整列法評価のためのランダム順列の生成

6F－5

二村良彦　青木健一　遠藤貢一　大谷啓記　白井千恵子
早稲田大学　理工学部

1.はじめに

整列法(ソーティングアルゴリズム)の性能評価のための入力列として、従来は多くの場合に一様乱数列が利用されてきた。しかし現実の問題としてソーティングを扱う場合には、与えられた数列はほとんど整列済みであることが多いと言われている[6]。従って、ソーティングアルゴリズムの性能評価をする際には、入力列の整列度合を表す各種の事前整列性測度[3]を制御しながら入力列を生成する必要がある。実際ある種の整列性測度で見ると一様乱数は非常に偏った値を持ち、整列法の性能評価には不適である[4]。

本稿では(1)事前整列性測度のうち、葉数(数列内で隣接する要素より小さい要素の個数)を制御し、任意に与えられた葉数mを持つ順列を一様に生成する方法、(2)生成された順列の一様性を検定する方法および(3)検定結果について報告する。葉数の事前整列性測度としての妥当性については[4]で述べた。

2.葉数を制御したランダム順列の生成法

REM(数列の長さ－数列に含まれる最長上昇列の長さ)および撹乱要因(葉数－1)等の事前整列性測度については、それらを制御して生成された順列を用いた整列法の性能評価実験が既に報告されている[1, 2]。しかしそこでは生成された順列の妥当性(指定された測度を持った順列が、偏った性質を持たないこと)に関する議論はなされていない。

ここでは、長さn、葉数mの順列をk個生成する3つの方法について報告する。そのうちの2つはO(n*max(m*k))時間かつn*mスペース、およびO(k*n*m)時間かつn-mスペースで、順列を一様に生成することが理論的に保証されている。残りの1つはO(k*n)時間かつnスペースであるが、一様性を保証しない簡便法である。

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。ただし、u節とは、隣接する要素の片方のみが自分より大きい要素のことである。

最初の方式は、下記の性質1に基づく。

性質1：集合{1,2,3,･･･,n}の順列tの葉数がm、かつtはt'に1を挿入して作られているとする。この時、1がt'の葉についている確率P(n,m)は、2m*L(n-1,m)/L(n,m)である。ただしL(n,m)は長さn葉数mの順列の総数を表す[4]。

方式1：長さn葉数mの順列を次の再帰的手順により作る：(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、葉数mのt'を生成し、t'の葉に1を挿入する。r>P(n,m)ならば、葉数m-1のt'を生成し、t'のu節に1を挿入する。挿入箇所はランダムに選ぶ。

ここで大きなnに対しては、L(n,m)がオーバーフローにより計算不可能であるが、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)を直接計算しないことができる[4]。

定理1：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
ただしただしu=n-2m+1。（証明は省略）

n*mの配列を利用した動的計画法により、F(n,m)はO(n*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おいては再計算の必要が無い(ただしmが小さい場合にF(n-1,m)-1で桁落ちが起こり、深刻な数値誤差が発生するので注意)。

上記の事柄により、方式1ではO(n*max(m,k))時間で順列をk個生成することができる。しかし、整列法の評価のように大きなnを扱う場合にはスペースネックになる。そこで、次の方式2が有効になる。

方式2：まずn-mスペースの方法でF(n,m)を計算し、次に長さn葉数mの順列を次の再帰的手順により作る：(1)0≦r≦1なる一様乱数rを生成する。(2)r≦P(n,m)なら葉数mのt'を生成し、t'の葉に1を挿入する。r>P(n,m)なら葉数m-1のt'を生成し、t'の

Random Permutation Generator for Evaluating Sorting Algorithms, by Yosihiko FUTAMURA, Kenichi AOKI, Koichi ENDO, Hirofusa OTANI, Chieko SHIRAI, School of Science and Engineering, Waseda University

u節に1を挿入する。(3)P(n,m)およびn-m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。

上記の方式2は、O(k*n*m)時間である。

次に、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の順列を次の手順により作る：(1)要素nだけからなる長さ1の順列をsとする。(2)sの長さがnより小さい間は処理(3)を行う。sの長さがnになったら処理を終了する。(3)n-(sの長さ)をiとする。sの葉数がmの時、あるいはsがu節を持たなければ、iをsの葉に挿入する。sの葉数がm-iならば、iをsのu節に挿入する。その他の場合は処理(4)を行う。(4)sの葉数およびu節の数をそれぞれm'およびu'とする。この時、iを2 m'/(2m'+u')の確率でsの葉に挿入し、u'/(2 m'+u')の確率でsのu節に挿入する。

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、指標の列についてカイ2乗検定を行った。またその指標列の葉数も調べた。(一様乱数列の葉数は約n/3。ただしnは乱数列の長さ[4]。)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、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,...,|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。

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（$0 \leq u_0 \leq n-2m$）かまたは左からm_0番目の葉位置（$0 \leq m_0 \leq 2m-1$）に付いているものとする。また$I(n)=1$かつ$u_1=n-2m+2$

$I(t)=I(t')+u_0\, L(n-1,m-1)$ if $t' \in T(n-1,m-1)$

$I(t)=I(t')+u_1 L(n-1,m-1)+m_0 L(n-1,m)$

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

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50000個生成し、指標付け関数Iにより指標付けして1と2880の間の乱数列とし、カイ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16667)。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の順列をk個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	2861.4	16691.0
2	k*n*m	n－m	2861.4	16691.0
3	k*n	n	11178.2	16708.3
浅野	k*n*log n	n	25140.0	16652.3

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])。しかし方式2では、確率の計算式中で桁落ちによる数値誤差が発生し、大きなnとm（例えばn=4095,m=512)に対しては正確な計算が目下不可能である。誤差を補正しながら計算すると最悪$O(k*n*m^2)$時間かかる。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。しかし例えば長さ4095かつ葉数512の順列は4000！個のオーダーあるので、生成された順列の一様性を検定することは不可能である。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では結果に差が見られなかった[5]。

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ことが期待される高速な簡便生成法も示した。生成法の高速化と桁落ちの防止が今後の課題である。

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]ESTIVILL-CASTRO and WOOD:A survey of adaptive sorting algorithns,ACM Computing Surveys,Vol.24,No.4,December,1992.

[3]COOK and KIM:Best Sorting Algorithm for Nearly Sorted Lists,C.ACM,Vol.23,No.11,1980.

[4]二村,二村,遠藤,平井:LOAS:葉数に関して最適な適応整列法、日本ソフトウェア科学会第11回大会C1-1、1994.

[5] 二村,遠藤,平井,青木:適応整列法の評価,情報処理学会第50回大会4J-10,1995.

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

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