Departamento de Ciência da Computação – IME-USP

MAC 115 – Introdução à Ciência da Computação

Instituto de Física - Segundo Semestre de 2005

Primeiro Exercício-Programa (EP1) Entregue até 2/10/2005, pelo Panda

Um jogo de sorte (azar?)

Neste jogo, dois jogadores, Alice e Beto, escolhem cada um um padrão de `0`s e `1`s, de mesmo comprimento. Daí, joga-se uma moeda honesta até um dos padrões aparecer (digamos, `0` significa coroa e `1` significa cara). Por exemplo: suponha que o padrão de Alice é `1011` e o padrão de Beto é `0110`. Suponha que, ao lançarmos a nossa moeda várias vezes, obtemos

`100100111001010100110...`

(Isto é, cara, coroa, coroa, cara, coroa, etc.) Então Beto ganhou, já que o padrão dele apareceu antes do padrão de Alice.

Para simular a moeda, voce deve usar as funções `rand()` e `srand()`, como neste exemplo:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ int N, i, semente;
  scanf("%d %d", &N, &semente);
  srand(semente);
  for (i=0; i<N; i++) printf("%d", rand()%2);
  printf("\n");
  return 0;
}
```

Por simplicidade, consideraremos apenas os seguintes padrões:

$$a_n = (1100)^n \qquad b_n = (1001)^n \qquad c_n = (0011)^n \qquad d_n = (0110)^n,$$

onde nossa convenção é que $a_1 = 1100$, $a_2 = 11001100$, $a_3 = 110011001100$, etc.

Nosso objetivo é estimar as seguintes quantidades, através de simulações. Defina A_n como sendo o número esperado de lances de moeda necessários até que ocorra o padrão a_n. Defina as quantidades B_n, C_n, e D_n analogamente. Podemos estimar A_n para qualquer n fixo, simulando lances de moeda usando `rand()` até que o padrão em questão ocorra pela primeira vez. Naturalmente, podemos repetir este processo um número M de vezes. Se na i-ésima vez foram necessários N_i lances da moeda para o padrão a_n ocorrer, então podemos estimar A_n pela média dos N_i, isto é, podemos estimar A_n por $M^{-1} \sum_{1 \le i \le M} N_i$.

Naturalmente, podemos da mesma forma obter estimativas para B_n, C_n, e D_n. Note que estas estimativas vão ser melhores para valores maiores de M. A primeira parte de seu programa deve receber como entrada inteiros positivos n e M e uma semente s. Seu programa deve então estimar A_n, B_n, C_n, e D_n, como descrito acima.

A segunda parte de seu programa deve estimar a probabilidade de Alice vencer o jogo se Alice escolhe um dos quatro padrões acima e Beto escolhe um outro padrão. Seu programa deve estimar esta probabilidade para os pares

$$(a_n, b_n), \qquad (b_n, c_n), \qquad (c_n, d_n), \qquad (d_n, a_n).$$

Isto é, Alice escolhe a_n e Beto escolhe b_n em um jogo, Alice escolhe b_n e Beto escolhe c_n no segundo jogo, etc; há portanto quatro probabilidades que seu programa deve estimar. Denotemos por p_1, p_2, p_3, e p_4 estas quatro probabilidades, respectivamente. Para estimar $p_1, \ldots, p_4$, seu programa deve simular o jogo M vezes para cada par, onde M é também dado pelo usuário.

Entrada

A entrada será formada pelos inteiros positivos n, M, e s (s deve ser usado como a semente de `rand()`, através da chamada `srand(s)`).

Saída

A saída deve ser constituída de suas estimativas para A_n, B_n, C_n, D_n, p_1, p_2, p_3, e p_4.

Observação

Se você notar que alguns dos $A_n, B_n, \ldots, p_1, p_2, \ldots$ coincidem, você pode diminuir o número de valores a serem estimados. (Por exemplo, como se comparam A_n e C_n?)

Observações Importantes sobre os Exercícios-Programas

Todos os exercícios-programas devem ter o seguinte cabeçalho:

```
/*********************************************************/
/* Fulano de Tal                                         */
/* Exercicio-Programa xx                                 */
/* Curso yy  - Turma zz  -- Professor:  Ciclano de Tal   */
/* Computador: .....                                     */
/* Compilador: .....                                     */
/*********************************************************/
```

1. O exercício-programa é **estritamente individual**. Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO.
2. **Exercícios atrasados não serão aceitos.**
3. Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO.
4. É **muito importante** que seu programa tenha comentários e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A avaliação dos exercícios-programas levará isto em conta.
5. Guarde com você até o final do semestre, em seus disquetes, os arquivos de seus programas, pois eles podem ser solicitados pelo professor caso surja alguma dúvida.
6. Entregue o seu programa pelo sistema Panda.