

Lista 1 - MAC0122 Princípios de Desenvolvimento de Algoritmos - POLI

Prof. Ronaldo Fumio Hashimoto

1 Exercícios

Os exercícios de 1 a 10 correspondem, respectivamente, aos exercícios de 3.1 a 3.10 do livro de Robert Sedgwick [1]. O exercício 11 foi adaptado de [2].

1. Encontre os maiores e os menores números que podem ser representados usando os tipos `int`, `long int`, `short int`, `float` e `double` em seu ambiente de programação.
2. Teste o gerador de números aleatórios do seu sistema gerando N inteiros aleatórios entre 0 e $r - 1$ usando `rand() % r` e calculando a média e o desvio padrão para $r = 10, 100$ e 1000 e $N = 10^3, 10^4, 10^5$ e 10^6 .
3. Teste o gerador de números aleatórios do seu sistema gerando N números aleatórios do tipo `double` entre 0 e 1, transformando-os em valores inteiros entre 0 e $r - 1$ ao multiplicá-los por r e truncando o resultado, e calculando a média e o desvio padrão para $r = 10, 100$ e 1000 e $N = 10^3, 10^4, 10^5$ e 10^6 .
4. Faça os exercícios 2 e 3 acima para $r = 2, 4$ e 16 .
5. Implemente as funções necessárias ao programa abaixo (Programa 3.2 do livro) para que ele possa ser usado com *bits* (números que só tomam valores 0 ou 1) aleatórios.

Programa 3.2

```
#include <math.h>
#include <stdlib.h>
#include <stdio.h>

typedef int Number;
Number randNum ()
{ return rand(); }

main(int argc, char *argv[])
{ int i, N = atoi(argv[1]);
  float m1 = 0.0, m2 = 0.0;
  Number x;
  for (i = 0; i < N; i++)
  {
    x = randNum();
    m1 += ((float) x)/N;
    m2 += ((float) x*x)/N;
  }
  printf("      Average: %f\n", m1);
  printf("Std. deviation: %f\n", sqrt(m2-m1*m1));
}
```

6. Defina uma **struct** (estrutura) adequada para a representação de uma carta de baralho.
7. Escreva um programa cliente que usa o tipo de dado definido abaixo (Programas 3.3 e 3.4 do livro) para as seguinte tarefa: Leia uma sequência de pontos (pares de números de pontos flutuantes) da entrada padrão, e encontre o ponto mais próximo do primeiro.

Programa 3.3

```
typedef struct { float x; float y; } point;
float distance(point, point);
```

Programa 3.4

```
#include <math.h>
#include "Point.h"
float distance(point a, point b)
{ float dx = a.x - b.x, dy = a.y - b.y;
  return sqrt(dx*dx + dy*dy);
}
```

-

(a) Escreva um programa `RandomWalker.c` que toma um inteiro N pela linha de comando e simula a movimentação de um caminhante aleatório para N passos. Depois de cada passo, imprima a localização do caminhante aleatório, tratando o poste de luz como a origem $(0,0)$. Também, imprima o quadrado da distância final do caminhante à origem.

```

./RandomWalker 10      ./RandomWalker 20
(0,-1)                 (0,1)
(0,0)                  (-1,2)
(0,1)                  (0,2)
(0,2)                  (1,2)
(-1,2)                 (1,3)
(-2,2)                 (0,3)
(-2,1)                 (-1,3)
(-1,1)                 (-2,3)
(-2,1)                 (-3,3)
(-3,1)                 (-3,2)
squared distance = 10  (-4,2)
                        (-4,1)
                        (-3,1)
                        (-3,0)
                        (-4,0)
                        (-4,-1)
                        (-3,-1)
                        (-3,-2)
                        (-3,-3)
                        squared distance = 18

```

- (b) Escreva um programa `RandomWalkers.c` que toma dois argumentos N e T inteiros pela linha de comando. Em cada um dos T experimentos independentes, simule um passeio aleatório de N passos e calcule o quadrado da distância. Exiba a distância quadrada média (A média de T distâncias ao quadrado).

```

$ ./RandomWalkers 100 100000      $ ./RandomWalkers 400 100000
mean squared distance = 100.15086  mean squared distance = 401.22024

$ ./RandomWalkers 100 100000      $ ./RandomWalkers 800 100000
mean squared distance = 99.95274   mean squared distance = 797.5106

$ ./RandomWalkers 200 100000      $ ./RandomWalkers 1600 100000
mean squared distance = 199.57664  mean squared distance = 1600.13064

```

Conforme N aumenta, é esperado que um caminhante aleatório termine cada vez mais longe da origem. Mas quão distante? Use `RandomWalkers` para formular uma hipótese de como a distância quadrada média cresce em função de N . Use $T = 100000$ tentativas para obter uma estimativa suficientemente acurada.

Observação: Este processo é uma versão discreta de um fenômeno natural conhecido como movimento browniano. Isso serve como um modelo científico para um surpreendente número de processos físicos tais como: dispersão de tinta fluindo em água, formação de cadeia de polímeros em química e disparo em cascata de neurônios no cérebro.

Referências

- [1] R. Sedgewick, “Algorithms in c—third edition,” p. 82, 1998.
- [2] B. Sedgewick, “Computer Science 123 Programming Assignments.” <http://www.cs.princeton.edu/courses/archive/fall10/cos126/assignments/loops.html>, 2010. [Online; acessado dia 15 de março de 2017].