MPEI 2018-2019

Aula 12

Funções de dispersão

Aplicações

Motivação

- Em muitos programas de computador torna-se necessário aceder a informação através de uma chave
  - Exemplo:
    - Obter nome associado a um número de telefone

- Em Java, por exemplo, temos estruturas de dados como HashMap e Hashtable

Um dicionário simples: `Hashtable`

- Para criar uma *Hashtable*:

```
import java.util.*;
Hashtable table = new Hashtable();
```

- Para colocar elementos (par chave-valor) na Hashtable, usa-se:

```
table.put(chave, valor);
```

- Para obter um valor:

```
valor = table.get(chave);
```

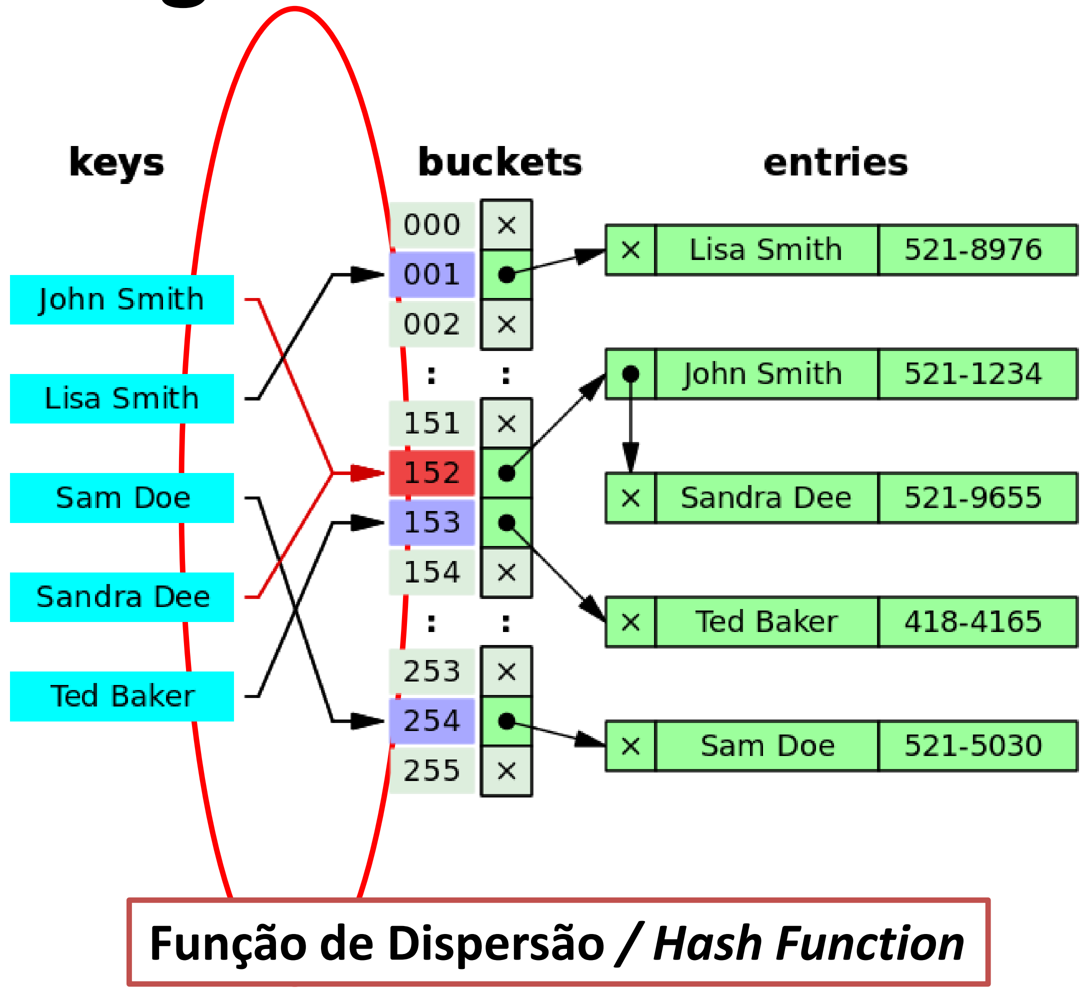
Exemplo de utilização de uma Hashtable

```
import java.util.*;

public class HashtableUser {

    public static void main(String[] args) {
        Hashtable<String, String> table =
                                           new Hashtable<String,
String>();

        table.put("one", "um");
        table.put("two", “dois”);
        table.put("three", "três");

        System.out.println("two -> " + table.get("two"));
        System.out.println("deux -> " + table.get("deux"));
    }
}
```

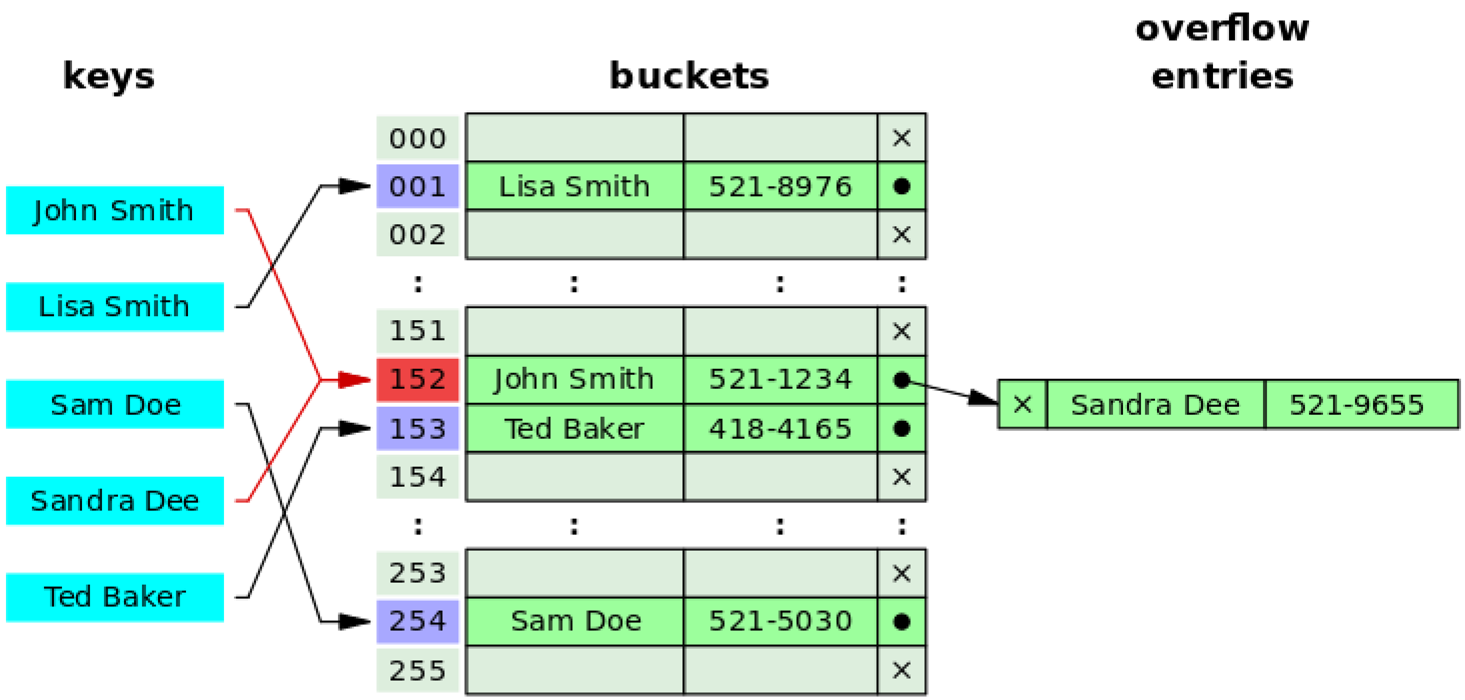
```
two -> dois
deux -> null
```

Implementação comum
Separate chaining with linked lists

- As chaves são transformadas em posições num array
  - usando uma função
- Cada posição do array é o início de uma lista ligada



Outra implementação
Separate chaining with list head cells



Funções de dispersão / Hash functions

- Uma função de dispersão (hash function) **mapeia chaves para um número pequeno de inteiros** (buckets)

- Uma função de dispersão ideal mapeia as chaves em inteiros de uma forma aleatória,
  - De forma a que os valores sejam igualmente distribuídos, mesmo quando existem regularidades nas chaves

- O processo pode ser dividido em dois passos:
  - Mapear a chave para um inteiro
  - Mapear o inteiro para um conjunto limitado (de inteiros)
    - Os denominados buckets

Existem muitas funções de dispersão

- Com diferentes graus de complexidade
- E diferenças de desempenho
  - Em geral dependente da aplicação
    - Pode ser útil testar várias

- Nesta aula apenas apresentamos alguns exemplos
  - E sugestões para procura de mais Informação

Funções de dispersão simples (para inteiros)

- As funções seguintes mapeiam uma única chave inteira (k) num número inteiro pequeno h(k), o número da posição/*bucket*

- **Método da divisão**
  - Escolher um primo que não seja próximo de uma potência de 2
  - h(k) = k mod m

  - m é o número de posições (igual ao tamanho da tabela), que deve ser um número primo

  - Funciona muito mal para muitos tipos de padrões nas chaves

Funções de dispersão simples (para inteiros)

- **Variante de Knuth para a Divisão**
  - h(k) = k(k+3) mod m
  - m é o número de posições (igual ao tamanho da tabela), que deve ser um número primo
  - É suposto ter muito melhor desempenho que o método anterior

demoKnuth.m

Função de dispersão de uma sequência de caracteres

- Como sabemos uma sequência de caracteres (String) é em geral representada como uma sequência de inteiros
- A função de dispersão para Strings tem por entrada uma sequência de inteiros k=k1,..., ki, ….,kn e produz um número inteiro pequeno h(k).

- Os algoritmos para este tipo de entrada assumem que os inteiros são de facto códigos de caracteres, fazendo uso do seguinte:

- Em muitas linguagens um caracter é representado em 8 bits
- O código ASCII apenas usa 7 desses 8 bits
- Desses 7, os caracteres comuns apenas usam os 6 menos significativos
  – E o mais significativo desses 6 indica essencialmente se é maiúscula ou minúscula, muitas vezes pouco relevante

- Em consequência os algoritmos concentram-se na preservação do máximo de informação dos 5 bits menos significativos de cada número, fazendo muito menos uso dos 3 bits mais significativos

Função de dispersão de uma sequência de caracteres (String)

- Começar por inicializar h (h=0)
- Percorrer a sequência de inteiros (representando caracteres)
  - Combinando os inteiros ki, um por um, a h
    - Os algoritmos diferem na forma como combinam ki com h
- O resultado final é dado por h mod m

- Nota: Na utilização dos algoritmos seguintes, as entradas ki são inteiros sem sinal (unsigned int)
  - A utilização de representações de inteiros com sinal pode resultar em comportamentos estranhos

Alguns algoritmos – variante CRC

- Fazer um shift circular de 5 bits para a esquerda ao h.
- De seguida fazer XOR de h com ki.

CRC variant: Do a 5-bit left circular shift of h. Then XOR in ki. Specifically:

```
highorder = h & 0xf8000000    // extract high-order 5 bits from h
                              // 0xf8000000 is the hexadecimal representation
                              //   for the 32-bit number with the first five
                              //   bits = 1 and the other bits = 0
h = h << 5                    // shift h left by 5 bits
h = h ^ (highorder >> 27)     // move the highorder 5 bits to the low-order
                              //   end and XOR into h
h = h ^ ki                    // XOR h and ki
```

Alguns Algoritmos– PJW hash

- Deslocar (shift) h 4 bits para a esquerda
- Adicionar ki
- Mover 4 bits mais significativos

PJW hash (Aho, Sethi, and Ullman pp. 434-438): Left shift h by 4 bits. Add in ki. Move the top 4 bits of h to the bottom. Specifically:

```
// The top 4 bits of h are all zero
h = (h << 4) + ki                // shift h 4 bits left, add in ki
g = h & 0xf0000000               // get the top 4 bits of h
if (g != 0)                      // if the top 4 bits aren't zero,
   h = h ^ (g >> 24)             //   move them to the low end of h
   h = h ^ g
// The top 4 bits of h are again all zero
```

Exemplo de uma função completa

- Mapeia uma string de comprimento arbitrário num inteiro (>=0)
- DJB31MA

```
uint hash(const uchar* s, int len, uint seed)
{
   uint h = seed;
   for (int i=0; i < len; ++i)
       h = 31 * h + s[i];
   return h;
}
```

  – Fonte: Paulo Jorge Ferreira “MPEI – summary” 2014

Outro exemplo (em Matlab)

```
function hash=string2hash(str,type)
% This function generates a hash value from a text string
%
% hash=string2hash(str,type);
%
% inputs,
%   str : The text string, or array with text strings.
% outputs,
%   hash : The hash value, integer value between 0 and 2^32-1
%   type : Type of has 'djb2' (default) or 'sdbm'
%
% From c-code on : http://www.cse.yorku.ca/~oz/hash.html
 .....
```

- From: http://www.mathworks.com/matlabcentral/fileexchange/27940-string2hash/content/string2hash.m

demoStr2Hash.m

Exemplo de uso de string2hash

```
>> demoStr2Hash
Chave=        António   ->  Hash Code = 36
Chave=        Antónia   ->  Hash Code = 22
Chave=         Manuel   ->  Hash Code  = 48
Chave=           Manu   ->  Hash Code   = 21
Chave=        Manuela   ->  Hash Code = 88
Chave=          Vitor   ->  Hash Code = 33
```

Universal hashing

- O esquema de hashing universal selecciona uma função h de entre uma família de funções de tal forma que a probabilidade de colisão entre quaisquer duas chaves distintas seja 1/n
  - Sendo n o número de resultados diferentes desejados para a função

- Estes métodos asseguram (probabilisticamente) que a aplicação da função de dispersão se comportará como se estivéssemos a usar uma função aleatória, para qualquer distribuição da entrada

- Podem utilizar mais operações do que uma função específica

- Uma implementação faz parte da package "Data Structures" de *Brian Moore (2014)*
  - *As funções HashCode e InitHashFunction functions encontram-se em: http://www.mathworks.com/matlabcentral/fileexchange/45123-data-structures/content/Data%20Structures/Hash%20Tables/HashTable.m*

InitHashFunction()

```
%
function InitHashFunction(this)
    % Set prime parameter
    ff = 1000; % fudge factor
    pp = ff * max(this.m + 1,76);
    pp = pp + ~mod(pp,2); % make odd
    while (isprime(pp) == false)
        pp = pp + 2;
    end
    this.p = pp; % sufficiently large prime number

    % Randomized parameters
    this.a = randi([1,(pp - 1)]);
    this.b = randi([0,(pp - 1)]);
    this.c = randi([1,(pp - 1)]);
end
```

HashCode()

```
    function hk = HashCode(this,key)
        % Convert character array to integer array
        ll = length(key);
        if (ischar(key) == false)
            % Non-character key
            HashTable.KeySyntaxError();
        end
        key = double(key) - 47; % key(i) = [1,...,75]

        %
        % Compute hash of integer vector
        %
        % Reference: http://en.wikipedia.org/wiki/Universal_hashing
        %            Sections: Hashing integers
        %                      Hashing strings
        %
        hk = key(1);
        for i = 2:ll
            % Could be implemented more efficiently in practice via bit
            % shifts (see reference)
            hk = mod(this.c * hk + key(i),this.p);
        end
        hk = mod(mod(this.a * hk + this.b,this.p),this.m) + 1;
    end
end
```

Propriedades

- Requer-se, em geral, que as funções de dispersão satisfaçam algumas propriedades, como:

- Serem determinísticas

- **Uniformidade**:
  - Uma boa função de dispersão deve mapear as entradas esperadas de forma igual por toda a gama de valores possíveis para a sua saída

  - Todos os valores possíveis para a função de dispersão devem ser gerados com aproximadamente a mesma probabilidade

Como ter k funções de dispersão ?

Possíveis soluções:

1. Ter mesmo k funções diferentes
2. Usar funções customizáveis (definindo uma família de funções) e usando parâmetros diferentes
3. Usar a mesma função de dispersão e processar a chave por forma a ter k chaves diferentes baseadas na chave original

Exemplo (Matlab):

```
for i=1:k
    str= [str num2str(i)];
    h=HashCode(hash,m,str);
end
```

Propriedades (continuação)

- As k funções de dispersão devem cumprir um requisito adicional:
- Produzir resultados **não-correlacionados**
- Esta propriedade é muito importante e é aconselhável verificá-la/avaliá-la em trabalhos envolvendo várias funções

“Teste” de funções de dispersão

- Um teste simples e básico consiste em:

1. Gerar um conjunto grande de chaves (pseudo)aleatórias

2. Processar todas essas chaves com as k funções de dispersão
   - Guardando os resultados produzidos (*hash codes*)

3. Analisar o histograma de cada função de dispersão
   - Para verificar a uniformidade da distribuição dos *hash codes*

4. Calcular, visualizar e analisar as correlações entre os resultados produzidos pelas várias funções de dispersão

Alguns links

- http://www.mathworks.com/matlabcentral/fileexchange/27940-string2hash/content/string2hash.m
- http://www.mathworks.com/matlabcentral/fileexchange/45123-data-structures/content/Data%20Structures/Hash%20Tables/HashTable.m
- http://www.cse.yorku.ca/~oz/hash.html
- http://programmers.stackexchange.com/questions/49550/which-hashing-algorithm-is-best-for-uniqueness-and-speed
- https://www.cs.hmc.edu/~geoff/classes/hmc.cs070.200101/homework10/hashfuncs.html
- https://en.wikipedia.org/wiki/Hash_function#Properties
- https://en.wikipedia.org/wiki/Universal_hashing
- http://www.i-programmer.info/programming/theory/2664-universal-hashing.html

Aplicação interessante de soma de variáveis aleatórias (e geração de números aleatórios)

Contadores estocásticos

Motivação

- Evitar contadores grandes quando o volume de dados é grande
  - Por exemplo: na contagem de células
- Como um contador de n bits contará no máximo até 2^n eventos, será este o limite a ultrapassar

Primeira solução

- Para duplicar o número de eventos que se podem contar, incrementa-se o contador com probabilidade 1/2 cada vez que ocorre um evento

- A ideia *é incrementar o contador metade das vezes (em média)*

…

- Com base na função rand() podemos agora tomar decisões aleatórias com probabilidade 1/2 e portanto construir uma função para incrementar (ou não) o contador:

```
if (rand() < 0.5) then
    incrementar_contador
endif
```

Em octave/ Matlab

- Podemos facilmente simular o resultado após 100 eventos:

```
% gera 100 var aleatórias indep em [0,1]
x = rand(1, 100);

% calcular quantas são < 0.5
n = sum(x < 0.5);
```

- n representará o valor do contador após os 100 eventos

contadores1.m

Qual é o valor médio do contador após k eventos?

- O contador é uma variável aleatória, determinada por uma sucessão de experiências aleatórias

- Associando uma variável aleatória a cada evento, de forma a representá-lo probabilisticamente
- Seja X_i a variável aleatória que representa o incremento i, com valor 1 se o contador foi incrementado, e valor zero caso contrário.
- Como $P(X_i = 0)$ e $P(X_i = 1)$ são iguais a 1/2, tem-se

- $E[X_i] = 0 \times P(X_i = 0) + 1 \times P(X_i = 1) = \frac{1}{2}$

Valor médio

- O valor do contador após k eventos é a soma dos k incrementos, $S = X_1 + X_2 + \cdots + \ X_k$

- E o valor médio:
- $E[S] = E[X_1 + X_2 + \cdots + \ X_k]$
- $= E[X_1] + E[X_2] + \cdots + E[X_k]$
- $= \frac{1}{2} + \cdots + \frac{1}{2} = \frac{k}{2}$

- Como o valor médio do contador após k eventos é $k/2$, o número de eventos pode ser estimado através do dobro do número registado pelo contador

Variância

- A variância de um qualquer dos X_i é
- $Var(X_i) = E\left[{X_i}^2\right] - \ (E[X_i])^2$

- $E\left[{X_i}^2\right] = 0^2 \times P(X_i = 0) + 1^2 \times P(X_i = 1)$
- $= \frac{1}{2}$

- $Var(X_i) = \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$

Variância (continuação)

- Como as variáveis X_i são independentes, a variância de S é
- $Var(S) = Var(X_1 + X_2 + \cdots + X_k)$
- $= Var(X_1) + Var(X_2) + \cdots + Var(X_k)$
- $= \frac{1}{4} + \cdots + \frac{1}{4} = \frac{k}{4}$
- O que implica $\sigma = \frac{\sqrt{k}}{2}$
- Para n=10000 teremos:
  - média 5000
  - desvio padrão 50

Comparar com simulação (contadores1.m)

Distribuição de probabilidade

- Pode calcular-se a probabilidade de, após k eventos, o valor do contador ser n.

- Fixemos $k = 4$:
- Teremos $X_1, X_2, X_3\ e\ X_4$
  – Variáveis binárias que descrevem se o contador é incrementado ou não após o evento 1,2,3 e 4
- O que nos dá 16 possibilidades (2^4)

X_1	X_2	X_3	X_4	valor do contador
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	2
0	1	0	0	1
0	1	0	1	2
0	1	1	0	2
0	1	1	1	3
1	0	0	0	1
1	0	0	1	2
1	0	1	0	2
1	0	1	1	3
1	1	0	0	2
1	1	0	1	3
1	1	1	0	3
1	1	1	1	4

- É agora fácil determinar as probabilidades, por contagem:
- $p(0) = \frac{1}{16}$
- $p(1) = \frac{4}{16}$
- $p(2) = \frac{6}{16}$
- $p(3) = \frac{4}{16}$
- $p(4) = \frac{1}{16}$

Generalizando

- Sendo p a probabilidade de incrementar e $1 - p$ a probabilidade de não incrementar …
- A probabilidade de observar uma soma igual a n após k experiências é:

$$p(n) = \binom{k}{n} p^n \ (1 - p)^{k-n}$$

Variante 1

- Como proceder para alargar mais a gama do contador?

- Imaginemos, por exemplo, que se quer multiplicar por 64 essa gama. A solução natural é incrementar com probabilidade $1/64$ em vez de ½

- O valor médio de X_i será agora $\frac{1}{64}$
- E $E[S] = \quad \dots . = \frac{k}{64}$
- Neste caso, o número de eventos pode ser estimado por $64\, n$, sendo n o valor do contador

Segunda solução

- Neste caso o contador é incrementado com probabilidade cada vez menor à medida que o seu valor aumenta:
- quando o contador contém *n*, a probabilidade de um incremento é 2^{-n}

Eventos	*Valor do contador*	*Número de eventos*
x	1	1
x		
x	2	3
x		
x		
x		
x	3	7
x		
x		
x		
x		
x		
x		
x		
x	4	15

Para mais informação

- Ver notas do ano lectivo 2014/2015 de autoria do Prof. Paulo Jorge Ferreira
  - (disponíveis no elearning da UC)

E se a soma for de outros tipos de variáveis ?

Soma
e combinação linear
de variáveis aleatórias

Motivação

- Se somarmos duas variáveis aleatórias X_1 e X_2 quais as características da variável aleatória $S = X_1 + X_2$?

  – Em termos de momentos ?
    - Em especial média e variância

  – Em termos de função de distribuição ?

- E no caso geral $S_n = X_1 + X_2 + \cdots + X_n$?

Média da soma de n variáveis

- Sejam $X_1, X_2, \dots, X_n$, n variáveis aleatórias e $S_n = X_1 + X_2 + \cdots + X_n$ a sua soma
- Teorema: **A média da soma de n variáveis é igual à soma das médias**
- Demonstração

$$
\begin{aligned}
E\left[S_n\right] &= \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \left(\sum_{j=1}^{n} x_j\right) f_{X_1 \dots X_n}(x_1, \dots, x_n) dx_1 \dots dx_n = \\
&= \sum_{j=1}^{n} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} x_j f_{X_1 \dots X_n}(x_1, \dots, x_n) dx_1 \dots dx_n = \\
&= \sum_{j=1}^{n} \int_{-\infty}^{+\infty} x_j f_{X_j}(x_j) dx_j = \sum_{j=1}^{n} E\left[X_j\right]
\end{aligned}
$$

Variância da soma de n variáveis

- Considerando da mesma forma $S_n = X_1 + X_2 + \cdots + X_n$:
- Teorema: **A variância da soma de n variáveis aleatórias é dada pela soma de todas as variâncias e covariâncias**

$$Var(S_n) = \sum_{i=1}^{n} Var(X_i) + \sum_{j=1}^{n} \sum_{\substack{k=1 \\ j \neq k}}^{n} Cov(X_j, X_k)$$

- Demonstração:

$$\begin{aligned} Var(S_n) &= E\left[\sum_{j=1}^{n} (X_j - E[X_j]) \sum_{k=1}^{n} (X_k - E[X_k])\right] = \\ &= \sum_{j=1}^{n} \sum_{k=1}^{n} E[(X_j - E[X_j])] \, E[(X_k - E[X_k])] \end{aligned}$$

Variância da soma de n variáveis

- Se as variáveis **são independentes**, $Cov\left(X_j, X_k\right) = 0$, para todo o $j \neq k$, pelo que:

- $\boldsymbol{Var(S_n) = \sum_{i=1}^{n} Var(X_i)}$
  - **Variância da soma igual a soma das variâncias**

- Se para além de independentes forem **identicamente distribuídas (IID)**
  e tivermos $E[X_i] = \mu$ e $Var(X_i) = \sigma^2, i = 1,2, \ldots, n$
  a média e variância da soma são dadas por:

- $\mathrm{E}[S_n] = n\,\mu$ e $Var(S_n) = n\,\sigma^2$

Função de distribuição da soma de 2 variáveis aleatórias independentes

- Caso discreto (2 v.a. Discretas X e Y)
- Fazendo $\mathrm{Z} = X + Y$
- $p_Z(z) = P(X + Y = z)$

$=\sum_{\{(x,y)|x+y=z\}} P(X = x, Y = y)$

$$= \sum_x P(X = x, Y = z - x)$$

$= \sum_x p_X(x) \quad p_Y(z - x)$; devido à indep.

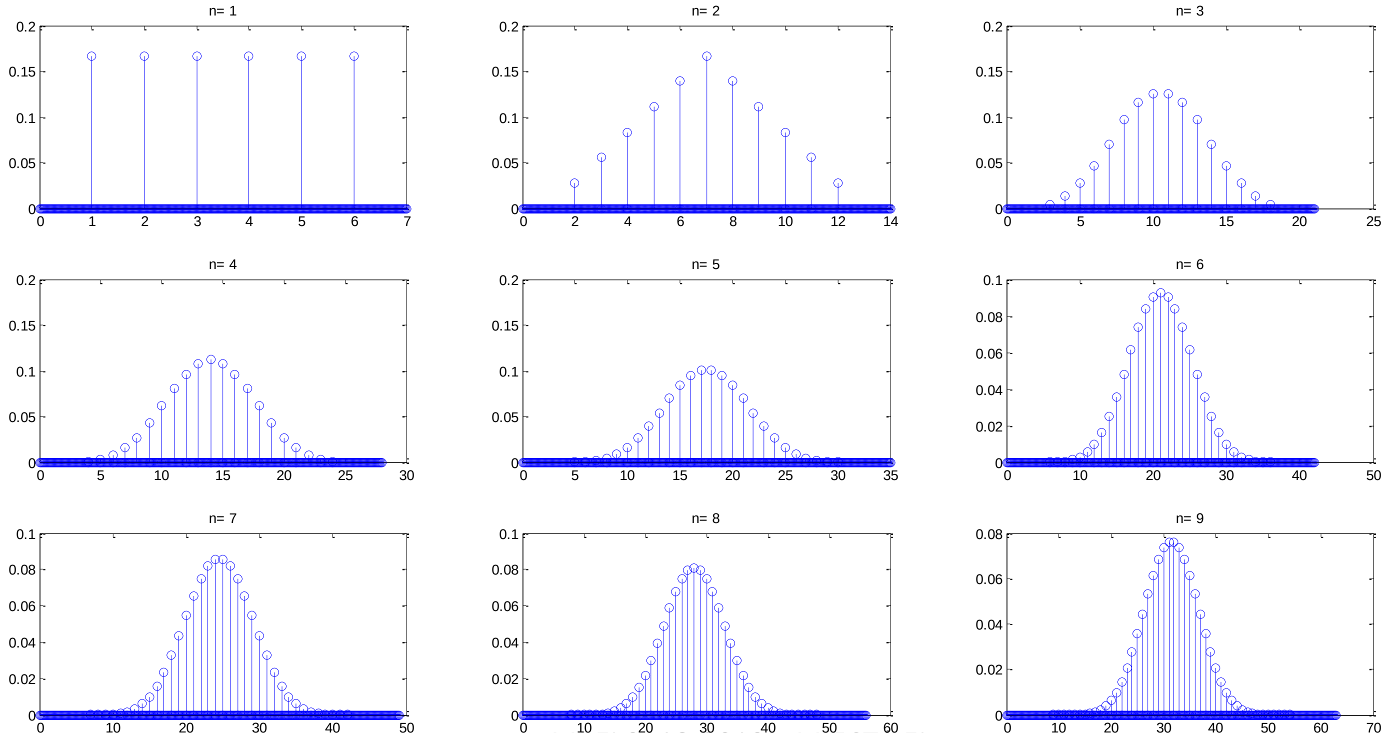
$= p_X\ (x)\ *\ p_Y(z)$

demoConvolucao.m

- Que é a **convolução** discreta de p_X e p_Y

Exemplo (em Matlab)

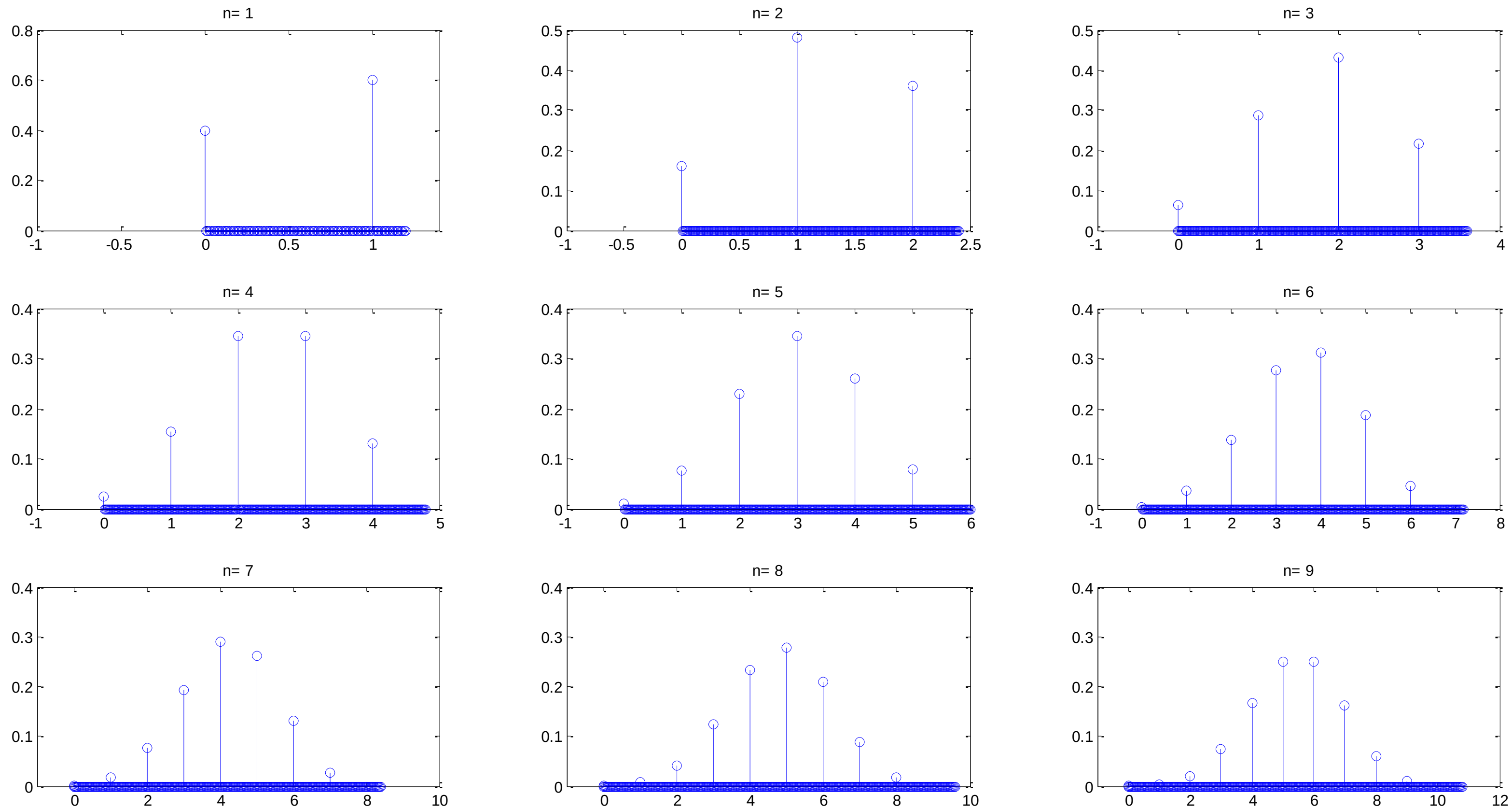
- Usando conv() e a pmf relativa à variável *X* correspondente à soma de *n* lançamentos de um dado honesto (*n*=1, 2, …, 9)



Outro exemplo

demoConvolucaoMoeda.m

- Sendo *X* relativa ao número de caras em *n* lançamentos de uma moeda não honesta
  - com probabilidade de cara = 0,6



Caso contínuo

- Sendo X e Y independentes e contínuas
- Fazendo novamente $\mathrm{Z} = X + Y$
- Para obter a função densidade prob. de Z , primeiro obtém-se a f. densidade conjunta de X e Z e depois integra-se

$$
\begin{aligned}
F_{Z|X}(z \mid x) &= \mathbf{P}(Z \leq z \mid X = x) = \mathbf{P}(X + Y \leq z \mid X = x) = \mathbf{P}(x + Y \leq z \mid X = x) = \mathbf{P}(Y \leq z - x \mid X = x) \\
&= \mathbf{P}(Y \leq z - x) \qquad \text{using the independence of } X \text{ and } Y \\
&= F_Y(z - x)
\end{aligned}
$$

$$
f_{Z|X}(z \mid x) = \frac{d}{dz} F_{Z|X}(z \mid x) = \frac{d}{dz} F_Y(z - x) = f_Y(z - x)
$$

$$
\begin{aligned}
f_Z(z) &= \int_{-\infty}^{\infty} f_{X,Z}(x,z)\,dx = \int_{-\infty}^{\infty} f_X(x) f_{Z|X}(z \mid x)\,dx \\
&= \int_{-\infty}^{\infty} f_X(x) f_Y(z - x)\,dx \equiv f_X * f_Y(z)
\end{aligned}
$$

Obtém-se através da convolução, agora contínua

- Para mais informação, ver, por exemplo:
- https://engineering.purdue.edu/~ipollak/ece302/SPRING12/notes/23_GeneralRVs-8_Sums.pdf

Combinações lineares de variáveis aleatórias

- Os **resultados anteriores generalizam-se** facilmente para o caso de termos uma soma pesada (combinação linear) $Y_n = c_1X_1 + c_2X_2 + \cdots + c_nX_n$
  - Em que $c_1, c_2, \ldots, c_n$ são constantes

- $E[Y_n] = c_1E[X_1] + c_2E[X_2] + \cdots + c_nE[X_n]$
- $Var(Y_n) = \sum_{i=1}^{n} {c_i}^2\, Var(X_i) + \sum_i \sum_{j\ (\neq i)} c_i c_j Cov(X_i, X_j)$

- Se independentes $Var(Y_n) = \sum_{i=1}^{n} {c_i}^2\, Var(X_i)$

Exemplo de aplicação

- Um semicondutor tem 3 camadas.
- Se a espessura das várias camadas tiver uma variância de 25, 40 e 30 nanómetros quadrados, qual a variância da espessura das 3 camadas ?

- Considerando X_1, X_2, X_3 como as espessuras das camadas e independência
- $Var(X_1 + X_2 + X_3) = Var(X_1) + var(X_2) + var(X_3) = 95$
- Mostrando a propagação das variâncias de todas as camadas para o resultado final
  - Os problemas somam-se ☹

Funções de variáveis aleatórias

A soma (simples ou pesada) de variáveis aleatórias é um caso particular

Funções de v. a. múltiplas

- Muitas vezes encontramos **problemas em que temos uma transformação das v. a.** $X_1, X_2, \ldots, X_n$ que produz variáveis aleatórias $Y_1, Y_2, \ldots, Y_m$
- O caso mais simples é termos uma função escalar de várias variáveis aleatórias

$$Z = g(X_1, X_2, \ldots, X_n)$$

- A função de distribuição acumulada de Z é determinada como sabemos calculando a probabilidade do conjunto $\{Z \leq z\}$

  i.e. A região R_z do espaço n-dimensional tal que

  $R_z = \{\mathbf{x}: g(\mathbf{x}) \leq \mathbf{z}\}$ $\qquad \mathbf{x} = (\boldsymbol{x_1}\ \boldsymbol{x_2}\ \ldots\ \boldsymbol{x_n})$
- Logo
  - $F_Z(z) = \int_{\mathrm{x} \in R_Z} \ldots \int f_X(\mathrm{x})\, d\mathrm{x}$

Expectância de funções de v. aleatórias

- Expectância de uma função de 2 var. aleatórias

  $Z = g(X,Y)$

  $$E[Z] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y) f_{X,Y}(x,y)\, dx\, dy$$

- Para o caso de variáveis discretas :
  - $E[Z] = \sum_i \sum_n g(x_i, y_n)\, p_{X,Y}(x_i, y_n)$

- Resultado generalizável para uma função com um número arbitrário de variáveis aleatórias

  $Z = g(\mathbf{X})$ em que $\mathbf{X}$ (bold) é um vector

  $$E[Z] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}$$

Exemplo

- $Z = g(X,Y) = X + Y \quad i.e.$ Soma de 2 v.a.
- $E[Z] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x+y) f_{X,Y}(x,y)\, dx\, dy$
- $= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x\, f_{X,Y}(x,y)\, dx\, dy + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y f_{X,Y}(x,y)\, dx\, dy$
- $= \int_{-\infty}^{\infty} x\, f_X(x)\, dx + \int_{-\infty}^{\infty} y f_Y(y)\, dy$
- $= \boldsymbol{E(X) + E(Y)}$
- Resultado válido quer as variáveis **sejam independentes ou não**
- Mostra (conjuntamente com a propriedade de escala, $E(aX) = aE(X)$) que a expectância é um **operador linear**

Momentos de funções de variáveis aleatórias

- Momento de ordem n de uma função escalar de um vector aleatório:

  $Z = g(\mathbf{X})$ em que $\mathbf{X}$ (bold) é um vector

  $$E[\boldsymbol{Z}^{\boldsymbol{n}}] = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \boldsymbol{g}^{\boldsymbol{n}}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}$$

- Aplicando para obter a variância temos:
- $\mathrm{Var}[Z] = E[Z^2] - E^2[Z]$
- $= \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \boldsymbol{g}^{2}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x} - \left(\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \boldsymbol{g}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}\right)^{\mathbf{2}}$