THE RANDOM GENERATION OF UNDERDIAGONAL WALKS

E. Barcucci, R. Pinzani, R. Sprugnoli
Dipartimento di Sistemi e Informatica
Università di Firenze, Firenze (Italy)

ABSTRACT

In this paper, we propose an algorithm for the random generation of underdiagonal walks. We consider the plane walks which are made up of different kinds of east, north-east and north steps and which start from the origin and remain under the main diagonal. The algorithm is very simple: it randomly generates plane walks and refuses the walks crossing the diagonal. We prove that the algorithm works in linear time with respect to the walks' length when the number of different kinds of east steps is greater than, or equal to, the number of different kinds of north steps. Finally, some results of our experiments are reported in order to support our theoretical results with empirical evidence.

1. INTRODUCTION

Various studies (see, for instance, [5, 6, 7]) have been made on one-dimensional walks made up of three kinds of unitary steps: right (r), left (l) and in-place (s). Particular attention has been given to walks that begin at the origin O and never go to its left. This means that in each subwalk which begins at the origin O, the number of right-hand steps must be greater than, or equal to, the number of left-hand steps.

These walks can be codified by means of some words defined on an alphabet having three symbols, such as $\{r, l, s\}$. If we denote the number of letters r (l) in word w by $|w|_r$ ($|w|_l$), then the language of the words codifying the walks is:

$$\mathcal{L} = \{w \in \{r, l, s\}^* \mid |w'|_r \geq |w'|_l \text{ for each } w' \text{ prefix of } w\}$$

The words $w \in \mathcal{L}$ are also called Motzkin left factors because they are the prefixes of the Motzkin words, i.e., words $w \in \{r, l, s\}^*$, such that $|w|_r = |w|_l$ and $|w'|_r \geq |w'|_l$ for each w' prefix of w. This kind of walk has been very thoroughly studied because these walks correspond biunivocally to single-rooted directed animals [6].

The walks on the line can also be represented two-dimensionally by recording the time on the abscissa and the moves on the ordinates. As a result, the elementary steps corresponding to the right, left and in-place steps are north-east, south-east and east, respectively. For example, the *rrslsrl* walk can be represented as in Figure 1. In this case, the condition that a one-dimensional walk mustn't go to the left of the origin means that it mustn't go under the x axis.

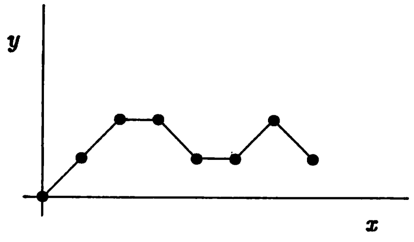

Fig. 1 - The 7-walk corresponding to *rrslsrl*

Another way of representing this kind of walk on a plane is to consider the east, north, and north-east steps as elementary steps corresponding to r, l, s. In this case, the walk corresponding to the preceding example is represented in Figure 2.

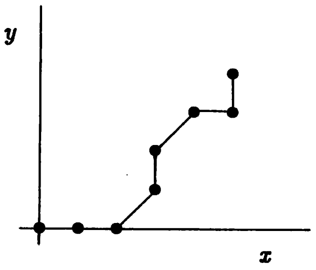

Fig. 2 - The underdiagonal walk corresponding to *rrslsrl*

In this case, the condition that the one dimensional walk mustn't go to the left of O means that the walk mustn't go over the $y = x$ diagonal. This is why these walks are also called ***underdiagonal*** walks.

In [1] we introduced a linear algorithm for the random generation of single-rooted directed animals whose first step is to generate a word in the $\mathcal{L}$ language. In the present study, we examine a more general class of underdiagonal walks made up of several kinds of east, north and north-east steps and we introduce an algorithm that generates them randomly. This algorithm is shown to be linear when the number of east steps is greater than, or equal to, the number of north steps.

2. UNDERDIAGONAL WALKS

We examine underdiagonal walks in which a kinds of east steps, b kinds of north-east steps and c kinds of north steps can be used. This type of walk is usually called *coloured walk*. Walks made up of n steps, (n-long walks or n-walks), can be codified by n-long words (n-words) on the alphabet $\mathcal{A} = \{x_1, x_2, \ldots, x_a, z_1, \ldots, z_b, y_1, \ldots, y_c\}$, in which the x_i, z_i and y_i represent the east, north-east and north steps.

For a walk to be underdiagonal, the number of north steps in each subwalk beginning at the origin must be less than, or equal to, the number of east steps. Consequently, if $w \in \mathcal{A}^*$ is a word that codifies an underdiagonal walk, we obtain

$$\sum_{k=1}^{c} |w'|_{y_k} \leq \sum_{k=1}^{a} |w'|_{x_k}$$

for each w' prefix of w.

It is then relatively simple to provide a non-ambiguous context-free grammar that generates the language $\mathcal{S}$ of the words codifying the underdiagonal walks. The non-terminal symbol D corresponds to the underdiagonal walks ending on the main diagonal; as noted previously, the x and y symbols should balance. The non-terminal symbol F corresponds to the walks not ending on the main diagonal; each of these walks can be uniquely decomposed into a sequence of D-walks joined together by some positive number of x steps. The total number of these x steps represents the distance of the ending point of the walk from the main diagonal. Finally, the non-terminal symbol S corresponds to all the underdiagonal walks, each of which may end on the main diagonal (a D-walk) or not (an F-walk).

$$\begin{aligned}
&S ::= D \mid F\\
&F ::= DAD \mid DAF\\
&D ::= \epsilon \mid BD \mid ADCD\\
&A ::= x_1 \mid x_2 \mid \ldots \mid x_a\\
&B ::= z_1 \mid z_2 \mid \ldots \mid z_b\\
&C ::= y_1 \mid y_2 \mid \ldots \mid y_c
\end{aligned}$$

By applying Schutzenberger's method [8], we obtain the following system:

$$\begin{aligned}
&S(t) = D(t) + F(t)\\
&F(t) = A(t)D^2(t) + A(t)D(t)F(t)\\
&D(t) = 1 + B(t)D(t) + A(t)C(t)D^2(t)
\end{aligned}$$

$$A(t) = at$$
$$B(t) = bt$$
$$C(t) = ct$$

from which we can get $D(t)$ and $S(t)$. The expression for $D(t)$ is the following:

$$D(t) = \frac{1-bt-\sqrt{\Delta}}{2act^2}$$

in which $\Delta = (1-(b-2\sqrt{ac})t)\,(1-(b+2\sqrt{ac})t)$.

$D(t) = \sum_n d_n t^n$ is the generating function of the words that codify the underdiagonal n-walks that end on the diagonal, that is, the w words, such that:

$$\sum_{k=1}^{c} |w|_{y_k} = \sum_{k=1}^{a} |w|_{x_k} \quad \text{and}$$

$$\sum_{k=1}^{c} |w'|_{y_k} \leq \sum_{k=1}^{a} |w'|_{x_k} \text{ for each } w' \text{ prefix of } w$$

It is worth noting that in the $D(t)$ function, a and c only appear as the product ac. This implies that the number of n-walks with a east steps, b north-east steps and c north steps is equal to the number of n-walks with c east steps, b north-east steps and a north steps. In order to verify this, it is sufficient to run the walks backwards.

For the generating function of the underdiagonal n-walks we obtain the following expression:

$$S(t) = \frac{1}{2at}\,\frac{1-(b+2a)t-\sqrt{\Delta}}{(a+b+c)t-1}$$

Because of this generating function, when $a,\ b,\ c > 0$ there is no closed form for $s_n = [t^n]\ S(t)$. Therefore, we now determine an asymptotic expression for s_n which will be important for our remarks on the random generation of underdiagonal walks in Section 3. Since:

$$s_n = \frac{1}{2a}\,[t^{n+1}]\,\frac{1-(b+2a)t-\sqrt{\Delta}}{(a+b+c)t-1} \tag{2.1}$$

let us consider this last function. It presents three singularities: a pole at $t = (a+b+c)^{-1}$ and two algebraic singularities at $t = (b+2\sqrt{ac})^{-1}$ and $t = (b-2\sqrt{ac})^{-1}$. By our hypothesis on $a,\ b,\ c$, the last singularity cannot have minimum modulus; besides, since $a+c-2\sqrt{ac} = (\sqrt{a}-\sqrt{c})^2 \geq 0$, we always have $a+b+c \geq b+2\sqrt{ac}$, and the equality holds if and only if $a = c$. So $t = (a+b+c)^{-1}$ is the singularity of minimum modulus unless $a = c$. We have:

$$\sqrt{\Delta}\,\Big|_{t=1/(a+b+c)} = \frac{|c-a|}{a+b+c}$$

and when $c > a$ the numerator in (2.1) is annulled for $t = (a+b+c)^{-1}$. Hence, for $c > a$, this quantity is not a pole for the function $S(t)$. We thus have three different cases, according to whether a is greater than, equal to or less than c.

<u>case $a > c$</u>

Since in this case $t = (a+b+c)^{-1}$ is a first order pole, we can apply Darboux's method and we immediately obtain:

$$s_n \sim \frac{1}{2a}\left(1-\frac{b+2a}{a+b+c}-\frac{a-c}{a+b+c}\right)(a+b+c)^{n+1} = \frac{a-c}{a}(a+b+c)^n \qquad (2.2)$$

We could now remove the pole from the generating function and obtain a more precise approximation for s_n by the method of subtracted singularities. However, as we shall see in Section 3, formula (2.2) is usually sufficient for our purposes.

<u>case $a = c$</u>

In this case, $a+b+c = b+2\sqrt{ac} = b+2a$ and $\Delta = (1-(b-2a)t)(1-(b+2a)t)$. The function $S(t)$ simplifies to:

$$S(t) = \frac{1}{2at}\left(\sqrt{\frac{1-(b-2a)t}{1-(b+2a)t}}-1\right)$$

The minimum modulus singularity is now the algebraic singularity at $t=(b+2a)^{-1}$. In this case, we need a more precise approximation of s_n and, therefore, instead of applying the Darboux's method, we develop $S(t)$ around the dominating singularity and obtain an asymptotic development:

$$tS(t) \sim \frac{1}{2a}\sqrt{\frac{1-(b-2a)t}{1-(b+2a)t}} \sim$$

$$\sim \frac{1}{a}\sqrt{\frac{a}{b+2a}}\left((1-(b+2a)t)^{-1/2}+\frac{b-2a}{8a}(1-(b+2a)t)^{1/2}-\frac{(b-2a)^2}{128a^2}(1-(b+2a)t)^{3/2}\right)$$

We can now extract the coefficient of t^{n+1} from this expression:

$$s_n \sim \frac{1}{\sqrt{a(b+2a)}}\frac{(b+2a)^{n+1}}{\sqrt{\pi(n+1)}}\left(1-\frac{b}{16a(n+1)}+\frac{16a^2-3b^2}{512a^2(n+1)^2}\right) \qquad (2.3)$$

When $a = b = c = 1$, this formula coincides with the one found in [1], which approximates the number of single-rooted directed animals having $n+1$ nodes.

<u>case $a < c$</u>

This is the most difficult case. As observed before, the dominating singularity is the algebraic singularity at $t = (b+2\sqrt{ac})^{-1}$. As in the case $a = c$, the best way to obtain an asymptotic approximation for s_n is to develop $S(t)$ around this singularity. The use of a formal system, such as MAPLE [3], may help in performing the necessary computations. In any case, we obtain the following asymptotic development:

$$tS(t) \sim -\frac{4\sqrt{ac}\sqrt{b+2\sqrt{ac}}}{a(\sqrt{c}-\sqrt{a})^2}(1-\gamma t)^{1/2}\left(1+\left(\frac{b-2\sqrt{ac}}{8\sqrt{ac}}+\frac{a+b+c}{(\sqrt{c}-\sqrt{a})^2}\right)(1-\gamma t)+\right.$$

$$\left.+\left(\frac{(a+b+c)^2}{(\sqrt{c}-\sqrt{a})^4}+\frac{b-2\sqrt{ac}}{8\sqrt{ac}}\frac{a+b+c}{(\sqrt{c}-\sqrt{a})^2}-\frac{(b-2\sqrt{ac})^2}{128ac}\right)(1-\gamma t)^2+\ldots\right)$$

where γ denotes the reciprocal of the dominating singularity, i.e., $\gamma = b+2\sqrt{ac}$. In order to simplify our notations, let us set:

$$K=\frac{4\sqrt{ac}\sqrt{b+2\sqrt{ac}}}{a(\sqrt{c}-\sqrt{a})^2} \qquad C_1=\frac{b-2\sqrt{ac}}{8\sqrt{ac}}+\frac{a+b+c}{(\sqrt{c}-\sqrt{a})^2}$$

$$C_2=\frac{(a+b+c)^2}{(\sqrt{c}-\sqrt{a})^4}+\frac{b-2\sqrt{ac}}{8\sqrt{ac}}\frac{a+b+c}{(\sqrt{c}-\sqrt{a})^2}-\frac{(b-2\sqrt{ac})^2}{128ac}$$

We then find the asymptotic approximation:

$$s_n=\frac{2K}{n+1}\binom{2n}{n}\left(\frac{\gamma}{4}\right)^{n+1}\left(1-\frac{3C_1}{2n-1}+\frac{15C_2}{(2n-1)(2n-3)}+O(n^{-3})\right)=$$

$$=\frac{K}{2}\frac{(b+2\sqrt{ac})^{n+1}}{(n+1)\sqrt{\pi n}}\left(1-\frac{1+12C_1}{8n}+O(n^{-2})\right) \qquad (2.4)$$

The main value will be used in Section 3, whilst the corrections will be useful in the remarks of Section 4.

3. RANDOM GENERATION OF UNDERDIAGONAL WALKS

In order to generate a random underdiagonal n-walk, we only have to generate a random n-word in $\mathcal{S}$. We propose the following algorithm:

1) The letters of the word are generated one after another by taking them out of $\{x_1, x_2, \ldots, x_a, z_1, \ldots, z_b, y_1, \ldots, y_c\}$ (for example, by generating a random integer number k included between 1 and $a+b+c$ and then by taking x_k if $k \leq a$, z_{k-a} if $a < k \leq a+b$ and y_{k-a-b} if $k > a+b$);

2) The difference $\sum_{i=1}^{a} |w|_{x_i} - \sum_{j=1}^{c} |w|_{y_j}$ is taken into account;

3) If this difference becomes negative before generating n letters, the prefix generated up to then is discarded and we start from the beginning again;

4) If n letters are generated before the difference is less than zero, they constitute the word desidered.

We want to calculate the average number of generated letters necessary for obtaining an n-word (i.e., the expected number of calls to a random number generation routine *random*). It will be shown that if $a \geq c$, the algorithm is linear and that it is not so if $a < c$.

Generally speaking, if we want to obtain an n-word in $\mathcal{S}$, some words shorter than n not belonging to $\mathcal{S}$ are generated first and then an n-word in $\mathcal{S}$ is generated.

Let $p_{n,k}$ be the probability that k ($k \geq n$) calls to *random* must be made to generate an n-word in $\mathcal{S}$. If we determine the probability generating function $P_n(t) = \sum_k p_{n,k} t^k$ for each $n \in \mathbb{N}$, we are able to evaluate the average number avg_n of characters generated in order to obtain an n-word in $\mathcal{S}$ because $avg_n = P'_n(1)$. By using the same method, we could also calculate the variance because $var_n = P''_n(1) + P'_n(1) - (P'_n(1))^2$.

We use D_k for denoting the language of the words that codify the k-walks ending on the diagonal, that is the k-words whose number of x letters is equal to its number of y letters. We now go on to examine the languages $N_1, N_2, N_3, \ldots$ of the negative words, that is, the k-words ($k \leq i$ for N_i) that interrupt the generation of a word in $\mathcal{S}$. We have:

$$
\begin{aligned}
&C ::= y_1 \mid y_2 \mid \ldots \mid y_c \\
&N_1 ::= C \\
&N_2 ::= C \mid D_1 C \\
&N_3 ::= C \mid D_1 C \mid D_2 C \\
&\ldots \\
&N_n ::= C \mid D_1 C \mid D_2 C \mid \ldots \mid D_{n-1} C
\end{aligned}
$$

from which we get

$$N_n(t) = c \sum_{k=0}^{n-1} d_k t^{k+1} \qquad \text{where } d_k = [t^k]\, D(t)$$

When we want to generate a word in $\mathcal{S}$ by means of the algorithm, we generate a (possibly empty) sequence of words in N_n followed by a word in $\mathcal{S}$. As a result, the language L_n of the sequence of letters generating the codifications of underdiagonal n-walks is defined by:

$$R_n ::= \epsilon \mid N_n R_n$$
$$L_n ::= R_n S_n$$

in which S_n is the language of n-words in $\mathcal{S}$. We thus get:

$$R_n(t) = \frac{1}{1-N_n(t)} = \frac{1}{1-c\sum_{k=0}^{n-1} d_k t^{k+1}}$$

and

$$L_n(t) = \frac{s_n t^n}{1-c\sum_{k=0}^{n-1} d_k t^{k+1}}$$

Since every letter of $\mathcal{A}$ is generated with a probability of $1/(a+b+c)$, we can obtain the probability generating function as follows:

$$P_n(t) = L_n(t/(a+b+c)) = \frac{s_n t^n}{(a+b+c)^n - c\sum_{k=0}^{n-1} d_k (a+b+c)^{n-k-1} t^{k+1}}$$

We now prove that $P_n(1) = 1$. We can write

$$S(t) = \frac{1}{2at}\frac{1-(b+2a)t-\sqrt{\Delta}}{(a+b+c)t-1} = \frac{1}{1-(a+b+c)t} - ct\frac{1-bt-\sqrt{\Delta}}{2act^2}\frac{1}{(a+b+c)t-1}$$

From this it follows that:

$$s_n = (a+b+c)^n - c\sum_{k=0}^{n-1} d_k (a+b+c)^{n-k-1}$$

and therefore $P_n(1) = 1$.

If we indicate $Q_n(t) = (a+b+c)^n - c\sum_{k=0}^{n-1} d_k (a+b+c)^{n-k-1} t^{k+1}$, we can write $P_n(t) = s_n t^n / Q_n(t)$. Therefore $Q_n(1) = s_n$ and:

$$avg_n = P_n'(1) = \frac{n s_n Q_n(1) - s_n Q_n'(1)}{Q_n^2(1)} = \frac{n s_n^2 - s_n Q_n'(1)}{s_n^2} = n - \frac{Q_n'(1)}{s_n}$$

From the preceding formula for $Q_n(t)$ we immediately find:

$$Q_n'(1) = -c\sum_{k=0}^{n-1}(k+1)\ d_k (a+b+c)^{n-k-1} = -q_n$$

This allows us to go on to the generating function $q(t) = \sum_{k=0}^{\infty} q_n t^n$, which is the convolution of the derivative of $D(t)$ with the geometric series $(1-(a+b+c)t)^{-1}$. In fact:

$$q(t) = ct\frac{\mathrm{d}}{\mathrm{d}t}(tD(t))\,\frac{1}{1-(a+b+c)t} = \frac{1-bt-\sqrt{\Delta}}{2at\sqrt{\Delta}\,(1-(a+b+c)t)} \tag{3.1}$$

Here we used the explicit formula for $D(t)$ found in Section 2 and a great deal of routine computations performed by computer.

As for $S(t)$, here again we have three singularities and since the numerator of $q(t)$ is not annulled for $t = (a+b+c)^{-1}$, it may seem that we only have two cases. Actually, the remarks to be done and the exact results obtained for $a > c$ and $a < c$ differ from each other very significantly and we are now going to examine the three different cases, as we did in Section 2.

<u>case $a > c$</u>

The dominating singularity for $q(t)$ is a first order pole at $t = (a+b+c)^{-1}$. The residual at this point is easily found and we have:

$$q_n \sim \frac{c}{a(a-c)}\,(a+b+c)^{n+1}$$

This value is to be divided by s_n, as computed for the corresponding case in Section 2. We immediately find:

$$avg_n \sim n + \frac{c(a+b+c)}{(a-c)^2} \tag{3.2}$$

that is, the expected number of calls to *random* is n plus a constant per generation.

<u>case $a = c$</u>

In this case, the dominating singularity is the algebraic singularity at $t = (b+2a)^{-1}$, and the formula for $q(t)$ simplifies as follows:

$$2atq(t) = \frac{1-bt}{(1-(b+2a)t)^{3/2}(1-(b-2a)t)^{1/2}} - \frac{1}{1-(a+b+c)t}$$

Again, we develop the first term around the singularity and find:

$$\frac{1-bt}{(1-(b+2a)t)^{3/2}(1-(b-2a)t)^{1/2}} = \sqrt{\frac{a}{b+2a}}\,(1-(b+2a)t)^{3/2}\Big(1+$$

$$+\frac{3b+2a}{8a}\,(1-(b+2a)t) - \frac{(6a+5b)(b-2a)}{128a^2}\,(1-(b+2a)t)^2 + \ldots\Big)$$

We can now extract the coefficient of t^{n+1} and obtain:

$$q_n \sim \frac{2n+3}{2\sqrt{a(b+2a)}}\binom{2n+2}{n+1}\left(\frac{b+2a}{4}\right)^{n+1}\left(1+\frac{3b+2a}{8a(2n+3)}+\frac{(6a+5b)(b-2a)}{128a^2(2n+3)(2n+1)}+\ldots\right)+$$

$$-\frac{(b+2a)^{n+1}}{2a}$$

At this point, obtaining avg_n by formula (3.1) is routine procedure:

$$avg_n = 2n - \frac{1}{2}\sqrt{\frac{b+2a}{a}}\sqrt{\pi(n+1)} + \frac{6a+b}{4a} + o(1) \tag{3.3}$$

This means that the expected number of calls to *random* is something less than $2n$ per generation.

case $a < c$

As in the case of $a > c$, the dominating singularity is the first order pole at $t = (a+b+c)^{-1}$, but the condition $a < c$ implies some difference in the constant. In fact, we have:

$$q_n \sim \frac{(a+b+c)^{n+1}}{c-a}$$

By using the value of s_n found in formula (2.4), we have:

$$avg_n \sim n+(n+1)\sqrt{n}\left(\frac{a+b+c}{b+2\sqrt{ac}}\right)^{n+1}\frac{\sqrt{\pi}\,2a(\sqrt{c}-\sqrt{a})^2}{\sqrt[4]{ac}\,\sqrt{b+2\sqrt{ac}}\,(c-a)}\left(1+\frac{1+12C_1}{8n}\right) \tag{3.4}$$

Consequently, the expected number of calls to *random* grows exponentially with n, and our method cannot be used to randomly generate this kind of underdiagonal walks in an efficient way.

It may be interesting to observe that by using the method of subtracted singularities it is possible to obtain a more accurate approximation of q_n. This explicitly shows that the error introduced by the value of q_n also introduces an error in the order of $O((b+2\sqrt{ac})^n)$, which is exponentially smaller than the main value:

$$q_n \sim \frac{(a+b+c)^{n+1}}{c-a} - \frac{\sqrt{b\sqrt{ac}+2ac}}{2a(\sqrt{c}-\sqrt{a})^2}\left(\frac{b+2\sqrt{ac}}{4}\right)^{n+1}\binom{2n+2}{n+1}+$$

$$+\frac{(c+a)(b+2\sqrt{ac})^{3/2}}{4a\sqrt[4]{ac}\,(\sqrt{c}-\sqrt{a})^4}\left(\frac{b+2\sqrt{ac}}{4}\right)^{n+1}\frac{1}{2n+1}\binom{2n+2}{n+1}$$

4. EXPERIMENTAL RESULTS

We performed a series of experiments both to check the validity of our approach to the random generation of underdiagonal walks and to give empirical evidence that we can actually generate random walks in linear time when $a \geq c$. In our experiments, we use the average value of the difference:

$$\delta(w) = \sum_{i=1}^{a} |w|_{x_i} - \sum_{j=1}^{c} |w|_{y_j}$$

as an indicator to prove the randomness of the generated words; $\delta(w)$ corresponds to the distance from the main diagonal (measured on the x axis) of the end point of the walk codified by w. We computed the average value of $\delta(w)$ for the n-words in $\mathcal{S}$, that is

$$\delta_n = \sum_{w \in S_n} \delta(w)/s_n$$

by using another application of Schutzenberger's method to the grammar of Section 2. We introduce a new indeterminate u, which counts every x symbol positively and every y symbol negatively and we obtain:

$$\begin{aligned}
&S(t,u) = D(t,u) + F(t,u)\\
&F(t,u) = A(t,u)D^2(t,u) + A(t,u)D(t,u)F(t,u)\\
&D(t,u) = 1 + B(t)D(t,u) + A(t,u)C(t,u)D^2(t,u)\\
&A(t,u) = atu\\
&B(t) = bt\\
&C(t,u) = ctu^{-1}
\end{aligned}$$

In this way, the coefficient $s_{n,k}$ of $t^n u^k$ in $S(t,u)$ represents the number of n-walks ending at a distance k from the main diagonal and it is easy to evaluate $S(t,u)$:

$$S(t,u) = \frac{1-(b+2au)t-\sqrt{(1-bt)^2-4act^2}}{2at(atu^2+btu+ct-u)}$$

What we need is the sum $\sum_{k=0}^{n} ks_{n,k}$, representing the total distance from the main diagonal of all the n-long walks. The simplest way to obtain this quantity is to note that:

$$\sum_{k=0}^{n} ks_{n,k} = [t^n]\left.\frac{\partial}{\partial u}S(t,u)\right|_{u=1}$$

By performing the necessary computations, we eventually find:

$$\sum_{k=0}^{n} ks_{n,k} = [t^n]\left(\frac{1}{1-(a+b+c)t} - \frac{(1-(b+2a)t)\left(1-(b+2a)t-\sqrt{\Delta}\right)}{2at(1-(a+b+c)t)^2}\right) \tag{4.1}$$

At this point, we only need to extract the coefficient of t^n from this generating function. The singularities are the same as for $S(t)$, and, therefore, we have three different cases according to whether a is less than, equal to, or greater than c. We discuss these three cases separately.

<u>case $a > c$</u>

The dominating singularity of (4.1) is $t = (a+b+c)^{-1}$, a second order pole. By using Darboux's method, we easily obtain:

$$\sum_{k=0}^{n} ks_{n,k} \sim \frac{a-c}{a}(a+b+c)^{n-1}(n+1) + (a+b+c)^n$$

and, therefore, by dividing by s_n:

$$\delta_n = \frac{a-c}{a+b+c}(n+1) + \frac{a}{a-c} + o(1)$$

a quantity linearly increasing with n.

We performed a series of experiments by generating 100,000 n-words for $n = 1000$ and various combinations of a, b, c ($a > c$). The results are summarized in Figure 3, and the experimental data agree with the expected values to a large extent.

			calls to *random*		distance from the diagonal	
a	b	c	expected	experim.	expected	experim.
2	1	1	1004	1004.38	252.25	252.89
3	1	1	1001.25	1001.25	401.9	401.07
3	1	2	1012	1012.07	169.83	170.70
3	2	2	1014	1014.18	146	146.80
4	1	1	1000.67	1000.67	501.83	500.73
4	2	3	1027.00	1026.88	115.22	117.27
4	3	2	1004.50	1004.52	224.44	224.17
5	1	1	1000.44	1000.44	573.25	571.96
5	3	4	1048	1047.91	88.42	91.38
5	4	3	1009	1009.04	169.33	169.78

Fig. 3 - Experimental results for $a > c$

We wish to point out that, by formula (3.2), when the constant $K = c(a+b+c)(a-c)^{-2}$ is high, and we limit ourselves to experiments with $n \approx K$, we obtain seemingly erroneous results. In Figure 4 we show some examples of this.

				calls to *random*		distance from the diagonal	
a	b	c	n	expected	experim.	expected	experim.
20	20	17	1000	1107.67	1101.81	59.35	63.71
			10000	10107.67	10107.61	533.04	537.88
20	10	18	1000	1216.00	1180.06	51.71	58.18
			10000	10216.00	10213.46	426.81	436.62
20	20	18	1000	1261.00	1207.61	44.52	50.29
			10000	10261.00	10259.92	354.86	363.03
20	1	19	1000	1760.00	1369.73	45.03	50.94
			10000	10760.00	10729.70	270.03	290.13
20	10	19	1000	1931.00	1401.69	40.43	44.75
			10000	10931.00	10880.27	224.10	243.65
20	20	19	1000	2121.00	1433.90	36.97	39.63
			10000	11121.00	11072.43	189.51	206.81

Fig. 4 - Experimental results for $a > c$

<u>case $a = c$</u>

As we already know, when $a = c$ we have $a+b+c = b+2\sqrt{ac} = b+2a$, and (4.1) has an algebraic, dominating singularity at $t = (b+2a)^{-1}$. The same formula (4.1) simplifies and we obtain:

$$\sum_{k=0}^{n} ks_{n,k} = [t^n]\left(\frac{1}{1-(b+2a)t} - S(t)\right) = (b+2a)^n - s_n$$

Then we divide by s_n (formula (3.3)):

$$\delta_n = \frac{(b+2a)^n}{s_n} - 1 \sim \sqrt{\frac{a}{b+2a}}\sqrt{\pi(n+1)}\left(1 + \frac{b}{16a(n+1)} + \frac{5b^2-16a^2}{512a^2(n+1)^2}\right) - 1$$

and this time the average distance from the main diagonal only increases as $\sqrt{n}$ does. This is also the case of Motzkin words, and the previous formula for δ_n coincides with the one given in [1] when we set $a = b = c = 1$.

In this case, too, the experimental results agree with the expected values, as shown in Figure 5. It is worth noting that the words in the first two lines in the table correspond to the directed animals over the square and triangular lattices [7] respectively.

			calls to *random*		distance from the diagonal	
a	b	c	expected	experim.	expected	experim.
1	1	1	1953.19	1956.65	31.38	31.37
1	2	1	1945.92	1846.10	27.04	26.15
1	4	1	1933.82	1926.40	21.90	22.12
1	8	1	1914.83	1914.24	16.74	16.84
10	1	10	1960.90	1959.30	37.70	37.84
10	5	10	1957.30	1945.50	34.47	34.52
10	10	10	1953.19	1955.86	31.38	31.44
10	15	10	1949.42	1948.94	28.98	29.04
20	15	20	1955.20	1946.96	32.82	32.87
20	20	20	1953.19	1948.79	31.38	31.47

Fig. 5 - Experimental results for $a = c$

case $a < c$

As usual, this is the most difficult case. Formula (4.1) can be written as:

$$\frac{2at(1-(a+b+c)t) - (1-(b+2a)t)\left(1-(b+2a)t-\sqrt{\Delta}\right)}{2at(1-(a+b+c)t)^2}$$

where the numerator is annulled for $t = (a+b+c)^{-1}$. By using de l'Hospital's rule, we discover that $t = (a+b+c)^{-1}$ is not a simple pole either, and, therefore, the dominating singularity is $t = (b+2\sqrt{ac})^{-1}$. We can now either expand everything around this point or use the implicit function method as described by Bender (see [2], theorem 5). This only gives the main term but it also somewhat reduces the number of necessary computations, which could be performed by computer. In any case, we find:

$$\sum\nolimits_{k=0}^{n} ks_{n,k} \sim \frac{4\sqrt{ac}\,\sqrt{b+2\sqrt{ac}}}{\sqrt{a}(\sqrt{c}-\sqrt{a})^3}\,\frac{(b+2\sqrt{ac})^{n+1}}{(n+1)\sqrt{\pi(n+1)}}$$

Therefore, we divide by s_n and obtain:

$$\delta_n = \frac{2\sqrt{a}}{\sqrt{c}-\sqrt{a}} + o(1)$$

This means that the average distance from the main diagonal tends to become constant. In Figure 6, we summarize some of our experiments. Obviously, avg_n increases exponentially with n and the generation time may become rather prohibitive. Moreover, we can see in Figure 6 that experimental data are significantly different from expected values but this however is not particularly surprising.

			calls to *random*		distance from the diagonal	
a	b	c	expected	experim.	expected	experim.
5	6	6	39916.4	56133.3	20.95	14.99
7	8	8	9557.0	14736.6	28.97	17.79
9	1	10	8227.9	12767.5	36.97	22.28
9	10	10	4984.7	7978.0	36.97	19.91
14	1	15	3312.5	5370.5	56.98	26.54
14	15	15	2589.7	4210.6	56.98	23.09
19	1	20	2350.4	3833.0	76.99	29.22
19	10	20	2150.2	3502.9	76.98	26.90
19	20	20	2002.7	3315.8	76.98	24.86
19	30	20	1898.4	3140.6	76.99	23.36

Fig. 6 - Experimental results for $a < c$

Let us consider the correction introduced in formula (3.4); it gives us an idea of the values of n, for which it is possible to obtain some agreement between experimental data and both expected values.

We can claim that experimental data and expected values agree when, for example, the relative error is within 12.5% or 1/8. So we should have:

$$\frac{12C_1+1}{8n} < \frac{1}{8}$$

and we can define n_{min} as the minimum value of n for which this inequality holds. So n_{min} represents the minimal length of the words to be generated in order to achieve the agreement required. By feeding n_{min} into formula (3.4), we find the average number of calls to *random* necessary for generating a word of length n_{min}.

Unfortunately, $n_{min} = \lceil 12C_1+1 \rceil$ is minimized when c is much larger than a, but in this case the quantity $(a+b+c)/(b+2\sqrt{ac})$ is maximized and we have to perform a large number of calls to *random*. For example, let us consider the case $a = b = 1$. When $c = 2$, we find $n_{min} = 279$ but we also have $avg_n > 262{,}000{,}000$. Consequently, it is practically impossible to generate a single word, not to say 10,000 words. If we increase c to 9, we find $n_{min} = 32$ (a very small quantity!) but $avg_n > 216{,}000{,}000$. The (relatively) best situation is achieved for $c = 4$ when we have $n_{min} = 71$ and avg_n

something more than 112,000,000 calls to *random*. In any case, we must conclude that it is actually impossible to provide empirical evidence for our formula (3.4).

5. CONCLUSIONS

We have shown that a very simple algorithm allows us to generate a random underdiagonal walk with a kinds of east steps, b kinds of north-east steps and c kinds of north steps in expected linear time, whenever $a \geq c$. This actually means that m generations of n-walks are performed in time $O(mn)$. Besides, no extra space is required by the generation routine, except for the space required for the string to be produced.

When $a < c$, the same algorithm works in exponential time and hence it is of no practical interest. This is so because there are some general methods for randomly generating n-strings in context-free languages, which perform in polynomial time (see Cohen and Hickey [4]). Therefore, finding out a linear time algorithm for generating random underdiagonal walks with $a < c$ is still an open question.

REFERENCES

[1] E. Barcucci, R. Pinzani, R. Sprugnoli, "Génération aléatoire des animaux dirigés", Atélier Franco-Québecois de Combinatoire, Bordeaux, 1991.

[2] E. A. Bender, "Asymptotic methods in enumeration", SIAM Rev., 16 (1974), 485-514.

[3] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, S. M. Watt, Maple V: Language Reference Manual, Springer-Verlag, New York, 1991.

[4] J. Cohen, T. Hickey, "Uniform random generation of strings in context-free languages", SIAM J. Comput., 12 (1983), 645-655.

[5] R. Donaghey, L. W. Shapiro, "Motzkin numbers", Journal of Combinatorial Theory A, 23 (1977), 291-301.

[6] D. Gouyou-Beauchamps, G. Viennot, "Equivalence of the two-dimensional directed animals problem to a one-dimensional path problem", Advances in Applied Mathematics, 9 (1988), 334-357.

[7] J. G. Penaud, "Une nouvelle bijection pour les animaux dirigés", Rapport LaBRI 89-45, Bordeaux.

[8] M. P. Schutzenberger, "Context-free languages and pushdown automata", Information and Control, 6 (1963), 246-264.