SIAM J. COMPUT.
Vol. 33, No. 2, pp. 379–392

THE MINIMUM ALL-ONES PROBLEM FOR TREES*

WILLIAM Y. C. CHEN†, XUELIANG LI†, CHAO WANG†, AND XIAOYAN ZHANG†

Abstract. The minimum all-ones problem was shown to be NP-complete for general graphs. Therefore, it becomes an interesting problem to identify special classes of graphs for which one can find polynomial time algorithms. In this paper we consider this problem for trees. First, for any solution to the all-ones problem for a tree, we give a characterization of the elements in the solution by introducing the concept of the quasi all-ones problem. Then we give the enumeration for the number of solutions in a tree. By using the minimum odd (even) sum problem as subprocess, we obtain a linear time algorithm for the minimum all-ones problem for trees. We also get a linear time algorithm for finding solutions to the all-ones problem in a unicyclic graph.

Key words. lamp lighting problem, all-ones problem, graph algorithm, time complexity

AMS subject classifications. 05C85, 05C70, 90C27, 68Q25, 68R10

DOI. 10.1137/S0097539703421620

1. Introduction. The term *all-ones problem* was introduced by Sutner [9]. The problem has applications in linear cellular automata (see [10] and the references therein) and is cited as follows: Suppose each square of an $n \times n$ chessboard is equipped with an indicator light and a button. If the button of a square is pressed, the light of that square will change from off to on, and vice versa; the same happens to the lights of all the edge-adjacent squares. Initially all lights are off. Now, consider the following questions: is it possible to press a sequence of buttons in such a way that in the end all lights are on? This is referred to as the *all-ones problem*. If there is such a solution, how can we find it? And finally, how can we find a solution that presses as few buttons as possible? This is referred to as the *minimum all-ones problem*. All the above questions can be asked for arbitrary graphs. Here and in what follows, we consider connected simple undirected graphs only. One can deal with disconnected graphs component by component. For all terminology and notation on graphs, we refer to [6]. An equivalent version of the all-ones problem was proposed by Peled in [7], where it was called the *lamp lighting problem*. The rule of the all-ones problem is called the σ^+ rule on graphs, which means that a button lights not only its neighbors but also its own light. If a button lights only its neighbors but not its own light, this rule on graphs is called the σ rule.

The all-ones problem has been extensively studied recently; see Sutner [11, 12], Barua and Ramakrishnan [1], and Dodis and Winkler [2]. Using linear algebra, Sutner [10] proved that it is always possible to light every lamp in any graph by the σ^+ rule. Lossers [5] gave another beautiful proof also by using linear algebra. A graph-theoretic proof was given by Erikisson, Eriksson, and Sjöstrand [3]. So, the existence of solutions of the all-ones problem for general graphs was solved already. Galvin [4] gave a graph-theoretic algorithm of linear time to find solutions for trees. In [8], Sutner proved that the minimum all-ones problem is NP-complete in general. Therefore, it becomes an interesting question to identify special classes of graphs for

*Received by the editors January 25, 2003; accepted for publication (in revised form) November 3, 2003; published electronically February 18, 2004. This work was done under the auspices of the "973" Project on Mathematical Mechanization and the NSFC.
http://www.siam.org/journals/sicomp/33-2/42162.html
†Center for Combinatorics and LPMC, Nankai University, Tianjin 300071, People's Republic of China (x.li@eyou.com).

which one can find polynomial time algorithms. It is the main result of this paper that there exists a linear time algorithm for the minimum all-ones problem for trees.

In graph-theoretic terminology, a solution to the all-ones problem with σ^+-rule can be stated as follows: Given a graph $G = (V, E)$, where V and E denote the node-set and the edge-set of G, respectively, a subset X of V is a solution if and only if for every node v of G the number of nodes in X adjacent to or equal to v is odd. Such a subset X is called an *odd parity cover* in [10]. So, the all-ones problem can be formulated as follows: Given a graph $G = (V, E)$, does a subset X of V exist such that for all nodes $v \in V - X$, the number of nodes in X adjacent to v is odd, while for all nodes $v \in X$, the number of nodes in X adjacent to v is even? If there exists a solution, how can one find it with minimum cardinality?

This paper is organized as follows. In section 2, we give, for any solution to the all-ones problem for a tree, a characterization of the elements in the solution by introducing the concept of the quasi all-ones problem. This leads to an enumeration for the number of solutions in a tree. In section 3, we give a linear time algorithm to the minimum all-ones problem for trees. In the concluding section, section 4, we give a linear time algorithm for constructing solutions to the all-ones problem in a unicyclic graph. An open problem on the all-colors problem is also proposed, generalizing the concept of the all-ones problem.

2. Characterization and enumeration of solutions for trees. It is easy to see that if a given graph G can be partitioned into two disjoint subgraphs G_1 and G_2 such that G_1 is Eulerian and every node of G_2 is adjacent to an odd number of nodes of G_1, then by pressing all the buttons on the nodes of G_1 all the lights will be on, and vice versa. However, it is very difficult to find an Eulerian subgraph with such a property in a large graph G. Sutner [9] posed the question of whether there is a graph-theoretic method to find a solution for the all-ones problem for trees. Galvin [4] solved this question in the following way: Consider a rooted tree, drawn like a family tree, with the root at the top. The nodes will be divided into 3 classes: outcasts, oddballs, and rebels. The classification is defined inductively, from the bottom up, as follows:

- All of the childless nodes or leaves are rebels.
- A node, other than a leaf, is called a *rebel* if it has no oddball children and an even number of its children are rebels.
- A node is called an *oddball* if it has no oddball children and an odd number of its children are rebels.
- A node is called an *outcast* if at least one of its children is an oddball.

We sometimes simply call a node *r-type, b-type, or o-type* if it belongs to the rebel class, the oddball class, or the outcast class, respectively. For examples, see Figure 1.

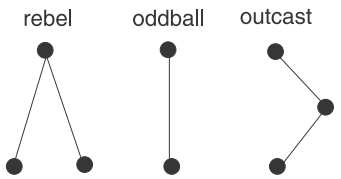

FIG. 1. *The roots of the rooted trees are r-type, b-type, and o-type.*

GALVIN ALGORITHM [4]. Membership in a solution C is defined inductively from the top down. An outcast is excluded from the membership in C. A rebel will be

a member of C if and only if its parent is not a member of C; in particular, if the parentless node (the root) is a rebel, then it will be a member of C. The oddballs will join C in whatever numbers are needed to make their parents' closed neighborhood contain an odd number of members. It is easy to check that C is a solution.

In this section, from the idea of the Galvin algorithm we determine whether or not the root of a tree is in a solution of the all-ones problem and the quasi all-ones problem, defined next. This will be used in enumerating the number of solutions for the all-ones problem. This will also be used in solving the minimum all-ones problem for trees and in constructing solutions to the all-ones problem for unicyclic graphs.

DEFINITION 2.1. *For a rooted tree, the quasi all-ones problem is to find a subset C of nodes such that for every node v except for the root of the tree, the number of nodes in C adjacent to v or equal to v is odd, while for the root, the number is even. C is called a solution to the quasi all-ones problem.*

THEOREM 2.1. *For a rooted tree,*

(1) *if the root is a rebel, then*
   (1.1) *the all-ones problem has a solution if and only if the root belongs to the solution;*
   (1.2) *the quasi all-ones problem has a solution if and only if the root does not belong to the solution;*
(2) *if the root is an oddball, then*
   (2.1) *the all-ones problem has a solution no matter whether the root belongs to the solution;*
   (2.2) *the quasi all-ones problem does not have any solution no matter whether the root belongs to the solution;*
(3) *if the root is an outcast, then*
   (3.1) *the all-ones problem has a solution if and only if the root does not belong to the solution;*
   (3.2) *the quasi all-ones problem has a solution if and only if the root does not belong to the solution.*

Proof. We prove this theorem by induction on the depth s of the rooted tree, which is defined as the maximal distance from root to leaves. In particular, a rooted tree with only one node is of depth 0. It is easy to see that if the root of a tree is a rebel, then the depth can be any nonnegative integer. For a tree with an oddball root, the depth will be at least 1, while for a tree with an outcast root, the depth will be at least 2.

If a rooted tree is of depth 0, then the root must be a rebel node. Any solution to the all-ones problem must contain the root and any solution to the quasi all-ones problem does not contain the root, which means that (1.1) and (1.2) hold when the depth is 0. It is also easy to check that (2.1) and (2.2) hold when the (least possible) depth is 1, and (3.1) and (3.2) hold when the (least possible) depth is 2.

Next, suppose that for any rooted tree whose depth is less than s, all the statements of the theorem are true. Then, for a rooted tree whose depth is s, we distinguish three cases.

Case 1. *The root is a rebel.* Assume that the children of the root are $t_1^{(r)}, \ldots, t_{2k}^{(r)}$, $t_{2k+1}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 0$, the subtree rooted at $t_i^{(r)}$ $(1 \leq i \leq 2k)$, denoted by $T_i^{(r)}$, is a subtree with a rebel root and with depth less than s, and the subtree rooted at $t_j^{(o)}$ $(2k+1 \leq j \leq m)$, denoted by $T_j^{(o)}$, is a subtree with an outcast root and with depth less than s. Then, from the induction hypothesis, for an outcast-rooted tree the all-ones problem or quasi all-ones problem has a solution if and only if the outcast

root does not belong to the solution. So we can ignore the case for outcast-rooted subtree.

Suppose that the all-ones problem for the rebel-rooted tree has a solution, denoted by C. If the root $r \notin C$, then $C(T_i^{(r)})$ $(1 \le i \le 2k)$ (which is the restriction of C on the subtree $T_i^{(r)}$) is the solution to the all-ones problem for the rebel-rooted subtree $T_i^{(r)}$ with depth less than s. From the induction hypothesis, $t_i^{(r)} \in C(T_i^{(r)})$, and so $t_i^{(r)} \in C$. However, we know that the number of rebel children of the rebel root is even. So, the root cannot be covered odd times by C, and hence C is not an odd parity cover, a contradiction. Thus, if the all-ones problem for a rebel-rooted tree with depth s has a solution, then the root must belong to the solution.

Conversely, consider each rebel-rooted subtree $T_i^{(r)}$ $(1 \le i \le 2k)$ and each outcast-rooted subtree $T_j^{(o)}$ $(2k+1 \le j \le m)$, whose root $t_i^{(r)}, t_j^{(o)}$ is a rebel child and an outcast child of the root r, respectively. Then the quasi all-ones problem for each of them has a solution, denoted by $C(T_i^{(r)}), C(T_j^{(o)})$, respectively. Note that

$$t_i^{(r)} \notin C(T_i^{(r)}) \ (1 \le i \le 2k), \qquad t_j^{(o)} \notin C(T_j^{(o)}) \ (2k+1 \le j \le m).$$

It is easy to check that $C = \{r\} \bigcup (\bigcup_{i=1}^{2k} C(T_i^{(r)})) \bigcup (\bigcup_{j=2k+1}^{m} C(T_j^{(o)}))$ is a solution to the all-ones problem for the original rebel-rooted tree with depth s. So (1.1) holds when the depth is s.

Similarly we can prove (1.2).

Case 2. *The root is an oddball.* Assume that the children of the root are $t_1^{(r)}, \ldots, t_{2k-1}^{(r)}, t_{2k}^{(o)}, \ldots, t_m^{(o)}$, where $k \ge 1$, the subtree rooted at $t_i^{(r)}$ $(1 \le i \le 2k-1)$, denoted by $T_i^{(r)}$, is a rebel-rooted subtree with depth less than s, and the subtree rooted at $t_j^{(o)}$ $(2k \le j \le m)$, denoted by $T_j^{(o)}$, is an outcast-rooted subtree with depth less than s. Similar to the above discussion, we can ignore the case for outcast-rooted subtree.

Suppose that the quasi all-ones problem for the oddball-rooted tree has a solution, denoted by C. If the root $r \in C$ (or $r \notin C$), then $C(T_i^{(r)})$ $(1 \le i \le 2k-1)$ is a solution to the quasi all-ones problem (or the all-ones problem) for the rebel-rooted subtree $T_i^{(r)}$ with depth less than s. From the induction hypothesis, we have that $t_i^{(r)} \notin C(T_i^{(r)})$ (or $t_i^{(r)} \in C(T_i^{(r)})$), and so $t_i^{(r)} \notin C$ (or $t_i^{(r)} \in C$). However, we know that the number of rebel children of the oddball root is odd. So, the root is covered odd times by C, which means that C is a solution to the all-ones problem for the original oddball-rooted tree, a contradiction. Thus (2.2) holds when the depth is s.

Next, we prove (2.1). Consider each rebel-rooted subtree $T_i^{(r)}$ $(1 \le i \le 2k-1)$ and each outcast-rooted subtree $T_j^{(o)}$ $(2k \le j \le m)$, whose root $t_i^{(r)}, t_j^{(o)}$ is a rebel child and an outcast child of the root r, respectively. We discuss the following two cases.

First, the quasi all-ones problem for each of them has a solution, denoted by $C(T_i^{(r)}), C(T_j^{(o)})$, respectively. Note that

$$t_i^{(r)} \notin C(T_i^{(r)}) \ (1 \le i \le 2k-1), \qquad t_j^{(o)} \notin C(T_j^{(o)}) \ (2k \le j \le m).$$

It is easy to check that $C = \{r\} \bigcup (\bigcup_{i=1}^{2k-1} C(T_i^{(r)})) \bigcup (\bigcup_{j=2k}^{m} C(T_j^{(o)}))$ is a solution to the all-ones problem for the original oddball-rooted tree with depth s. Here the root belongs to the solution.

Second, the all-ones problem for each rebel-rooted subtree $T_i^{(r)}$ $(1 \le i \le 2k-1)$ has a solution, denoted by $C(T_i^{(r)})$. Then the all-ones problem for each outcast-rooted subtree $T_j^{(o)}$ $(2k \le j \le m)$ has a solution, denoted by $C(T_j^{(o)})$. Note that

$$t_i^{(r)} \in C(T_i^{(r)}) \ (1 \le i \le 2k-1), \qquad t_j^{(o)} \notin C(T_j^{(o)}) \ (2k \le j \le m).$$

It is easy to check that $C = (\bigcup_{i=1}^{2k-1} C(T_i^{(r)})) \bigcup (\bigcup_{j=2k}^{m} C(T_j^{(o)}))$ is a solution to the all-ones problem for the original oddball-rooted tree with depth s. Here the root does not belong to the solution. Thus (2.1) holds when the depth is s.

Case 3. *The root is an outcast.* Assume that the children of the root are $t_1^{(b)}, \ldots, t_k^{(b)}, t_{k+1}^{(r)}, \ldots, t_l^{(r)}, t_{l+1}^{(o)}, \ldots, t_m^{(o)}$, where $k \ge 1$, the subtree rooted at $t_i^{(b)}$ $(1 \le i \le k)$, denoted by $T_i^{(b)}$, is an oddball-rooted subtree with depth less than s, the subtree rooted at $t_i^{(r)}$ $(k+1 \le i \le l)$, denoted by $T_i^{(r)}$, is a rebel-rooted subtree with depth less than s, and the subtree rooted at $t_i^{(o)}$ $(l+1 \le i \le m)$, denoted by $T_i^{(o)}$, is an outcast-rooted subtree with depth less than s. A similar argument can cover the proof of this case. □

From the theorem, we have the following remarks.

Remark 2.1.

1. For any solution to the all-ones problem for a rooted tree,
   (a) if the root is a rebel, it must belong to the solution;
   (b) if the root is an outcast, it cannot belong to the solution;
   (c) if the root is an oddball, both cases are possible; i.e., it may or may not belong to the solution.
2. If there exists a solution to the quasi all-ones problem for a rooted tree, then the root cannot be an oddball and
   (a) if the root is a rebel, then it cannot belong to the solution;
   (b) if the root is an outcast, then it cannot belong to the solution.

Remark 2.2. If there exists a solution to the all-ones problem or the quasi all-ones problem for a rooted tree, then

1. if the root is a rebel or an oddball, both cases are possible; i.e., it may or may not belong to the solution;
2. if the root is an outcast, it cannot belong to the solution.

From the above clear analysis, we can get the following enumeration result.

THEOREM 2.2. *If a rooted tree has p oddball nodes and q outcast nodes, then the number of solutions to the all-ones problem for the tree is 2^{p-q}.*

Proof. From Theorem 2.1 and Remarks 2.1 and 2.2, we can deduce the following facts: First, for a rebel node v, if its parent node is not contained in a solution C, then v is contained in C, whereas if its parent node is contained in C, then v is not contained in C. Second, the outcast nodes cannot be contained in any solution.

If the root of the tree is a rebel or an outcast node, then every outcast node needs one of its oddball children to match its rebel children so that the outcast node will be lighted without itself in the solution. So, at each outcast node, its oddball children have degrees of freedom equal to the number of oddball children minus 1. Therefore, the number of solutions to the all-ones problem for a tree is exactly 2^{p-q}. If the root of the tree is an oddball, since the root can have two choices, i.e., in a solution, or not, we have that the number of solutions is $2(2^{p-1-q}) = 2^{p-q}$. The proof is complete. □

From the results so far, we can say that the all-ones problem for trees has a satisfactory solution. It is natural to ask about the minimum all-ones problem for

trees. The minimum all-ones problem is NP-complete for general graphs [8]. However, it can be solved easily for some special classes of graphs; for example, a path with n nodes has an optimal solution with $\lceil \frac{n}{3} \rceil$ nodes, and a cycle with n nodes has an optimal solution with $\frac{n}{3}$ nodes if $n = 0 \mod 3$, and n nodes otherwise. For an arbitrary tree, no such succinct formula has been known for the number of nodes in an optimal solution. However, we can ask whether there is a polynomial time algorithm for trees. At first look, we cannot see if there is such an algorithm, because from our Theorem 2.2 we know that the number of solutions could be exponentially large. However, we do obtain a polynomial time algorithm for trees by using the characterization in Theorem 2.1 and Remarks 2.1 and 2.2. Actually, what we get is a linear time algorithm.

3. The minimum all-ones problem for trees. In order to give our algorithm, we need to introduce a new problem, called the *minimum odd (even) sum problem*, which is described as the following linear program.

For the matrix $M_{2\times n} = (m_{ij})_{2\times n}$, $i \in \{0,1\}$, $j \in \{1,2,\cdots,n\}$, $m_{ij} \in Z^+$, the *minimum odd sum problem* is defined as

$$\min \sum_{j=1}^{n} m_{0j}x_{0j} + m_{1j}x_{1j},$$

$$\begin{cases} \sum_{j=1}^{n} x_{1j} = 1 \mod 2 \\ x_{0j} + x_{1j} = 1, \quad j = 1,2,\ldots,n \\ x_{ij} \in \{0,1\}, \quad i \in \{0,1\}. \end{cases}$$

Note that $m_{0j}x_{0j} + m_{1j}x_{1j}$ is equal to m_{1j} if $x_{1j} = 1$, or m_{0j} if $x_{1j} = 0$. So $m_{0j}x_{0j} + m_{1j}x_{1j}$ can be written as $m_{x_{1j}j}$. For convenience, we replace x_{1j} by y_j. Then it is easy to see that the above linear program is equivalent to the following one:

$$\min \sum_{j=1}^{n} m_{y_j j},$$

$$\begin{cases} \sum_{j=1}^{n} y_j = 1 \mod 2 \\ y_j \in \{0,1\} \end{cases}$$

Algorithm for the minimum odd sum problem.

Input. A matrix $M_{2\times n}$.

Step 1. Choose a minimum element from every column of $M_{2\times n}$ (if both elements in a column are the same, choose one of the them). Then sum up the first subscripts of all the chosen elements, denoted by S. If $S = 1 \mod 2$, go to Step 3; otherwise, go to Step 2;

Step 2. Calculate the absolute value of the difference of the two elements in every column. Choose one of the columns with the minimum absolute values. In this column, we choose the hitherto unselected element and forget about the chosen element, then go to Step 3;

Step 3. Sum up all the chosen elements, which gives the optimal value $\min \sum_{j=1}^{n} m_{y_j j}$.

Theorem 3.1. *The above algorithm correctly solves the minimum odd sum problem, and the time complexity is linear.*

Proof. The first statement of the theorem is proved as follows. Since the minimum odd sum problem asks for a unique element from every column, our greedy algorithm

picks up the minimum element from every column. If the sum of the first subscripts of all the chosen elements satisfies that $\sum_{j=1}^{n} y_j = 1 \bmod 2$, then the sum of all the chosen elements is exactly the optimal value $\min \sum_{j=1}^{n} m_{y_j j}$. If $\sum_{j=1}^{n} y_j \neq 1 \bmod 2$, we only need to adjust the elements slightly so that the sum of the first subscripts of all the chosen elements satisfies that $\sum_{j=1}^{n} y_j = 1 \bmod 2$. Because we adjust elements in the column where the minimum absolute value of the difference of the two elements is attained from Step 2, it is easy to see that the chosen elements after adjusting have the minimum sum among the feasible solutions to the odd sum problem; i.e., the chosen elements consist of an optimal solution.

For the second statement of the theorem, since every step uses linear time, the total time is $O(n)$. The proof is complete. □

From the above discussion, we can enumerate the number of optimal solutions to the minimum odd sum problem. Suppose that the absolute value of the difference of the two elements in the ith column is $|d_i|$.

If $\min\{|d_i| \mid i = 1, 2, \ldots, n\} = s > 0$, then

1. if the sum of the first subscripts of all the chosen elements satisfies that $\sum_{j=1}^{n} y_j = 1 \bmod 2$ in Step 1, it is straightforward to see that the problem has a unique optimal solution;

2. if $\sum_{j=1}^{n} y_j \neq 1 \bmod 2$, the only possible ways to adjust the chosen elements have to be done in the set of the columns $\{i \mid |d_i| = s,\ i = 1, 2, \ldots, n\}$, say, r such columns in all. Since we can do the adjustment in any one of the r such columns, the problem has r optimal solutions.

If $\min\{|d_i| \mid i = 1, 2, \ldots, n\} = s = 0$, and supposing $|\{i \mid |d_i| = 0,\ i = 1, 2, \ldots, n\}| = r$, then

1. if $\sum_{j=1}^{n} y_j = 1 \bmod 2$, the only possible ways to adjust the chosen elements have to be done in an even number of the columns $\{i \mid |d_i| = 0,\ i = 1, 2, \ldots, n\}$. So, the problem has T_0 optimal solutions, where $T_0 = \binom{r}{0} + \binom{r}{2} + \binom{r}{4} + \cdots = 2^{r-1}$;

2. if $\sum_{j=1}^{n} y_j \neq 1 \bmod 2$, the only possible ways to adjust the chosen elements have to be done in an odd number of the columns $\{i \mid |d_i| = 0,\ i = 1, 2, \ldots, n\}$. So, the problem has T_1 optimal solutions, where $T_1 = \binom{r}{1} + \binom{r}{3} + \binom{r}{5} + \cdots = 2^{r-1}$.

Replacing $\sum_{j=1}^{n} y_j = 1 \bmod 2$ in the minimum odd sum problem by $\sum_{j=1}^{n} y_j = 0 \bmod 2$, we then get a new problem, called the *even sum problem*. It can be solved in the same way as above. The details are omitted.

Now we give our linear time algorithm to the minimum all-ones problem for trees. The algorithm uses induction on the number of layers of a tree and the minimum odd or even sum algorithm as subprocess.

First of all, we give the definition of layers for a rooted tree as follows: The ith *layer* of the tree is composed of the nodes with distance i from the root for $i = 0, 1, 2, \ldots$. Suppose the tree has s layers. Then for any $i < s$, every node except the leaves in the ith layer can be considered the root of a small tree with depth 1, which is simply called a *small tree* in what follows. We divide the small trees into the following three types.

Type I. A type I small tree has an r-type root. For such a small tree, we can assume that the children of its root are $t_1^{(r)}, \ldots, t_{2k}^{(r)}, t_{2k+1}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 0$, the subtree rooted at $t_i^{(r)}$ $(1 \leq i \leq 2k)$ is denoted by $T_i^{(r)}$, and the subtree rooted at $t_j^{(o)}$ $(2k + 1 \leq j \leq m)$ is denoted by $T_j^{(o)}$. An example of type I small trees is shown in Figure 2(a).

Type II. A type II small tree has a b-type root. For such a small tree, we can

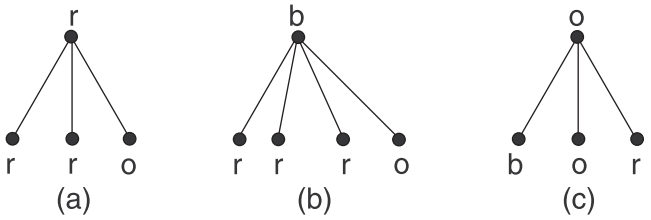

FIG. 2. *Examples of small trees of types* I, II, *and* III.

assume that the children of its root are $t_1^{(r)}, \ldots, t_{2k-1}^{(r)}, t_{2k}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 1$, the subtree rooted at $t_i^{(r)}$ $(1 \leq i \leq 2k-1)$ is denoted by $T_i^{(r)}$, and the subtree rooted at $t_j^{(o)}$ $(2k \leq j \leq m)$ is denoted by $T_j^{(o)}$. An example of type II small trees is shown in Figure 2(b).

Type III. A type III small tree has an o-type root. For such a small tree, we can assume that the children of its root are $t_1^{(b)}, \ldots, t_k^{(b)}, t_{k+1}^{(r)}, \ldots, t_l^{(r)}, t_{l+1}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 1$, the subtree rooted at $t_i^{(b)}$ $(1 \leq i \leq k)$ is denoted by $T_i^{(b)}$, the subtree rooted at $t_i^{(r)}$ $(k+1 \leq i \leq l)$ is denoted by $T_i^{(r)}$, and the subtree rooted at $t_i^{(o)}$ $(l+1 \leq i \leq m)$ is denoted by $T_i^{(o)}$. An example of type III small trees is shown in Figure 2(c).

By executing our algorithm layer by layer, from the bottom up, we are going to tag each node v in the present layer with a pair of sets of nodes. Then we can get an optimal solution in linear time. If v is a leaf, it is tagged by Step 0 of our algorithm; if not, its tagged pair of sets can be obtained from the following three cases.

Case 1. For every r-type leaf t_i^r of the small trees rooted at v, since the leaf is in the previous layer, we have an optimal solution $C_1(T_i^{(r)})$ to the all-ones problem for the subtree rooted at $t_i^{(r)}$ and an optimal solution $C_2(T_i^{(r)})$ to the quasi all-ones problem for the same subtree.

Case 2. For every b-type leaf t_i^b of the small trees rooted at v, we have an optimal solution $C_1(T_i^{(b)})$ to the all-ones problem for the subtree rooted at t_i^b such that the root of the subtree belongs to $C_1(T_i^{(b)})$ and an optimal solution $C_2(T_i^{(b)})$ to the all-ones problem for this subtree such that the root of the subtree does not belong to $C_2(T_i^{(b)})$.

Case 3. For every o-type leaf t_i^o of the small trees rooted at v, we have an optimal solution $C_1(T_i^{(o)})$ to the all-ones problem for the subtree rooted at t_i^o and an optimal solution $C_2(T_i^{(o)})$ to the quasi all-ones problem for the subtree.

Note that the above pair for every leaf of a tree is clearly determined at the beginning of our algorithm.

ALGORITHM FOR THE MINIMUM ALL-ONES PROBLEM FOR TREES.

Input. A rooted tree T with s layers, and a pair $\{C_1(v), C_2(v)\}$ of sets for each node v of the tree.

Step 0. Initially, for every leaf $t^{(r)}$ of T, set $\{\{t^{(r)}\}, \emptyset\}$, which means that $\{t^{(r)}\}$ is the optimal solution to the all-ones problem for the subtree with the single node $t^{(r)}$, and $\emptyset$ is the optimal solution to the quasi all-ones problem for the single node subtree.

Step 1. Inductively generate the pair for every node of T layer by layer, from the bottom up, till we arrive at the root of the tree. Suppose that the present layer is the ith layer. If $i \geq 0$, go to Step 2; otherwise, go to Step 4;

Step 2. We distinguish the following three cases to generate the pairs. For every small tree rooted in the ith layer, the algorithm works as follows:

1. The tree is type I. Denote its root by r^*. Suppose that the children of r^* are $t_1^{(r)}, \ldots, t_{2k}^{(r)}, t_{2k+1}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 0$. We already knew that $C_1(t_i^{(r)}) = C_1(T_i^{(r)})$ and $C_2(t_i^{(r)}) = C_2(T_i^{(r)})$ for every r-type leaf as a root for the subtree $T_i^{(r)}$, where $1 \leq i \leq 2k$, and $C_1(t_i^{(o)}) = C_1(T_i^{(o)})$ and $C_2(t_i^{(o)}) = C_2(T_i^{(o)})$ for every o-type leaf as a root for the subtree $T_i^{(o)}$, where $2k+1 \leq i \leq m$. Then set
$$C_1(r^*) = \{r^*\} \bigcup \left(\bigcup_{i=1}^{2k} C_2(t_i^{(r)}) \right) \bigcup \left(\bigcup_{j=2k+1}^{m} C_2(t_j^{(o)}) \right)$$
as the optimal solution to the all-ones problem of the subtree rooted at r^*, and set
$$C_2(r^*) = \left(\bigcup_{i=1}^{2k} C_1(t_i^{(r)}) \right) \bigcup \left(\bigcup_{j=2k+1}^{m} C_1(t_j^{(o)}) \right)$$
as the optimal solution to the quasi all-ones problem of the subtree rooted at r^*.
2. The tree is type II. Denote its root by b^*. Suppose that the children of b^* are $t_1^{(r)}, \ldots, t_{2k-1}^{(r)}, t_{2k}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 1$. We already knew that $C_1(t_i^{(r)}) = C_1(T_i^{(r)})$ and $C_2(t_i^{(r)}) = C_2(T_i^{(r)})$ for every r-type leaf as a root for the subtree $T_i^{(r)}$, where $1 \leq i \leq 2k-1$, and $C_1(t_i^{(o)}) = C_1(T_i^{(o)})$ and $C_2(t_i^{(o)}) = C_2(T_i^{(o)})$ for every o-type leaf as a root for the subtree $T_i^{(o)}$, where $2k-1 \leq i \leq m$. Then set
$$C_1(b^*) = \{b^*\} \bigcup \left(\bigcup_{i=1}^{2k-1} C_2(t_i^{(r)}) \right) \bigcup \left(\bigcup_{j=2k}^{m} C_2(t_j^{(o)}) \right)$$
as the optimal solution to the all-ones problem of the subtree rooted at b^* such that b^* belongs to the optimal solution, and set
$$C_2(b^*) = \left(\bigcup_{i=1}^{2k-1} C_1(t_i^{(r)}) \right) \bigcup \left(\bigcup_{j=2k}^{m} C_1(t_j^{(o)}) \right)$$
as the optimal solution to the all-ones problem of the subtree rooted at b^* such that b^* does not belong to the optimal solution.
3. The tree is type III. Denote its root by o^*. Suppose that the children of o^* are $t_1^{(b)}, \ldots, t_k^{(b)}, t_{k+1}^{(r)}, \ldots, t_l^{(r)}, t_{l+1}^{(o)}, \ldots, t_m^{(o)}$, where $k \geq 1$. Use the pairs of sets on the nodes $t_1^{(b)}, \ldots, t_k^{(b)}$ to make a two-dimensional matrix $C_{2\times k} = (c_{ij})_{2\times k}$ such that $|C_2(t_i^{(b)})|$ is the value of the element c_{0i} in $(c_{ij})_{2\times k}$ and $|C_1(t_i^{(b)})|$ is the value of the element c_{1i}.

Remark 3.1. From Theorem 2.1 and Remarks 2.1 and 2.2, any solution to the all-ones problem (or the quasi all-ones problem) of the subtree rooted at o^* cannot contain the o-type root o^*. This means that all the r-type children of the o-type

root must be contained in the solution. Because the solution must contain an odd (or even) number of children of the o-type root, we have to employ our minimum odd sum algorithm or the minimum even sum algorithm to choose some of the b-type children into an optimal solution, according to the parity of $l-k$.

Step 2 (cont'd). If $l-k$ is even (odd), we use the minimum odd (even) sum algorithm to choose the elements in $(c_{ij})_{2\times k}$. Suppose that the union of the elements chosen from the 0th row in $(c_{ij})_{2\times k}$ is $\bigcup_{p=1}^{n_1} c_{0j_p}$, and the union of the elements chosen from the first row in $(c_{ij})_{2\times k}$ is $\bigcup_{q=1}^{n_2} c_{1j_q}$, where $n_1+n_2=k$. Then we set

$$C_1(o^*)=\left(\bigcup_{p=1}^{n_1} C_2(t_{j_p}^{(b)})\right)\bigcup\left(\bigcup_{q=1}^{n_2} C_1(t_{j_q}^{(b)})\right)\bigcup\left(\bigcup_{i=k+1}^{l} C_1(t_i^{(r)})\right)\bigcup\left(\bigcup_{i=l+l}^{m} C_1(t_i^{(o)})\right)$$

as the optimal solution to the all-ones problem of the subtrees rooted at o^*. Next, we use the minimum even (odd) sum algorithm to choose the elements in $(c_{ij})_{2\times k}$. Suppose that the union of the elements chosen from the 0th row in $(c_{ij})_{2\times k}$ is $\bigcup_{p=1}^{n_1} c_{0j_p}$, and the union of the elements chosen from the first row in $(c_{ij})_{2\times k}$ is $\bigcup_{q=1}^{n_2} c_{1j_q}$, where $n_1+n_2=k$. Then we set

$$C_2(o^*)=\left(\bigcup_{p=1}^{n_1} C_2(t_{j_p}^{(b)})\right)\bigcup\left(\bigcup_{q=1}^{n_2} C_1(t_{j_q}^{(b)})\right)\bigcup\left(\bigcup_{i=k+1}^{l} C_1(t_i^{(r)})\right)\bigcup\left(\bigcup_{i=l+l}^{m} C_1(t_i^{(o)})\right)$$

as the optimal solution to the quasi all-ones problem of the subtrees rooted at o^*.

Step 3. $i := i-1$, go to Step 1;

Step 4. We are now ready to give an optimal solution for the rooted tree T from the pair on the root by distinguishing the following three cases: (i) If the root is of r-type, denoted by r^*, then the $C_1(r^*)$ of the pair is an optimal solution. (ii) If the root is of b-type, denoted by b^*, then the $C_1(b^*)$ of the pair is a candidate for the optimal solution such that b^* belongs to the candidate solution, and the $C_2(b^*)$ of the pair is another candidate for the optimal solution such that b^* does not belong to the candidate solution. Now, compare the values of $|C_1(b^*)|$ and $|C_2(b^*)|$. Suppose that $|C_t(b^*)| = \min\{|C_1(b^*)|, |C_2(b^*)|\}$, $t\in\{1,2\}$. Then we choose $C_t(b^*)$ as an optimal solution. (iii) If the root is of o-type, denoted by o^*, then the $C_1(o^*)$ of the pair is an optimal solution.

THEOREM 3.2. *The above algorithm outputs an optimal solution to the all-ones problem of a given tree T, and the time complexity is linear.*

Proof. In Step 0, we regard every leaf in the bottom of the tree as a subtree whose unique optimal solution to the all-ones problem and the quasi all-ones problem contains exactly the node itself and nothing, respectively. Then the initial values of all the leaves of the tree can be completely determined. The algorithm now proceeds inductively on the number of layers of the tree. Then from the method for constructing solutions in the proof of Theorem 2.1 and from Remarks 2.1, 2.2, and 3.1, it is easy to conclude that the algorithm ensures that all the leaves v of the small trees of all types I, II, and III in every layer have recorded the right information, i.e., the pairs $\{C_1(v), C_2(v)\}$. From these pairs of sets, we can choose the optimal solution for the given tree according to the type of the root of the tree. The first statement of the theorem is thus proved.

For the second statement of the theorem, it is not hard to see that for every layer, the algorithm uses time linear in the number of nodes in the layer, even though

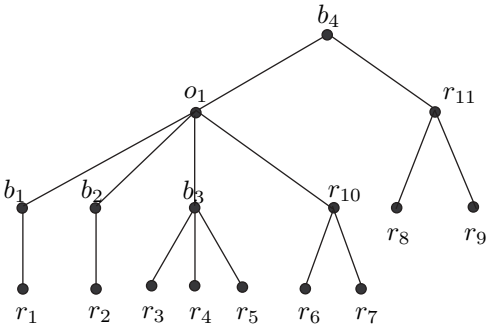

FIG. 3. *Labeling the type for every node.*

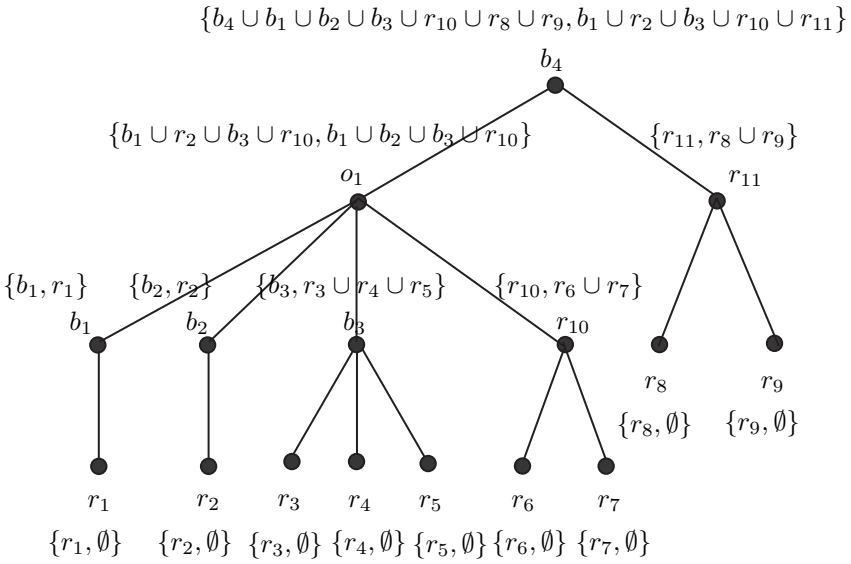

FIG. 4. *An example of our algorithm for the minimum all-ones problem for trees.*

sometimes the minimum odd or even sum algorithm has to be used. Therefore, the total time used by the algorithm is linear for the minimum all-ones problem of the given tree. The proof is complete. □

An example to show our algorithm at work is given in Figures 3 and 4. For every node in Figure 3, the label, by ignoring its subscript, is the type of that node. In Figure 4 we simply use x to denote the set $\{x\}$ with a single element x. Initially, our algorithm sets a pair $\{v, \emptyset\}$ for every leaf v. Then, from the bottom up, the pair for each node in every layer can be generated. Note that the children of o_1 are b_1, b_2, b_3, and r_{10}, and the pairs on the b-type nodes can form a matrix as described in our algorithm: $P_{23} = \left(\begin{smallmatrix} r_1 & r_2 & r_3 \cup r_4 \cup r_5 \\ b_1 & b_2 & b_3 \end{smallmatrix}\right)$. The corresponding numerical matrix is $C_{23} = \left(\begin{smallmatrix} 1 & 1 & 3 \\ 1 & 1 & 1 \end{smallmatrix}\right)$, which will be used in the minimum odd (even) sum algorithm. Note that only r_{10} is an r-type node. Then use the minimum even sum algorithm to get $b_1 \cup r_2 \cup b_3$ (possibly, $r_1 \cup b_2 \cup b_3$) union r_{10} and form $C_1(o_1)$. Use the minimum odd

sum algorithm to get $b_1 \cup b_2 \cup b_3$ (possibly, $r_1 \cup r_2 \cup b_3$) union r_{10} and form $C_2(o_1)$. In the end, by comparing $|C_1(b_4)| = 7$ with $|C_2(b_4)| = 5$, we get an optimal solution $C_2(b_4) = b_1 \cup r_2 \cup b_3 \cup r_{10} \cup r_{11}$. The details about the pair $\{C_1(v), C_2(v)\}$ for every node v are recorded in Figure 4.

4. Concluding remarks. Although the existence of solutions to the all-ones problem for general graphs was proved by linear algebraic methods (see [10, 5]), in [10] Sutner asked whether there is a graph-theoretic proof for the existence. Eriksson, Eriksson, and Sjöstrand [3] gave such a proof. However, how to find a solution efficiently by graph-theoretic algorithms remains unknown. Although, based on the result in [3], one can get a graph-theoretic algorithm inductively, the time complexity is not polynomial, which is upper bounded by $O(n!)$. It is easy to see that for the empty graph with n nodes, their algorithm runs in time $O(n!)$. So, to find a graph-theoretic algorithm of polynomial time for general graphs, some other ideas are needed. For trees, Galvin [4] gave a graph-theoretic algorithm of linear time. In this concluding section, based on the discussion in section 2 we would like to give such an algorithm of linear time for unicyclic graphs.

For convenience, we say that the truth value of a node in G is 1 if it belongs to the solution to the all-ones problem (or the quasi all-ones problem) for G, and 0 otherwise. Recall that a graph G is called *unicyclic* if it contains a unique cycle. In other words, we can regard a unicyclic graph as a cycle attached with each node to a rooted tree, called a *suspended tree*. Note that the depth of a suspended tree can be 0. For simplicity, we say that a node t in the cycle has the same type as the type of the root t of the suspended tree.

Algorithm for unicyclic graphs.

Input. A unicyclic graph G, each node in the unique cycle being labeled by types.

Step 1. If none of the nodes on the cycle is an outcast, then let the truth values of all nodes on the cycle be 1; i.e., take the union of the solutions, each of which is a solution to the all-ones problem for each suspended tree whose root belongs to the solution. Then the union is a solution to the all-ones problem for the whole unicyclic graph.

Step 2. If there are outcast nodes, we fix an order to the nodes on the cycle. Then we cut the cycle by deleting the edge between an outcast node u and the node v before it on the cycle. The unicyclic graph becomes a tree with root v, denoted as T, and the type of every node on the original cycle will be changed as in Figure 5, where the changing of the type of each node is from on the original cycle to on the tree T, and the changing rule is just the same as that in the Galvin method.

By Galvin's algorithm, we can construct a solution X for the tree T. Then we add an edge to connect the node u and the node v in T; then the tree T returns to the original unicyclic graph with a solution X. Because u is an outcast node, no matter whether it is in the unicyclic graph or in the tree, it will not belong to the solution, and hence will not affect the other nodes' truth values, while u's on or off status will probably be affected by v if v belongs to X. If v does not belong to X, then X is a solution to the all-ones problem for the unicyclic graph; otherwise, we only need to change the solution to the all-ones problem into the solution to the quasi all-ones problem, or the other way around for the suspended tree with root u according to the construction method in the proof of Theorem 2.1; then the modified solution is a solution to the all-ones problem for the unicyclic graph. An example is shown in Figure 6.

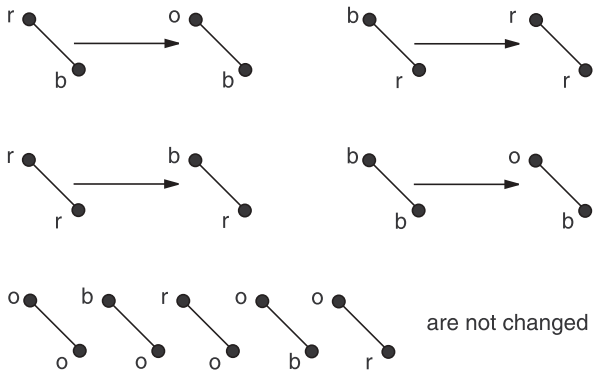

FIG. 5. *Changing of types.*

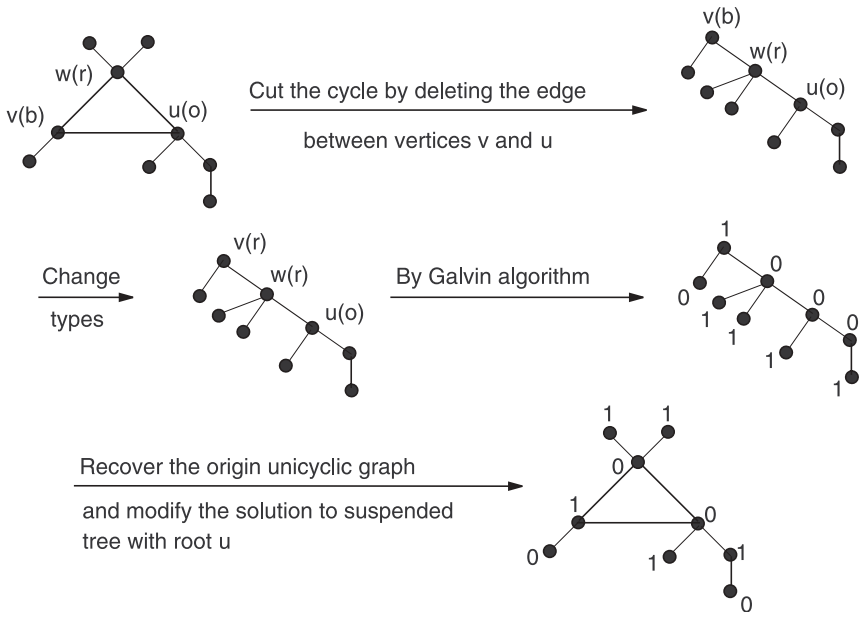

FIG. 6. *An example of our algorithm for unicyclic graphs.*

So, we get the following result.

THEOREM 4.1. *The above algorithm outputs solutions to the all-ones problem for unicyclic graphs, and the time complexity is linear.*

To end this paper, we propose the following problem.

All-colors problem. The so-called *all-colors problem* on graphs is described as follows, which is a natural generalization for the all-ones problem:

For any node of a graph G, it has a color value between 0 and $r-1$. If a node is pressed one time, then the color values of the node and its neighbors are added by 1 under the meaning of modular r. If the initial status is that the color value of every node is 0, then we ask how to press some nodes (maybe many times) to make the color value of every node equal to $r-1$ (or any fixed k such that $1 \leq k \leq r-1$) under

the meaning of modular r. If we ask that the sum of color values of all nodes attains the minimum, the problem is called the *minimum all-colors problem*.

Acknowledgment. The authors are greatly indebted to the referees for their invaluable suggestions and comments, which have substantially improved the presentation of the paper.

REFERENCES

[1] R. Barua and S. Ramakrishnan, *σ-game, σ^+-game and two-dimensional additive cellular automata*, Theoret. Comput. Sci., 154 (1996), pp. 349–366.
[2] Y. Dodis and P. Winkler, *Universal configurations in light-flipping games*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM, New York, SIAM, Philadelphia, 2001, pp. 926–927.
[3] H. Eriksson, K. Eriksson, and J. Sjöstrand, *Note on the lamp lighting problem*, Adv. in Appl. Math., 27 (2001), pp. 357–366.
[4] F. Galvin, *Solution to problem* 88-8, Math. Intelligencer, 11 (2) (1989), pp. 31–32.
[5] O. P. Lossers, *Solution to problem* 10197, Amer. Math. Monthly, 100 (1993), pp. 806–807.
[6] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimizations: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[7] U. Peled, *Problem* 10197, Amer. Math. Monthly, 99 (1992), p. 162.
[8] K. Sutner, *Additive automata on graphs*, Complex Systems, 2 (1988), pp. 1–28.
[9] K. Sutner, *Problem* 88-8, Math. Intelligencer, 10 (3) (1988), p. 101.
[10] K. Sutner, *Linear cellular automata and the Garden-of-Eden*, Math. Intelligencer, 11 (2) (1989), pp. 49–53.
[11] K. Sutner, *The σ-game and cellular automata*, Amer. Math. Monthly, 97 (1990), pp. 24–34.
[12] K. Sutner, *σ-automata and Chebyshev-polynomials*, Theoret. Comput. Sci., 230 (2000), pp. 49–73.