アルゴリズム 44－6
（1995．3．17）

移動視点からの多角形可視部分計算

小澤孝夫　内田直樹
龍谷大学理工学部数理情報学科
大津市瀬田大江町　520-21

本文では並列的に配置された凸多角形物体の列に沿って移動する視点を考える。まず、視点からの多角形可視部分を計算するための簡単化された基本的なモデルとそれに関する可視性問題を設定し、次いで、それらの問題を解くアルゴリズムと計算結果を示す。

凸多角形は頂点と頂点をむすぶ辺から構成されるので、その可視部分を求めるのは、頂点の可視性を考え、辺の可視性は頂点の可視性に基づいて計算すればよい。従って、可視性計算のための基本的モデルとして、xy平面においてy軸に平行に配置された半直線物体の列およびそれらの半直線物体と交差しない線上を移動する視点から構成されるというモデルを考える。さらに、それらの半直線物体の可視性は、特定の半直線物体に対して視点からの視線を遮る半直線物体はどれかを求めるという問題に帰着できるので、この問題を解く線形時間アルゴリズムを示す。また、解を可視木と呼ばれるデータ構造に構成する。ランダムな半直線物体列に対して得られた計算結果は極めて特徴的なものである。

COMPUTING THE PORTION OF POLYGONS VISIBLE FROM A MOVING VIEWPOINT

Takao Ozawa and Naoki Uchida
Department of Applied Mathematics and Informatics
Ryukoku University
Seta, Ohtsu, Siga 520-21

In this paper we consider a viewpoint moving along a row of polygonal objects. In order to compute the visible portion of such objects we introduce a simplified model and define visibility problems in terms of the model. We then gpresent solution algorithms for the problems and finally give some computational results.

First, we note that a polygon consists of verticies and edges and that the visibility of the edges can be found from the visibility of the vertices. The abovementioned simplified model for finding the visibility of vertices is constituted by a row of half-lines in the xy-plane which are all in parallel to the y-axis, and the viewpoint which moves on the line not crossing any of the half-lines. The visibility of the half-lines can be obtained by finding, for each of the half-lines, the half-lines which lie in front of it and shut off the ray to the viewpoint. We present a linear-time algorithm for solving the problems of finding such half-lines. The solution is stored in a data structure called the visibility tree. We also present some computational results which are obtained for randomly generated half-line rows and which seem quite characteristic to our model.

1．はじめに

コンピュータグラフィックスにおける基本的な問題の一つは可視性問題である。本文では、道路に沿って並んだと建物と道路上を移動するカメラを想定し、カメラから見える建物部分を求めるというような移動視点からの可視性問題をとりあげる。ここで示すアルゴリズムは上記のような3次元空間における実際問題をかなり簡単化した基本的な可視性問題を解くオフラインアルゴリズムであるが、例えば、あらかじめ知られている物体間をロボットが移動し、見えてくる物体によって自己の位置を知るという問題や、その他のさまざまな可視性問題に対する拡張は容易であると考えられる。

一般的な物体の可視部分計算は、デプスバッファリング法やスペースパーティショニング法などコンピュータグラフィックスにおいて確立されたアルゴリズムによってなされるが適当と考えられる[1) 2)]が、移動視点からの可視性問題に対してこれらのアルゴリズムが必要とする計算量とメモリ量は極めて大きいので、上記のような限定された問題に対しては、異なった接近法も有効であるといえよう。

2．問題の設定

1.に述べた簡単化された問題では、図1に示すような2次元平面（xy-平面）上で並列的に配置された不透明な凸多角形とそれに沿って移動する視点を考える。図1においてLは移動視点Qの軌跡で、簡単のため直線で示してある。凸多角形はその頂点とそれらを結ぶ辺で表現されるので、その可視部分を求めるのは、まず頂点の可視性を考え、辺の可視部分は頂点の可視性に基づいて計算すればよい。さらに、頂点の可視性は、図2に示すような頂点を端点とする平行な半直線物体を考え、頂点と頂点を結ぶ直線とLとの交点を計算することにより求められる。

まず、xy-平面上に配置されたn個の頂点P[j] (j=1, 2, ..., n) およびP[j]を端点としてy軸と平行に下方に延ばされた半直線H[j](j=1, 2, ..., n)とからなる半直線物体の順序列SEQ=(P[j], H[j] / j=1, 2, ..n)が与えられているものとする。ただし、頂点P[j] の座標は(x[j], y[j]), x[j]>0, y[j]>0 で、2点P[j] とP[i] についてj>i ならx[j] >x[i] である。順序列SEQに左方無限遠点にある頂点P[0]と半直線H[0]を追加して得られる順序列を SEQL = {P[j], H[j] / j=0, 1, 2, ..., n} と記す。双対的に、頂点P[0]の代わりに右方無限遠点にある頂点P[f]と半直線H[f]を追加して得られる順序列を SEQR = {P[j], H[j] / j=1, 2, ..., n, f} と記す。

議論の簡単化のために線Lはx軸に平行な直線でそのy座標はどのy[j](j=1, 2, .., n)より大きいとしておく。図2に示すように、視点Qが線L上を左から右に移動するとき、頂点Pが見え始めるのは、頂点Pと頂点PLを結ぶ直線とLとの交点ALにQが来たときであり、頂点Pが見えなくなるのは、頂点Pと頂点PRを結ぶ直線と線Lとの交点ARにQが到達したとき以後である。従って、SEQの頂点P[j]に対して次のようなな条件を満足する点を頂点P[j]の左制限点と呼ぶことにする。

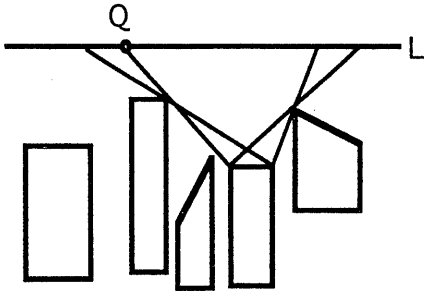


図1

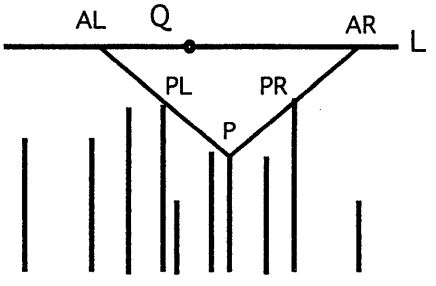


図2

条件LC：（1）頂点P[j]と頂点P[i]を結ぶ直線はどの半直線とも交わらない；（2）iは（1）を満たす最小のインデックスである。

頂点P[j]に対する左制限点をPL[j]と記し、そのインデックスをPI[j]と記すことにする。双対的に、SEQの頂点P[j]に対して、条件LCの（2）における「最小」を「最大」に変更して定義される点をP[j]の右制限点と呼ぶことにする。

まず、次のような基本的な問題を定義する。

問題PL（PR）：SEQの頂点P[j] (j=1, 2, ..., n) に対する左（あるいは右）制限点を求めよ。

次にSEQの半直線部分について考える。図3では、視点が線L上の点ALを越した時から頂点Pの下方に延ばした半直線Hが見え始めるが、その下方の一部は、頂点PLから下方に延ばした半直線HLによって視線が遮られて見えない。すなわち、視点と頂点PLを結ぶ直線が、半直線Hと交わる点より下の部分は見えない。さらに、頂点PLと頂点PBを結ぶ直線が線Lと交わる点をABとすると、点ABを視点が通り過ぎた後、視線を遮るのは頂点PBから下方に延ばした半直線HBである。同様に、頂点PBとPCを結ぶ直線が線Lと交わる点をACとすると、点ACを視点が通り過ぎた後、視線を遮るのは頂点PCから下方に延ばした半直線HCである。頂点PLとPBを結ぶ直線あるいは頂点PBと頂点PCを結ぶ直線が、半直線Hと交わる点（図3の点P1あるいは点P2のような点）を半直線H上の左臨界点と呼ぶことにする。さらに、頂点Pに対する左制限点PLと同様、左臨界点P1あるいはP2に対する頂点PBあるいは頂点PCをそれぞれそれらの左制限点と呼ぶことにする。双対的に、H[j]における右臨界点、および右臨界点に対する右制限点を定義できる。

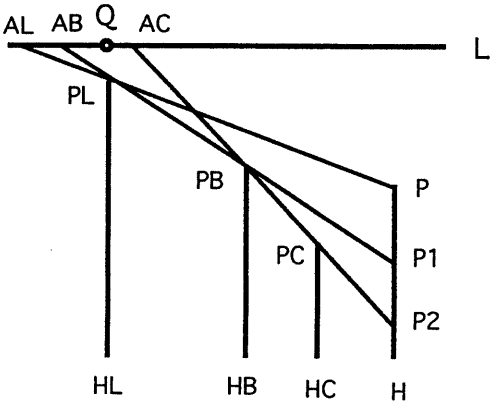


図3

問題CL（CR）：SEQの半直線H[j] (j=1, 2, ..., n) 上の左（あるいは右）臨界点およびそれらに対する左（あるいは右）制限点を求めよ。

半直線H[j]上の左臨界点をP1[j]、P2[j]、...と記すことにする。図3からも分かるように、移動する視点から見える半直線H[j]の部分は、左臨界点に対する左制限点が分かっていれば容易に求めることができる。すなわち、視点と左制限点を結ぶ直線がHと交わる点より上がHの可視部分である。

制限点や臨界点は半直線物体間の位置関係からのみ求められるが、次の問題は視点についてのそれである。

問題ODL: 視点Qが移動する際に左側から見えるSEQの頂点の順序および頂点が見え始める線L上の視点位置を求めよ。

問題ODLは、視点が半直線物体の真上に来るまでを考えているが、視点が半直線物体の真上を通りすぎた後を考えて、問題ODLに双対的な問題ODRを定義できる。

問題ODR: 視点Qが移動する際に右側から見えるSEQの点の順序および頂点が見えなくなる線L上の視点位置を求めよ。

上の問題では半直線物体を考えたが、凸多角形物体の場合は上記の問題を修正あるいは拡張して可視部分を求めればよい。例えば、図3においては、半直線Hは点Pからその真下に延びているが、点Pから左下方へ斜めに延びる直線上の左臨界点に対する左制限点は、点Pから真下に延びた半直線上の左臨界点に対する左制限点と同じである。図4参照。点Pから右下方へ斜めに延びる半直線の左側からの可視部分もこれらの左制限点を考慮して求めることができる。線Lに対する制限も緩めることができる。

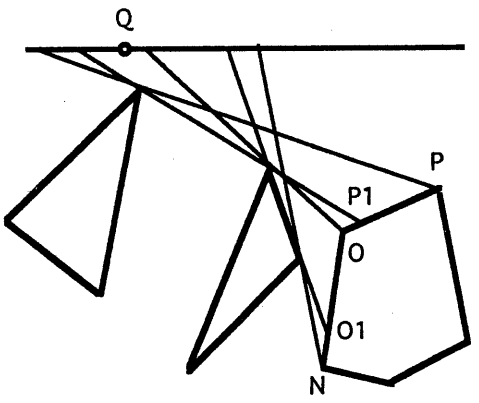


図4

3．可視木と問題PLの解法アルゴリズム

順序列SEQLに含まれる頂点に対する左制限点を明示するため、および半直線物体の挿入あるいは削除があったときの制限点の変更を取り扱うためのデータ構造として、左可視木を次のように定義する。

（1）左可視木は根付多分木であって、そのノードはSEQLの頂点に1対1に対応する。従って、頂点P[j]に対応するノードのラベルもP[j]とする。

（2）左可視木の根は頂点P[0]に対応する。

（3）ノードP[j]（ただし、P[j]!=P[0]）の親ノードは頂点P[j]の左制限点PL[j]に対応する。

（4）同一の親を持つノード（兄弟ノード）にはSEQLにおける頂点の順序に応じた順序付けがなされ、インデックスjの値が小さいほど左にくるものとする。

双対的に、右可視木を定義できる。その根は頂点P[f]に対応し、ノードP[j]（ただし、P[j]!=P[f]）の親ノードはP[j]の右制限点に対応する。

頂点P[j]と頂点P[i]を結ぶ直線の勾配を与える関数は

$$slope(j, i)=(y[j]-y[i])/(x[j]-x[i]) \qquad (1)$$

と定義される。ただし、slope(j, 0)=0とする。

左可視木に関しては次のような補題が成り立つ。

[補題1] ノードP[j]の親ノードPL[j]は、左可視木においてノードP[j-1]からノードP[0]に至るパス上（両端のノードを含む）にある。

（証明）ノードP[j-1]からノードP[0]に至るパス上にある親と子のノードをそれぞれP[a]とP[b]とする。P[a]の子でパス上にないノードP[c]を考え、これに対応する頂点P[c]は頂点P[j]の左制限点でありえないことを示す。まず、ノードP[c]がP[b]より右にある兄弟ノードなら、兄弟ノードに付けられた順序からc>jであるから、頂点P[c]が頂点P[j]の左制限点でありえない。また、ノードP[c]がP[b]より左にある兄弟ノードなら、図5に示すように、頂点P[c]が頂点P[b]の左制限点でないことから、P[c]はP[j]の左制限点でもありえない。従って、補題1が成立する。

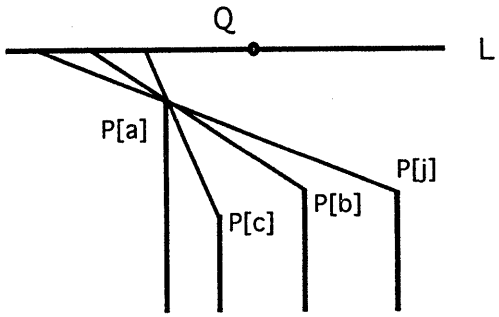


図5

補題1を考慮すると、ノードP[j-1]からP[0]に至るパス上のノードを順次たどれば頂点P[j]の左制限点を求めることができる。パス上にあるノードP[i]について考えてみると、図6(a)に示すように、

$$slope(j, i) > slope(i, PI[i]). \qquad (2)$$

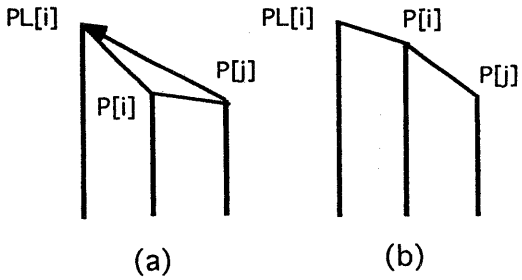


図6

なら頂点P[i]は頂点P[j]の左制限点でないので、ノードP[i]から更にその親ノードPL[i]に対する式（2）のテストに進む。図6(b)のように、式（2）が満たされないと、頂点P[i]と頂点PL[i]を結ぶ直線の上部にはSEQLの点が存在しないので、P[i]はP[j]の左制限点である。

上に述べたことをアルゴリズムにまとめると次のようになる。

[アルゴリズム　SOL-PL]
入力　SEQL
出力　左可視木
(1) /*初期設定*/　{可視木の根P[0]を設定し、ノードP[1]をノードP[0]にその子として接続する。
PI[1]=PI[0]=0;
(2) for(j=2; j<=n; j++){
(2-1) i=j-1;
(2-2) while(slope(j, i) >slope(j, PI[i])){
i=PI[i];}
(2-3) PI[j]=i; ノードP[j]をノードP[i]にその子として接続する。}

[定理1]　アルゴリズム　SOL-PLによりO(n)の計算量で問題PLの解を求め、可視木を構成できる。

（証明）アルゴリズム　SOL-PLのステップ(2-2)の繰り返しは補題1にいうパス上のノードの探索である。計算量について考えると、まず、ステップ(2-1)の繰り返し回数は明らかにn-1である。次に、ステップ(2-2)の条件テストの回数は2n以下であることを示す。可視木構成の際に各ノードが条件テストのために訪問される回数を考えると、子となったノードからの訪問回数は1であり、子とならなかったノードからの訪問回数は1以下である（P[i]がP[j]の親とならなければ、P[j]はP[i]の親あるいは祖先に接続され、P[i]はP[j]より後に来るノードP[k](k>j)によって訪問されることはない）。P[1]を除く子ノードの総数はn-1であるから、前者の訪問回数もn-1である。一方、ノードの総数がnであるから後者の訪問回数の総和はn以下である。

右可視木は左可視木と双対的なアルゴリズムにより構成できる。また、左可視木が与えられると、それから構成することもできる。

4．可視木と問題CLの解法

図7(a)に示すように、PL[j]=P[a]で、ノードP[j]のすぐ左に兄弟ノードP[c]があるとする。図7(b)のように、頂点P[a]と頂点P[c]を結ぶ直線が線Lおよび半直線H[j]と交わる点をそれぞれA1[j]およびP1[j]とすると、P[j]がA[j]からP[a]越しに見えた後、P1[j]がA1[j]からP[c]越しに見えてくることになる。更に、ノードP[c]の最も右の子をノードP[d]とし、頂点P[c]と頂点P[d]を結ぶ直線が線LおよびH[j]と交わる点をそれぞれA2[j]およびP2[j]とすると、P1[j]がA1[j]からP[c]越しに見えた後、P2[j]がA2[j]からP[d]越しに見えてく

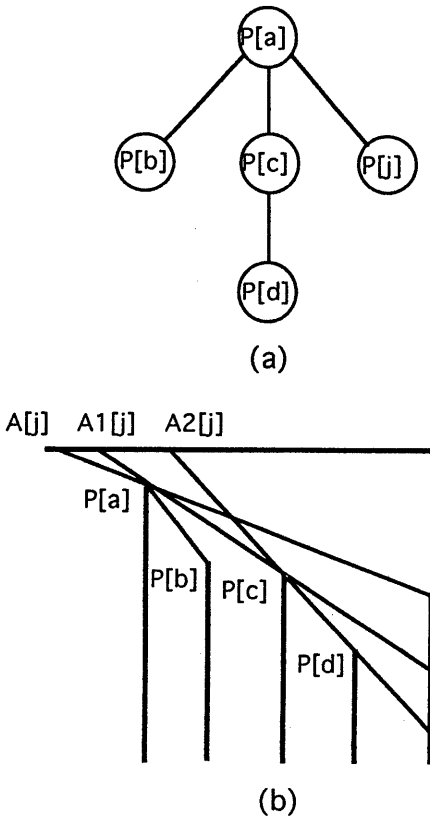


図7

ることになる。ノードP[j]のすぐ左にある兄弟ノードP[c]以外のP[b]のような兄弟ノードは考慮する必要がない。

左可視木の各ノードに左臨界点に対する左制限点を記憶するスタックを準備できるとすれば、図3と図6から分かるように、可視木構成の際に左臨界点に対する左制限点をスタックに入れていける。すなわち、このスタックをS[j]として、アルゴリズムSOL-PLのステップ(2-1)と(2-2)を次のように変更すれば問題CRの解が可視木構成時に得られる。

```
(2-1)   i=j-1;   S[j]を空にする;
(2-2)   while(slope(j, i)>slope(j, PI[i])){
            P[i]をS[j]に入れる;
            i=PI[i];}
```

SOL-PLにより可視木が構成された後では、可視木においてノードP[j]のすぐ左にある兄弟ノードから順次最も右にある子をたどって行けば、半直線H[j]についての左臨界点に対する左制限点を求めることができる。

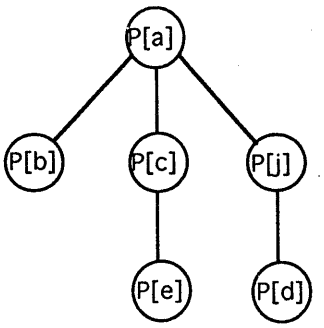
5. 可視木を用いた問題ODLの解法

視点Qが移動する際、ある頂点が左側から見え始める線L上の視点位置は、その頂点とその左制限点を結ぶ直線の延長が線Lと交わる点であるから、このような交点をSEQLのすべての頂点に対して求め、その座標をソートすれば、視点から見える頂点の順序を知ることができる。

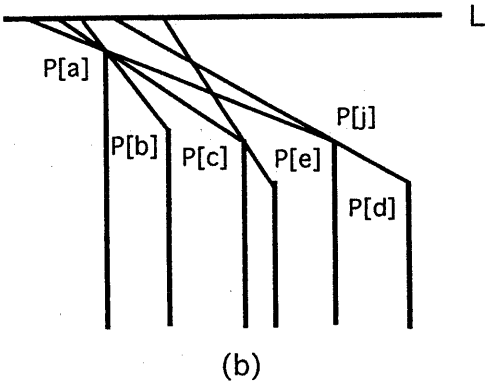
視点から見える頂点の順序は、左可視木から次のようにしても求められる。

まず、根に付くノードに対応する頂点が最初に見えるのは明らかである。次に、これらのノードの最も右にある子ノードに対応する頂点のうちのどれかが見え始めるのであるから、それらの頂点とその左制限点を結ぶ直線の延長が線Lと交わる交点を求め、それらをソートして次に見える頂点の候補の順序列を作る。これをCANDとする。さらに、CANDに含まれるある頂点Pが見え始めたとすると、Pに対応するノードPのすぐ左にある兄弟ノード（もしあれば）とPの最も右にある子ノード（もしあれば）とに対応する頂点がCANDに付け加わえられることになる。

一般的に、図8(a)に示すような部分木がある場合、図8(b)から分かるように、頂点P[j]が見え始めたとすると、P[c]とP[d]が次に見え始める頂点の候補となる。どちらの頂点が見えるのかを決めるために、P[c]あるいはP[d]とその左制限点を結ぶ直線の延長が線Lと交わる点を求め、交点の座標値に応じて頂点をCANDに挿入する。さらに、図8(b)のように頂点P[c]が見え始める場合は、CANDに頂点P[b]とP[e]が挿入されることになる。



(a)



(b)

図8

[アルゴリズム　SOL-ODL]
入力　左可視木
出力　視点から見える頂点の順序列ODL

補助　次に見える頂点候補の順序列CAND

(1) 左可視木の根の子ノードに対応する頂点をCANDに入れる。

(2) while(CANDに頂点が存在する){

(2-1) CANDの先頭にある頂点（Pとする）をCANDから削除し、ODLの末尾に置く。

(2-2) Pに対応する可視木のノードの左側に兄弟ノードがあれば、それに対応する頂点をCANDに挿入する。

(2-3) Pに対応する可視木のノードに子ノードがあれば、最も右の子ノードに対応する頂点をCANDに挿入する。}

上のアルゴリズムにおいて、頂点PをCANDに挿入する際には、Pとその左制限点を結ぶ直線の延長が線Lと交わる点を求め、交点の座標値に応じて挿入することになる。

6．アルゴリズムの適用例

図9に半直線物体とそれから構成された可視木の例を示す。

また、ランダムの高さと間隔を持った半直線物体列を発生し、それに可視木構成アルゴリズムVISI-TREEを適用した結果を下記のようになった。

物体数：n=10,000、発生列数：10,000

物体の存在領域：0<x[j]<100,000, 0<y[j]<Y

左制限点を求めるために可視木において訪問したノード数の合計

Y= 1,000:　平均：22,621　標準偏差：36.4

Y= 5,000:　平均：22,643　標準偏差：37.0

Y= 10,000:　平均：22,645　標準偏差：37.5

Y= 50,000:　平均：22,592　標準偏差：37.0

Y=100,000:　平均：22,593　標準偏差：36.0

ノードP[j]からは、ノードP[j-1]とP[j]の左制限点になった点に対応するノードを必ず訪問するので、必ず訪問しなければならないノード以外には、平均2.26-2.0=0.26個のノードを訪問していることになる。訪問したノード数の合計の平均に対して標準偏差がきわめて小さいので、上記のことは、どの物体列に対してもがいえ、欄数により発生した物体列の持つ特定の性質から導けると考えられる。

5．おわりに

計算幾何学の問題として、かなり簡単化した移動する視点からの可視性問題を設定し、それを解くオフラインアルゴリズムと可視性を表わす可視木というデータ構造をを提案した。アルゴリズムは簡単なものであるが、それを拡張して種々の計算幾何学の問題[3) 4)]などに適用できると考えられる。そのような拡張や計算結果に対する検討、また、アルゴリズムの並列化が今後の課題である。

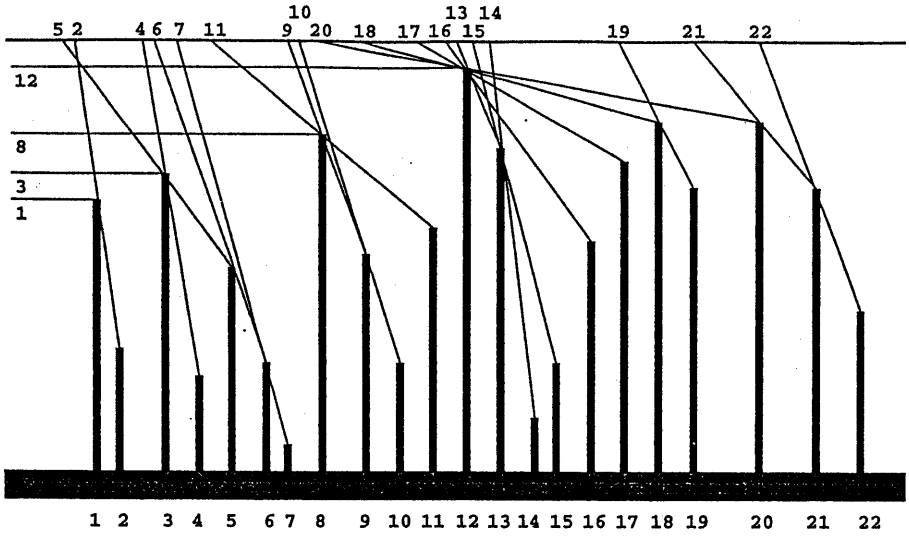
文献

1）S. Harrington: Computer Graphics, McGraw-Hill Book Company(1987)

2）G. Glaeser: Fast Algorithms for 3D-Graphics, Springer-Verlag(1994)

3）浅野哲夫: 計算幾何学, 朝倉書店(990)

4）今井浩, 今井桂子: 計算幾何学, 共立出版(1994)


10
13 14
5 2
4 6 7 11
9 20 18 17 16 15
19 21 22
12
8
3
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

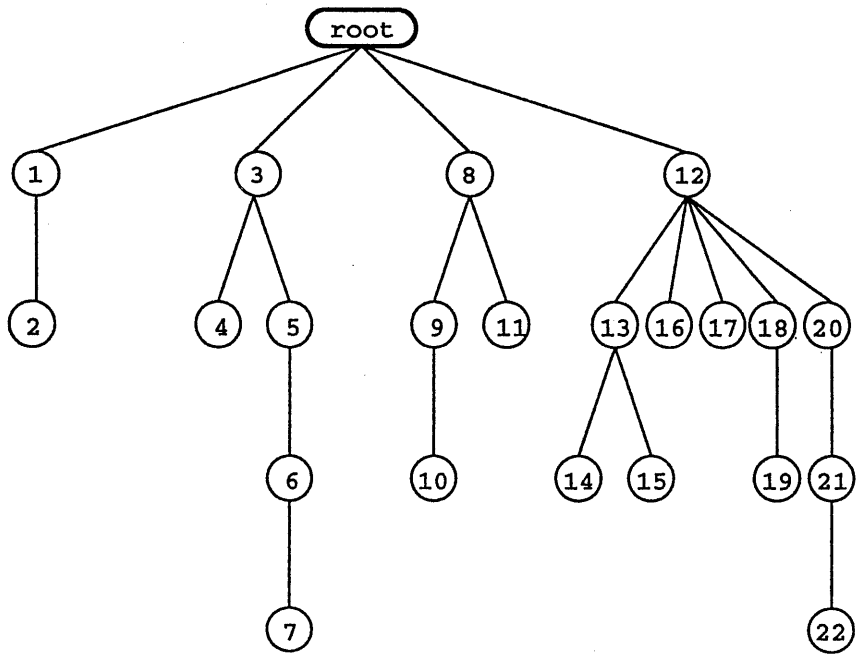

root
1
3
8
12
2
4
5
9
11
13
16
17
18
20
6
10
14
15
19
21
7
22

図　9