Rooted Tree Sequence Problems

Rooted Tree Sequence Problems

Masaya TAKAHASHI (Department of Electronics and Information Science, Fukuoka J. College of Technology)

SUMMARY

Let D be a directed tree. If $\deg^-(r)=0$ for some vertex $r \in D$, and $\deg^-(v)=1$ for any vertex $v \in D$ with $v \neq r$, then r is called a *root* and D is called a *rooted tree*. A sequence of nonnegative integers $S=(s_1, s_2, \cdots, s_n)$ is a *rooted tree sequence* if there is a rooted tree with vertices $v_1, v_2, \cdots, v_n$ such that $\deg^+(v_j)=s_j$ for each $j=1, 2, \cdots, n$. The rooted tree sequence problem is: Given a sequence of nonnegative integers, determine whether it is a rooted tree sequence or not. In this paper, I consider several variations of the rooted tree sequence problem and give linear time algorithms.

Key words: *minium depth, leveled rooted tree, tournament tree, score sequence, optimal and optimum condition*

1. Introduction

Let D be a directed tree. If $\deg^-(r)=0$ for some vertex $r \in D$, and $\deg^-(v)=1$ for any vertex $v \in D$ with $v \neq r$, then r is called a *root* and D is called a *rooted tree* ($\deg^-(v)$ is the indegree of v). A sequence of nonnegative integers $S=(s_1, s_2, \cdots, s_n)$ is a *rooted tree sequence* if there is a rooted tree with vertices $v_1, v_2, \cdots, v_n$ such that $\deg^+(v_j)=s_j$ for each $j=1, 2, \cdots, n$ ($\deg^+(v_j)$ is the outdegree of v_j). The rooted tree sequence problem is: Given a sequence of nonnegative integers, determine whether it is a rooted tree sequence or not. The rooted tree sequence problem was considered as the special case of graphical degree sequence problems by Menon [2]. The graphical degree sequence problems and the variations of them have been considered by Havel [6], Erdös and Gallai [7], Takahashi, Imai and Asano [9,10], Landau [11] and others [3,4,5].

In this paper, I consider several variations of the rooted tree sequence problem and give linear time algorithms. Furthermore, I consider the score sequence problem of a tournament tree and give linear time algorithms.

平成6年10月27日受付

2. Rooted Tree Sequence Problem

In this section, I consider the rooted tree sequence problem. I first recall the previous results. Menon [2] gave Proposition 2. 1 in the following. (The proposition is introduced in a standard book of graph theory [3].)

Proposition 2. 1: Let $S=(s_1, s_2, \cdots, s_n)$ be a sequence of nonnegative integers. Then S is a directed tree sequence if and only if $\sum_{j=1}^{n} s_j = n-1$.

Based on Proposition 2. 1, I can determine whether S is a rooted tree sequence or not in $O(n)$ time.

I can assume without loss of generality that $s_{p(1)} \geqq s_{p(2)} \geqq \cdots \geqq s_{p(n)}$ (p is a pernutation on $\{1, 2, \cdots, n\}$). Then $s_{p(1)} \geqq 1$ and $s_{p(n)} \geqq 0$ hold. By the following algorithm, it is clear that a rooted tree D with

S as a rooted tree sequence can be obtained.

Algorithm CRT:

Step 1: $u_1 := 0$ and $q := 1$.

Step 2: *For* $j := 2$ *to* n *do* the following.

(a) $u_j := 0$, $u_q := u_q + 1$ and add edge $(v_{p(q)}, v_{p(j)})$.

(b) *If* $u_q = s_q$ *then* $q := q + 1$ and $u_q := 0$.

Since the sorting of S to satisfy $s_{p(1)} \geqq s_{p(2)} \geqq \cdots \geqq s_{p(n)}$ requires only $O(n)$ time [1], if S is a rooted tree sequence then a rooted tree D with S can be constructed in $O(n)$ time.

In the following, I consider variations of the rooted tree sequence problem and present linear time algorithms.

2. 1 Minimum Depth Problem

Let D be a rooted tree with root r. For any path $P = (v_0, v_1), (v_1, v_2), \cdots, (v_{k-1}, v_k)$ of D, k is called a *distance* and denoted by $d(v_0, v_k)$. I define that $d(v_0, v_0) = 0$. For each vertex $v \in D$, $\max\{d(r, v)\}$ is also called a *depth* of D. Then I consider the *minimum depth problem*: Given a rooted tree sequence $S = (s_1, s_2, \cdots, s_n)$, construct a rooted tree D with S as a rooted tree sequence such that a depth of D is minimum.

Let x be the lower bound of D with S as a rooted tree sequence. If S has a special from, I present the lower bound as follows.

Proposition 2. 2: Let $S = (s_1, s_2, \cdots, s_n)$ be a rooted tree sequence with $s_1 \geqq s_2 \geqq \cdots \geqq s_n$. Then the following (1) and (2) hold:

(1) If $s_1 \geqq (n-1)/2$ then $x = \lceil (n-1)/s_1 \rceil$,

(2) If $s_1 = s_2 = \cdots = s_r \geqq 1$, $s_{r+1} = \cdots = s_n = 0 (1 \leqq r \leqq n-1)$ and $\sum_{j=0}^{q-1} \{n-1)/r\}^j + 1 \leqq n \leqq \sum_{j=0}^{q} \{n-1)/r\}^j$ hold for some integer q, then $x = q$. Especially, if $s_1 = 2$ then $x = \lfloor \log_2 r \rfloor + 1$. □

It is easy to prove Proposition 2. 2, and I will omit the proof here. In general case, I can obtain the following proposition.

Proposition 2. 3: Let $S = (s_1, s_2, \cdots, s_n)$ be a rooted tree sequence with $s_1 \geqq s_2 \geqq \cdots \geqq s_n$. Then the lower bound x can be found and a rooted tree D with S such that a depth of D is x can be constructed, by the following algorithm.

Algorithm CRT-1:

Step 1: $u_1 := 0$, level $(v_1) := 0$ and $q := 1$.

Step 2: *For* $j := 2$ *to* n *do* the following.

(a) $u_j := 0$, level $(v_j) :=$ level $(v_q) + 1$ and $u_q := u_q + 1$.

(b) add edge (v_q, v_j) and *if* $u_q = s_q$ *then* $q := q + 1$ and $u_q := 0$.

Step 3: $x :=$ lebel (v_n).

Proof: Let D be a rooted tree with S such that a depth of D is x'', and let D_1 be a rooted tree obtaibed by Algorithm CRT-1.

Suppose that D has a vertex v_j satisfying $d(r, v_j) >$ level (v_j) for some $1 \leqq j < q \leqq n$. Then D has a vertex v_q satisfying $d(r, v_g) <$ level (v_g) for some $1 \leqq j < q \leqq n$. Since $j < q$, $s_j \geqq s_q$ hold.

Assume $s_j = s_q$. I construct a directed tree D' obtained by swapping v_j and v_q. Let x' be a deaph of D'. Then $x' = x''$ holds.

Assume $s_j > s_q$. For each $t = 1, 2, \cdots, s_j$, let w_t be a vertex such that D has an edge (v_j, w_t), $sD(w_t)$ be a rooted subtree with root w_t, and sdp (w_t) be a depth of $sD(w_t)$. Then I can assume sdp $(w_{k(1)}) \geqq$ sdp $(w_{k(2)}) \geqq \cdots \geqq$ sdp $(w_{k(s_j)})$ (k is a permutation on $\{1, 2, \cdots, s_j\}$). I construct a rooted tree D' obtained by swapping $v_j \cup \{(v_j, w_{k(t)}) | t = 1, 2, \cdots, s_j - s_q\} \cup \{sD(w_{k(t)}) | t = 1, 2, \cdots, s_j - s_q\}$ and v_q. Let x' be a depth of D'. Then $x' \leqq x''$ holds.

Suppose that $v_j \in D$ satisfies $d(r, v_j) =$ level (v_j) for each $j = 1, 2, \cdots, n$. Assume that D does not have an edge (v_j, v_a) such that (v_j, v_a) in D_1 for some $1 \leqq j < a \leqq n$. Then D has an edge (v_j, v_b) such that (v_j, v_b) not in D_1 for some $1 \leqq j < b \leqq n$, and has an edge (v_q, v_a) such that (v_q, v_a) not in D_1 for some $1 \leqq q < a \leqq n$. (It is clear that level $(v_j) =$ level (v_q).) I construct a rooted tree $D' = D \cup \{(v_j, v_a), (v_q, v_b)\} - \{(v_j, v_b), (v_q, v_a)\}$. Let x' be a depth of D'. Then $x' = x''$ holds.

For the argument above, see Example 2. 1. By setting $D:=D'$, $x'':=x'$, and repeating the argument above, I can finally obtain $D=D_1$ with S as a rooted tree sequence such that a depth x of D_1 is minimum. □

Example 2. 1: Let $S=(4, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0)$, D be any rooted tree with S (see Fig. 2. 1), and D_1 be a rooted tree obtained by Algorithm CRT-1 with S (see Fig. 2. 2).Then depth of D is 3 and depth of D_1 is 2.

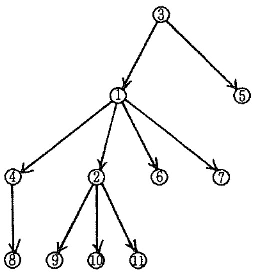

Fig. 2. 1.

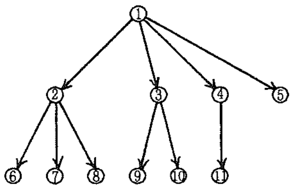

Fig. 2. 2.

(1) $d(r, v_1)=1>\text{level}(v_1)=0$ holds in D. Then $d(r, v_3)=0<\text{level}(v_3)=1$, $d(r, v_1)=1>d(r, v_3)=0$ and $1<3$ hold for v_3 in D. By swapping $\{v_1\}\cup\{(v_1, v_2), (v_1, v_4)\}\cup\{sD(v_2), sD(v_4)\}$ and v_3, I can obtain a rooted tree D' such that a depth of D' is 2 (see Fig. 2. 3). Set $D:=D'$.

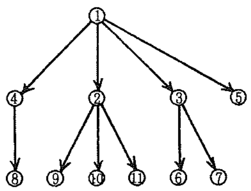

Fig. 2. 3.

(2) I construct a rooted tree $D'=D\cup\{(v_2, v_6), (v_3, v_9)\}-\{(v_2, v_9), (v_3, v_6)\}$ (see Fig. 2. 4). Then a depth of D' is 2. Set $D:=D'$. By repeating same operations, I can finally obtain $D=D_1$ with S such that a depth of D_1 is 2. □

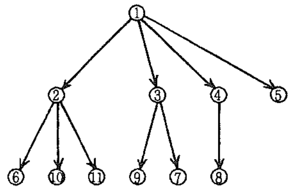

Fig. 2. 4.

It is easy to see that Algorithm CRT-1 requires only $O(n)$ time. Thus, based on Proposition 2. 3, I have the following theorem.

Theorem 2. 1: For a rooted tree sequence $S=(s_1, s_2, \cdots, s_n)$ with $s_1\geqq s_2\geqq\cdots\geqq s_n$, a rooted tree with minimum depth and S, can be constructed in $O(n)$ time.

2. 2 Leveled Rooted Tree Ssequence Problem

Let $S=(s_1, s_2, \cdots, s_n)$ be a rooted tree sequence, and D be a rooted tree with S. Then D is called a *leveled rooted tree* if and only if v_1 is a root of D and $d(v_1, v_j)\leqq d(v_1, v_q)$ for each $1\leqq j<q\leqq n$.

Example 2. 2: Let $S=(2, 3, 1, 0, 0, 0, 2, 1, 0, 0)$, D_1 (see Fig. 2. 5) and D_2 (see Fig.2. 6) be rooted trees. D_1 is a leveled rooted tree since v_1 is a root of D_1 and $d(v_1, v_j)\leqq d(v_1, v_q)$ for each $1\leqq j<q\leqq n$. However D_2 is not so since $d(v_1, v_3)=2>d(v_1, v_4)=1$.□

Then I consider the *leveled rooted tree sequence problem:* Given a rooted tree sequence $S=(s_1, s_2, \cdots, s_n)$, determine whether S is a leveled rooted tree sequence (i. e., there is a leveled rooted tree with vertex set $V=\{v_1, v_2, \cdots, v_n\}$ such that $\deg^+(v_j)=s_j$ for each $j=1, 2, \cdots, n$). Then the following proposition holds.

Proposition 2. 4: Let $S=(s_1, s_2, \cdots, s_n)$ be a

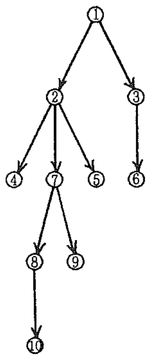

Fig. 2. 5.

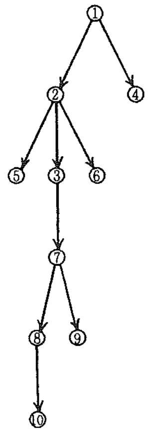

Fig. 2. 6.

rooted tree sequence (which is a randum nonnegative integer sequence). Then S is a leveled rooted tree sequence if and only if $\sum_{j=1}^{k} s_j \geqq k$ for each $k=1, 2, \cdots, n-1$, with equality holding for $k=n-1$.

Proof: Necessity can be obtained as follows. Let $S=(s_1, s_2, \cdots, s_n)$ be a leveled rooted tree sequence and D be a leveled rooted tree with S. I use an induction on the distance. For v_1, $d(v_1, v_1)=0$ and $s_1 \geqq 1$ since v_1 is a root of D. Thus $\sum_{j=1}^{1} s_j \geqq 1$ holds. Let x be the depth of D and r be any nonnegative integer with $r<x$. Let $z=\max\{j \mid d(v_1, v_j)=r-1\}$ and $t=\max\{j \mid d(v_1, v_j)=r\}$. Then $t<n$ and $\sum_{j=1}^{z} s_j \geqq t-1$. Assume that $\sum_{j=1}^{k} s_j \geqq k$ for each $k=1, 2, \cdots, t$, as the hypothesis of the induction. Let $q=\sum_{j=z+1}^{t} s_j$. Then $\sum_{j=1}^{t} s_j = t+q-1$, and $\max\{j \mid d(v_1, v_j)=r+1\}=t+q$. Suppose that $r+1<x$. Then $t+q<n$ and $\sum_{j=t+1}^{t+q} s_j \geqq 1$. Thus $\sum_{j=1}^{t+b} s_j \geqq t+q-1$ for each $b=1, 2, \cdots, q-1$, and $\sum_{j=1}^{t+q} s_j \geqq t+q$ hold. Suppose that $r+1=x$. Then $t+q=n$ and $\sum_{j=t+1}^{n} s_j=0$. Thus $\sum_{j=1}^{t+b} s_j = t+q-1=n-1$ holds for each $b=1, 2, \cdots, n-t-1$. This completes the proof of necessity.

Sufficiency can be obtained as follows. I use an induction on k. Assume $k=1$. Let D_1 be a rooted tree with vertex v_1. Then D_1 has no edge and is a leveled rooted tree. Assume $k=q$ for some integer $1 \leqq q \leqq n-1$. Suppose that D_q be a leveled rooted tree with vertices $v_1, v_2, \cdots, v_q$, as the hypothesis of the induction. Then D_q has only $q-1$ edges. However, $\sum_{j=1}^{q} s_j \geqq q$ (if $q=n-1$ then $\sum_{j=1}^{n-1} s_j = n-1$). Thus D_q has a vertex v_b satisfying $\deg^+(v^b) < s_b$ for some $1 \leqq b \leqq q$. Let $t=d(v_1, v_q)$ and $r=\min\{j \mid \deg^+(v_j) < s_j\}$. Then $d(v_1, v_r)=t-1$ or $d(v_1, v_r)=t$ since D_q is a leveled rooted tree. Furthermore, if $d(v_1, v_r)=t$ then $\deg^+(v_r)=0$. Thus I can construct a rooted tree $D_{q+1}=D_q+(v_r, v_{q+1})$ with vertices $v_1, v_2, \cdots, v_{q+1}$. Then, if $d(v_1, v_r)=t-1$ then $d(v_1, v_{q+1})=t$ else $d(v_1, v_{q+1})=t+1$. Hence $d(v_1, v_j) \leqq d(v_1, v_b)$ holds for each $1 \leqq j < b \leqq q+1$ and D_{q+1} is a leveled rooted tree. This completes the proof of sufficiency. □

Based on proposition 2. 4, the following theorem can be obtained easily.

Theorem 2. 2: For a rooted tree sequence $S=(s_1, s_2, \cdots, s_n)$ which is a randum nonnegative integer sequencce, it can be determined in $O(n)$ time whether S is a leveled rooted tree sequence or not.

Next I present an algorrithm for actually constructing a leveled rooted tree for a given leveled rooted tree sequence based on the following proposition.

Proposition 2. 5: Let $S=(s_1, s_2, \cdots, s_n)$ be a rooted tree sequence. Let $T=(t_1, t_2, \cdots, t_{n-1})$ be defined by using $q=\max\{j \mid s_j \geqq 1\}$ as follows.

$$t_j=\begin{cases} s_j-1 & \text{if } j=q, \\ s_j & \text{otherwise.} \end{cases}$$

Then S is a leveled rooted tree sequence if and only if T is a leveled rooted tree sequence and $s_n=0$.

Proof: Sufficiency can be obtained as follows. Let T be a leveled rooted tree sequence and $s_n=0$. Then $\sum_{j=1}^{n-1} t_j = n-2$ and $\sum_{j=1}^{k} t_j \geqq k$ for each $j=1, 2, \cdots, n-2$ (if $k=n-2$ then $\sum_{j=1}^{n-2} t_j = n-2$), by Proposition 2.4. Thus $\sum_{j=1}^{k} s_j = \sum_{j=1}^{k} t_j \geqq k$ for each $j=1, 2, \ldots q-1$, and $\sum_{j=1}^{k} s_j = \sum_{j=1}^{k} t_j + 1 \geqq k+1$ for each $j=q, q+1, \cdots, n-2$. Furthermore, $\sum_{j=1}^{n-1} s_j = \sum_{j=1}^{n-1} t_j$

$+1=(n-2)+1=n-1$ and $\sum_{j=1}^{n} s_j=n-1$ since $s_n=0$. Hence $\sum_{j=1}^{n} s_j=n-1$ and $\sum_{j=1}^{k} s_j \geqq k$ for each $j=1, 2, \cdots, n-1$ (if $k=n-1$ then $\sum_{j=1}^{n-1} s_j=n-1$), and therefore, S is a leveled rooted tree sequence.

Necessity can be obtained as follows. Let S be a leveled rooted tree sequence and D be a leveled rooted tree with S. Since $\sum_{j=1}^{n} s_j=n-1$ and $\sum_{j=1}^{n-1} s_j=n-1$ by Proposition 2. 4, $s_n=0$ holds. Let $d(v_1, v_n)=r$ in D. Suppose that $(v_q, v_n) \in D$. Then $d(v_1, v_q)=r-1$ in D and $D'=D-v_n$ is also a leveled rooted tree with T. Suppose that $(v_q, v_n) \notin D$. Then D has a vertex v_z such that $s_z \geqq 1$, $d(v_1, v_z)=r-1$ and (v_z, v_n) in D. Since D is a leveled rooted tree, $d(v_1, v_q)=r-1$ and $z<q$. Furthermore, D has a vertex w such that $\deg^+(w)=0$, $d(v_1, w)=r$ and (v_q, w) in D. Thus $D'=D \cup \{(v_q, v_n), (v_z, w)\} - \{(v_g, w), (v_z, v_n)\}$ is also a leveled rooted tree with S. By setting $D:=D$, $D'=D-v_n$ is also a leveled rooted tree with T. □

Based on Proposition 2. 5, I can obtain the following algorithm CLRT for constructing a leveled rooted tree D having S as a leveled rooted tree sequence.

Algorithm CLRT.

Step 1: $u_1=0$, level$(v_1):=0$ and $q:=1$.

Step 2: *For* $j:=2$ *to* n *do* the following.

(a) $u_j:=0$, level$(v_j):=$ level$(v_q)+1$ and $u_q:=u_q+1$.

(b) *add edge* (v_q, v_j).

(c) *while* $u_q=s_q$ do $q:=q+1$ and $u_q:=0$.

It is easy to that Algorithm CLRT correctly constructs a leveled rooted tree D with S as a leveled rooted tree sequence and that it takes $O(n)$ time. Thus the following theorem can be obtained.

Theorem 2. 3: For a leveled rooted tree sequence $S=(s_1, s_2, \cdots, s_n)$, a leveled rooted tree with S can be obtained in $O(n)$ time.

3. Score Sequence Problem of a Tournament Tree

Let T be a rooted tree with n leaves. (Note that any vertex $v \in T$ is called a *leaf* if and only if $\deg^+(v)=0$ and $\deg^-(v)=1$.) Then T is called a *tournament tree* if and only if the following (1) through (3) are satisfied:

(1) For a root $r \in T$, $\deg^+(r)=2$ and $\deg^-(r)=0$,

(2) For any vertex $v \in T$ which is not a root and is not a leaf, $\deg^+(v)=2$ and $\deg^-(v)=1$,

(3) T has $2n-1$ vertices and $2n-2$ edges.

Let T be a tournament tree with n leaves and $W=\{w_1, w_2, \cdots, w_n\}$ be a set of n elements. Then a champion of W is selected by using the following method:

(1) Each element $w_j (1 \leqq j \leqq n)$ of W is set to a leaf of T. There is an one-to-one correspondence between W and leaves of T,

(2) For a vertex $v \in T$ which is not a leaf, and two edges $(v, u_1), (v, u_2)$, assume $u_1=w_j$ and $u_2=w_q$ $(j \neq q, 1 \leqq j \leqq n, 1 \leqq q \leqq n)$. Then w_j is a winner then set $v:=w_j$ else $v:=w_q$.

It is clear that a root of T is champion of W.

Example 3. 1: Let $W=(w_1, w_2, \cdots, w_7)$ be a set of seven elements and T be a tournament tree with seven leaves (see Fig. 3. 1). Assume that each element of W is assigned to leaves of T (see Fig. 3. 1). Suppose that (w_j, w_q) means "w_j win to w_q." If $(w_4, w_6), (w_1, w_4), (w_2, w_7), (w_1, w_3), (w_5, w_2)$, and (w_1, w_5) then I can obtain a result as shown in Fig. 3. 2. □

Let win(w_j) be a number of win of w_j and lose(w_j) be a number of lose of w_j for each $j=1, 2, \cdots, n$. Then $\sum_{j=1}^{n}$ win$(w_j)=n-1$, lose$(w_q)=0$ for some $q, 1 \leqq q \leqq n$, and lose$(w_j)=1$ for each $j, 1 \leqq j \leqq n, j \neq q$. Thus (win$(w_1)$, win$(w_2), \cdots,$ win(w_n)) is a rooted tree sequence and T has $2n-2=2($

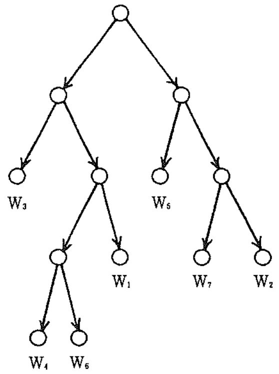

Fig. 3. 1.

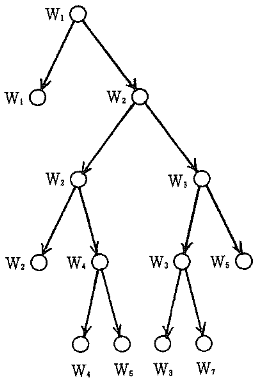

Fig. 3. 2.

$\sum_{j=1}^{n}$ win $(w_j))$ edges.

Let $S=(s_1, s_2, \cdots, s_n)$ be a rooted tree sequence and let $s_j=$win(w_j) for each $j=1, 2, \cdots, n$. Then it is clear that there is a tournament tree T with S. Let r be a root of T, and let level$(w_j)=\min\{d(r, w_j)\}$ and index$(w_j)=j$ for each $j=1, 2, \cdots, n$. For a vertex $v \in T$ with $v \neq r$ and an edge (u, v), u is called a *parent* and denoted prt(v). Then T is called a *leveled tournament tree* if and only if the following conditions (C1) and (C2) are satisfied:

(C1) w_1 is a root of T (i. e., w_1 is a champion of W),

(C2) (lebel$(w_j)<$lebel(w_q)) or (lebel$(w_j)=$lebel(w_q) and index(prt(w_j)) $<$index(prt(w_q))) for each $1 \leqq j < q \leqq n$.

Example 3. 2: Let $S_1=(1, 2, 2, 1, 0, 0, 0)$ and $S_2=(3, 3, 0, 0, 0, 0, 0)$. Let D_1 (see Fig 3. 3) be tournament tree with S_1 as a rooted tree sequence and D_2 (see Fig. 3. 4) be tournament tree with S_2 as a rooted tree sepuence. D_1 is a leveled tournament tree since conditions (C1) and (C2) described above are satisfied. However D_2 is not so since level$(w_4)=3>$ level$(w_5)=2$ and index (prt(w_3)) $=$index$(w_2)=2>$index (prt(w_5)) $=$index$(w_1)=1$.
□

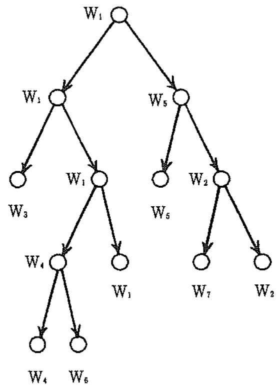

Fig. 3. 3.

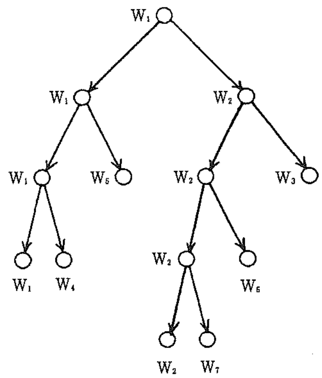

Fig. 3. 4.

In this section, I consider the *score sequence problem of a leved tournament tree*: Given a rooted tree sepuence $S=(s_1, s_2, \cdots, s_n)$, determine whether S is a score sequence of a leveled tournament tree (i. e., there is a set of n elements $W=$

$(w_1, w_2, \cdots, w_n)$ such that win$(w_j)=s_j$ for each $j=1, 2, \cdots, n$, and there is a leveled tournament tree with n leaves). Furthermore, I consider variations of the score sequence problem of a leveled tounament tree and present efficient algorithms.

3. 1 Characterization

I assume without loss of generality that $n \geqq 2$. Let leaf$(w_j)=\max\{d(r, w_j)\}$ for each $j=1, 2, \cdots, n$, where r is a root of a tournament tree T. Then $s_j=$win$(w_j)=$leaf$(w_j)-$level(w_j) holds for each $j=1, 2, \cdots, n$. Then the following proposition holds.

Proposition 3. 1: Let $S=(s_1, s_2, \cdots, s_n)$ be a rooted tree sequence (which is a randum non-negative integer sequence). Then S is a score sequence of a leveled tournament tree if and only if $\sum_{j=1}^{k} s_j \geqq k$ for each $k=1, 2, \cdots, n-1$, with equality holding for $k=n-1$.

Proof: Necessity can be obtained as follows. Let $S=(s_1, s_2, \cdots, s_n)$ be a score sequence of a leveled tournament tree ane T be a leveled tournament tree with S. Let q be any integer with $1 \leqq q \leqq n-1$, and $T'=T-\{(u, v) \mid u, v \in \{w_{q+1}, \cdots w_n\}\}$. Then T' has at least $q+1$ leaves and $2\sum_{j=1}^{q}$ win(w_j) edges. Thus $\sum_{j=1}^{q}$win$(w_j)=\sum_{j=1}^{q} s_j \geqq q$ holds. Suppose that $q=n-1$. Since $s_n \geqq 0$, $\sum_{j=1}^{n} s_j=n-1$ and $\sum_{j=1}^{n-1} s_j \geqq n-1$ by the discussion discribed above, I can obtain $\sum_{j=1}^{n-1} s_j \geqq n-1$. This completes the proof of necessity.

Sufficiency can be obtained as follows. I use an induction of k. Assume $k=1$. Let T_1 be a rooted tree with win$(w_1)+1=s_1+1$ vertices, s_1 edges and only one leaf. However, T_1 has s_1 vertices v with $\deg^+(v)=1$. Thus I can construct a rooted tree $T_2=T_1+(r_1, r_2)+D_2$, where r_1 is a root of T_1, D_2 is a rooted tree with s_2+1 vertices, s_2 edges and only one leaf, and r_2 is a root of D_2. Then T_2 sataisfies the leveled tournament tree rule. Assume $k=q$ for some integer $1 \leqq q \leqq n-1$. Suppose that T_q be a rooted tree with the leveled tournament tree rule, $\sum_{j=1}^{q} s_j+q$ vertices, $\sum_{j=1}^{q} s_j+q+1$ edges and q leaves as the hypothesis of the induction (see Example 3. 3). However, T_q can have $2\sum_{j=1}^{q} s_j$ edges. Thus, since $\sum_{j=1}^{q} s_j \geqq q$ (if $q=n-1$ then $\sum_{j=1}^{n-1} s_j=n-1$), T_q has at least one vertex v with $\deg^+(v)=1$. I can construct a rooted tree $T_{q+1}=T_q+(u_q, r_{q+1})+D_{q+1}$, where u_q is chosen from a vertex set $V'=\{v \mid \deg^+(v)=1$ in $T_q\}$ such that level(u_q) and index(u_q) are minimum respectively in V', D_{q+1} is a rooted tree with $s_{q+1}+1$ vertices, s_{q+1} edges and only one leaf, and r_{q+1} is a root of D_{q+1}. Then T_{q+1} satisfies the leveled tournament tree rule by the hypothesis of induction. This completes the proof of sufficiency. □

Example 3. 3: Let $S=(3, 2, 1, 0, 0, 1, 0, 0)$ be a rooted tree sequence and $S'=(3, 2, 1)$ be a subsequence of S. Then T_3 (see Fig. 3. 5) is a rooted tree with the leveled tournament tree rule, $(3+2+1)+3=9$ vertices, $9-1=8$ edges and three leaves. □

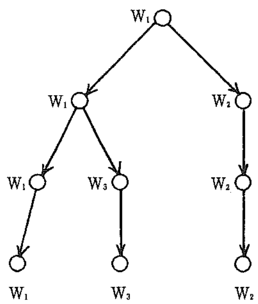

Fig. 3. 5.

Based on Proposition 3. 1, the following theorem can be obtained easily.

Theorem 3. 1: For a rooted tree sequence $S=(s_1, s_2, \cdots, s_n)$ which is a randum nonnegative integer sequence, it can be determined in $O(n)$ time whether S is a score sequence of a leveled tournament tree or not.

Next I present an algorithm for actually constructing a leveled tournament tree for a given score sequence of a leveled tournament tree. In the algorithm, L_j is initialized empty and represented by a doubly-linked list and $\mathrm{pre}_j(v)$ and $\mathrm{suc}_j(v)$ denote the previous element and the next element of $v \in L_j$ for each $j=0, 1, 2, \cdots, n-1$.

Algorithm CLTT.

Begin

1. $q:=1$; $d:=0$; $t_q:=w_1$;
   insert t_q into L_d as the last element; $q:=q+1$;
2. For $j:=2$ to n do begin
   (a) If L_d is empty then $d:=d+1$;
   $v:=$the first element of L_d;
   delete v from L_d;
   (b) $t_q:=v$; $\deg^+(v):=\deg^+(v)-1$;
   If $\deg^+(v)\geqq 1$ then insert t_q into L_{d+1} as index $(\mathrm{pre}_{d+1}(t_q)) <$ index$(t_q) <$ index$(\mathrm{suc}_{d+1}(t_q))$;
   (c) $t_{q+1}:=w_j$;
   If $s_j \geqq 1$ then insert t_{q+1} into L_{d+1} as index $(\mathrm{pre}_{d+1}(t_{q+1})) <$ index$(t_{q+1}) <$ index$(\mathrm{suc}_{d+1}(t_{q+1}))$;
   (d) add two edges (v, t_q) and (v, t_{q+1});
   $q:=q+2$ end

End.

It is easy see that Algorithm CLTT correctly constructs a leveled tournament tree T with S as a score sequence of a leveled tournament tree and that it takes $O(n)$ time. Thus the following theorem can be obtained.

Theorem 3. 2: For a score sequence of a leveled tournament tree $S=(s_1, s_2, \cdots, s_n)$, a leveled tournament tree with S can be obtained in $O(n)$ time.

3. 2 Optimal Condition and Optimum Condition

Let T be a leveled tournament tree with n leaves. For any leaf v of T, $d(r, v) \leqq \lceil \log_2 n \rceil$ holds if and only if T is *optimal*, where r is a root of T and $\lceil \log_2 n \rceil$ is the least integer which is equal or greater than $\log_2 n$. (If T is optimal then T must have a leaf v with $d(r, v)=\lceil \log_2 n \rceil$.) Especially, for any leaf v of T, $d(r, v)=\lceil \log_2 n \rceil$ or $d(r, v)=\lfloor \log_2 n \rfloor$ holds if and only if T is *optimum*, where $\lfloor \log_2 n \rfloor$ is the greatest integer which is equal or less than $\log_2 n$. Then I consider the *optimal* (*optimum, respectively*) *condition*: Given a score sequence of a leveled tournament tree $S=(s_1, s_2, \cdots, s_n)$, determine whether S is optimal (optimum) (i. e., there is a set of n elements $W=(w_1, w_2, \cdots, w_n)$ such that $\mathrm{win}(w_j)=s_j$ for each $j=1, 2, \cdots, n$, and there is an optimal (optimum) leveled tournament tree S and n leaves).

First I can obtain the following proopsition.

Proposition 3. 2: Let $S=(s_1, s_2, \cdots, s_n)$ be a score sequence of a leveled tournament tree. Assume $n=2^r$ for some integer r. Then S is optimal (optimum) if and only if $s_j=r-\lceil \log_2 j \rceil$ holds for each $j=1, 2, \cdots, n$. □

It is easy to prove the proposition, and I will omit a proof here.

Next I consider the $2^r < n < 2^{r+1}$ case for some integer r. Then I can obtain the following proposition.

Proposition 3. 3: Assume $2^r < n < 2^{r+1}$ for some integer r. Let $S=(s_1, s_2, \cdots, s_n)$ be a score sequence of a leveled tournament tree. Then S is optimum if and only if

$$s_j=\begin{cases} r+1-\lceil \log_2 j \rceil \text{ or } r-\lceil \log_2 j \rceil & \text{if } 1 \leqq j \leqq 2^r, \\ 0 & \text{if } 2^r+1 \leqq j \leqq n, \end{cases}$$

and $\sum_{j=1}^{k} s_j - \sum_{j=1}^{k}(r-\lceil \log_2 j \rceil)=n-2^r$ holds, where $k=2^r$.

Proof: Let $y=2^r$, let $U(y)=(u_1, u_2, \cdots, u_y)$ be a score sequence of a leveled tournament tree and let $X=(x_1, x_2, \cdots, x_{y+1})$ be defined by using $q=\min\{j \mid$

$s_j \neq u_j, 1 \leqq j \leqq y\}$ as follows,

$$x_j = \begin{cases} u_j + 1 & \text{if } j = q \text{ and } 1 \leqq j \leqq y, \\ u_j & \text{if } j \neq q \text{ and } 1 \leqq j \leqq y, \\ 0 & \text{if } j = y + 1. \end{cases}$$

Then it is clear that $U(y)$ is optimum if and only if X is optimum. Furthermore, by Proposition 3. 2, $U(y)$ is optimum if and only if $u_j = r - \lceil \log_2 j \rceil$ holds for each $j = 1, 2, \cdots, y$. Thus X is optimum if and only if

$$x_j = \begin{cases} r + 1 - \lceil \log_2 j \rceil \text{ or } r - \lceil \log_2 j \rceil & \text{if } 1 \leqq j \leqq y, \\ 0 & \text{if } j = y + 1, \end{cases}$$

and $\sum_{j=1}^{k} x_j - \sum_{j=1}^{k} (r - \lceil \log_2 j \rceil) = 1$ holds.

By setting $y := y + 1$ and $U(y) := X$, and repeating the argument above, I can finally obtain that $U(n) = X$ is optimum if and only if

$$u_j = \begin{cases} r + 1 - \lceil \log_2 j \rceil \text{ or } r - \lceil \log_2 j \rceil & \text{if } 1 \leqq j \leqq 2^r, \\ 0 & \text{if } 2^r + 1 \leqq j \leqq n, \end{cases}$$

and $\sum_{j=1}^{k} u_j - \sum_{j=1}^{k} (r - \lceil \log_2 j \rceil) = n - 2^r$ holds, where $k = 2^r$. Then $u_j = s_j$ holds for each $j = 1, 2, \cdots, n$. □

Furthermore I can obtain the following proposition.

Proposition 3. 4: Assume $2^r < n < 2^{r+1}$ for some integer r. Let $S = (s_1, s_2, \cdots, s_n)$ be a score sequence of a leveled tournament tree and let $U = (u_1, u_2, \cdots, u_k)$ be defined by the following algorithm, where $k = 2^{r+1}$. Then S is optimal if and only if U is optimum.

Algorithm DLOTT.

Begin

1. $u_j := -1$ for each $j = 1, 2, \cdots, n$; q:=1;
2. For $j := 2$ to n do begin
   (a) while $u_q \neq -1$ do begin
      for $p := u_q$ downto 1 do begin
         $b := 2^{r+1-p} + q$; $u_b := r + 1 - \lceil \log_2 b \rceil$ end:
      $q := q + 1$ end;
   (b) If $s_j \geqq r + 1 - \lceil \log_2 q \rceil$ then $u_q := s_j$
      else do begin
      $u_q := r + 1 - \lceil \log_2 q \rceil$;
      for $b := u_q$ downto 1 do begin
         $b := 2^{r+1-p} + q$; $u_b := r + 1 - \lceil \log_2 b \rceil$
      end end:
   (c) $q := q + 1$; If $q > k$ then halt end

End.

Example 3. 4: Let $S = (s_1, s_2, \cdots, s_n) = (2, 3, 1, 2, 1, 0, 1, 0, 0, 0, 0)$ and $S' = (s'_1, s'_2, \cdots, s'_n) = (2, 3, 1, 2, 1, 0, 0, 1, 0, 0, 0)$ be rooted tree sequences. Then $k = 16$, and $U = (u_1, u_2, \cdots, u_k) = (4, 3, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)$ and $U' = (u'_1, u'_2 \ldots, u'_k) = (4,3,2,2,1,1,1,1,0,1,0,0,0,0,0,0)$ are sequences defined from S and S' respectively, by Algorithm DLOTT. $u_1 = s_1 + 2$, $u_2 = s_2$, $u_3 = s_3 + 1$, $u_4 = s_4$, $u_6 = s_5$, $u_7 = s_6 + 1$, $u_8 = s_7$, $u_{10} = s_8$, $u_{12} = s_9$, $u_{14} = s_{10}$, $u_{16} = s_{11}$, and u_5, u_9, u_{11}, u_{13}, and u_{15} are new elements. Furthermore, $u'_1 = s'_1 + 1$, $u'_2 = s'_2$, $u'_3 = s'_3 + 1$, $u'_4 = s'_4$, $u'_6 = s'_5$, $u'_7 = s'_6 + 1$, $u'_8 = s'_7 + 1$, $u'_{10} = s'_8$, $u'_{12} = s'_9$, $u'_{14} = s'_{10}$, and u'_5, u'_9, u'_{11}, u'_{13}, u'_{15} and u'_{16} are new elements, however U' does not have an element corresponding to s'_{11}. U is optimum and U' is not optimum. Hence S is optimal and S' is not optimal. T_1 (see Fig. 3. 6) is a leveled tournament tree with S and T_2 (see Fig. 3. 7) is such a tree with U. □

It is also easy to prove Proposition 3. 4, and I

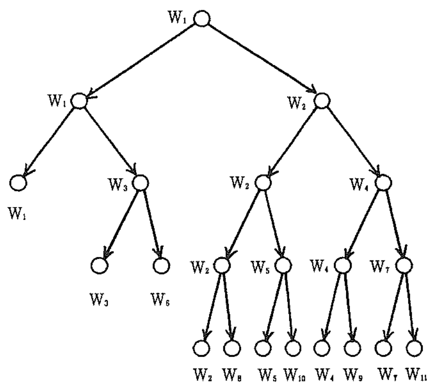

Fig. 3. 6.

will omit a proof here. Since $2^r < n < 2^{r+1}$, $2^{r+1} < 2n$ holds. Thus Algorithm DLOTT requires $O(n)$ time and it can be determined in $O(n)$ time

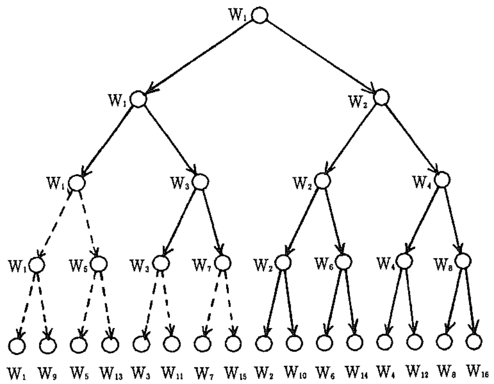

Fig. 3. 7.

whether U is optimum or not.

Hence, based on Proposion 3, 2 through 3. 4, the following theorem can be obtained easily.

Theorem 3.3: For a sequence of a leveled tournament tree $S=(s_1, s_2, \cdots, s_n)$, it can be determined in $O(n)$ time whether S is optimum or not, and it can be determined in $O(n)$ time whether S is optimal or not.

Furthermore, it is easy to see that I can construct an optimal (optimum) leveled tournament tree with S in $O(n)$ time by using Algorithm CLTT, if S is optimal (optimum).

4. Concluding Reamarks

I have considered variations of the rooted tree sequence problem and give linear time algorithms. In this paper, I have treated a sequence of non-negative integers.

In the following, assume that $S=(s_1, s_2, \cdots, s_n)$ is a sequence of positive integers. Then I can easily obtain a rooted tree with $m+1$ vertices and a leveled rooted tree with $m+1$ vertices from S respectively, where $m=\sum_{j=1}^{n} s_j$ and assume $\deg^+(v_j)=0$ and $\deg^-(v_j)=1$ for each $j=n+1, n+2, \cdots, m+1$. Such trees can be constructed in $O(m)$ time. Furthermore I can easily obtain a leveled tournament tree with $m+1$ leaves such that $\mathrm{win}(w_j)=s_j$ for each $j=1, 2, \cdots, n$, and $\mathrm{win}(w_j)=0$ for each $j=n+1, n+2, \cdots, m+1$. Such a tree also can be constructed in $O(m)$ time. For the optimum condition, I can obtain the following proposition.

Proposition 4. 1: Assume $2^{r-1} < m+1 \leqq 2^r$ for some integer r. Let $S=(s_1, s_2, \cdots, s_n)$ be a positive integer sequence. Then S is optimum if and only if $s_j = r-1-\lceil \log_2 j \rceil$ or $s_j = r-\lceil \log_2 j \rceil$ for each $j=1, 2, \cdots, n$, and $\sum_{j=1}^{n} s_j - \sum_{j=1}^{n}(r-1-\lceil \log_2 j \rceil) = m+1-2^{r-1}$ hold. □

For the optimal condition, the result like Proposition 3. 4 can be obtained.

I want to consider an efficient algorithm for generating a leveled rooted tree sequence for further investigation.

References

[1] Aho, A. V., Hopcroft, J. E. and Ullman, J. D., *"The Design and Analysis of Computer Algorithms,"* Addison-Wesley, 1974.

[2] Menon, V. V., *On the existence of trees with given degrees,* Sankhya, Series A, 26 (1964), pp63-68.

[3] Chen, Y.K., *"Applied Graph Theory* (Second revised edition)," North-Holland, 1976.

[4] Behzard, M., Chartrand, G. and Foster, L. L.-, *"Graphs and Digraphs,"* Prindle, Weber and Schmidt, 1979.

[5] Harary, F., *"Graph Theory,"* Addison Wesley, 1969.

[6] Havel, V., *A remark on the existence of finite graphs* (Hungarian), Casopis Pest., Mat., vol.80, pp477-480, 1955.

[7] Erdös, P. and Gallai, T., *Graphs with prescribed degrees of vertices* (Hungarian), Mat. Lapok, vol.11, pp264-274, 1960.

[8] Hakimi, S. L., *On the realizability of a set of integers as degrees of the vertices of a graph,* J. SIAM Appl. Math. vol.10, pp496-506, 1962.

[9] Takahashi, M., Imai, K. and Asano, T., *Graphical Degree Sequence Problems*, IEICE Trans. Fundamentals., vol.E77-A., No.3 (March), pp546-552, 1994.

[10] Asano, T., *An O(nloglogn) Time Algorithm for Constructing a Graph of Maximum Connectivity with Prescribed Degrees,* Technical Report, ISETR 93-02, Department of Information and System Engineering, Chuo University, 1993.

[11] Landau, H. G., *On dominance relations and the structure of animal societies,* III., *The condition for a score structure,* Bulletin for Mathematical Biophysics, vol.15, pp143-148, 1953.