A, A_1, A_2, \ldots, A_m = B$. The labels A_i and A_{i+1} have only one different element from n components.

To store intermediate data in the following algorithm A3, we use a stack structure with the LIFO maintenance order.

When we transfer the disks, we use free pegs. In the relevant operations on labels the so-called free number should be used. We should have enough free numbers, temporarily used in the construction of paths.

Algorithm 3.

Source data: The vertices labeled $A = a_1 a_2 \ldots a_n$ and $B = b_1 b_2 \ldots b_n$, where $[a_i], [b_i] \in \{1, \ldots, k\}, k > n$; a set Z; a set of numbers $W \neq \emptyset$; a stack $S = \emptyset$.

Output data: The shortest path $(A, B) = A_0 = A, A_1, A_2, \ldots, A_m = B$.

1. $A = a_1 a_2 \ldots a_n$, $B = b_1 b_2 \ldots b_n$, Z, $W \neq \emptyset$, a stack $S = \emptyset$, $A = A_0$, $m = 0$, $i := n$.

2. If $i = 0$, then go to step 5. Otherwise, we go from right to left in A_m. If $a_i \in Z$, then $i := i - 1$ and go to step 2.

Otherwise, we compare $[a_i]$ and $[b_i]$.

The following three situations are possible.

2.1. $[a_i] \neq [b_i] = l$.

If $\forall b_i(b_i \notin Z)\exists j(j < i)([b_i] \neq l)$ and $\forall a_i(a_i \notin Z)\exists j(j < i)([a_j] \neq [a_i])$ and $\forall a_i(a_i \neq l)$, then we change the value of the component a_i and have $[a_i] = l$.

We have a new label A_{m+1}, a new set W and $Z := Z \cup \{ai\}$. We change $i := i - 1$ and go to step 2.

2.2. $[a_i] = q, [b_i] = l, q \neq l$.

To the left of the component $a_i = a_{i1}$, we have the components $a_{i2}, a_{i3}, \ldots, a_{is}$ such that $[a_{i2}] = [a_{i3}] = \ldots = [a_{is}] = q$, and to the left of the component $b_i = b_{i1}$, we have the components $b_{i2}, b_{i3}, \ldots, b_{it}$ such that $[b_{i2}] = [b_{i3}] = \ldots = [b_{it}] = l$.

In this case, we change the content of components $a_i = a_{i1}, a_{i2}, \ldots, a_{i(s-1)}$ twice. The exception is made for the component a_{is}, which content we shall change one-off later. First, we change the content of each component $a_i = a_{i1}, a_{i2}, \ldots, a_{i(s-1)}$ on a free number $w \in W$. Thereafter, we write it down $a_i = a_{i1}, a_{i2}, \ldots, a_{i(s-1)}$ to a stack S and create new labels. Later we change the set $W := W'$ and go to step 4.

2.3. $[a_i] = [b_i] = l$.

Then $[a_i] = l* \in W$. Thereafter, we create a new label A_{m+1}, a new W and modernize a stack S. Then $j = i$ and go to step 3.

3. $j := j - 1$. If $j = 0$, then $i := i - 1$ and go to step 4.

Otherwise, we analyze a_j.

If $[a_j] = [a_i] = l$, we compare $[a_j]$ and $[b_j]$.

If $[a_j] = [b_j] = l$, then $[a_j] = l* \in W$, we create a new label A_{m+1} and new sets W, S.

If $[a_j] \neq [a_i] = l$, we compare $[b_j]$ and $[a_i]$. If $[b_j] \neq [a_i]$, go to step 3.

If $[bj] = [ai]$, then $S := \{a_j\} \cup S$ and go to step 3.

4. If $S = \emptyset$, then $i := i - 1$ and go to step 2.

We analyze the next element from the stack S.

Our actions are similar to those in step 2.

We have new labels and new elements for S, Z, W.

Thereafter, $S := S - \{a*\}$ and go to step 4.

5. If $\mid Z \mid= n$, go to step 6. Otherwise, $i := i - 1$ and go to step 2.

6. Stop. We have the shortest path (A, B) :

$$A = A_0 \to A_1 \to A_2 \to \ldots \to A_m = B.$$

Example. Let us build the shortest path between vertices with labels A and B in the graph $S(10,9)$ with 10^9 vertices.

1. $A = 10\,1\,3\,1\,2\,9\,9\,4\,3$, $B = 8\,1\,2\,1\,2\,4\,4\,4\,3$, $S = \emptyset$, $W = \{5,6,7,8\}$, $Z = \{a_2, a_4\}$.

2. $i = 9$. We analyze a_9. We have $[a_9] = [b_9] = 3$ and $[a_3] = 3$. We should change this component by a free number $[a_9] = 5$. Then $A_1 = 10\,1\,3\,1\,2\,9\,9\,4\,5$, $W = \{6,7,8\}$, $S = (a_9)$, $Z = \{a_2, a_4\}$.

3. We analyze a_3, because $[a_3] = [a_9] = 3$. We cannot change $[a_3] = 2$, because $[a_5] = 2$, $S = (a_3, a_9)$.

4. We analyze the element a_3 from the stack S. We cannot change $[a_3] = 2$, because $[a_5] = 2$. We cannot change $[a_5] = 2$, because $[b_3] = 2$. We have $A_2 = 10\,1\,3\,1\,6\,9\,9\,4\,5$, $W = \{7,8\}$, $S = (a_5, a_9)$, $Z = \{a_2, a_4\}$.

We analyze the element a_3 once again. We can change $[a_3] = 2$ and have $A_3 = 10\,1\,2\,1\,6\,9\,9\,4\,5$, $W = \{3,7,8\}$, $S = (a_5, a_9)$, $Z = \{a_2, a_3, a_4\}$.

We analyze the element a_5 from the stack S and have $A_4 = 10\,1\,2\,1\,2\,9\,9\,4\,5$, $W = \{3,6,7,8\}$, $S = (a_9)$, $Z = \{a_2, a_3, a_4, a_5\}$.

We analyze the element a_9 from the stack S and have $A_5 = 10\,1\,2\,1\,2\,9\,9\,4\,3$, $W = \{5,6,7,8\}$, $S = ()$, $Z = \{a_2, a_3, a_4, a_5, a_9\}$.

2. We analyze a_8. As $[a_8] = [b_8] = 4$, we should change this component on a free number $[a_8] = 5$. Then $A_6 = 10\,1\,2\,1\,2\,9\,9\,5\,3$, $W = \{6,7,8\}$, $S = (a_8)$,
$Z = \{a_2, a_3, a_4, a_5, a_9\}$.

3. We have $[b_6] = 4$ and $S = (a_6, a_7, a_8)$.

4. We cannot change $[a_6] = 4$, because $[a_7] = 9$. We should change $[a_7] = 6$. Then $S = (a_7, a_8)$ and $A_7 = 10\,1\,2\,1\,2\,9\,6\,5\,3$, $W = \{7,8\}$, $S = (a_7, a_8)$, $Z = \{a_2, a_3, a_4, a_5, a_9\}$. Thereafter, we can change $[a_6] = 4$. Then $A_8 = 10\,1\,2\,1\,2\,4\,6\,5\,3$, $A_9 = 10\,1\,2\,1\,2\,4\,4\,5\,3$, $A_{10} = 10\,1\,2\,1\,2\,4\,4\,4\,3$, $A_{11} = 8\,1\,2\,1\,2\,4\,4\,4\,3$, $W = \{5,6,7,9,10\}$, $S = \emptyset$, $Z = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$.

5. $\mid Z \mid = 9$. $d(A,B) = 11$.

From our estimation (1) with $n_0 = 2$ and $n_1 = 3$, the minimal distance between vertices with labels A and B for our example should be equal to $d(A,B) = 2(9-2) - 3 = 11$. With the help of the algorithm A3 we have built the path with the length 11. So the constructed path is the shortest one.

References

[1] P.K. Stockmeyer. The Tower of Hanoi: A Bibliography, September 2005. http://www.cs.win.edu/ pkstoc/biblio2.pdf

[2] S. Novikov. About shortest paths between nodes of the graph Hanoi. *Scientific Issues, Catholic University in Ružomberok, Mathematica*, **II**, 51–58, 2009.

[3] A. Hinz. Shortest paths between regular states of the Tower of Hanoi. *Information Sciences*, **63**, 173–181, 1992.