MAC 2166 – Introdução à Ciência da Computação
Grande área Elétrica – Primeiro Semestre de 2021

Segundo Exercício-Programa — O jogo do bixo

> Em Polilândia, seus mais recentes cidadãos são carinhosamente chamados de *bixos*, e cada um deles recebe na matrícula um número de identificação, semelhante ao que é feito no clássico *jogo do bicho*. Ao contrário do famoso jogo de azar, porém, este exercício-programa é um jogo de perícia onde você é desafiado a apostar alto à procura das mais intrigantes anomalias!

Por um lado, no clássico jogo de azar criado pelo barão João Batista Vianna Drummond em 1892 são enumerados de 1 a 25 a mesma quantidade de animaizinhos, os quais o fundador do Jardim Zoológico do Rio de Janeiro tornou objeto de sorteios para atrair visitantes e superar a crise financeira da recém fundada República. Por outro lado, no *jogo do bixo* descrito neste exercício-programa são cerca de 256 alunos matriculados nas turmas de Introdução a Computação oferecidas em C que são enumerados e sorteados.

Abstraindo um pouco destes dois exemplos, definimos um natural n tal que os números a serem sorteados sejam aqueles do intervalo fechado $[1..n] = \{1, 2, \ldots, n\}$. No caso do jogo criado pelo fundador do Jardim Zoológico do Rio, n é 25; no caso da bixarada de Polilândia, o n adotado é 256.

Ademais, mais de um sorteio é possivelmente realizado no jogo do bixo, de forma que denotamos por $k \geq 1$ a *quantidade de sorteios realizados*. Observe que o maior e o menor dos k sorteados definem as extremidades do menor intervalo que contém todos eles. O comprimento d deste menor intervalo constitui o *diâmetro* dos sorteados e vale sempre um mais a diferença entre o maior e o menor dos k números.

Por exemplo, para três sorteios a gerar os números 19, 8 e 17, o maior sorteado é 19, o menor é 8, e o diâmetro d é $1 + (19 - 8) = 12$, o comprimento do intervalo $[8..19]$.

Observe que $k = 1$ implica que $d = 1$, qualquer que seja o sorteio. Ademais, se puder haver repetições nos sorteios, a sequência de k enes $n, n, \ldots, n$ também possui diâmetro $d = 1$. Em contrapartida, $d \geq k$ se um número já sorteado não puder ser novamente sorteado.

De fato, quando se fala numa sucessão de sorteios, é necessário saber se há ou não a possibilidade de repetições, o que acontece se a bola sorteada é retornada à urna empregada no sorteio, por exemplo. É um exercício simples de Combinatória demonstrar que o número de possíveis sequências de k sorteios de inteiros de $[1..n]$ é: n^k, se são permitidas repetições; $\frac{n!}{(n-k)!} = n(n-1)\ldots(n-k+1) = n^k + p(n, k-1)$, onde $p(n, k-1)$ é um polinômio em n de grau $k-1$ com coeficiente do termo de maior grau $\frac{k(k-1)}{2}$, se nenhuma repetição é permitida. Para simplificar as computações, **supomos neste EP que os sorteios possam ser repetidos**, o que deve alterar pouco os resultados se $n \gg k^2$, i.e., se n for significativamente maior que k^2.

Se a história do jogo do bicho é cheia de conflitos entre os promotores dos sorteios e a polícia, a história do jogo do bixo deve desenvolver naquele que aceita abraçar este simples desafio a capacidade de desenvolver uma ferramenta robusta que permita periciar a lisura de uma sucessão de sorteios. Por exemplo, se uma sequência de sorteios produz a sequência $1, 2, 3, \ldots, k$ ou mesmo a sequência com k enes, além da alegria dos premiados comparável ao de uma criança que se lambuza comendo marmelada, há que se esperar pela acusação de fraude, eterna fonte de conflitos e intervenções policiais! Afinal, qual a chance de que o diâmetro dos sorteados seja pequeno? Que diria a este respeito a perícia científica?

***

Uma abordagem frequentemente utilizada tentaria inferir conhecimento a partir da observação de um grande número, N digamos, de diâmetros de sequências de k sorteios, para em seguida aplicar ao caso concreto que temos diante de nós. As computações massivas que devem ser realizadas num grande conjunto de dados, porém, requerem o uso de um 'cérebro eletrônico' que, entre outras coisas, deve calcular os N diâmetros associados a estes kN sorteios.

```
CÉREBRO ELETRÔNICO
Gilberto Gil (1969)
```

```
O cérebro eletrônico faz tudo
Faz quase tudo
Faz quase tudo
Mas ele é mudo

O cérebro eletrônico comanda
Manda e desmanda
Ele é quem manda
Mas ele não anda

Só eu posso pensar
Se Deus existe
Só eu
Só eu posso chorar
Quando estou triste
Só eu
Eu cá com meus botões
De carne e osso
Eu falo e ouço
Eu penso e posso
Eu posso decidir
Se vivo ou morro por que
Porque sou vivo
Vivo pra cachorro e sei
Que cérebro eletrônico nenhum me dá socorro
No meu caminho inevitável para a morte
Porque sou vivo
Sou muito vivo e sei

Que a morte é nosso impulso primitivo e sei
Que cérebro eletrônico nenhum me dá socorro
Com seus botões de ferro e seus
Olhos de vidro
```

Se há quem diga que o 'cérebro eletrônico' faz tudo, a verdade é que ele faz tudo que um cérebro humano foi capaz de programá-lo a fazer. Ademais, não poucas vezes a realidade oferece bem menos que estes kN sorteios. Não poucas vezes ela até oferece, mas requer que se esteja disposto a pagar pela obtenção de tantos dados. Para contornar problemas como estes, um recurso frequentemente utilizado pelo cérebro humano que tenta extrair conhecimento a partir de tanta observação da realidade é criar uma realidade virtual com o uso de simulações onde os sorteios são obtidos a partir de geradores de números pseudoaleatórios, tais como os *geradores congruenciais lineares*. Para gerar uma sequência de valores aparentemente sorteados de forma aleatória, um tal gerador parte de uma *semente*, denotada por r_0, e lança mão da recorrência r parametrizada em constantes previamente fixadas m, a e c que é definida pela seguinte equação recorrente:

$$r_{n+1} = (a \times r_n + c) \bmod m. \tag{1}$$

Por exemplo, para m, a, c e r_0 valendo respectivamente 8, 5, 3, 7, a sequência de inteiros em [0..7] gerada pela recorrência definida na equação 1 começa com os doze seguintes números:

6, 1, 0, 3, 2, 5, 4, 7, 6, 1, 0, 3. Observe que a sequência produzida por este gerador tem período 8, o valor de m. De fato, a sequência originada por um gerador congruencial linear é sempre periódica e o comprimento deste período é um divisor de m. Os geradores congruenciais lineares são muito bem estudados e admitem uma implementação bastante simples e rápida. Cada sorteio de um número de 1 a n é então simulado a partir de uma função f que mapeia $[0..m-1]$ em $[1..n]$ e aplica um r devolvido pelo gerador congruencial linear em

$$f(r) = \frac{r \cdot n}{m} + 1. \tag{2}$$

De fato, $r < m$ implica que $rn < mn$. Por conseguinte, $f(r) - 1 = (r \cdot n)/m < (m \cdot n)/m = n$. Por outro lado, $r \geq 0$ implica que $f(r) \geq f(0) = 1$. Assim, os valores gerados pelo total de sorteios forma uma sequência total com kN valores sorteados no intervalo $[1..n]$, abaixo dispostos matricialmente de modo a formar em cada uma das N linhas uma sequência com k valores sorteados:

$$\begin{bmatrix} f(r_1), & f(r_2), & \dots & f(r_k), \\ f(r_{k+1}), & f(r_{k+2}), & \dots & f(r_{2k}), \\ \vdots & \vdots & \vdots & \vdots \\ f(r_{k(N-1)+1}), & f(r_{k(N-1)+2}), & \dots & f(r_{kN}). \end{bmatrix} \tag{3}$$

Observe que a i-ésima sequência com k números de $[1..n]$ sorteados com repetição que se forma na i-ésima das N linhas da matriz da fórmula (3) é $f(r_{k(i-1)+1}), f(r_{k(i-1)+2}), \dots f(r_{ki})$. A partir desta sequência computa-se o i-ésimo diâmetro.

Retomando o exemplo acima e supondo que os parâmetros n, k, e N sejam respectivamente 8, 3 e 5, as cinco sequências de três sorteios em [1..8] correspondem às três primeiras colunas das cinco linhas a seguir, as quais exibem em seu final os respectivos diâmetros.

```
    7    2    1   ->    7
    4    3    6   ->    4
    5    8    7   ->    4
    2    1    4   ->    4
    3    6    5   ->    4
```

* * *

Como **primeira parte** deste exercício programa, imprima na saída as N primeiras sequências com k números de $[1..n]$ sorteados com repetição que são formadas pelas N linhas da matriz da fórmula (3) e para cada uma delas compute e imprima seu respectivo diâmetro. Cada valor sorteado da forma $f(r_i)$ deve ser computado como segundo as equações 1 e 2. O formato de saída deve ser o mesmo do exemplo que acabamos de ver, onde cada valor numérico foi impresso com formato `"%5d"`, para garantir alinhamento em cada coluna. Antes da matriz expandida com a coluna dos diâmetros, imprima duas linhas de cabeçalho obedecendo estritamente ao formato dado a seguir.

```
Recorrencia   r <-- ( r x 5 + 3 ) mod 8   a partir da semente 7.
As 5 sequencias de 3 sorteios (com repeticao) em [ 1 .. 8 ] e seus diametros:
```

De forma a imprimir este cabeçalho e a matriz expandida com a coluna dos diâmetros, escreva uma função com o protótipo:

```
void PrimeiraParte( int n, int k, int N, int m, int a, int c, int r_0 );
```

Embora não devolva nenhum valor (como especifica a palavra reservada `void`), é nesta função (procedimento) que deve ser impressa a saída semelhante à do exemplo acima, depois de duas linhas iniciais a descrever os parâmetros. Observe que os parâmetros de `m`, `a`, `c`, `r_0` definem a recorrência especificada na equação 1, ao passo que os parâmetros `n`, `m` são usados no cálculo da função f presentes em cada elemento da matriz $\mathtt{N} \times \mathtt{k}$.

Informação como a provida na coluna dos diâmetros, principalmente, pode ser utilizada de modo a inferir quão provável é a obtenção de uma sequência de k sorteios com diâmetro d.

***

Abstraindo das falcatruas, definamos $S(n, k, d)$ o conjunto de sequências de k sorteios com repetição no intervalo $[1..n]$ que possuem diâmetro d. É assim natural que investiguemos sobre o tamanho de $S(n, k, d)$. A respeito desta questão, é simples verificar que

$$S(n, k, 1) = \{s_1, s_2, \ldots, s_k, \text{sequências tais que: } s_1 \in [1..n] \text{ e } s_i = s_1, \text{para todo } i \in [2..k]\}.$$

Daí, $|S(n, k, 1)| = n$: são n as sequências de k sorteios com repetição que possuem diâmetro 1.

Fixemos agora um natural $d > 1$ e seja $s = s_1, s_2, \ldots, s_k$ uma sequência qualquer de $S(n, k, d)$. Seja $m \in [1..n - d + 1]$ o menor destes k valores e seja a o menor índice em $[1..k]$ tal que $s_a = m$. Há por conseguinte um menor índice b em $[1..k] \setminus \{a\}$ tal que $s_b = m + d - 1$ seja o maior dos k inteiros da sequência s. Seja i um índice qualquer em $[1..k] \setminus \{a, b\}$. Há quatro casos a considerar:

1. Para todo $i > \max(a, b)$, temos que $s_i \in [s_a..s_b]$. Este intervalo fechado possui d naturais.
2. Para todo $i < \min(a, b)$, temos que $s_i \in (s_a..s_b)$. Note que este intervalo aberto exclui s_a e s_b e possui $d - 2$ naturais.
3. Supondo $a < b$ e i tal que $a < i < b$, temos que $s_i \in [s_a..s_b)$. Note que este intervalo exclui s_b e possui $d - 1$ naturais.
4. Supondo $b < a$ e i tal que $b < i < a$, temos que $s_i \in (s_a..s_b]$. Note que este intervalo exclui s_a e possui $d - 1$ naturais.

Fixados $\bar{a}, \bar{b} \in [1..k]$ dois índices distintos, podemos nos restringir ao conjunto das sequências de k sorteios com repetição $s \in S(n, k, d)$ cujo menor índice do menor valor sorteado é $\bar{a}$ e cujo menor índice do maior valor sorteado é $\bar{b}$. Denotamos por $S(n, k, d, \bar{a}, \bar{b})$ uma tal restrição de $S(n, k, d)$.

(Pode parecer uma complicação desnecessária exigir a minimalidade de a e de b, contudo quem quiser examinar com perícia e precisão os desafios que terão de enfrentar os investigadores do jogo do bixo não pode se assustar com a complexidade da combinatória que a realidade pode oferecer. De fato, para um índice i distinto de a e de b, é justamente em função de sua comparação com estes índices assim definidos, conforme os quatro casos acima analisados, que se determinam os possíveis valores de $s_i - m$.)

Dados i, a e b índices *distintos* em $[1..k]$, definimos

$$\delta(i, a, b) = \begin{cases} 0, & \text{se } i > a \text{ e } i > b; \\ 2, & \text{se } i < a \text{ e } i < b; \\ 1, & \text{se } i \text{ está entre os dois outros índices.} \end{cases}$$

São ao todo $d - \delta(i, a, b)$ os possíveis valores de s_i, todos no intervalo $[m..m + d - 1]$, podendo acontecer a exclusão das extremidades, como visto acima. Temos assim a seguinte identidade:

$$|S(n, k, d, a, b)| = (n - d + 1) \prod_{i \in [1..k] \setminus \{a,b\}} (d - \delta(i, a, b)). \tag{4}$$

Ademais, como $d - 2 \leq (d - \delta(i, a, b)) \leq d$, temos que

$$(n - d + 1)(d - 2)^{k-2} \leq |S(n, k, d, a, b)| \leq (n - d + 1)d^{k-2} < nd^{k-2}. \tag{5}$$

Ora,

$$S(n, k, d) = \bigcup_{a \in [1..k]} \bigcup_{b \in [1..k] \setminus \{a\}} S(n, k, d, a, b). \tag{6}$$

Se levarmos em conta que são k os possíveis valores de $a \in [1..k]$ e que são $k - 1$ os possíveis valores de $b \in [1..k] \setminus \{a\}$, a desigualdade 5 acima obtida permite-nos concluir que

$$k(k - 1)(d - 2)^{k-2}(n - d + 1) \leq |S(n, k, d)| < k^2 d^{k-2} n. \tag{7}$$

A título de curiosidade, definindo $T(n, k, d)$ como o conjunto de sequências de k sorteios *sem* repetição no intervalo $[1..n]$ que possuem diâmetro d, pode-se de forma análoga deduzir que:

$$(k - 1)^2(d - k)^{k-2} n < k(k - 1)\frac{(d - 2)!}{(d - k)!}(n - d + 1) = |T(n, k, d)| < k^2 d^{k-2} n. \tag{8}$$

* * *

Sejam dados inteiros $n > 1$ e $d \geq 1$. Como **segunda parte** deste exercício programa, imprima todas as sequências com $k = 3$ números de $[1..n]$ sorteados com repetição e que possuem diâmetro d. Para tanto, escreva uma função com o protótipo:

```
void SegundaParte( int n, int d );
```

A geração das sequências de $S(n, 3, d)$ deve realizar este procedimento implicitamente descrito na seção anterior:

1. Siga a caracterização de $S(n, k, d)$ da equação 6 conforme a ordenação dela induzida:

   (a) primeiro, conforme valores crescentes de a, de 1 a 3;

   (b) segundo, conforme valores crescentes de b, de 1 a 3, excluindo-se o caso $b = a$.

   Fixados índices distintos $a, b \in \{1, 2, 3\}$, a geração das sequências de $S(n, 3, d, a, b)$ deve ser feita segundo o procedimento usado na seção anterior, com a semântica de a e de b vista na seção acima, até culminar com a caracterização de $|S(n, 3, d, a, b)|$ à equação 4. Assim,

   (a) escolha $m \in [1..n - d + 1]$, na ordem crescente;

   (b) sejam $s_a = m$ e $s_b = m + d - 1$;

   (c) seja i o único índice em $\{1, 2, 3\} \setminus \{a, b\}$;

(d) escolha $s_i \in [s_a..s_b]$, na ordem crescente, mas eventualmente despreze alguma extremidade do intervalo conforme $i < a$ ou $i > b$, tal como visto na seção acima;

(e) fixados $a, b, m, s_a, s_b, i, s_i$, imprima uma linha com s_1, s_2 e s_3, nesta ordem.

(f) conte quantas sequências foram geradas e impressas;

(g) calcule os limites oferecidos nas desigualdades (7);

A seguir vemos um exemplo das impressões realidadas dentro de `SegundaParte` para $n = 4$ e $d = 2$.

```
Listagem das sequencias de S( 4 , 3 , 2 ) (com repeticao):
1 2 1
1 2 2
2 3 2
2 3 3
3 4 3
3 4 4
1 1 2
2 2 3
3 3 4
2 1 1
2 1 2
3 2 2
3 2 3
4 3 3
4 3 4
2 2 1
3 3 2
4 4 3
Total de 18 sequencias.
Limites: 0 e 72
```

O formato de saída deve ser o mesmo do exemplo que acabamos de ver.

Apesar de parecer complexo o enunciado, a verdade é que o algoritmo já está elaborado e descrito. Só falta escrever em C.

Com relação ao exemplo acima, as seis primeiras sequências correspondem ao caso $(a, b) = (1, 2)$, as três seguintes ao caso $(a, b) = (1, 3)$, as seis seguintes ao caso $(a, b) = (2, 1)$, as três seguintes ao caso $(a, b) = (3, 1)$. A cada escolha de (a, b), temos um número de sequências que é múltiplo de 3 porque esta é a quantidade de escolhas para $m = s_a$. Já o número de possibilidades para a escolha de s_i são 2, 1 ou 0. Os casos em que $a, b \in \{2, 3\}$ e implicam $i = 1$ e nenhuma possibilidade de escolha para s_i.

***

Seria possível gerar $S(n, k, d)$ para um k qualquer, mas isto praticamente requereria uma técnica de programação que não é vista nesta disciplina. NÃO USE RECURSOS DA C NÃO VISTOS EM SALA, em particular neste EP, não use vetores ou números reais ou funções da biblioteca que não as vistas.

Para integrar estas duas partes, o programa principal de seu EP deve primeiro pedir a parte desejada para em seguida pedir os parâmetros necessários e transferir a execução à função apropriada. Abaixo temos duas execuções completas em que cada uma das partes foi acionada. Seu EP deve também reproduzir o mesmo comportamento da função `main` exposto neste exemplo.

```
mac2166@polilandia: ./ep2
Digite a tarefa desejada: 1 para primeira; 2 para segunda.
1
Digite parametros n, k, N, m, a, c, r_0:
8 3  5   8 5 3   7
Recorrencia   r <-- ( r x 5 + 3 ) mod 8   a partir da semente 7.
As 5 sequencias de 3 sorteios (com repeticao) em [ 1 .. 8 ] e seus diametros:
    7     2     1   ->     7
    4     3     6   ->     4
    5     8     7   ->     4
    2     1     4   ->     4
    3     6     5   ->     4
mac2166@polilandia: ./ep2
Digite a tarefa desejada: 1 para primeira; 2 para segunda.
2
Digite parametros n, d:
4 2
Listagem das sequencias de S( 4 , 3 , 2 ) (com repeticao):
1 2 1
1 2 2
2 3 2
2 3 3
3 4 3
3 4 4
1 1 2
2 2 3
3 3 4
2 1 1
2 1 2
3 2 2
3 2 3
4 3 3
4 3 4
2 2 1
3 3 2
4 4 3
Total de 18 sequencias.
Limites: 0 e 72
mac2166@polilandia:
```

Para a impressão de cada linha com 3 elementos da sequência, sugere-se usar função `Cospe` fornecida a seguir.

```
/*
 * Cospe: função recebe três índices i, j e k
 * e o i-ésimo, o j-ésimo e o k-ésimo elementos
 * de uma sequência e imprime os três elementos
 * segundo ordem crescente dos índices. Ex: para
 * parâmetros  3, 2, 1, 4, 6, 5, a função imprime:
 *   5  6  4
 */
void Cospe( int i, int j, int k, int s_i, int s_j, int s_k ) {
  if ( i < j )
    if ( j < k )
      printf( "%d %d %d\n", s_i, s_j, s_k );
    else
      printf( "%d %d %d\n", s_i, s_k, s_j );
  else if (i < k)
    if ( j < k )
      printf( "%d %d %d\n", s_j, s_i, s_k );
    else
      printf( "%d %d %d\n", s_k, s_i, s_j );
  else
    if ( j < k )
      printf( "%d %d %d\n", s_j, s_k, s_i );
    else
      printf( "%d %d %d\n", s_k, s_j, s_i );
  return;
}
```

Bom trabalho a todos. De um bom trabalho na solução para o jogo do bixo depende a solução para o problema da contravenção e da fraude mundiais.