Universidade de São Paulo

Instituto de Física de São Carlos

Introdução à Física Computacional

—

Projeto 2

Levy Bruno do Nascimento Batista − 11212550

Prof. Francisco Castilho Alcaraz

São Carlos

09/2021 − 10/2021

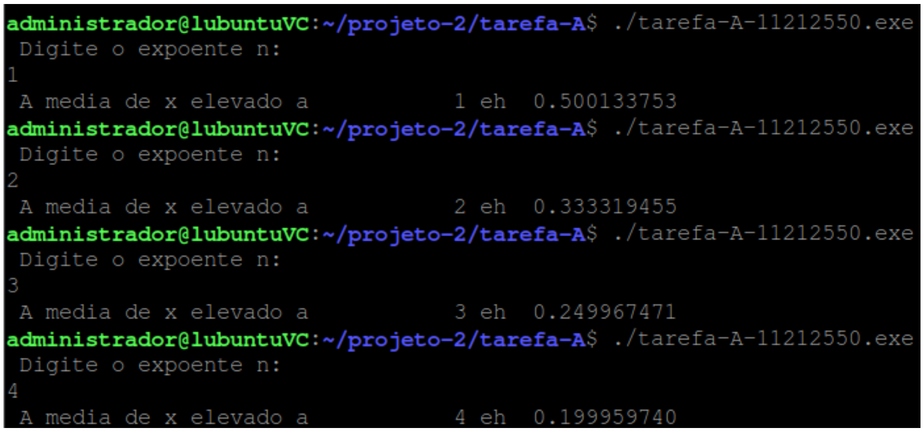
1 Tarefa A

Nessa primeira tarefa, foi escrito um código capaz de calcular a média da distribuição originada a partir do gerador de números pseudo-aleatórios associado à função rand(); no caso, a *seed* utilizada, de valor 5, foi definida como um parâmetro, assim como o número de valores utilizados para calcular a média, sendo m = 1000000. A entrada do programa consiste no expoente inteiro de x^n, e as respectivas saídas para os 4 primeiros inteiros estão mostradas abaixo.

tarefa-A-11212550.f:

```
      program tarefaA

c     o valor da seed foi definido como parâmetro e pode ser alterado abaixo
      parameter (m = 1000000, iseed = 5)
      write(*,*)'Digite␣o␣expoente␣n:'
      read(*,*)n
      valor = rand(iseed)

      xmedia = 0
      do i = 1, m
c       o valor de n é informado no terminal pelo usuário
        xmedia = xmedia + valor**n
        valor = rand()
      end do

      write(*,*)'A␣media␣de␣x␣elevado␣a', n, 'eh', xmedia/m

      end
```

Figura 1: Saídas do programa para n = 1, 2, 3 e 4.



```
administrador@lubuntuVC:~/projeto-2/tarefa-A$ ./tarefa-A-11212550.exe
 Digite o expoente n:
1
 A media de x elevado a           1 eh  0.500133753
administrador@lubuntuVC:~/projeto-2/tarefa-A$ ./tarefa-A-11212550.exe
 Digite o expoente n:
2
 A media de x elevado a           2 eh  0.333319455
administrador@lubuntuVC:~/projeto-2/tarefa-A$ ./tarefa-A-11212550.exe
 Digite o expoente n:
3
 A media de x elevado a           3 eh  0.249967471
administrador@lubuntuVC:~/projeto-2/tarefa-A$ ./tarefa-A-11212550.exe
 Digite o expoente n:
4
 A media de x elevado a           4 eh  0.199959740
```

Fonte: gerado pelo autor

Esses resultados estão de acordo com aqueles que são esperados para uma distribuição contínua de 0 a 1, em que qualquer número neste intervalo tem a mesma probabilidade de ser "sorteado". Note que essas médias podem ser calculadas a partir da seguinte expressão.

$$< x^n >= \int_0^1 x^n dx = \frac{1}{n+1} \tag{1}$$

2 Tarefa B

2.1 B1

Já nessa tarefa, a ideia é analisar a distribuição de $M = 1000000$ de andarilhos após darem $N = 1000$ passos, no caso unidimensional. Veja que "M"e "N"são definidas como parâmetros, assim como a probabilidade p de um andarilho dar um passo à direita, que nessa primeira parte é $p = \frac{1}{2}$. A distribuição será representada pelo *array* "ipos", de forma que "2l"indica uma possível posição depois de N passos e "ipos(l)"a respectiva quantidade de andarilhos nessa posição, sendo "l"o índice que varia de $-\frac{N}{2}$ a $\frac{N}{2}$. Perceba que, como um andarilho só pode parar em uma posição par, é preferível utilizar um *array* com esses limites ao invés de -N a N.

`tarefa-B-11212550.f`:

```
      program tarefaB

c     M -> número de andarilhos, N -> número de passos
c     p -> probabilidade de dar um passo à direita
      parameter (M = 1000000, N = 1000, p = 1.0/2)
      integer*8 l, ipos
c     array que guarda a quantidade de andarilhos em cada posição
      dimension ipos(-N/2:N/2)
      open(1, file='1saida-B-11212550')
      xmed = 0
      xqmed = 0
      do k = -N/2, N/2
        ipos(k) = 0
      end do

      do i = 1, M
        ix = 0
c       a seed muda a cada iteração para gerar um conjunto diferente
        valor = rand(i)
        do j = 1, N
c         verifica para qual direção será o passo do andarilho
          if (valor.lt.p) then
            ix = ix + 1
          else
            ix = ix - 1
          end if
          valor = rand()
        end do
c       atualiza a quantidade de andarilhos na posição que o atual parou
        ipos(ix/2) = ipos(ix/2) + 1
      end do

      do l = -N/2, N/2
c       calcula <x> e <x  >
        xmed = xmed + 2*l*ipos(l)
        xqmed = xqmed + (2*l)**2*ipos(l)
        write(1,*)2*l, ipos(l)
      end do

      write(*,*)'O␣valor␣medio␣de␣x␣eh:', xmed/M,
     +          'e␣o␣de␣x  ␣eh:', xqmed/M

      close(1)
      end
```

Depois de posicionar aleatoriamente todos os andarilhos, o programa calcula os valores de $< x >$ e $< x^2 >$ para o p dado, sendo a saída desse primeiro teste mostrada a seguir.

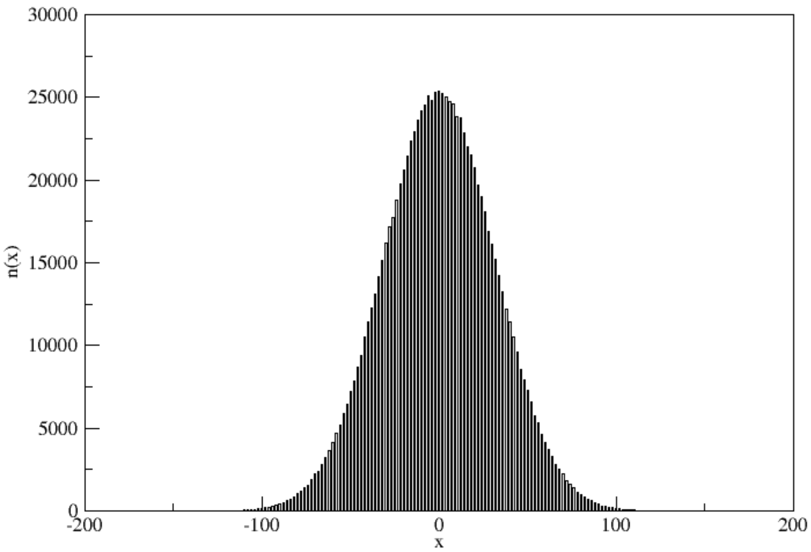
Figura 2: Saída para $p = \frac{1}{2}$.

```
administrador@lubuntuVC:~/projeto-2/tarefa-B$ gfortran -o tarefa-B-11212550.ex
e tarefa-B-11212550.f
administrador@lubuntuVC:~/projeto-2/tarefa-B$ ./tarefa-B-11212550.exe
 O valor medio de x eh:   2.28100009E-02 e o de x² eh:   996.166443
```

Fonte: gerado pelo autor

Além disso, durante a execução, o programa "escreve"em um arquivo de saída a posição x e a quantidade $n(x)$ de andarilhos que estão nessa posição. A partir disso, é possível construir o histograma da Figura 3. Os limites escolhidos para "plotar"o gráfico foram reduzidos porque $n(x)$ vai a 0 para $|x| < 200$; é válido destacar também que a curva formada pelo histograma se assemelha a uma curva gaussiana.

Figura 3: Histograma de $n(x)$ por x.



Fonte: gerado pelo autor

2.2 B2

Variando o parâmetro "p"no código exposto na subseção anterior, percebe-se que os valores de $< x >$ e $< x^2 >$, e consequentemente a distribuição $n(x)$ por x, vão sendo alterados. Analiticamente, podemos escrever que a probabilidade de um andarilho dar n_d passos para a direita, sendo $n_d + n_e = N$ e $p + q = 1$, é:

$$P(n_d) = \frac{N!}{n_d! n_e!} p^{n_d} q^{n_e} \tag{2}$$

Daí, o valor de $< n_d >$ é:

$$< n_d >= \sum_{n_d=0}^{N} n_d P(n_d) = p \frac{\partial (p+q)^N}{\partial p}$$

Ou seja:

$$< n_d >= Np \tag{3}$$

De forma análoga, temos que a média de passos n_e para a esquerda, com probabilidade "q", é:

$$< n_e >= Nq$$

Logo:

$$< x >=< n_d > - < n_e >= N(p - q) \tag{4}$$

Por outro lado, podemos escrever também que $x = n_d - n_e = 2n_d - N$. Daí:

$$< x^2 >= 4 < n_d^2 > -4N < n_d > +N^2 \tag{5}$$

Já sabemos que $< n_d >= Np$, agora resta calcular $< n_d^2 >$:

$$< n_d^2 >= \sum_{n_d=0}^{N} n_d^2 P(n_d)$$

$$< n_d^2 > - < n_d >= p^2 \frac{\partial^2 (p+q)^N}{\partial p^2}$$

Ou seja:

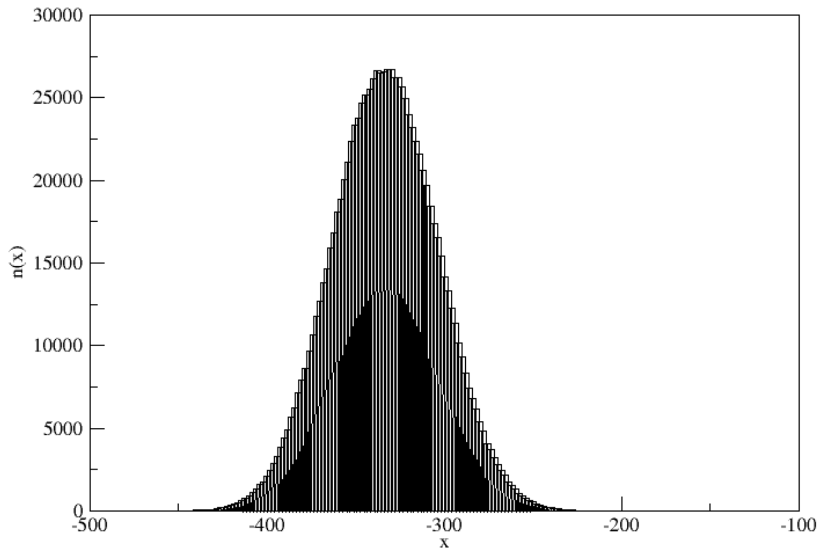
$$< n_d^2 >= Np + p^2 N(N-1) = N^2 p^2 + Npq \tag{6}$$

Substituindo em (5):

$$< x^2 >= 4Npq + N^2 - 4N^2 pq \tag{7}$$

Os valores de $< x >$ e $< x^2 >$ calculados pelo programa e os respectivos histogramas para $p = \frac{1}{3}$, $p = \frac{1}{4}$ e $p = \frac{1}{5}$ são mostrados a seguir.

Figura 4: Histograma de $n(x)$ por x para $p = \frac{1}{3}$.



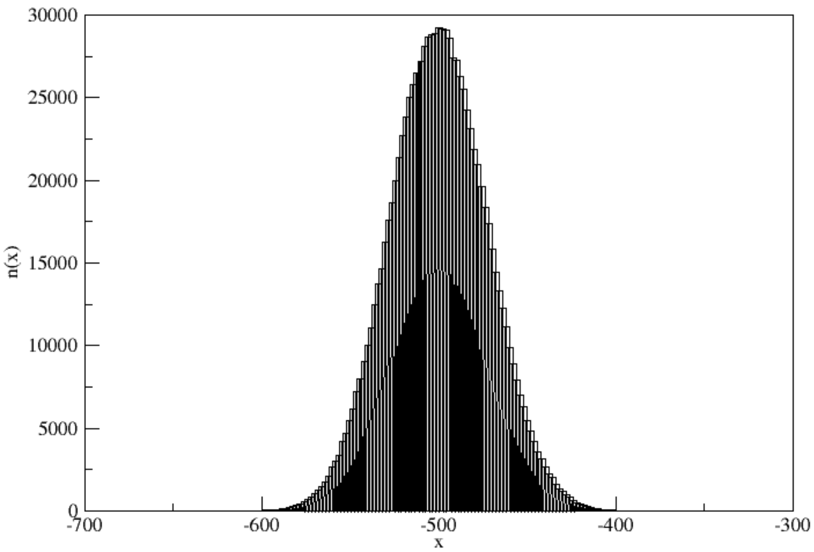
Fonte: gerado pelo autor

Figura 5: Saída para $p = \frac{1}{3}$.

```
administrador@lubuntuVC:~/projeto-2/tarefa-B$ gfortran -o tarefa-B-11212550.ex
e tarefa-B-11212550.f
administrador@lubuntuVC:~/projeto-2/tarefa-B$ ./tarefa-B-11212550.exe
 O valor medio de x eh:  -333.319916     e o de x² eh:   111989.094
```

Fonte: gerado pelo autor

Figura 6: Histograma de $n(x)$ por x para $p = \frac{1}{4}$.



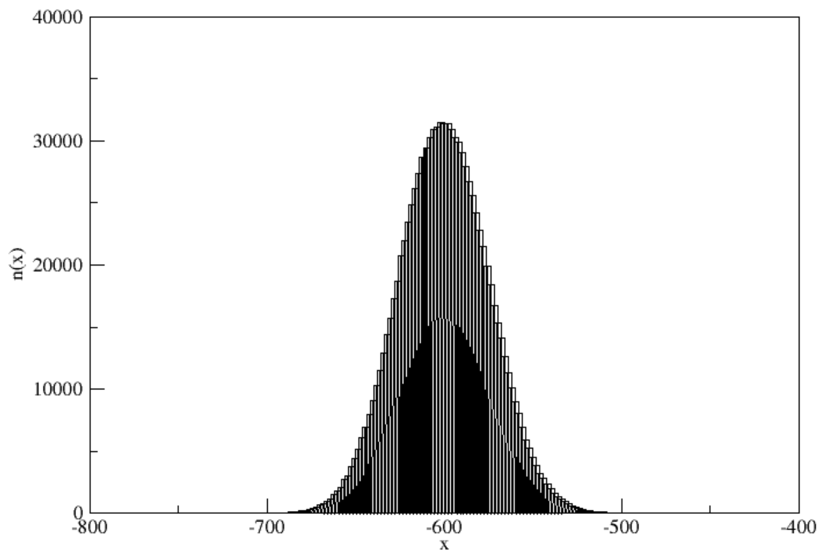
Fonte: gerado pelo autor

Figura 7: Saída para $p = \frac{1}{4}$.

```
administrador@lubuntuVC:~/projeto-2/tarefa-B$ gfortran -o tarefa-B-11212550.ex
e tarefa-B-11212550.f
administrador@lubuntuVC:~/projeto-2/tarefa-B$ ./tarefa-B-11212550.exe
 O valor medio de x eh:  -499.990479     e o de x² eh:   250738.328
```

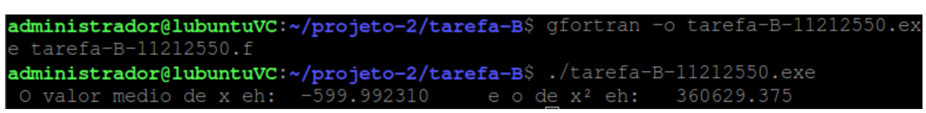
Fonte: gerado pelo autor

Figura 8: Histograma de $n(x)$ por x para $p = \frac{1}{5}$.



Fonte: gerado pelo autor

Figura 9: Saída para $p = \frac{1}{5}$.



Fonte: gerado pelo autor

3 Tarefa C

Generalizando a ideia da seção anterior para o caso bidimensional, temos que:

$$< \vec{r} > = < x > \mathbf{i} + < y > \mathbf{j} \tag{8}$$

Da mesma forma, temos:

$$< \vec{r}^2 >=< x^2 > + < y^2 > \tag{9}$$

Assim, foi escrito um programa capaz de calcular $< \vec{r} >$ e $< \vec{r}^2 >$ indiretamente, isto é, calculando $< x >$, $< y >$, $< x^2 >$ e $< y^2 >$, além de gerar um arquivo de saída com as coordenadas de cada andarilho após dar N passos. No caso do código abaixo, temos que M = 10000 (número de andarilhos), N vai variando de 10 à 10^6, passando apenas por potências de 10, e a probabilidade p de um andarilho ir para qualquer uma das 4 direções é a mesma.

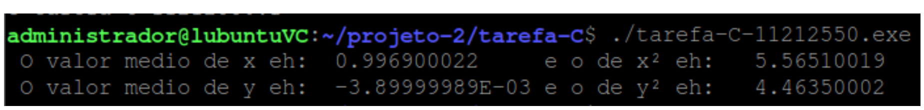
tarefa-C-11212550.f:

```
      program tarefaC

c     no caso bidimensional, p assume o valor 1/4
      parameter (M = 10000, N = 10, p = 1.0/4)
      integer*8 k, l, tmpx, tmpy
c     agora, a quantidade em cada posição é armazenada numa matriz
      dimension ipos(-N:N, -N:N)
      open(1, file='1saida-C-11212550')
      xmed = 0
      xqmed = 0
      ymed = 0
      yqmed = 0
      do k = -N, N
        do l = -N, N
          ipos(k, l) = 0
        end do
      end do

      do i = 1, M
        ix = 0
        iy = 0
        valor = rand(i)
        do j = 1, N
c         verifica para qual das 4 direções foi o passo dado
          if (valor.lt.p) then
            ix = ix + 1
          else if (valor.lt.2*p) then
            ix = ix - 1
          else if(valor.lt.3*p) then
            iy = iy + 1
          else
            iy = iy - 1
          end if
          valor = rand()
        end do
c       adiciona 1 na posição onde o anadarilho parou
```

```fortran
        ipos(ix, iy) = ipos(ix, iy) + 1
        write(1,*)ix, iy
      end do

      do icont = -N, N
c       armazena quantos andarilhos estão na reta x = icont
        tmpx = 0
c       armazena quantos andarilhos estão na reta y = icont
        tmpy = 0
        do jcont = -N, N
          tmpx = tmpx + ipos(icont, jcont)
          tmpy = tmpy + ipos(jcont, icont)
        end do
c       calcula <x>, <y>, <x  > e <y  >
        xmed = xmed + icont*tmpx
        xqmed = xqmed + icont**2*tmpx
        ymed = ymed + icont*tmpy
        yqmed = yqmed + icont**2*tmpy
      end do

      write(*,*)'O␣valor␣medio␣de␣x␣eh:', xmed/M,
     +          'e␣o␣de␣x  ␣eh:', xqmed/M
      write(*,*)'O␣valor␣medio␣de␣y␣eh:', ymed/M,
     +          'e␣o␣de␣y  ␣eh:', yqmed/M

      close(1)
      end
```

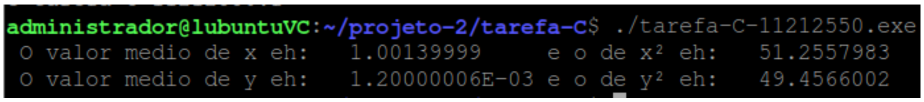
Com isso, observa-se as seguintes saídas do programa para cada N testado

Figura 10: Saída para $N = 10$.



Fonte: gerado pelo autor

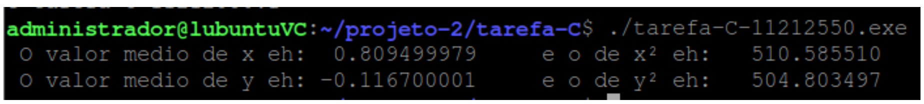
Figura 11: Saída para $N = 10^2$.



```
administrador@lubuntuVC:~/projeto-2/tarefa-C$ ./tarefa-C-11212550.exe
 O valor medio de x eh:   1.00139999     e o de x² eh:   51.2557983
 O valor medio de y eh:   1.20000006E-03 e o de y² eh:   49.4566002
```

Fonte: gerado pelo autor

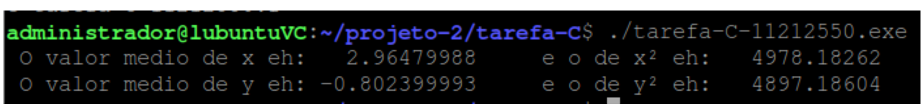
Figura 12: Saída para $N = 10^3$.



```
administrador@lubuntuVC:~/projeto-2/tarefa-C$ ./tarefa-C-11212550.exe
 O valor medio de x eh:  0.809499979     e o de x² eh:   510.585510
 O valor medio de y eh: -0.116700001     e o de y² eh:   504.803497
```

Fonte: gerado pelo autor

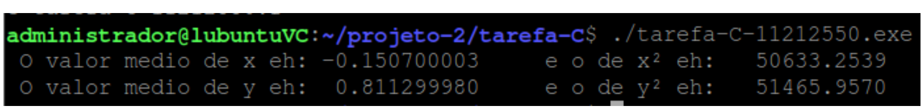
Figura 13: Saída para $N = 10^4$.



```
administrador@lubuntuVC:~/projeto-2/tarefa-C$ ./tarefa-C-11212550.exe
 O valor medio de x eh:   2.96479988     e o de x² eh:   4978.18262
 O valor medio de y eh: -0.802399993     e o de y² eh:   4897.18604
```

Fonte: gerado pelo autor

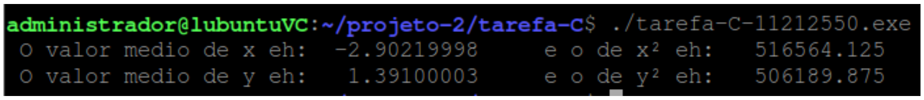
Figura 14: Saída para $N = 10^5$.



```
administrador@lubuntuVC:~/projeto-2/tarefa-C$ ./tarefa-C-11212550.exe
 O valor medio de x eh: -0.150700003     e o de x² eh:   50633.2539
 O valor medio de y eh:  0.811299980     e o de y² eh:   51465.9570
```

Fonte: gerado pelo autor

Figura 15: Saída para $N = 10^6$.



Fonte: gerado pelo autor

De posse desses dados, monta-se a Tabela 1 que dá os valores de $| < \vec{r} > | = \sqrt{< x >^2 + < y >^2}$, $< \vec{r}^2 >$ e $\Delta^2 =< \vec{r}^2 > - < \vec{r} >< \vec{r} >$.

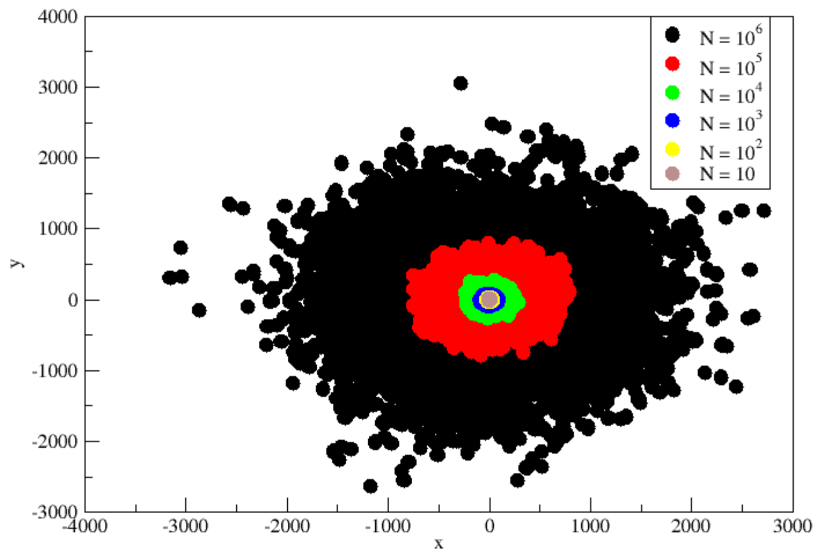
Tabela 1: Valores médios relevantes.

Passos	$\mid < \vec{r} > \mid$	$< \vec{r}^2 >$	Δ^2
10	0,997	10,029	9,035
10^2	1,001	100,713	99,711
10^3	0,817	1015,389	1014,722
10^4	3,072	9875,369	9865,932
10^5	0,825	102099,211	102098,530
10^6	3,218	1022754,000	1022743,644

Fonte: gerado pelo autor

Além disso, é possível analisar o problema observando o espaço que os andarilhos ocupam após N passos, como está mostrado na Figura 16.

Figura 16: Diagrama de posições ocupadas após N passos.



Fonte: gerado pelo autor

Observe que essa situação pode ser usada para modelar o processo de difusão de moléculas em diversos meios, como o que ocorre quando se coloca uma gota de leite em um copo de café, por exemplo. O aumento do N, nesse caso, representaria uma passagem de tempo, de forma que as moléculas vão se espalhando pelo meio.

4 Tarefa D

Nessa última parte, o objetivo é calcular a entropia de cada configuração analisada no item anterior, mostrando que ela tende a aumentar, o que está intimamente relacionado com a chamada "flecha do tempo", uma vez que um aumento no valor do parâmetro N representa o transcorrer de um certo intervalo de tempo. Observe que o código abaixo é bastante similar ao da seção anterior, o que muda é que, após posicionar todos os andarilhos (ou moléculas) na matriz, o programa vai computando a contribuição na entropia, e não nos valores médios de posição, de cada par (x,y). Lembrando que é necessário sempre checar se a célula está vazia, pois isso pode ser indesejável na hora de executar a função log(). Destaca-se também que o tamanho do reticulado utilizado foi de uma célula da matriz em que a quantidade de andarilhos em cada posição é guardada.

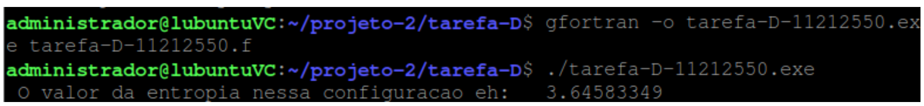
tarefa-D-11212550.f:

```
      program tarefaD

c     até o cálculo da entropia de fato, o código é o mesmo da tarefa-C
      parameter (M = 10000, N = 10, p = 1.0/4)
      integer*8 icont, jcont, k, l, ipos, ix, iy
      dimension ipos(-N:N, -N:N)

      s = 0
      do k = -N, N
        do l = -N, N
          ipos(k, l) = 0
        end do
      end do

      do i = 1, M
        ix = 0
        iy = 0
        valor = rand(i)
        do j = 1, N
          if (valor.lt.p) then
            ix = ix + 1
          else if (valor.lt.2*p) then
            ix = ix - 1
          else if(valor.lt.3*p) then
            iy = iy + 1
          else
            iy = iy - 1
          end if
          valor = rand()
        end do
        ipos(ix, iy) = ipos(ix, iy) + 1
      end do

      do icont = -N, N
        do jcont = -N, N
c         verifica se a posição está vazia
          if (ipos(icont, jcont).ne.0) then
c           caso não esteja, computa a contribuição dela na entropia
            s = s - ipos(icont, jcont)*log(1.0*ipos(icont,
     &          jcont)/M)
          end if
```

```fortran
          end do
        end do

        write(*,*)'O valor da entropia nessa configuracao eh:',
     &    s/M

        end
```

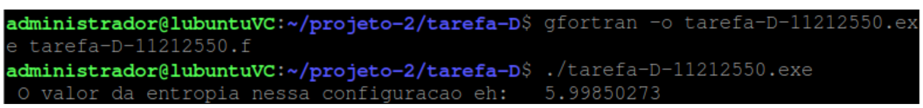
A seguir estão as saídas geradas pelo programa para cada valor de N testado.

Figura 17: Valor da entropia para N = 10.



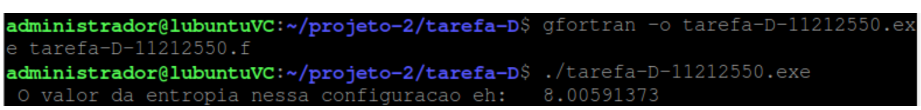
Fonte: gerado pelo autor

Figura 18: Valor da entropia para N = 10^2.



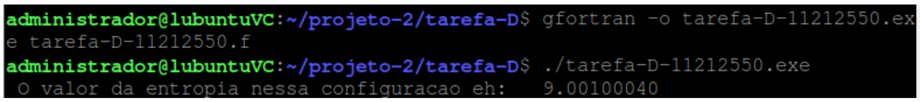
Fonte: gerado pelo autor

Figura 19: Valor da entropia para N = 10^3.



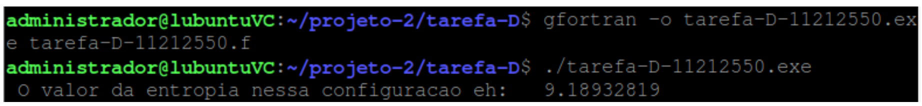
Fonte: gerado pelo autor

Figura 20: Valor da entropia para N = 10^4.



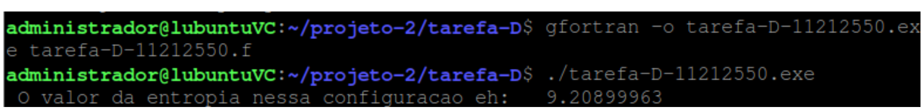
Fonte: gerado pelo autor

Figura 21: Valor da entropia para N = 10^5.



Fonte: gerado pelo autor

Figura 22: Valor da entropia para N = 10^6.



Fonte: gerado pelo autor

É observado que a entropia cresce mais rapidamente quando N é menor, e depois que ultrapassa um certo valor, que no caso é por volta de 9, ela cresce pouco, mesmo variando sempre o mesmo fator na ordem de grandeza.

Um detalhe importante, válido tanto para essa seção quanto para a anterior, é que, a partir de $N = 10^4$, o programa dá falha de segmentação, devido ao excesso de memória utilizado ao tentar criar a matriz que armazena as quantidades em cada posição. A solução encontrada para "fugir"desse problema foi, mesmo aumentando N, não alterar da mesma forma as dimensões da matriz, uma vez que os andarilhos não vão muito longe e todas as células cortadas nesse processo teriam valor 0, o que não altera o valor da entropia, de $< \vec{r} >$ e de $< \vec{r}^2 >$.