RESUMO

São referidas técnicas de geração de números aleatórios e pseudo-aleatórios em computador e apresentadas algumas aplicações concretas.

índice

1.introdução

A importância dos números aleatórios e pseudo-aleatórios provem da multiplicidade das suas aplicações algumas das quais iremos mencionar neste artigo. Historicamente, parece dever-se a J. E. Mayer a ideia de utilizar números aleatórios para resolver certos tipos de problemas, muitos dos quais de natureza puramente determinista; estas técnicas de resolução de problemas são, normalmente, designados por métodos de Monte-Carlo, tendo surgido, durante a segunda guerra mundial, como meio de simular o comportamento dos neutrões no interior dos materiais radioactivos.

1.1- Um método de Monte-Carlo

Como exemplo ilustrativo dos métodos de Monte-Carlo, consideremos o seguinte problema: calculo do integral definido de uma função f(x) no intervalo [a,b].
Como e sabido, no caso que acabamos de esquematizar, o valor do integral reduz-se a área representada a tracejado na figura 1.

Um método (de Monte-Carlo) para resolver' o problema proposto, e o seguinte: espalhemos, aleatoriamente, pontos no rectângulo abCD (teremos, assim, uma densidade de pontos constante dentro do referido rectângulo) ; contamos o número de pontos gerados que se encontram abaixo da curva da função fi esse numero de pontos e proporcional ao valor do integral desejado.

Obviamente, (b-a).M representa a área de rectangulo abCD.

Este método e intuitivamente aceitável, o mesmo se passando com muitos outro deste tipo. Mais adiante, referiremos maneiras de gerar números aleatórios em computador.

Convém, no entanto, acentuar que estes processos de cálculo dão, apenas, resultados aproximados, sendo os erros cometidos tanto mais importantes quanto menor for o número total de valores aleatórios gerados. Existem, contudo, muitas situações em que não há soluções analíticas (ou mesmo, aproximadas; que possam ser obtidas a custa dos métodos usuais da Analise Numérica) nessas condições, os métodos de Monte-Carlo podem ser uma ajuda preciosa, caso sejam convenientemente utilizados.

-14-½

2.Métodos históricos de geração de números aleatórios

Normalmente, podemos referir. que não faz muito sentido a expressão "numero aleatório"; com efeito não podemos dizer que um de terminado valor e um numero aleatório. Podemos é falar de sequência de números aleatórios (independentes), se cada um deles foi obtido casualmente e o conhecimento de um deles não permitir obter qualquer informação sobre os outros que constituem a sequência; normalmente quando se fala de sequências de números aleatórios, quer-se indicar, adicionalmente, que se trata de numero com distribuição uniforme (cada um tem a mesma probabilidade de ser gerado que qualquer outro), a não ser que, explicitamente, se indique outra distribuição.

2.1- Métodos Manuais

2.1.1- “Chapéu Alto"
Consiste em seleccionar (com reposição) discos ou bolas numeradas continadas em urnas ou quaisquer outros receptáculos apropriados (por exemplo. chapéus), depois de bem misturadas.

Este método teve bastante importância no desenvolvimento dos trabalhos sobre Estatística realizados, na University College, sob a direcção de Karl Pearson; sendo um método manual, tem, contudo, muitas inconvenientes: e lento e trabalhoso e, principalmente, não existe possibilidade de garantir uma mistura adequada, pelo que, muitas vezes, as sequências geradas não são realmente aleatórias.

2.1.2 - Tabelas de dígitos aleatórios

São obtidas através da selecção aleatória de alguns dos dígitos que constituem os números que surgem em certas tabelas: funções trigonométricas logaritmos, listas telefónicas, etc. Esta ideia deve-se a L. Tippett, que publicou uma tabela de 10400 dígitos aleatórios, retirados dos digitos finais de números que apareciam em tabelas de relatórios de recenseamentos.

Historicamente, o método que veio substituir o do "chapéu alto", tornando muito mais rápida e fiável a realização de experiências baseadas no uso dos números aleatórios.

2.2- Métodos mecânicos

2.2.1- A Máquina de Kendall e Babington-Smith

Era constituída por um disco (com sectores numerados) que girava movido por um motor eléctrico e que parava em instantes arbitrariamente escolhidos por um observador, sendo' tomada nota do número indicado por um ponteiro; o método aleatório de escolha do instante de paragem consistia em deslocar,

às cegas, um eléctrodo através de um labirinto em grafite, desenhado sobre uma folha de papel, até que o eléctrodo fizesse contacto eléctrico com o labirinto.

Foram obtidos deste modo, 100000 dígitos aleatórios. Para testar a máquina, os autores desta desenvolveram una serie de testes estatísticos, que ainda hoje são usados.

2.2.2- A maquina analógica de Beer

Stafford Beer desenvolveu esta máquina para estudar o comportamento de filas de espera; continha uma parte que constituía uma versão mecânica do "chapéu alto", parte essa que. é reproduzida na figura 2.

Deixa-se cair uma bola num crivo, que vibra sobre um cone rotativo, e aquela irá entrar num dos cem receptáculos existentes na base do cone. Os movimentos da peneira e do cone fazem com que os receptáculos tenham a mesma probabilidade de serem seleccionados, resultando, para os números aleatórios gerados, uma distribuição uniforme entre 1 e 100.

2.3-. Métodos electrónicos

Existem muitas maquina electrónicas para gera números aleatórios; algumas das mais importantes são: ERNIE (que selecciona os prémios da lotaria da Grã-Bretanha), Rand Corporation Machine (através da qual foi obtida a tabela "A Million Random Digits"), OS Electronic Probability Generator) RCA Random Number Generator (que usa um dispositivo sugerido por J. Vom Neumann), etc.

De um modo geral , todas essas maquinas usam o mesmo principio: existe uma fonte (material radioactivo, lâmpadas de neon, diodos termionicos) que produz um ruído aleatório, ruído esse que é convertido numa sequência de impulsos que vão comandar um contador cíclico; o envio de impulsos e suspenso a intervalos regulares para exame do estado do contador. Este estado o resto da divisão inteira do número de impulsos ocorridos durante o intervalo de tempo entre interrupções, pelo modulo do contador (número total de estados do mesmo).

Embora haja dez contadores, só são gerados nove dígitos distintos (cada um é obtido por subtracção dos conteúdos de dois contadores sucessivos); o objectivo e permitir o funcionamento de maquina, mesmo quando se avaria um dos geradores de impulsos.

2.4- Análise dos métodos referidos

Todos os métodos mencionados, até agora, apresentam inconvenientes, caso encaremos a sua utilização em computadores, porque são lentos ou produzem sequências de números que, por vezes, não tem as características de aleatoriedade requeridass (manuais e mecânicos); todos, porque tem custos elevados quer seja em termos de trabalho humano, quer em termos de preço do dispositivos). Mas um inconveniente ainda mais serio é a não reprodutibilidade das sequências: com efeito, na fase de desenvolvimento dos programas, é necessário verificar os cálculos; em certas aplicações precisamos de avaliar hipóteses alternativas, pelo que será conveniente testarmos as alternativas em condições análogas. Uma maneira de ultrapassarmos este problema seria gravarmos, num suporte físico apropriado (banda ou disco magnético), esses números aleatórios gerados e lê-los, no programa, cada vez que precisássemos deles; contudo, para alem do espaço necessário para armazenar os valores, teríamos o problema da lentidão da leitura, pois , como e de conhecimento geral, o processamento de informação na unida de central do computador e varias ordens de grandeza mais rápido que a troca de informação com as unidades periféricas.

3. Métodos de geração de números pseudo-aleatorios

Com o advento dos computadores, as deficiências mencionadas, anteriormente, acerca dos métodos de geração de números aleatórios', levaram ao aparecimento de metodos mais eficientes para utilização em computador, fazendo uso das operações aritméticas deste.

Deve-se a John V. Neumann o primeiro destes métodos: o "Mid-square", em que cada número da sequência ". obtido, a partir do anterior, elevando-se este ao quadrado e aproveitando, depois, os dígitos do meio do resultado encontrado.

Por exemplo, consideremos, como primeiro numero
X1=81 x12= 6561 x1=56

Se continuarmos, obtemos a seguinte sequência 81, 56, 13, 16, 25, 62, 84, 5, 25, 62,....

Esta série de valores ilustra um dos problemas comuns a todos os métodos de geração de números pseudo-aleatórios: mais cedo ou mais tarde, há um valor de sequência que se repete, limitando um ciclo que se repete indefinidamente (neste caso, o comprimento do ciclo é quatro).

Outra objecção que se pode por a estes métodos é a seguinte: até que ponto uma sequência de números, gerados por um processo puramente determinístico, pode ser considerada aleatória? Uma justificação pragmática consiste em reconhecer que não se trata de números aleatórios, mas acrescentar que eles se comportam como tal (especialmente dos pontos de vista da equiprobabilidade e independência). Portanto, embora não o sejam, esses números parecem ser realmente aleatórios: daí o seu nome: pseudo-aleatórios.

3.1- Métodos baseados no aproveitamento de certos dígitos de um resultado

1.1.1 -Mid-square

Já foi mencionado antes; a expressão de recorrência que o sintetiza e a seguinte:

Xn+1 : int(x2/bd) - b2d .int (x2n/b2d)

Onde b é a base de numeração usada, d o número de dígitos considerados e int (x) a parte inteira do número x.

Tem, como inconveniente, o produzir ciclos relativamente curtos e favorecer os mais pequenos valores possíveis para xn+1

3.1.2- "Mid-product"

Trata-se de uma extensão do método anterior e do proposto tendo em vista reduzir os defeitos assinalados para aquela. Parte-se de dois numeros, x1 e x2, que se multiplicam, aproveitando-se os dígitos intermédios para obter x3; o processo repete-se depois, 'em x2 e x,3 para obter x4, etc.

A expressão de recorrência que o define é

Xn+2 = int(xn*xn+1/bd) . int (xn*xn+1/b3d)

b, d e int(x) tem os significados já assinalados anteriormente.

Embora reduza os inconvenientes do método anterior', não os elimina, pelo que também não e muito utilizado.

3.1.3 Potenciação
Pode ainda considerar-se como uma alteração do método proposto por Von Neumann, sendo necessário inicializar 3 parâmetros: o valor inicial da série x1 e as constantes e e c.

Tem a seguinte expressão de recorrência:
Xn+1 = (c+xn)e – int (c+xn)e

Onde C é uma constante real e e um expoente inteiro.

Nesta forma, este método gerará números pseudo-aleatórios entre 0 e 1

3.2 Métodos congruenciais

Tendo reconhecido que uma sequência de números pseudo -aleatórios e ciclica, D. H. Lehmer sugeriu, em 1943, que se usasse a teoria dos muremos para tentar fazer esse período o mais longo possível. Hoje em dia, a quase totalidade dos geradores; de números pseudo-aleatorios baseia-se nesta ideia.

3.21 – Método congruencial linear (de Lehmer)

Tem, como expressão geral
Xn+i= (f.xn+a) modulo (m)

Onde f e um factor multiplicativo, a o acréscimo e m o modulo.

É necessário, adicionalmente, fixar o primeiro elemento da sequência, x1

Lembramos que a expressão x: (y) módulo (n) significa: x e congruente com y, modulo z, e indica que a diferença x-y é múltipla inteira de z. mais informalmente, podemos dizer que y é o resto da divisão inteira de x por z

geraremos a seguinte sequência

3, 1, 7, 9, 3, 1,

Verifiquemos que e necessária uma escolha criteriosa dos parâmetros para obtermos uma sequência com periodo suficientemente longo.

OBS: O método inicialmente proposto por Lehmer tinha c=0 (método congruencial multiplicativo) que e um pouco rápido do que se fizermos c ¹= (método congruencial misto), em contrapartida, o primeiro tem um período ligeiramente mais curto: que o segundo.

Como a dissemos, os parâmetros que definem este método tem de possuir um determinado conjunto de propriedades, se quisermos obter sequências com características satisfatórias:

a)Escolha do módulo, m: como é natural, convém que m seja o maior possível, embora, na pratica, o seu valor esteja dependente da dimensão da palavra do computador.

5)Escolha do factor multiplicativo f, e do acréscimo a: a fixação de m limita, desde logo, a dimensão máxima do período da sequência (que será m); no entanto, para que esse valor máximo Seja atingido, f e a tem de satisfazer as condições enunciadas nos seguintes teoremas:

Teorema 1: uma sequência congruencial linear tem um período do comprimento m se e só se (com c¹0):

i)a e primo em relação a m;

ii)f-1 é um múltiplo inteiro de cada m dos números primos que dividam m;

iii) f-1 é múltiplo Inteiro de 4, se m for, também, múltiplo inteiro de 4.

Teorema 2: O maior período possível, com c:0, é x(m)1 que se verificara caso:

i)x1 seja primo em relação a m;

ii) f seja um elemento primitivo, módulo m.

Definições:

-Ordem de f modulo m (com f e m, primos): menor inteiro, x para o qual

fx= (1) módulo (m)

-Elemento primitivo, módulo m: qualquer valor de 1 que conduza a máxima ordem possível, módulo m.

-Ordem de um elemento primitivo, x(m): máxima ordem possível, módulo m.

NOTA:estes teoremas estão demonstrados em Knuth [1] , onde vem referido, igualmente, como obter elementos primitivos, modulo m.

3.2.2- Outros métodos

Embora, como se disse, o método congruencial linear seja o mais usado, tem surgido outras sugestões, como sejam:

Métodos congruenciais quadráticos
Xn+1 = (ax2n + bxn +c) módulo (m)

-Métodos congruenciais aditivos

Xn+1 = (xn+xn-k) módulo (m)

Este último necessita de k valores inicais para poder funcionar; quando k=1, produz uma sequência de Fibonacci, que não apresenta resultados satisfatórios.

No entanto, por uma razão ou outra (pouca eficiência, necessidade de memorizar um certo número de valores, pouca qualidade estatística), não tem tido grande aceitação.

3.3- Uma concretização Pratica destes métodos

Como dissemos, na obra, já citada, de Knuth vem indicado:

modos de obter elementos primitivos, módulo m (ou seja, maneiras de garantir ciclos com o máximo período possível); no caso do modulo, m, poder ser expresso como uma potência de 2, m=2P, com p>4, o multiplicador, f, devera satisfazer uma das relações

f=(3) modulo (8) ou

f=(5) módulo (8)

Alem disso, f não pode ser muito pequeno, porque, então, a sequência apresentara uma correlação serial acentuada Existe uma fórmula , de Greemberger, que nos da os limites entre os quais varia o coeficiente de correlação serial, p (xi, xi+i):

1/f – (6a/fm) (1- a/m) +- f/m

Assim, para minimizarmos a correlação entre Elementos consecutivos da sequência, deveremos escolher valores, para f próximos de Öm

Uma escolha frequente parece ser f=2P/2 +3.

Vamos ilustrar o que acabamos de dizer com um exemplo retirado do Scientific Subroutine Package, desenvolvido pela IBM para o sistema 360 (rotina, em FORTRAN IV, RANDU):

```
SUBROUTINE RANDU (IX, IY, YFL)
IY =        ix * 65539

IF (IY) 5,6,6

5 IY = Iy + 2147483647 + 1

6 YFL = IY
YFL        YFL *4656613E-3

RETURN

END
```

IX é o elemento anterior da sequência (xn); IY, o elemento actual (xn+1): YFL, um número pseudo-aleatório com distribuição uniforme entre 0 e 1.

Como primeiro elemento da sequência, é sugerido um inteiro ímpar com um numero de dígitos não superior a nove

Ométodo utilizado nesta rotina é o congruencial multiplicativo

Xn+1 = (f.xn) módulo (m)

em que f= 65S39 (2.16 + 3) e m = 2147483648(2 31)

Esta rotina pode ser usada em qualquer computador binário que tenha uma palavra de 32 "bits"; como se sabe, os inteiros são normalmente, representados do seguinte modo:

-Os positivos, no seu equivalente binário, desde que não ultrapassem. 2 31 - 1, que corresponde a
Q 1111....11 = .2147483647 =2 31 -1

31 "bits"

-Os negativos, na forma de complemento para dois; por exemplo, -1 representar-se-ia do seguinte modo
1111111 . . . 11-1

32 "bits"

Quanto ao produto, módulo 232, ele é feito, automaticamente, pelo computador;. com efeito, na instrução IY = IX*65539, Se o resultado contiver mais de 32 dígitos binários, o só serão considerados os 32 menos significativos (é ignorado o 'overflow" em vírgula fixa). No entanto, pode acontecer que, dos 32 "bits" aproveitados pela maquina, o primeiro seja um (já vimos que, na forma de representação inteira normalmente utilizada, isso significa que o número é negativo); assim, torna-se necessário passar esse dígito a zero. Para conseguir isso, a rotina testa o resultado, para ver se e negativo, e, se o for, anula-o, do seguinte modo:

Seja 1000...01
32 bits
O resultado do produto (número negativo)

Somemos-lhe 0111....11 (21474836447)

Transporte desprezado 1, 0000....00

Somemos-lhe 0000....01 (1)
0000....01
Resultado final (apenas eliminamos o digito do sinal)

O objectivo deste conjunto de instruções e fazer o produto, modulo 231 (uma vez que o primeiro dígito e o do sinal), de ix por 65539.

Finalmente, e feita uma redução de escala dos números gerados:
de [0;2 31-1] para [0;1] (6ª instrução).

4.Testes estatísticos para números peuco-aleatorios

Depois de escolhermos um dado gerador de números pseudo-aleatórios, teremos que confirmar que os números se comportam, realmente, como aleatórios; é, pois, necessário aplicar alguns testes estatísticos para verificar a aleatoriedade das sequencias geradas. Esta é uma fase que e, normalmente, menosprezada, sendo comum verificar-se que as pessoas aceitam, cegamente, rotinas que colegas seus verificaram produzirem resultados aceitáveis em certas condições.

Muitas das técnicas usadas para testar sequências (Se números pseudo-aleatórios baseiam-se nos testes do Qui-quadrado (x2) e de Kolmogorov-Smirnov. No entanto, é possível efectuar simplificações nesses métodos, por forma a torná-los mais sugestivos. encontrando, mesmo, interpretações gráficas para alguns deles.

Para ilustrarmos este assunto, escolhemos três testes (simplificados), que iremos aplicar a rotina RANDU: de frequência, serial e da correlação serial.

Desenvolvemos, para isso, um programa FORTRAN IV, que passamos no 360/44 (IBM) do CCUL; a listagem do programa propriamente dito e os resultados de uma geração de 10000 valores são apresentados em anexo. Escolhemos, para primeiro elemento da sequência, o numero de aluno, no IST, do autor (14263).

4.1- Teste de Frequência

Como já dissemos, a rotina RANDU gera valores entre O e 1; se os números ao gerados "aleatoriamente", é lícito esperar que se distribuam uniformemente naquele intervalo, pelo que o dividimos em dez classes de idêntica dimensão (0,1) e contemos o número de valores gerados contidos, em cada uma das classes.

Os resultados obtidos aparecem na listagem já mencionada e estão de acordo com a hipótese feita (para ser mais exacto: aplicado o teste x2 , a concordância encontrada é demasiado boa para os níveis de significancia normalmente utilizados - 1, 5 e 10%; deveríamos, pois, ter empregado um teste mais potente).

4.2- Teste serial

Este tipo de testes é usado para avaliar a aleatoriedade entre grupos consecutivos de numeres na serie; se cada um dos grupos for constituído por dois números, poderemos considerar que se trata das coordenadas de um ponto num plano e, nesse caso, esses pontos deverão espalhar-se, uniformemente, no quadrado definido pelos quatro pontos (0;0), (O;1), (1;1) e (1;0). No anexo mencionado antes, está representado o gráfico que sintetiza este teste; o quadrado referido está limitado, dos lados esquerdo e inferior, pelo caracter "." e, dos lados Superior e direito, pelo símbolo "&"; cada um dos números que surge no gráfico indica o número de sobreposições no ponto que ele representa. Também neste teste, se

verifica, aparentemente, uma boa concordância com a hipótese feita.

Este teste pode ser generalizado para grupos, consecutivos, de três, quatro, etc, números pseudo-aleatorios.

NOTA:Tendo notado que o contraste obtido no diagrama de dispersão era pequeno, foi passado, novamente, o programa, agora com o Seguinte jogo de caracteres:

Antes Depois

Um único ponto 1+
Mais de um ponto (várias sobreposições)2,3,4,...0

4.3- Teste de correlação serial

Outro teste, vulgarmente utilizado, para estudar a correlação entre os valores gerados é o coeficiente de correlação serial (entre series de números intervalados de K unidades); se considerarmos a sequência gerada como um processo estocástico, os diversos coeficientes de correlação (deslocados de K=1, 2,...' etc., unidades) constituirão a função de auto-correlaçao do processo.

Estima-se o coeficiente de auto-correlação (deslocados de K), através de

Este coeficiente assume valores entre 1 e +1., havendo, nestes dois casos, dependência linear completa entre as duas series de valores deslocados de k. Em principio, para os diversos k dever-se-iam esperar valores de p próximos de zero- Além disso a representação gráfica de p não deveria obedecer a nenhum padrão regular

No anexo a este artigo, também estão referidos os valores obtidos para p em função de K, bem como a representação gráfica desses valores.

5. Algumas aplicações dos números pseudo-aleatórios

5.1 Métodos de Monte-Carlo (simulação)

Esta e, talvez, a mais importante aplicação, em computador, dos números pseudo-aleatórios. Com efeito, sob esta designação, estão englobados métodos que permitem a resolução de problemas de índole a complexidade extremamente variadas; desde calculo de integrais a estudos de comportamento de filas de espera passando pela simulação de processos físicos.

Iremos ver dois exemplos muito simples: o calculo de um integral definido e a simulação do movimento de um neutrão através de uma parede de chumbo.

5.1.1- Calculo de um integral definido

Escolhemos a função f(x):x21 no intervalo [o;1J; cada ponto é representado por dois numeres pseudo-aleatorios; gerados em sequência, correspondendo às coordenadas do ponto.

Foram gerados 5000, 10000, 15000 e 20000 pontos, obtendo-se os seguintes resultados:

Por aqui, podemos ver a lentidão de convergência do processo o que obriga à utilização de grandes volumes de dados.

5.1.2 - Simulação do comportamento de neutrões

Admitamos que quer íamos dimensionar a espessura da parede de chumbo destinada a controlar o movimento doe neutrões num reactor nuclear sabendo-se sue e constante a distancia, d, percorrida, pelo neutrão, entre choques sucessivos com os átomos de chumbo da parede e que e aleatória a direcção que ele toma, após cada um desses choques; admita-se, ainda, que cada neutrão só pode suportar doze choques, antes de ser absorvido.

Para resolver o problema, escolhe-se uma espessura inicial e considera-se um grande numero de neutrões; para cada um deles, geram-se as direcções que ele toma, na sequência de cada choque e verifica-se se conseguiu atravessar a parede; finalmente, calcula-se a percentagem de neutrões que conseguiu atravessar a parede Se esta percentagem for suficientemente pequena, esta resolvido o problema; se não, aumenta-se a espessura e repete-se o processo.

usamos 20000 neutrões e testamos o método para espessuras da pare de iguais a 2d, Sd e 10d.

Na figura 5 representamos as trajectórias dos três primeiros neutrões, para o caso ESP = 5d.

3.2- Hashing

O problema de procurar um elemento numa lista é extremamente importante em Informática, existindo um grande numero de algoritmos que sintetizam diversos modos de resolver; um de exemplos desse problema e o acesso directo (ou aleatório) a registos de um ficheiro em disco.

Dois dos métodos mais simples de pesquisa de uma lista, que poderiam ser usados neste caso, são a procura linear e a procura binária. O primeiro consiste cm comparar a identificação do elemento (neste exemplo seria a chave do registo) com a de todos os componentes da lista (registos do ficheiro), ate encontrar (ou não) uma idêntica. O segundo compara a identificação do elemento com a do componente que ocupar a posição media na lista - se existir una relação de menor entre ambas, a segunda metade da lista é eliminada; se for de maior, será a primeira metade que fica de fora; se forem iguais, a procura termina; o processo é depois repetido para a lista resultante, até se atingir uma sub-lista só com um elemento. Qualquer destes métodos é extremamente ineficiente para o caso em questão: com efeito, se n for o número de elementos da lista, o número de comparações necessárias é proporcional a n (linear) ou a log2n (binária).

Neste género de aplicações, e bastante útil o método de "hashing": técnica de construção (e pesquisa) de tabelas que usa minimizar o número de tentativas para encontrar um elemento numa tabela; enquanto os dois primeiros métodos referidos usavam a chave do registo para o procurarem no ficheiro, o de "hashing" calcula primeiro, uma função da chave, cujo resultado é o endereço do registo procurado.

OBS: o verbo inglês "to hash" significa "picar (carne)" ou "reduzir a pedaços".

As funções de "hashingt' normalmente utilizadas são de natureza "aleatória" (mais precisamente, baseiam-se no cálculo de números pseudo-aleatorios, a partir da chave): distribuem-se, assim uniformemente, as chaves a armazenar pelo registos disponíveis. Torna-se, no entanto, necessário solucionar dois problemas: o de escolha de uma função apropriada e o do possível aparecimento, a partir de chaves distintas, do mesmo endereço.

Para ilustrar a técnica, admitamos que queríamos criar um ficheiro, de acesso directo, dos, digamos, cinco

mil alunos de uma escola, de acordo com o numero do Bilhete de Identidade de cada um deles; o comprimento do registo era de 100 "bytes" (de oito "bits"); o meio de suporte a usar seria uma pilha de discos magnéticos, com 20 cilindros de 20 pistas, cada uma podendo conter 29 registos. Como já referimos, e devido a tratar-se de uma função aleatória, a transformação ("hashing") de chaves distintas pode conduzir a um mesmo endereço; por isso, devemos deixar alguns registos por preencher, para resolvermos essas colisões (também designadas por sinónimos): essa margem de segurança é chamada, por vezes, factor de carga e um dos valores usados é 0,8.

Nestas condições, o número de endereços disponiveis, em disco, será de 6250 (5000/0,8)

Consideremos, agora, o cálculo do endereço correspondente à chave 2318474 (BI do autor)

1)Começa-se por gerar pseudo-aleatoriamente um numero a partir da chave - calcula-se o resto da divisão inteira entre a chave e o número primo imediatamente Inferior ao numero de registos disponíveis (6247).

2)O numero obtido (337) vai-nos dar o endereço, em disco, do registo; Como existem 29 registos por pista, fazendo a respectiva divisão inteira, obtemos
637 29
257 28
05

Onumero 5 indica a posição relativa ocupada pelo registo na pista; como o registo 0 é usado pelo sistema, temos de lhe adicionar 1, pelo que o nosso registo e o sexto na pista a que pertence.

3)O quociente (28) indica o cilindro e a pista: como existem 20 pistas por cilindro

Nota: a função de "hashing" que escolhemos não é a única utilizada (outro método usa um produto em vez da divisão inteira).

Como já referimos atrás, um problema que pode surgir, neste método, é a ocorrência de colisões. Por exemplo, a chave 2324721 (2318474 + 6247) conduziria ao mesmo endereço; um dos modos de resolver este problema é armazenar este registo (o que provocou a colisão) no 1º registo livre a seguir ao endereço obtido (este método não é único). Quando da procura de um determinado registo, ela é iniciada no endereço obtido pela função de hashing e continua até ser encontrada uma chave idêntica (registo encontrado) ou um endereço vazio (procura infrutífera)

Como é obvio, a procura em "hashing" exige que a construção da lista tenha sido através da mesma técnica.

5.3 - Verificarão da redundância cíclica (Cyclic Redundancy Check-CRC) Trata-se de um método de detecção de erros usado, por exemplo, para verificar a integridade da informação gravada em disco ou transmitida através de um canal de telecomunicações.

Uma das aplicações mais importante é em redes de computadores, onde é usado para verificar a validade da informação contida nos blocos de mensagens trocadas entre os nós da rede. Nesta técnica, os bits da zona de dados da mensagem são considerados como representação binária de inteiros e são divididos por uma constante de valor fixado (segundo a norma da CCITT, essa constante é 10000000000001, que pode ser considerada como uma representação dos coeficientes de um polinómio de grau 16:x16+x12+x5+1); o quoficiente é ignorado e os 16 bits que constituem o resto da divisão inteira do bloco pela constante são aproveitados como caracter de verificação do bloco (BCC – b_LOCK Check Character): se o BCC transmitido é idêntico ao BCC calculado pela estação receptora, a mensagem é aceite; caso contrário, aquela pede a repetição da mensagem.

Como temos visto, o resto da divisão inteira de um número por outro apresenta caracteristicas pseudo-aleatórias, pelo que este método é bastante eficiente na detecção de erros de transmissão.