アルゴリズム 44－3
（1995．3．17）

整列法評価のためのランダム順列生成法

二村良彦　青木健一　大谷啓記　白井千恵子

早稲田大学 理工学部

与えられた長さnと葉数mを持つ順列を一様に生成することが理論的に保証された現実的な O((n-m)m)アルゴリズムとその高速簡便版であるO(n)アルゴリズム,生成された順列の一様性を検定する方法,および検定結果について報告する．ただし順列の葉数とは,順列内で隣接する要素より小さい要素の個数である．一様乱数列はある意味で非常に特殊な数列(常に約n/3個の葉を持つ)であり,整列法の評価データとしては不向きである．本稿提案の方式で順列を生成することにより整列法の正当な評価が可能となった．

Random Permutation Generator for Evaluating Sorting Algorithms

Yoshihiko FUTAMURA　**Kenichi AOKI**

Hirofusa OTANI　**Chieko SHIRAI**

School of Science and Engineering, Waseda University

We propose (1) an *O((n-m)m)* time algorithm for generating a random permutation with length *n* and *m Leaves* that is theoretically guaranteed to be uniform, and (2) an *O(n)* time algorithm for random permutation generation with specified *Leaves* that is not guaranteed to be uniform. Randomness testing methods for generators and chi-square test results are described. *Leaves* of a sequence are elements who have no larger neighbors in the sequence. Uniform random number sequences have very biased property to have (n+1)/3 leaves. Therefore they are not fair test data for evaluating sorting algorithms. Our generation method offers good test data for the evaluation.

1 はじめに

ソーティングアルゴリズム(整列法)の性能評価のために,従来は一様乱数列が利用されてきた[4]. しかし整列法にとって,一様乱数列はある意味で特殊な入力列である. 何故ならば一様乱数列の平均葉数は約n/3(ただしnは数列の長さ,そして葉数は数列内で隣接する要素より小さい要素の個数)である[2,3]. 現実の問題として数列のソーティングをあつかう場合には,与えられた数列が殆ど整列済みである場合が多いと言われている[4]. 整列済みに近い数列の葉数はn/3よりはるかに小さい. 従って整列法の性能評価をする際には葉数(あるいはその他の事前整列性測度[3])を制御しながら,数列を生成する必要がある.

本稿では,任意に与えられた葉数mを持つ順列を一様に(即ち等確率で)生成することが理論的に保証された2つの方法,および生成する順列の一様性は保証しないが0(n)時間かつスペースの高速簡便法について報告する. 生成すべき順列の個数をkとすると,前2者の計算量は

(1)(n-m)*max(k,m)時間かつ(n-m)*mスペースおよび

(2)k*(n-m)*m時間かつn-mスペースである.

それ等によって生成された順列に指標付けすることにより,順列の列を数列に変換し,その数列の一様性をχ自乗検定する方法および検定結果についても報告する.

葉数は筆者等が[2]で報告した撹乱要因と本質的に同じである(葉数=撹乱要因+1). 撹乱要因を制御して数列をランダムに生成するO(n log n)アルゴリズムに関する浅野[1]の報告がある. それは生成される順列の一様性を保証しない一種の簡便法であるが,その方法と本稿で提案する方法との比較についても触れる. 以下で用いる用語については[3]を参照されたい.

2 葉数を制御したランダム順列の生成法

以下においては長さn,葉数mの順列の個数を下記のL(n,m)で表す.

性質1:節数n,葉数mの順列(または対称ヒープ)の個数をL(n,m)とすれば,木の作り方より明らかに,それは下記の再帰方程式で定義できる:

(1) L(n,m)=0 if m<1 or n+1<2m

(2) $L(n,1)=2^{n-1}$

(3) L(n,m)=2m*L(n-1,m)+(n-2m+2)*L(n-1,m-1) if $1<m\leqq(n+1)/2$

L(n,m)の大きさの目安は次の定理により示される(証明は[3]を参照されたい).

定理1:$m\geqq 1$かつ$n+1\geqq 2m$ならば,

$L(n,m)\geqq((2m-2)/e)^{(n-1)/2}$

ちなみにL(3,2)=2, L(4,2)=16, L(5,2)=88, L(5,3)=16, L(6,2)=416, L(6,3)=272, L(7,3)=2880, $L(n,2)=2^{2n-3}-n2^{n-2}$である.

ここでは長さn,葉数mのL(n,m)個の順列を等確率で生成する方法をまず述べ,次に一様ではないが0(n)時間で順列を生成する方法を述べる. 最初の方法は下記の性質1に基づく.

定義1:集合{2,3,...,n}の任意の順列をt'とする. ここでt'に要素1を挿入する2つの操作に次のように名前を付ける.

(1)葉に挿入:t'の葉に隣接して挿入すること

(2)u節に挿入:t'のu節に隣接し,そのu節よりも小さい要素の反対側に挿入すること.

これは順列に対する対称ヒープあるいは順列木[3]に翻訳して考えれば明らかである. 葉に挿入する場合は葉の左右2通りがあるが,u節に挿入する場合は1通りしかない.

性質2:集合{1,2,3,...,n}の順列tの葉数がmかつtは集合{2,3,...,n}の順列t'に1を挿入して作られているとする. この時1がt'の葉に付いている確率P(n,m)は2m*L(n-1,m)/L(n,m)である.

方式1：長さn, 葉数mの順列を次の再帰的手順によって作る：

(1)P(n,m)を計算する.

(2)0<r≦1なる一様乱数rを生成する.

(3)r≦P(n,m)ならば集合{2,3,...,n}の順列で葉数mのものt'を生成し, t'の葉に1を挿入する. r>P(n,m)ならば集合{2,3,...,n}の順列で葉数m-1のものt'を生成し, t'のu節に1を挿入する. 挿入箇所は当然ランダムに選ぶ.

この方式に従って実際に順列を生成するためには問題が2点有る：

(1)順列中に1を挿入する際に常数時間で行わないと, P(n,m)の計算以外で$O(n^2)$時間掛かってしまう.

(2)P(m,n)の計算中にL(n,m)の計算が含まれる為, 大きなnに対してはオーバーフローにより計算不可能である.

問題点1を解決する為には, 順列をリンクドリストで表現し, 葉とu節を別々の配列で管理し(例えば後述の指標付け関数i_3におけるp_1とp_u), u節が減った時には対応する配列上の場所を, 配列の最後の要素の内容で埋めて配列に虫食いが出来ないようにする(葉についても同様に管理する). このようにすれば方式1における挿入箇所の選択および挿入を常数時間で行う事が可能である(計算量については表1参照).

問題点2を解決する為にプログラム変換の技術によりP(n,m)を次のように変換し, L(n,m)を直接計算しないで済ませた.

定理2：P(n,m)=1/F(n,m)ただし

(1)F(n,m)=∞ if m<1 or n<2m

(2)F(n,1)=1 if n>1.

(3)F(n,m)= 1+2(m-1)*F(n-1,m-1)/m if n=2m.

(4)F(n,m)=1+{(m-1)(u+1)*F(n-1,m-1)}/
{mu*F(n-1,m)/(F(n-1,m)-1)} if
otherwise ただしu=n-2m+1

(証明は付録1).

ちなみにFの値の大きさは実用の範囲である. 例えばF(1998,99)=1.000006, F(4001,2000)=811.0832. ただしFの値が1に近づくと定理2の(4)におけるF(n-1,m)-1の計算で深刻な桁落ちが発生し, 正しい値を求めることができなくなる. 従ってFの計算は下記の補助関数F1を用いて行う必要がある.

系1：F(n,m)=F1(n,m)+1, ただし

(1)F1(n,m)=∞ if m<1 or n<2m

(2)F1(n,1)=0 if n>1.

(3)F1(n,m)=2(m-1)*(F1(n-1,m-1)+1)/m
if n=2m.

(4)F1(n,m)={(m-1)(u+1)*(F1(n-1,m-1)+1)}/
{mu*(F1(n-1,m)+1)/F1(n-1,m)}
otherwise ただしu=n-2m+1

(n-m)*mの配列を利用した動的計画法によりF(n,m)は$O((n-m)*m)$時間で計算できる. またF(n,m)は一旦計算して, その中途結果F(i,j)(1≦i≦n, 1≦j≦m)と共に配列に格納しておけば, 同じ葉数の順列を生成する限りに於いては再計算の必要が無い. 順列がランダムに生成されたか否か検定する際には通常k≧10*L(n,m)>mであり, この方式でも順列1つあたり$O(n)$であるので時間的には問題は少ない. 配列の利用を工夫すると, スペース所要量も約(n-m)*m/2にすることが可能である.

しかし大きなnに対してF(n,m)を計算する場合にはこれでもメモリーネックになる. そこで計算時間は同じでもF(n,m)の中途結果を残す事は出来ないがn-mスペースで計算可能な方式2も開発した. この方式でk個の順列を生成する場合にはk*(n-m)*m時間かかる.

方式2：まずn-mスペースの方法でF(n,m)を計算する(付録2参照). 次に長さn, 葉数mの順列を次の再帰的手順によって作る：

(1)0<r≦1なる一様乱数rを生成する.

(2)r≦P(n,m)ならば集合{2,3,...,n}の順列で葉数mのものt'を生成し, t'の葉に1を挿入する. r>P(n,m)ならば集合{2,3,...,n}の順列で葉数m-1のも

のt'を生成し,t'のu節に1を挿入する.

(3)P(n,m)および2nスペースに残された情報に基づきP(n-1,m')を逆算する.ただし1が葉に挿入された時はm'=m,そうでなければm'=m-1.

上記の2方式が一様なランダム順列を生成することは理論的に保証できる.次には適当にランダムな順列をO(n)時間で生成する簡便法について述べる.

方式3:長さn,葉数mの順列を次の反復手続によって作る:

(1)要素nだけからなる長さ1の順列をtとする.

(2)tの長さがnより小さい間は処理(3)を行う.tの長さがnになったら処理を終了する.

(3) n-(tの長さ)をsとする.tの葉数がmあるいはtがu節を持たなければsをtの葉に挿入する.tの葉数がm-sならばsをtのu節に挿入する.その他の場合は処理(4)を行う.

(4)tの葉数およびu節の個数を各々m'およびu'とする.この時sを2m'/(2m'+u')の確率でtの葉に挿入し,u'/(2m'+u')の確率でtのu節に挿入する.

上記方式3において処理(3)を除去するだけで葉数を制御しないランダム順列の生成手順となる.

方式4:長さnの順列を次の反復手続きによって作る:

(1)要素nだけからなる長さ1の順列をtとする.

(2)tの長さがnより小さい間は処理(3)を行う.tの長さがnになったら処理を終了する.

(3)n-(tの長さ)をsとする.tの葉数およびu節の個数を各々m'およびu'とする.この時sを2m'/(2m'+u')の確率でtの葉に挿入し,u'/(2m'+u')の確率でtのu節に挿入する.

この方式で長さnの順列をk個生成するために要する時間はO(k*n)である.また所要スペースはO(n)である.

3 ランダム順列の検定

長さn葉数mを有する順列をランダムに生成する3方式を評価する為に,順列木に1からL(n,m)までの番号(木の指標と呼ぶ)を付け,指標の列についてχ^2検定を行った.またその指標列の葉数も調べた(順列に含まれる葉数の平均は(n+1)/3である[7]が,[3]で述べた性質5により,一様乱数列の葉数は約n/3).ここでは指標付けの方法および調査結果について述べる.以下では節数nのすべての順列木の集合を$T(n)$で表す.また節数n,葉数mのすべての順列木の集合を$T(n,m)$で表す.節数nの順序木の任意の集合Sの濃度を$|S|$,かつSから集合$\{1,2,\ldots,|S|\}$への任意の1-1 onto写像をindとする.このときSの任意の要素tに対して$ind(t)$を集合Sに関する木tの指標(index),そしてindを$T(n,m)$の指標付け関数と呼ぶ.

(1)$T(n,m)$の指標付け関数の例

$T(n,m)$の任意の要素をt,そしてtから節1を取り除いて得られる木をt'とする.この時下記の関数i_1は$T(n,m)$の指標付け関数である.ただし節1はt'の左からu_0番目のu節に付いている($0 \leqq u_0 \leqq n-2m$)かまたは左からm_0番目の葉位置($0 \leqq m_0 \leqq 2m-1$)に付いているものとする.また$i_1(n)=1$かつ$u_1=n-2m+2$(図1).

$$i_1(t)=i_1(t')+u_0 L(n-1,m-1) \quad \text{if} \quad t' \in T(n-1,m-1)$$
$$i_1(t)=i_1(t')+u_1 L(n-1,m-1)+m_0 L(n-1,m) \quad \text{if } t' \in T(n-1,m)$$

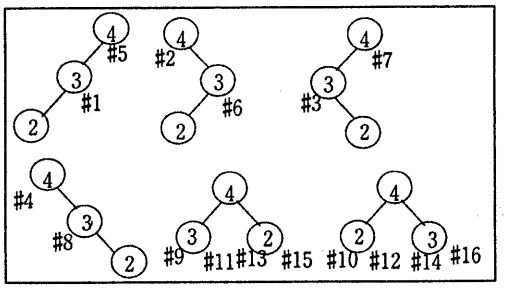

図1: 節数4,葉数2の順列木の指標付けの例
節数3,葉数1の木に要素1が葉として追加される場所および作られる木の指標を#1～#8で示した.また節数3,葉数2の木に要素1が葉として追加される場所および作られる木の指標を#9～#16で示した.

(2) $T(n)$の指標付け関数の例

$T(n)$の任意の要素をtとする．この時下記の関数i_2は$T(n)$の指標付け関数である．ただしmはtの葉数とする．

$$i_2(t)=i_1(t)+\sum_{j=1}^{m-1}L(n,j)$$

(3) $T(n,m)$の指標付け関数の別例

$T(n,m)$の任意の要素をt,そしてtから節1を取り除いて得られる木をt'とする．この時下記の関数i_3は$T(n,m)$の指標付け関数である．ここで節1はt'のu節u_0に付いている$(1\leqq u_0<n)$かまたは葉$m_0$$(1\leqq m_0<n)$に付いているものとする．節1が葉$m_0$の右側に付く場合には$left$=1とし,左側に付く時には$left$=0とする．また$i_3$(n)=1かつ$u_1=n-2m+2$.

$$i_3(t)=i_3(t')+(p_u(t')(u_0))*L(n-1,m-1)$$
$$\text{if } t'\in T(n-1,m-1)$$
$$i_3(t)=i_3(t')+u_1L(n-1,m-1)+(2p_l(t')(m_0)+left)$$
$$*L(n-1,m)\ \text{ if }\ t'\in T(n-1,m)$$

ただし$p_U(t')$および$p_l(t')$は各々t'のu節から$\{0,\ldots,n-2m\}$およびt'の葉から$\{0,\ldots,2m-1\}$への(t'によって一意的に決まる)1-1写像である．p_l(n)=$\{(1,0)\}$, p_U(n)=$\{\ \}$として,以下のように定義される．

$$p_l(t)=p_l(t')\cup\{(n,|p_l(t')|)\}$$
$$p_u(t)=(p_u(t')-\{(u_0,p_u(t')(u_0)),(x,|p_u(t')|-1)\})$$
$$\cup\{(x,p_u(t')(u_0))\}\ \text{ where }\ t'\in T(n-1,m-1)$$
$$p_l(t)=(p_l(t')-\{(m_0,p_l(t')(m_0))\})\cup\{(n,p_l(t')(m_0))\}$$
$$p_u(t)=p_u(t')\cup\{(m_0,|p_u(t')|\}\ \text{ where }\ t'\in T(n-1,m)$$

(4) $T(n)$の指標付け関数の別例

$T(n)$の任意の要素をtとする．この時下記の関数i_4は$T(n)$の指標付け関数である．ただしmはtの葉数とする．

$$i_4(t)=i_3(t)+\sum_{j=1}^{m-1}L(n,j)$$

$if\ t'\ \ T(n-1,m)$

順列生成方式1,2,3および先駆的業績である浅野方式[1]により,長さ7葉数3の順列(L(7,3)=2880)を50,000個生成し,指標付け関数i_1により指標付けして1と2,880の間の乱数列とし,χ^2乗検定を3回行った．表1はその平均値である．また3本の数列の葉数も1本当たりの平均を示した(葉数の理論的期待値は約16,667)．時間とスペースに関するビッグオーも示した．

表1：長さn葉数mの順列k個を生成する場合の比較

方式	時間	スペース	χ^2統計量	葉数
1	n*max{m,k}	(n-m)*m	2861.4	16691
2	k*n*m	n−m	2861.4	16691
3	k*n	n	11178.2	16708
浅野	k*n*log n	n	25140.0	16652

表1では,方式1と2は,予想通り良い検定結果を示している(統計量が2879に近いほど良い[7])．方式3と浅野方式は順列を一様に発生させるには不適であることが分かる．

しかし例えばn=4095,m=2048のとき,生成される乱数列は1516^{2047}以上の自由度を持ち,その一様性を検定する方法を我々は知らない．それ故に理論的に一様性を保証された生成法で順列を生成する必要がある．一方では,ある程度ランダム性が保証された生成法ならば,整列法の性能評価結果に大差を与えないと考えられる．実際,我々の実験に於いては方式2と3では性能評価結果に差が見られなかった[3]．ランダム順列の使用目的に応じた方法を選択することがシミュレーションの時間短縮のために重要である．

4 おわりに

葉数を制御したランダム順列の生成法で,理論的に正しさが保証されており,しかも現実的に計算可能な方法を2つ示した．指定された葉数の順列に指標付けする方法を開発し,χ^2検定および葉数の計測によりそれらの方式の正しさを実験的にも調べた．

生成される順列の一様性は保証されないが,整列法の評価実験には使用し得る高速な簡便生成法も示した．葉数を制御する場合にもO(n)時間で順列を一様に生成する方法の開発が今後の課題である(2分木の生成に関する[6]のような数値解析的なアプローチも検討中である)．最後に,確率計算における数値的誤差の指摘とその解決に協力を惜しまなかった二村研究室遠藤貢一氏に深謝します．

5 参考文献

[1]浅野：各種ソーティグアルゴリズムの実際的評価,情報処理学会アルゴリズム研究会30-7,92年11月.

[2]二村,筧,二村：対称ヒープの実現とその応用,情報処理学会アルゴリズム研究会28-7,92年7月.

[3]二村,二村,遠藤,平井:葉数最適整列法LOASとその実現法,情報処理学会アルゴリズム研究会44-2,95年3月.

[4]KNUTH：The Art of Computer Programming. Vol.3,Addison-Wesley, 1973.

[5]SEDGEWICK:Algorithms,Addison-Wesley,1989.

[6]SPRUGNORI:The generation of binary trees as a numerical problem, J.ACM,Vol.39,NO.2, 1992,317-327.

[7]SPRUGNORI:Properties of binary trees related to position, Comput. J., Vol.35, No.4,1992, 395-404.

付録1 F(n,m)の導出法

定義よりF(n,m)=L(n,m)/2mL(n-1,m)である．始めにh(n,m)=L(n,m)/L(n-1,m)を求め,次にF(n,m)=h(n,m)/2mを求める．以下ではu=n-2m+1とする．

h(n,m)=L(n,m)/L(n-1,m)

=2m+(u+1)L(n-1,m-1)/L(n-1,m)　(L(n,m)の展開)

ここでhを次のように定義し直す．

h(n,m)=2m+(u+1)g(n-1,m-1).....(1)

ただし,

g(n,m)=L(n,m)/L(n,m+1)

=h(n,m)/f(n,m+1).....(2)

f(n,m)=L(n,m)/L(n-1,m-1)

=2mL(n-1,m)/L(n-1,m-1)+u+1.....(3)

(1)より

g(n-1,m-1)=(h(n,m)-2m)/(u+1)

従ってu≠0ならば

g(n,m)=(h(n+1,m+1)-2m-2)/u .....(4)

また(2),(3)より

f(n,m)=2m/g(n-1,m-1)+u+1

=2m/g(n-1,m-1)+n-2m+2.....(5)

(4),(5)より

f(n,m)=2m(u+1)/(h(n,m)-2m)+u+1.....(6)

一方(2),(3)より

h(n,m)=f(n,m+1)*g(n,m)

従って(4),(6)より

h(n-1,m-1)=f(n-1,m)*g(n-1,m-1)

={2mu/(h(n-1,m)-2m)+u}*{h(n,m)-2m)/(u+1)}

従って右辺中のh(n,m)を左辺に持ってくれば,

h(n,m)

=2m+(u+1)h(n-1,m-1)/{2mu/(h(n-1,m)-2m)+u}

=2m+(u+1)h(n-1,m-1)/{uh(n-1,m)/(h(n-1,m)-2m)}

(最後の変換は2mu/(h(n-1,m)-2m)の割り算により小さな値が発生するのを避けるために行なった)．

従ってh(n,m)は下記により定義できる．

(1)h(n,m)=∞　if m<1 or n<2m

(2)h(n,1)=2　if n>1.

(3)h(n,m)=n+2h(n-1,m-1)　if n=2m.

(4)h(n,m)=2m+(u+1)h(n-1,m-1)/{uh(n-1,m)/(h(n-1,m)-2m)}　otherwise

ただしu=n-2m+1

次に上のh(n,m)の定義式に2mF(n,m)を代入した後,両辺を2mで割れば定理1におけるF(n,m)が得られる．

付録2 ランダム順列のO(n-m)スペース O((n-m)m)時間計算法

集合{i,...,n}のランダム順列(長さn-i+1,葉数m1)を葉数を制御しながら生成する際に,葉数を制御して生成された集合{i+1,...,n}のランダム順列(長さn-i)の葉またはu節に要素iを挿入する必要がある.挿入箇所を決める確率Pの計算の仕方には2通りある.

(1)順列を作りながら計算する.

(2)すべてのiについて,要素iを挿入する際の確率を前もって計算しておき,その結果を配列(例えばleafp(i)に格納しておく).順列を作る際にはその配列の値を調べる.

以下では後者の方法用に配列を作成する方法を述べる.挿入個所が葉かu節かを決めるには下記の関数setleafpを利用した次の手順を用いればよい(setleafpの中ではF(n,m)はD(n-m)に格納されていることに注意).この関数は新しい要素iを葉に付ける場合には配列要素leafp(i)を1に,そしてu節に付ける場合にはleafp(i)を0に設定する:

(1)m1=1ならleafp(i)=1

(2)n-i+1=2m1-1ならleafp(i)=0

(3)その他の場合は,0<r≦1なる乱数rを生成し,1/F(n-i+1,m1)≦rならleafp(i)=1,そうでなければleafp(i)=0とする.即ち

```
setleafp:
  if m1=1 then
     leafp(i)=1
  elseif n-i+1=2*m1-1 then
     leafp(i)=0
  elseif rnd≦1/D(n-i+1-m1) then
     leafp(i)=1
  else
     leafp(i)=0
  end if
```

leafp(1),...,leafp(n-1)の値が決まれば長さn葉数mのランダム順列が生成できる事は明らかである.以下ではn-mスペースを用いたO(m*n)時間のleafp決定手順を示す.

まずはじめにn-mスペースO(m*n)時間でF(n,m)を計算する手順を図2に示す.これは配列Hを利用してF(n,m)の計算から再帰除去をしたものである.そこではF(n,m)をH(n-m)に格納することに注意されたい.

長さn葉数mの順列を何回も繰返し生成するためにHは保存しておく.そしてn>iなるiに対するleafp(i)に必要な確率の計算をH(n-m)から逆向きに進める(図3参照).

その計算手順を図4に示した.そこではF(n,m)はD(n-m)に作られる.Fの定義よりF(n'-1,m-1)=(m*u*(F(n',m)-1)*F(n'-1,m))/((m-1)*(u+1)*(F(n'-1,m)-1)),m1=m-1,n'-1=n-i+1およびu=n'-2m+1である.またD(j)の作り方からD(j)=F(n',m)かつD(j-1)=F(n'-1,m)である.これよりu=j-m1およびPAD中のD(j)の式を得る.ただし本文で述べたように,このままで計算すると桁落ちにより深刻な計算誤差が発生するので実際の計算を系1で述べたF1を用いて行なう必要がある.

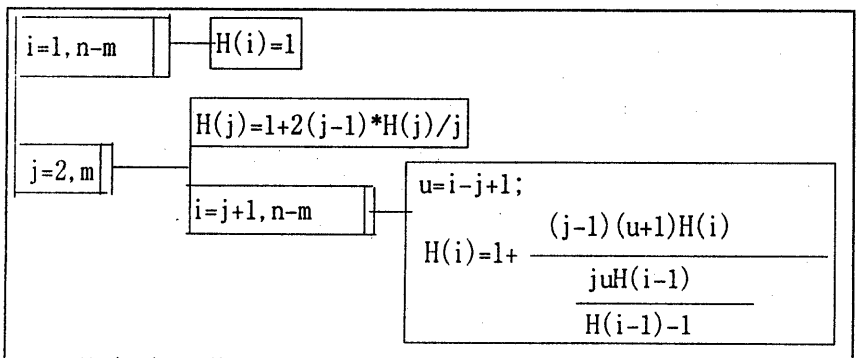

図2:関数F(n,m)を計算する反復型プログラム(配列H(n-m)にF(n,m)の値が入る).

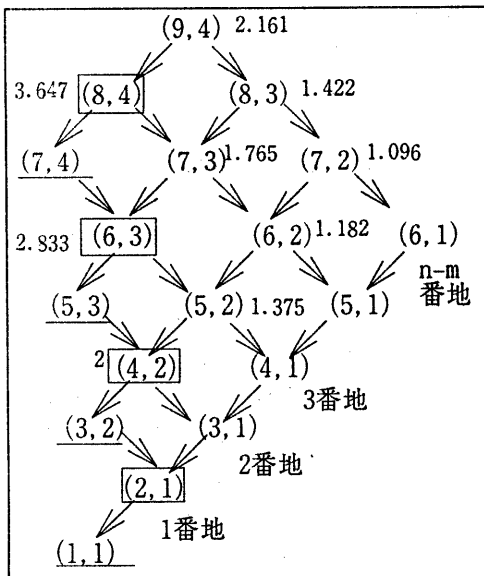

図3：下線を引かれた節に対するHの値は実際には計算されない。また四角で囲まれた節に対する値は壊されずにH中に残される。節に付けられた小さい数字はF(n,m)の値である。ちなみに
F(1998,99)=1.000006,
F(3001,1500)=608.44,
F(4001,2000)=811.0832

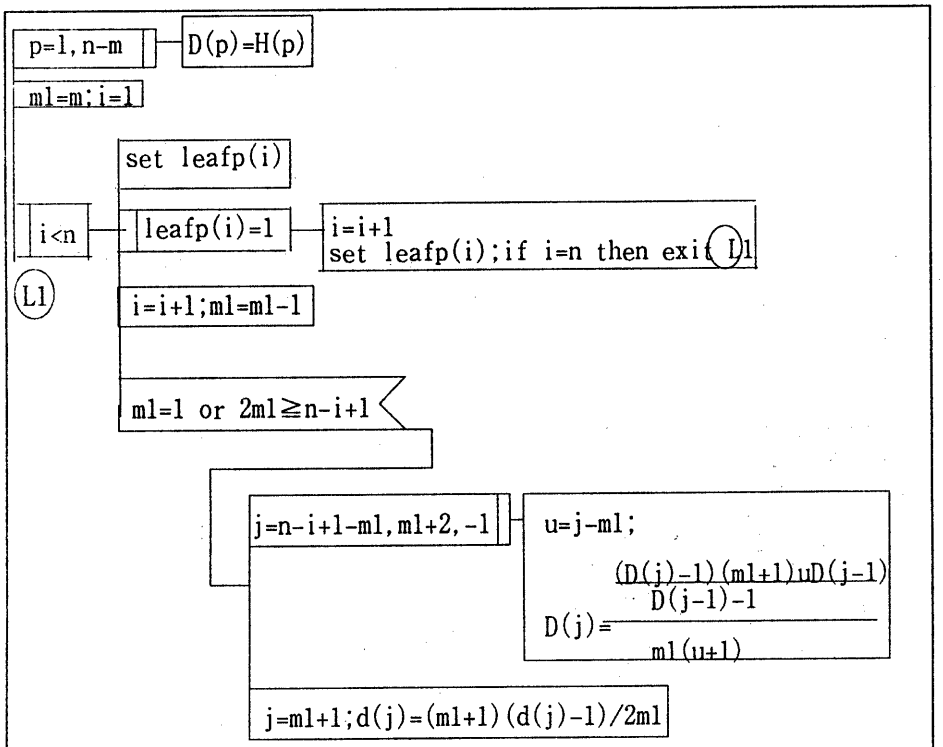

図4:追加する要素iを葉に付けるかu節に付けるかを決定するプログラム