アルゴリズム 36－5
（1993．11．25）

与えられた次数列を連結度最大のグラフで実現する $O(n \log\log n)$ 時間アルゴリズム

浅野孝夫
中央大学理工学部情報工学科

非負整数の列 $D=(d_1,d_2,...,d_n)$ を実現するグラフが存在するとき、D をグラフ次数列という。グラフ次数列 D に対して、D を実現する k-連結グラフは存在するが、$(k+1)$-連結グラフは存在しないとき、Dの連結度$\kappa(D)$ は kであるという。本論文では、与えられたグラフ次数列 Dに対して、Dを$\kappa(D)$-連結グラフで実現する $O(n \log\log n)$ の手間のアルゴリズムを与える。

An $O(n \log\log n)$ Time Algorithm for Constructing a Graph of Maximum Connectivity with Prescribed Degrees

Takao Asano
Department of Information and System Engineering
Chuo University, Bunkyo-ku, Tokyo 112, Japan

A sequence of nonnegative integers $D=(d_1,d_2,...,d_n)$ is *graphical* if there is a graph with degree sequence D. The connectivity $\kappa(D)$ of a graphical sequence D is defined to be the maximum integer k such that there is a k-connected graph with degree sequence D. In this paper, we present an $O(n \log\log n)$ time algorithm, for a given graphical sequence D, to construct a $\kappa(D)$-connected graph with degree sequence D.

1 Introduction

Graphs considered in this paper are all simple. A sequence of nonnegative integers $D = (d_1, d_2, ..., d_n)$ is *graphical* if there is a graph G with degree sequence D (i.e., d_i is the degree of vertex v_i of G for each $i = 1, 2, ..., n$.) A graphical sequence $D = (d_1, d_2, ..., d_n)$ is *k-connected* if there is a k-connected graph with degree sequence D. If a k-connected graphical sequence D is not $(k+1)$-connected then its connectivity $\kappa(D)$ is defined to be k. The (k-connected) graphical sequence problem is: Given a sequence D of nonnegative integers, determine whether it is (k-connected) graphical or not. The graphical sequence problem is one of the most fundamental problems in graph theory [1, 5, 9] and was first considered by Havel [6] and then considered by Erdös and Gallai [3] and Hakimi [4]. They gave simple characterizations which lead to efficient algorithms. The k-connected graphical sequence problems were considered by Berge [1]($k = 1, 2$), by Rao and Ramachandra Rao [7] ($k = 3$) and by Wang and Kleitman [12] ($k \geq 2$). They gave characterizations for a graphical sequence to be k-connected.

In this paper, we consider the following more general problem: Given a graphical sequence D, determine its connectivity $\kappa(D)$ and construct a $\kappa(D)$-connected graph with degree sequence D. We present efficient algorithms including an $O(n)$ time algorithm to determine the connectivity of a given graphical sequence $D = (d_1, d_2, ..., d_n)$ and an $O(m)$ time algorithm to construct a $\kappa(D)$-connected graph with degree sequence D ($m = \sum_{i=1}^{n} d_i/2$).

The $O(m)$ time algorithm for constructing a $\kappa(D)$-connected graph is concerned with the graphs represented explicitly. In some applications, however, implicitly represented graphs are satisfactory and faster algorithms may be required. For this reason, we also consider an implicit representation of graphs and present a faster $O(n \log\log n)$ time algorithm, for a given graphical sequence D, to construct a $\kappa(D)$-connected graph with degree sequence D

2 Connectivity of a Graphical Sequence

In this section, we present an efficient algorithm to determine the connectivity of a graphical sequence. We first recall the previous results. Havel [6] and Hakimi [4] gave Proposition 1(a) independently and Erdös and Gallai [3] gave Proposition 1(b) below [1, 5, 9].

Proposition 1. Let $D = (d_1, d_2, ..., d_n)$ be a sequence of integers with $n > d_1 \geq d_2 \geq ... \geq d_n \geq 0$ and let $D' = (d'_2, d'_3, ..., d'_n)$ be a sequence of integers obtained from D by setting $d'_i = d_i - 1$ $(i = 2, ..., d_1 + 1)$ and $d'_j = d_j$ $(j = d_1 + 2, ..., n)$. Then the following hold.
(a) D is graphical if and only if D' is graphical.
(b) D is graphical if and only if $\sum_{j=1}^{i} d_j \leq i(i-1) + \sum_{j=i+1}^{n} \min\{i, d_j\}$ for each $i = 1, 2, ..., n$.

For a sequence of integers $D = (d_1, d_2, ..., d_n)$ with $n > d_1 \geq d_2 \geq ... \geq d_n \geq 0$, if $\sum_{i=1}^{n} d_i$ is odd then D is not graphical. Thus, we assume throughout this paper that $\sum_{i=1}^{n} d_i$ is even and $m = \sum_{i=1}^{n} d_i/2$. For a graphical sequence D, it is trivial to obtain a graph G with degree sequence D in $O(n^2)$ time based on Proposition 1(a). A drawback to use Proposition 1(a) is that $d'_1 \geq d'_2 \geq ... \geq d'_{n-1}$ does not always hold. Thus we have to sort again to use Proposition 1(a) recursively. If the proposition is modified to avoid sorting in the following way, then the time complexity can be reduced to $O(m)$ [8].

Proposition 2. Let $D = (d_1, d_2, ..., d_n)$ be a sequence of integers with $n > d_1 \geq d_2 \geq ... \geq d_n > 0$. Let $h = d_n$, $x = \min\{j | d_j = d_h\}$, $y = \max\{j | j \leq n-1, d_j = d_h\}$ and

$$c_i = \begin{cases} d_i - 1 & \text{if } 1 \leq i \leq x-1 \text{ or } y-h+x \leq i \leq y, \\ d_i & \text{if } x \leq i \leq y-h+x-1 \text{ or } y+1 \leq i \leq n-1. \end{cases}$$

Then $c_1 \geq c_2 \geq ... \geq c_{n-1} \geq 0$ and D is graphical if and only if $C = (c_1, c_2, ..., c_{n-1})$ is graphical. Similarly, if we let $h' = d_1 + 1$, $x' = \min\{j | 2 \leq j, d_j = d_{h'}\}$, $y' = \max\{j | d_j = d_{h'}\}$ and

$$d'_i = \begin{cases} d_i - 1 & \text{if } 2 \le i \le x' - 1 \text{ or } y' - h' + x' \le i \le y', \\ d_i & \text{if } x' \le i \le y' - h' + x' - 1 \text{ or } y' + 1 \le i \le n, \end{cases}$$

then $d'_2 \ge d'_3 \ge ... \ge d'_n \ge 0$ and D is graphical if and only if $D' = (d'_2, d'_3, ..., d'_n)$ is graphical.

The above results are all without connectivity requirement. For k-connectivity requirement, the following Proposition 3 is in Berge [1] ($k = 1, 2$) and in Wang and Kleitman [12] ($k \ge 2$).

Proposition 3. Let $D = (d_1, d_2, ..., d_n)$ be a graphical sequence with $d_1 \ge d_2 \ge \cdots \ge d_n$. Then, for a positive integer k, D is k-connected if and only if (i) $d_n \ge k$ and (ii) $\sum_{j=1}^{n} d_j \ge 2(n + \sum_{j=1}^{k-1} d_j - k(k-1)/2 - 1)$.

Based on Proposition 3, we can determine whether a graphical sequence $D = (d_1, d_2, ..., d_n)$ is k-connected or not in $O(n)$ time. We can also compute the connectivity $\kappa(D)$ in $O(n)$ time as follows:

function $\kappa(D)$:integer; **begin**
$k := 1$; $a_0 := 0$; **for** $i := 1$ **to** n **do** $a_i := a_{i-1} + d_i$;
while $(d_n \ge k)$ **and** $(a_n \ge 2n + 2a_{k-1} - k(k-1) - 2)$ **do** $k := k + 1$;
$\kappa(D) := k - 1$ **end**;

Note that it can be determined whether a given sequence of integers D is graphical or not in $O(n)$ time based on Proposition 1(b) [8]. Thus the following theorem is obtained.

Theorem 1. For a sequence of integers $D = (d_1, ..., d_n)$, it can be determined whether D is graphical or not in $O(n)$ time, and if so, the connectivity $\kappa(D)$ can be computed in $O(n)$ time.

3 An $O(m)$ Time Algorithm

In this section, we will present an $O(m)$ time algorithm, for a graphical sequence D, to construct a $\kappa(D)$-connected graph with degree sequence D. Since the algorithm is based on a proof of Proposition 3 we first describe the proof, although it is almost the same as the proof given by Wang and Kleitman [12]. Note that their proof is confined to $k \ge 2$. Our proof is valid even for $k = 1$ and this is necessary since our algorithm treats a graphical sequence D with $\kappa(D) \ge 0$.

Proof of Proposition 3. **(Necessity)** Let G be a k-connected graph with degree sequence D and $U = \{v_1, v_2, ..., v_{k-1}\}$. Then (i) is trivially true and $G - U$ is connected. Thus, the number of edges in $G - U$ is greater than or equal to $-1+$ the number of vertices in $G - U$. Since the number of edges in $G - U$ is at most $\sum_{j=1}^{n} d_j/2 - (\sum_{j=1}^{k-1} d_j - (k-1)(k-2)/2)$, we have $\sum_{j=1}^{n} d_j - 2(\sum_{j=1}^{k-1} d_j - (k-1)(k-2)/2) \ge 2(n - (k-1) - 1)$ and (ii).

(Sufficiency) We prove the sufficiency by induction on n. Since D is graphical and satisfies (i) and (ii), we have $k \le d_n \le d_1 < n$ and thus $n \ge k+1$. If $n = k+1$, then $d_1 = \cdots = d_n = n-1 = k$ and the complete graph K_{k+1} is k-connected and has degree sequence D. Thus, we now assume that it is true for all integers less than $n \ge k + 2$. We divide into two cases: ($d_1 = n - 1$ and $k > 1$) and ($d_1 \le n - 2$ or $k = 1$).

Case I: $d_1 = n - 1$ **and** $k > 1$. Let $D' = (d'_2, ..., d'_n)$ be the same as in Proposition 2 ($d'_j = d_j - 1$ for each $j = 2, ..., n$). Since D' is graphical by Proposition 2 and satisfies (i) and (ii) of Proposition 3 for $k - 1$, D' is $(k-1)$-connected and there is a $(k-1)$-connected graph G' with degree sequence D' by induction hypothesis. The graph G obtained from G' by adding the edges joining new vertex v_1 and all vertices of G' is k-connected and has degree sequence D.

Case II: $d_1 \le n - 2$ **or** $k = 1$. Let $C = (c_1, c_2, ..., c_{n-1})$ be the same as in Proposition 2. If $k = 1$ and $d_n = 1$ then $d_1 \ge 2$ by (ii) and we have $c_{n-1} = d_{n-1} \ge 1$ and $\sum_{i=1}^{n-1} c_i = \sum_{i=1}^{n} d_i - 2d_n \ge 2(n-2)$. If $k = 1$ and $d_n \ge 2$ then $c_i \ge d_i - 1 \ge 1$ for all i and $\sum_{i=1}^{n-1} c_i \ge \sum_{i=1}^{n-2} d_i \ge 2(n-2)$. Thus, we have two cases (II-A): $c_{n-1} \ge k$ and (II-B): $c_{n-1} = k - 1$ and $k > 1$.

Case II-A: $c_{n-1} \ge k$. We have the following two cases II-A-1 and II-A-2:

Case II-A-1: $\sum_{j=1}^{n-1} c_j = \sum_{j=1}^{n} d_j - 2d_n \geq 2(n + \sum_{j=1}^{k-1} c_j - k(k-1)/2 - 2)$. In this case, C is graphical by Proposition 2 and k-connected by induction hypothesis, and there is a k-connected graph G' with degree sequence C. The graph G obtained by adding d_n edges joining vertex v_n and the vertices v_i of G' with $c_i = d_i - 1$ is k-connected and has degree sequence D

Case II-A-2: $\sum_{j=1}^{n-1} c_j < 2(n + \sum_{j=1}^{k-1} c_j - k(k-1)/2 - 2)$. Note that $k > 1$ and $d_1 \leq n-2$ by the above argument. We will prove that this case cannot occur.

Let h be an integer such that $c_{n-1} \geq h \geq k$. Let $\alpha = c_{k-1} - c_k$. Clearly $0 \leq \alpha \leq n-k-2$ since $n-2 \geq d_1 \geq d_n \geq k$. Thus, we have $(k-2)(n-2) \geq \sum_{j=1}^{k-2} c_j \geq -\alpha + \sum_{j=k+1}^{n-1} c_j - 2n + k(k-1) + 6$ $\geq h(n-k-1) - 2n + k(k-1) + 6 - \alpha = (k-2)(n-2) + (h-k)(n-k-1) + 2 - \alpha$ since $\sum_{j=1}^{n-1} c_j \leq 2(n + \sum_{j=1}^{k-1} c_j - k(k-1)/2 - 3)$. This implies that $c_{n-1} = h = k$ and $\alpha \geq 2$. Furthermore, $c_{n-1} = k$ implies $d_n \leq d_{n-1} \leq k+1$ and $\alpha \geq 2$ implies $d_{k-1} \geq d_k + 2$ and that $c_i = d_i - 1$ for all $i \leq k-1$. Thus, we have $(k-1)(n-2) \geq \sum_{j=1}^{k-1} d_j = \sum_{j=1}^{k-1} c_j + k - 1$ $\geq \sum_{j=k}^{n-1} c_j - 2n + k(k-1) + 6 + k - 1 = \sum_{j=k}^{n-1} d_j - (d_n - (k-1)) - 2n + k(k-1) + k + 5$. This implies that if $d_n = k+1$ then we have $(k-1)(n-2) \geq \sum_{j=k}^{n-1} d_j - (d_n - (k-1)) - 2n + k(k-1) + k + 5$ $\geq (k+1)(n-k) - 2n + k(k-1) + k + 3 = (k-1)(n-2) + k + 1$ and that if $d_n = k$ then we have by (ii) $\sum_{j=1}^{k-1} d_j \geq \sum_{j=k}^{n-1} d_j - (d_n - (k-1)) - 2n + k(k-1) + k + 5 \geq \sum_{j=1}^{k-1} d_j - 2 - d_n - (d_n - (k-1)) + k + 5$ $= \sum_{j=1}^{k-1} d_j + 2$. Thus, we have a contradiction in either case.

Case II-B: $c_{n-1} = k-1$ **and** $k > 1$. In this case, $d_n = d_{n-1} = k$ and $d_k = \cdots = d_n = k$ by the definition of C, since if $d_k > d_{n-1}$ then c_{n-1} would be $c_{n-1} = d_{n-1} = k$. We will give an algorithm II-B to construct a graph $G(D)$ with degree sequence D based on the algorithm proposed by Wang and Kleitman [12]. We divide into two cases: $d_1 \leq k+1$ and $d_1 \geq k+2$.

Algorithm II-B.

Case II-B-1: $d_1 \leq k+1$. We first consider the case $d_1 = k$. Then $d_i = k$ for all $i = 1, 2, ..., n$. Let $G(p, 2b)$ $(p > 2b)$ be the graph with vertex set $\{0, 1, ..., p-1\}$ and edge set $\{(i,j) | 0 < |(j-i) \bmod p| \leq b, 0 \leq i, j \leq p-1\}$. Here we assume $(j-i) \bmod p$ takes a value in $[-\lfloor p/2 \rfloor, \lfloor (p-1)/2 \rfloor]$. If p is even then let $H(2a, 2b+1)$ $(p = 2a > 2b+1)$ be the graph obtained from $G(2a, 2b)$ by adding edges $\{(i, i+a) | i = 0, 1, ..., a-1\}$. Note that $G(p, 2b)$ is $2b$-regular and $2b$-connected and also that $H(2a, 2b+1)$ is $(2b+1)$-regular and $(2b+1)$-connected. Thus, $G(n,k)$ (if k is even) or $H(n,k)$ (if k is odd) is a required graph $G(D)$ $(v_i = i-1)$.

Next we consider the case $d_1 = k+1$. Let f be the number such that $d_f = k+1$ and $d_{f+1} = k$. If k is even then f is even and the graph $G^+(n,k,f)$ obtained from $G(n,k)$ by adding edges $\{(i, i + \lfloor n/2 \rfloor) | i = 0, 1, ..., f/2 - 1\}$ is a required graph $G(D)$. If k is odd, then the graph $G^-(n, k+1, f)$ obtained from $G(n, k+1)$ by deleting edges $\{(2i, 2i+1) | i = 0, 1, ..., (n-f)/2 - 1\}$ is a required graph $G(D)$. These graphs $G^+(n,k,f)$, $G^-(n,k+1,f)$ have the following property:
(*) For each vertex subset U of G with $|U| \leq k$, if $G - U$ is disconnected then no connected component of $G - U$ consists of only vertices of degree $k+1$ in G.

Case II-B-2: $d_1 \geq k+2$. In this case let $D' = (d'_2, d'_3, ..., d'_n)$ be the sequence as in Proposition 2. We will show that $\sum_{j=2}^{n} d'_j \geq 2(n + \sum_{j=2}^{k-1} d'_j - (k-1)(k-2)/2 - 2)$. Suppose that $\sum_{j=2}^{n} d'_j = \sum_{j=1}^{n} d_j - 2d_1 < 2(n + \sum_{j=2}^{k-1} d'_j - (k-1)(k-2)/2 - 2)$. Let $\alpha = d'_{k-1} - d'_k$. Then we have $(k-3)(n-2) \geq \sum_{j=2}^{k-2} d'_j \geq -\alpha + \sum_{j=k+1}^{n} d'_j - 2n + (k-1)(k-2) + 6 \geq (k-1)(n-k) -$ $2n + (k-1)(k-2) + 6 - \alpha = (k-3)(n-2) + 2 - \alpha$. This implies $\alpha \geq 2$, $d_{k-1} \geq d_k + 2$ and that $d'_i = d_i - 1$ for all $2 \leq i \leq k-1$. However, this is a contradiction since $\sum_{j=2}^{k-1} d_j = \sum_{j=2}^{k-1} d'_j + k - 2$ $\geq \sum_{j=k}^{n} d'_j - 2n + (k-1)(k-2) + 6 + k - 2 = \sum_{j=k}^{n} d_j - (d_1 - (k-2)) - 2n + (k-1)(k-2) + k + 4$ $\geq \sum_{j=1}^{k-1} d_j - k(k-1) - (d_1 - (k-2)) + (k-1)(k-2) + k + 2 = \sum_{j=2}^{k-1} d_j + 2$.

Thus, D' satisfies (i) and (ii) for $k-1$ and D' is graphical and $(k-1)$-connected. The graph $G(D')$ obtained by the algorithm II-B can be shown to be $(k-1)$-connected and have degree sequence D'. Furthermore, $G(D')$ satisfies the property (*) above (with $k := k-1$). Thus, the graph $G(D)$ obtained from $G(D')$ by adding edges (v_1, v_i) between vertex v_1 and vertices v_i with $d'_i = d_i - 1$ is k-connected and has degree sequence D since only vertices of degree k may not be joined to v_1. $G(D)$ can be shown to satisfy the property (*).

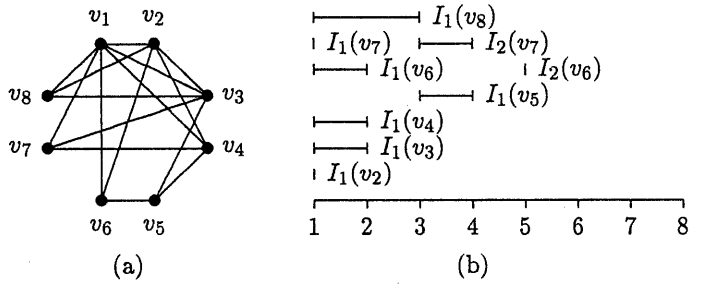
Case II-B-2-a. If $d'_k = k-1$ and $k-1 > 1$ then we have the case II-B for D' and call Algorithm II-B recursively.

Case II-B-2-b. If $d'_k = k$ or $d'_k = k-1 = 1$ then let $C' = (c'_2, c'_3, ..., c'_{n-1})$ be the sequence as in Proposition 2 (by regarding D' as D) and add edges (v_i, v_n) between vertex v_n and vertices v_i with $c'_i = d'_i - 1$. We repeat this (by regarding C' as D') until $d'_k = k-1$ or $d'_1 = 0$. Then we have the Case II-B for new D' and call Algorithm II-B recursively or $d_1 = 0$.

Thus we complete the proof of Proposition 3.

Since the proof of Proposition 3 is constructive, we can obtain an $O(m)$ time algorithm, for a graphical sequence D, to construct a $\kappa(D)$-connected graph with degree sequence D. In the algorithm below, L is initialized $L = \{j | d_j > d_{j+1}, j = 1, 2, ..., n-1\} \cup \{0, n\}$ and represented by a doubly-linked list with $pre[j] < j < suc[j]$ for each $j \in L$, where $pre[j]$ and $suc[j]$ denote the previous element and the next element of $j \in L$ (if $pre[j]$ ($suc[j]$) does not exist then the inequality $pre[j] < j$ ($j < suc[j]$) should be neglected). Note that $C = (c_1, c_2, ..., c_g)$ is initialized $C = D$ and then maintained to satisfy $c_\ell \geq c_2 \geq ... \geq c_g$. L is also maintained to satisfy $L = \{j | c_j > c_{j+1}\} \cup \{0, g\}$. Thus, $c_{pre[j]+1} = c_{pre[j]+2} = \cdots = c_j > c_{j+1} = c_{j+2} = \cdots = c_{suc[j]}$ for each $j \in L$ with $\ell \leq j < g$. $find(h)$ returns a maximum integer j with $c_j = c_h$ (thus, $c_j > c_{j+1}$).

Algorithm CG; {**comment** For a graphical sequence $D = (d_1, ..., d_n)$ with $d_1 \geq ... \geq d_n$, this constructs a $\kappa(D)$-connected graph $G(D)$ with degree sequence D}
begin
  $k := \kappa(D)$; $L := \{0, n\}$; **for** $i := 1$ **to** n **do** $c_i := d_i$;
  for $i := 1$ **to** $n-1$ **do if** $d_i > d_{i+1}$ **then** insert i into L;
  construct1_graph$(1, n)$
end.
procedure construct1_graph(ℓ, g);
{**comment** this constructs a $(k - \ell + 1)$-connected graph with degree sequence $(c_\ell, ..., c_g)$}
begin {**comment** $c_g \geq k - \ell + 1$}
  if $(c_\ell = g - \ell)$ **and** $(k > \ell)$ **then begin** {**comment** Case I}
    for $i := \ell + 1$ **to** g **do begin** add edge (v_ℓ, v_i); $c_i := c_i - 1$ **end**;
    delete ℓ from L; **if** $\ell + 1 < g$ **then** construct1_graph$(\ell + 1, g)$ **end**
  else
    if $c_k > k - \ell + 1$ **then begin** {**comment** Case II-A}
      if $g - 1$ is not in L **then** insert $g - 1$ into L; delete g from L;
      $j := find(c_g + \ell - 1)$; {**comment** $g - 1 \geq j \geq c_g + \ell - 1 > pre[j]$}
      create_edge(g); **if** $\ell < g - 1$ **then** construct1_graph$(\ell, g-1)$ **end**
    else {**comment** Case II-B: $j = g - 1$, $c_k = \cdots = c_g = k - \ell + 1$}
      construct the $(k - \ell + 1)$-connected graph $G(c_\ell, ..., c_g)$ by Algorithm II-B
end;
procedure create_edge(f);
begin
  if $j = c_f + \ell - 1$ **then for** $i := \ell$ **to** j **do begin** add edge (v_f, v_i); $c_i := c_i - 1$ **end**
  else begin
    $j_{new} := j - (c_f + \ell - 1) + pre[j]$; {**comment** $j_{new} - pre[j] = j - c_f - \ell + 1 > 0$}
    if $pre[j] \neq 0$ **then begin**
      for $i := \ell$ **to** $pre[j]$ **do begin** add edge (v_f, v_i); $c_i := c_i - 1$ **end**;
      if $(c_{pre[j]} = c_{pre[j]+1})$ **then** delete $pre[j]$ from L **end**;
    for $i := j_{new} + 1$ **to** j **do begin** add edge (v_f, v_i); $c_i := c_i - 1$ **end**;
    insert j_{new} into L; {**comment** j_{new} is inserted before j} **end**;
  if $(j < g - 1)$ **and** $(c_j = c_{j+1})$ **then** delete j from L
end;



(a) (b)

Fig. 1. (a) Graph G with degree sequence$D = (6,5,5,4,3,3,3,3)$
(b) An implicit representation of G by a set of intervals.

As noted in the algorithm, procedure construct1_graph(ℓ, g) constructs a $(k-\ell+1)$-connected graph with degree sequence $(c_\ell, c_{\ell+1}, ..., c_g)$, which can be easily observed if we consider the correspondence between procedure construct1_graph(ℓ, g) and the the proof of Proposition 3. Thus, Algorithm CG correctly constructs a $\kappa(D)$-connected graph with degree sequence D. The time complexity is clearly $O(m)$. Thus, we have the following theorem.

Theorem 2. For a graphical sequence $D = (d_1, d_2, ..., d_n)$ with $d_1 \geq d_2 \geq \cdots \geq d_n$, a $\kappa(D)$-connected graph with degree sequence D can be constructed in $O(m)$ time $(m = \sum_{i=1}^{n} d_i/2)$.

4 An $O(n \log\log n)$ Time Algorithm

In this section we present an $O(n \log\log n)$ time algorithm for constructing a $\kappa(D)$-connected graph represented implicitly for a given graphical sequence D. Before giving the algorithm, we consider an implicit representation of a graph by using an example.

Let $D = (6,5,5,4,3,3,3,3)$. Then D is 3-connected graphical. Based on Proposition 2 (not on Algorithm CG), we have the following sequence $(D_0 = D, D_1 = C)$:

$$D_0 = (6,5,5,4,3,3,3,3),\ D_1 = (5,4,4,4,3,3,3),\ D_2 = (4,4,3,3,3,3),$$
$$D_3 = (3,3,3,3,2),\ D_4 = (3,3,2,2),\ D_5 = (2,2,2),\ D_6 = (1,1).$$

The graph obtained in this way is shown in Fig.1(a). Note that we can represent vertices adjacent to vertex v_8 by an interval $I_1(v_8) = [1,3]$ when we obtain D_1 from D_0. Similarly, vertices adjacent to vertex v_7 can be represented by intervals $I_1(v_7) = [1,1]$, $I_2(v_7) = [3,4]$ when we obtain D_2 from D_1. Repeating this, we have a set of intervals shown in Fig.1(b). Thus,

$$E(v_i) = \{(v_j, v_i) | j < i, j \in I_1(v_i) \cup I_2(v_i)\} \cup \{(v_j, v_i) | j > i, i \in I_1(v_j) \cup I_2(v_j)\}$$

is the set of edges incident on vertex v_i in G ($I_2(v_i)$ and $I_2(v_j)$ may be empty).

A graph G represented by a set of intervals in this way is called *implicitly represented.* Note that, if necessary, the edge set $E(v_i)$ incident on vertex v_i can be obtained efficiently (in $O(|E(v_i)| + \log n)$ time) by solving the 1-dimensional range search and 1-dimensional point enclosure problems in computational geometry [2].

Degree sequence D can also be represented by a set of intervals with weights as follows. Let $\mathcal{J}(D)$ be the set of intervals obtained by partitioning the underlying set $[1,n]$ using the elements i with $d_i > d_{i+1}$. Thus, for each interval $J \in \mathcal{J}(D)$, $d_i = d_j$ if $i, j \in J$. We define the weight $w(J)$ of J to be d_i for any $i \in J$. For example, corresponding to $D = (6,5,5,4,3,3,3,3)$ we have

$$\mathcal{J}(D) = \{[1,1],[2,3],[4,4],[5,8]\}$$

with $w([1,1]) = 6, w([2,3]) = 5, w([4,4]) = 4, w([5,8]) = 3$. If $D = D_0$ is modified to $D_1 = (5,4,4,4,3,3,3)$, then $\mathcal{J}(D)$ and the weigths will be updated to

$$\mathcal{J}(D_1) = \{[1,1],[2,4],[5,7]\},\quad w([1,1]) = 5, w([2,4]) = 4, w([5,7]) = 3.$$

Thus, if we represent a degree sequence in terms of the corresponding set of intervals as above, we need three kinds of operations (*find*, *delete* and *split*) to efficiently manipulate such disjoint intervals. Here, $find(i)$ returns the maximum element in the interval containing i. $delete(i)$ unites the interval J containing i and the interval J' containing $i+1$ (old J and J' will be destroyed). If i is the maximum element of a current underlying set, then $delete(i)$ sets $J := J - \{i\}$ (and J will be removed if J becomes empty). $split(i)$ splits the interval J containing i into two intervals $J' := \{j \in J | j \leq i\}$ and $J'' := \{j \in J | j > i\}$ (old J will be destroyed and J'' will be removed if $J'' = \emptyset$). Initially $\mathcal{J}(D)$ can be obtained by $split(i)$ for each i with $d_i > d_{i+1}$. Thus, in the example above, $\mathcal{J}(D)$ is obtained by a sequence of $split(1), split(3), split(4)$. Similarly, $\mathcal{J}(D_1)$ is obtained from $\mathcal{J}(D)$ by a sequence of $delete(8), find(3), delete(3)$.

For each i, we initially set $\Delta(n) := d_n$ and $\Delta(i) := d_i - d_{i+1}$ for each $i \neq n$. Thus $d_i = \sum_{j=i}^{n} \Delta(j)$. In each iteration with the current degree sequence $C = (c_\ell, ..., c_g)$, Δ and $\Delta(i)$ are maintained to satisfy $\Delta = c_\ell$, $\Delta(g) = c_g$ (ℓ and g are the minimum and maximum elements of a current underlying set $[\ell, g]$) and $\Delta(i) = c_i - c_{i+1}$ for $\ell \leq i$
$leqg - 1$. Thus, $c_i = \sum_{j=i}^{g} \Delta(j)$ for all i. Note that $\Delta(i) > 0$ if and only if $find(i) = i$.

Now we will present an $O(n \log\log n)$ time algorithm for a graphical degree sequence $D = (d_1, d_2, ..., d_n)$ to constructa $\kappa(D)$-connected graph represented implicitly. Note that for the graphs $G(p, 2b)$, $H(p, 2b+1)$, $G^+(p, 2b, f)$ and $G^-(p, 2b, f)$ defined in Case II-B of the proof of Proposition 3, one can represent them implicitly by using intervals in $O(p)$ time. Similarly, it is an easy exercise to represent the graph in Case II-B with degree sequence $C = (c_\ell, ..., c_g)$ implicitly in $O(n)$ time (this will be done by procedure construct2_graph_implicitly(ℓ, g)). In the algorithm, we use data structures proposed in [10, 11] to support three operations *find*, *delete* and *split* described above as well as to support operations in the doubly-linked list representing the maximum elements of all current intervals.

Algorithm KCGIMPL; {**comment** For a graphical sequence $D = (d_1, ..., d_n)$ with $d_1 \geq ... \geq d_n$, this implicitly constructs a $\kappa(D)$-connected graph $G(D)$ with degree sequence D}
begin
  for $i := 1$ **to** $n-1$ **do begin** $\Delta(i) := d_i - d_{i+1}$; **if** $d_i > d_{i+1}$ **then** $split(i)$ **end**;
  $\Delta(n) := d_n$; $\Delta := d_1$; $k := \kappa(D)$; construct1_graph_implicitly$(1, n)$
end.
procedure construct1_graph_implicitly(ℓ, g); {**comment** this implicitly constructs a $(k - \ell + 1)$-connected graph $G(C)$ with degree sequence $C = (c_\ell, ..., c_g)$}
begin
  if $(\Delta = g - \ell)$ **and** $(k > \ell)$ **then begin** {**comment** Case I}
    $I_1(\ell) := [\ell + 1, g]$; $delete(\ell)$; $\Delta := \Delta - \Delta(\ell) - 1$;
    if $\ell + 1 < g$ **then** construct1_graph_implicitly$(\ell + 1, g)$ **end**
  else begin
    $j := find(\Delta(g) + \ell - 1)$; **if** $j = g$ **then** $j := g - 1$;
    {**comment** $\Delta(g) \geq k - \ell + 1$, $g - 1 \geq j \geq \Delta(g) + \ell - 1 > pre[j]$}
    if $(j \neq g - 1)$ **or** $(\Delta(g) + \Delta(g-1) > k - \ell + 1)$ **then begin** {**comment** Case II-A}
      $delete(g)$; $\Delta(g-1) := \Delta(g-1) + \Delta(g)$; {**comment** $find(g-1) = g-1$}
      create_edge$(g, \Delta(g))$; **if** $\ell < g - 1$ **then** construct1_graph_implicitly$(\ell, g-1)$ **end**
    else {**comment** Case II-B: $j = g-1$, $c_k = \cdots = c_g = k - \ell + 1$}
      construct2_graph_implicitly(ℓ, g) **end**
end;
procedure create_edge_implicitly(f, h);
begin
  if $j = h + \ell - 1$ **then begin** $I_1(f) := [\ell, j]$; $\Delta := \Delta - 1$ **end**
  else begin
    $j_{new} := j - (h + \ell - 1) + pre[j]$;
    if $pre[j] \geq 1$ **then begin**

$I_1(f) := [\ell, pre[j]]$; $I_2(f) := [j_{new}+1, j]$; $\Delta := \Delta - 1$; $\Delta(pre[j]) := \Delta(pre[j]) - 1$;
if $\Delta(pre[j]) = 0$ **then** *delete*$(pre[j])$ **end**
else {**comment** $pre[j] = 0$} $I_1(f) := [j_{new}+1, j]$;
$\Delta(j_{new}) := \Delta(j_{new}) + 1$; *split*$(j_{new})$ **end**;
$\Delta(j) := \Delta(j) - 1$; **if** $(j \neq g-1)$ **and** $(\Delta(j) = 0)$ **then** *delete*(j)
end;

The correctness of Algorithm CGIMPL can be easily shown by observing the correspondence between Alogrithms CG and CGIMPL. Since there are only $O(n)$ operations *find*, *delete* and *split* in the Algorithm CGIMPL, it takes $O(n \log\log n)$ time and $O(n)$ space by the data structure of van Emde Boas and Zijstra [10, 11] Thus we have the following theorem.

Theorem 3. For a graphical sequence $D = (d_1, d_2, ..., d_n)$ with $d_1 \geq d_2 \geq \cdots \geq d_n$, Algorithm CGIMPL implicitly constructs a $\kappa(D)$-connected graph with degree sequence D in $O(n \log\log n)$ time and in $O(n)$ space.

Concluding Remarks

We have presented an $O(n)$ time algorithm for computing the connectivity $\kappa(D)$ of a graphical sequence D and an $O(m)$ time algorithm for constructing a $\kappa(D)$-connected graph with degree sequence D. Furthermore, we have given $O(n \log\log n)$ time algorithm for implicitly constructing a $\kappa(D)$-connected graph with degree sequence D. The same technique will be also applied to other kinds of graphical sequence problems. We conclude by giving a few remarks. In EREW PRAM model, based on Propositon 3, we can easily obtain an $O(\log n)$ time parallel algorithm with $O(n/\log n)$ processors to determine whether a given sequence of nonnegative integers $D = (d_1, d_2, ..., d_n)$ is graphical or not and if so, compute its connectivity. It will be interesting to improve the time complexity of the algorithm for implicitly constructing a $\kappa(D)$-connected graph presented in this paper and find an $o(n \log\log n)$ time algorithm.

References

[1] C. Berge, *Graphes et Hypergraphes*, Dunod, 1970.

[2] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.

[3] P. Erdös and T. Gallai, Graphs with prescribed degrees of vertices (Hungarian), *Mat. Lapok*, 11 (1960), 264-274.

[4] S. Hakimi, On the realizability of a set of integers as degrees of the vertices of a graph, *J. SIAM Appl. Math.*, 10 (1962), 496-506.

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

[6] V. Havel, A remark on the existence of finite graphs (Hungarian), *Časopis Pěst. Mat.*, 80 (1955), 477-480.

[7] S.B. Rao and A. Ramachandra Rao, Existence of 3-connected graphs with prescribed degrees, *Pac. J. Math.*, 33 (1970), 203-207.

[8] M. Takahashi, K. Imai and T. Asano, *Graphical Degree Sequence Problems*, Technical Report TR93-AL-33-15, Information Processing Society of Japan, 1993.

[9] K. Thurlasirman and M.N.S. Swamy, *Graphs: Theory and Algorithms*, John Wilely & Sons, 1992.

[10] P. van Emde Boas, Preserving order in a forest in less than logarithmic time and linear space, *Information Processing Letters*, 6 (1977), 80-82.

[11] P. van Emde Boas and E. Zijstra, Design and implementation of an effective priority queue, *Math. System Theory*, 10 (1977), 99-127.

[12] D.L. Wang and D.J. Kleitman, On the existence of n-Connected Graphs with Prescribed Degrees, *Networks*, 3 (1973), pp.225-239.