Métodos Probabilísticos para Engenharia Informática

2019-2020

Informações sobre a cadeira

Objectivos da cadeira

- Desenvolver a capacidade de aplicar métodos probabilísticos em engenharia informática
  - Suportada no conhecimento de conceitos essenciais

- Complementar a formação em métodos determinísticos

  (da generalidade das outras UCs do MIECT e LEI)

Funcionamento da cadeira

- TPs (2x1.5 h) + PL (2 h) por semana

Teórico-Práticas:

- **Noções básicas de probabilidade**
- **Simulação**
- **Variáveis aleatórias e distribuições**
- **Aplicações representativas**
- **Cadeias de Markov**

Aulas Práticas

- Guiões para 1 (ou 2 aulas)
  - PL 01 - Probabilidades (e simulação)
  - PL 02 - Probabilidades condicionais
  - PL 03 - Variáveis aleatórias e Distribuições
  - PL 04 - Geração de Números aleatórios
  - PL 05 - Hash Functions e Simulação
  - PL 06 - Bloom filters
  - PL 07 - Similaridade
  - PL 08 - Cadeias de Markov

OT

- Por marcação, email para o respetivo docente até às 12 horas do dia da OT:
  - Terça-feira, Prof. António Teixeira, ajst@ua.pt
  - Quinta-feira, Prof. Carlos Bastos, cbastos@ua.pt
- Alternativamente, marcação direta com os respectivos docentes.

Faltas

- Há marcação de faltas nas TPs e nas PLs nos termos do Regulamento de Estudos da Universidade de Aveiro https://www.ua.pt/sga/PageText.aspx?id=4646

Equipa docente 2019-2020

- António Teixeira (ajst@ua.pt) – Regente Coordenador
  - TP1, 2ª e 3ª
  - P3
  - OT1 (3ª)
- Carlos Bastos (cbastos@ua.pt)
  - TP2, 2ª e 5ª
  - P7
  - OT2 (5ª)

- Amaro Sousa (asou@ua.pt)
  - P6, P2, P4, P5

Avaliação

- **Avaliação discreta**:
  - 25 % **TP** (exame escrito a realizar em data a anunciar, em princípio, na 3ª ou 4ª semanas de outubro)
  - 15 % **P** (avaliação do desempenho nas aulas práticas)
  - 35 % **P** (mini projecto e sua apresentação)
  - 25 % **P** (mini teste prático em computador)
    - Época de exames

Informação no PACO

AVALIAÇÃO DISCRETA

Definir como avaliação pr

Adicionar Elementos de Avaliação

componentes da UC	P	TP
peso da componente	**75%**	**25%**
Miniteste TP / minitest TP	0%	25%
Trabalho prático / Practical Assignment	35%	0%
Avaliação de desempenho nas aulas práticas / Evaluation of Performance in Practical Classes	15%	0%
Miniteste Prático / Practical Minitest *	25%	0%

*Os elementos assinalados com * decorrem na época de exames.*

Métodos probabilísticos para cursos de Engª. de Computadores e de Engª Informática?

Probabilidades para Informática ?

- **Muitos problemas** na área da Informática, Ciências da Computação e afins **contêm algum grau de aleatoriedade**

- Exemplos:
  - Quantos computadores estarão ligados ao longo do dia a uma determinada rede wireless?
  - Qual a palavra mais provável que um utilizador irá escrever ao escrever um SMS?
  - Quais as páginas da web que têm mais relevância para uma procura ?

Probabilidades para Informática ?

- Também se podem **resolver** muitos problemas usando abordagens não determinísticas …
  - Muitas vezes com vantagens em termos de, por exemplo, velocidade

Exemplos de Aplicação

- Algoritmos probabilísticos
  - Ordenação, Métodos de Monte Carlo e Las Vegas
- Simulação
  - Redes de dados, ataques informáticos …
- Análise probabilística de algoritmos
- Teste de Software
- Poupança de memória
  - Ex: Bloom filters, contadores aleatórios
- Programação probabilística

Algoritmos probabilísticos

- Algoritmos que efetuam decisões aleatórias durante a sua execução

- Exemplo: Quicksort com pivot decidido de forma aleatória

- Vantagens:
  - Para muitos problemas um algoritmo probabilístico é o mais simples, o mais rápido, ou ambos

Algoritmos probabilísticos:

exemplos áreas de aplicação

- Teoria de números
  - Teste de números primos
- Estruturas de dados
  - Procura, ordenação, geometria computacional…
- Identidades algébricas
  - Verificação de matrizes e polinómios
- Programação matemática
  - Programação linear
- Grafos
  - Caminho mais curto…
- Contagem e enumeração
  - Contagem de estruturas combinatórias
- Computação paralela e distribuída
  - Evitar deadlock, consenso distribuído
- …

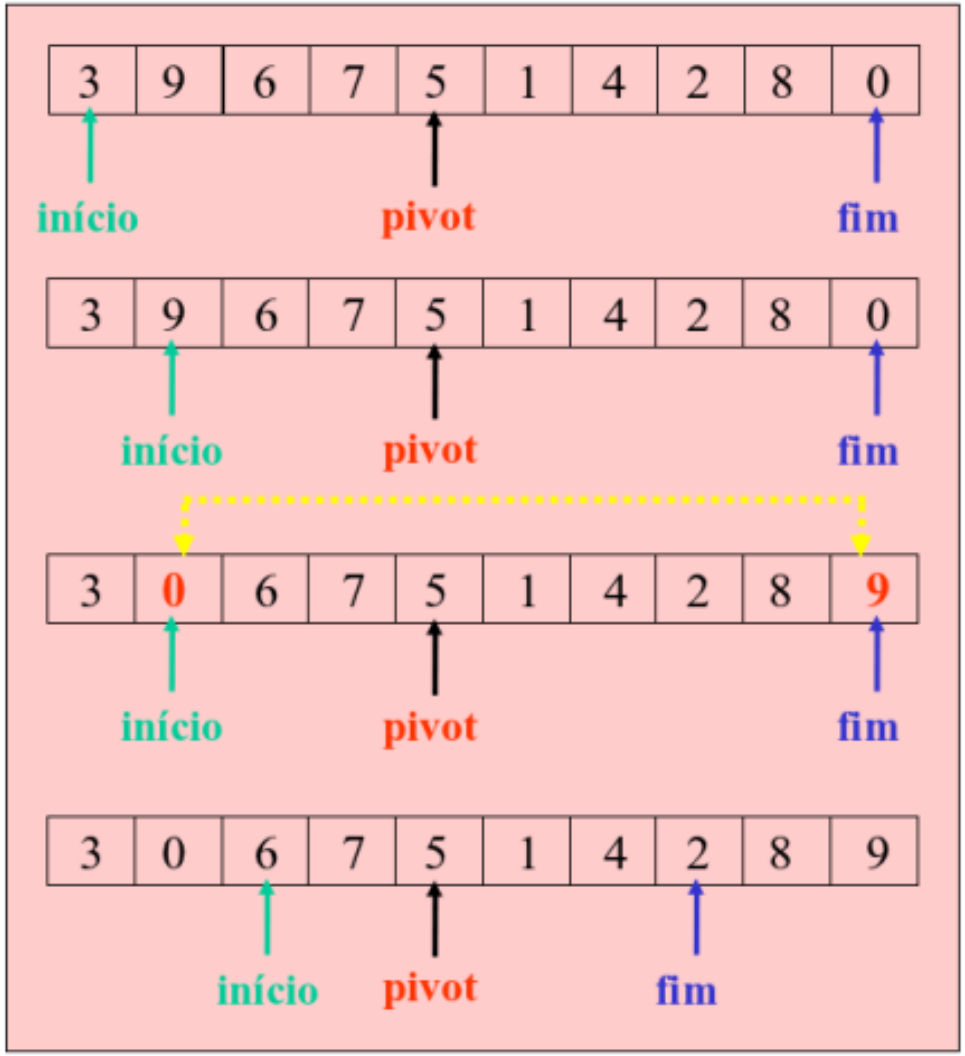
Quicksort

- O algoritmo **Quicksort** é um método de ordenação muito rápido e eficiente, inventado por C.A.R. Hoare em 1960
  - Quando visitou a Universidade de Moscovo como estudante,
  - Para traduzir um dicionário de inglês para russo, ordenando as palavras,
  - O objetivo era reduzir o problema original em subproblemas que possam ser resolvidos mais fácil e rapidamente.

Algoritmo

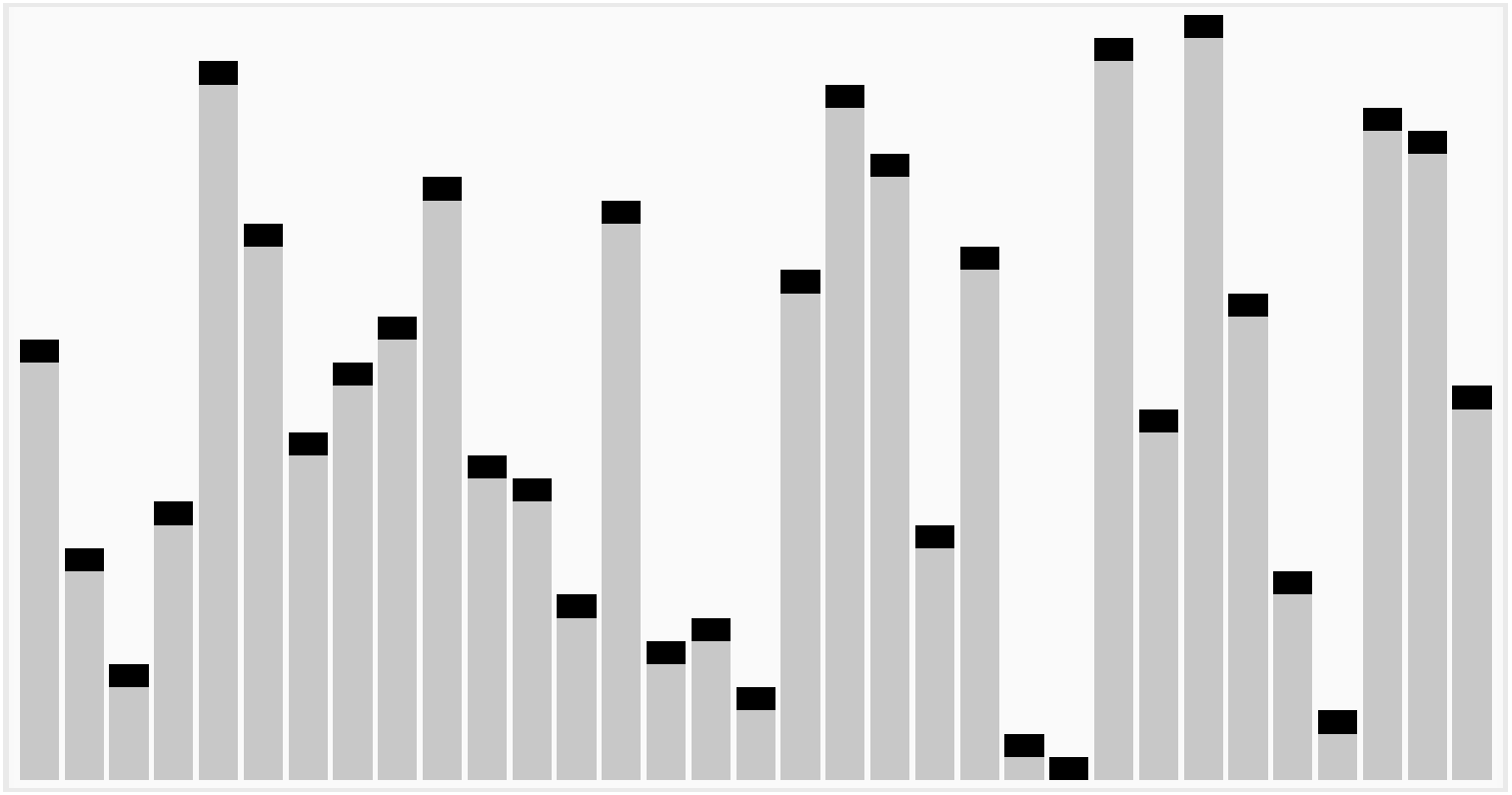
- Escolha um elemento da lista, denominado pivô;
- Rearranje a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele.
  - Ao fim do processo o pivô estará em sua posição final e haverá duas sublistas não ordenadas.
  - Essa operação é denominada partição
- Recursivamente, ordene a sublista dos elementos menores e a sublista dos elementos maiores;
  - O processo é finito

Partição



1. Escolher o pivot;
2. Movimentar o "inicio" até encontrar um elemento maior que o pivot;
3. Movimentar o "fim" até encontrar um elemento menor que o pivot;
4. Trocar o elemento encontrado no ponto 2 com o elemento encontrado no ponto 3;
5. Recomeçar o processo (i.e. voltar ao ponto 2) até que: "inicio" > "fim"

Quick Sort em acção



Código (Java)

- De P2

```
static void quickSort(int[] a, int start, int end) {
  assert validSubarray(a, start, end);
  int n = end-start;
  if (n < 2) // should be higher (10)!
    sequentialSort(a, start, end);
  else {
    int posPivot = partition(a, start, end);
    quickSort(a, start, posPivot);
    if (posPivot+1 < end)
      quickSort(a, posPivot+1, end);
  }
  assert isSorted(a, start, end);
}

static int partition(int[] a, int start, int end) {
  int pivot = a[end-1];
  int i1 = start-1;
  int i2 = end-1;
  while(i1 < i2) {
    do
      i1++;
    while(a[i1] < pivot);
    do
      i2--;
    while(i2 > start && a[i2] > pivot);
    if (i1 < i2)
      swap(a, i1, i2);
  }
  swap(a, i1, end-1);
  return i1;
}
```

Parte do código- Partição

```
static int partition(int[] a, int start, int end) {
  int pivot = a[end-1];
  int i1 = start-1;
  int i2 = end-1;
  while(i1 < i2) {
    do
      i1++;
    while(a[i1] < pivot);
    do
      i2--;
    while(i2 > start && a[i2] > pivot);
    if (i1 < i2)
      swap(a, i1, i2);
  }
  swap(a, i1, end-1);
  return i1;
}
```

Partição com pivot aleatório

```
int partitionRandomPivot(int[] a, int start, int end) {

   //  pivot part
     int randPosition= ((int) Math.floor(Math.random()*(end-start)))+start;
   System.out.printf("Pivot will be %d\n",a[randPosition]);

   swap(a,randPosition, end-1); // new :  save pivot at last position

   //  code below is the same
   int pivot=a[end-1];
   int i1 = start-1;
   int i2 = end-1;

   while(i1 < i2) {
              //  enquanto menor que pivot
              do
              i1++;
              while(a[i1] < pivot);
              //  enquanto maior que pivot e ...
               …

              if (i1 < i2){
                swap(a, i1, i2);
              }
   swap(a, i1, end-1);  //  restore pivot
   return i1;  //  <---
}
```

Análise probabilística de algoritmos

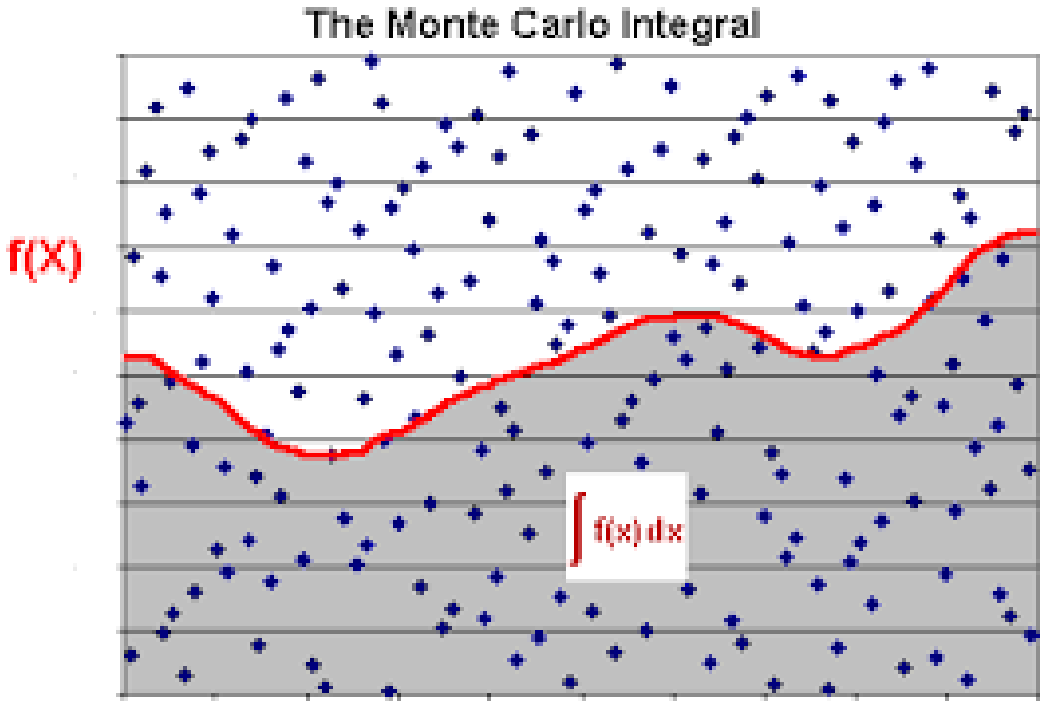
- Usa teoria de probabilidades para analisar o comportamento / desempenho de algoritmos (probabilísticos e determinísticos)

- Porquê ?
  - Naturalmente, algoritmos probabilísticos terão desempenho não determinístico
  - Também, o comportamento dos alg. determinísticos varia com as entradas
  - A análise probabilística permite estimar limites

- Exemplo:
  - Determinar a probabilidade de colisão de uma função de hash (utilizada, por exemplo, em HashMaps)

Exemplo Método Monte Carlo

- Aplicação: cálculo de valor de um integral



Probabilistic Programming

- **Probabilistic programming** is an emerging field that draws on probability theory, programming languages, and systems programming to provide concise, expressive languages for modeling and general-purpose inference engines that both humans and machines can use.

- Example of project: The MIT Probabilistic Computing Project aims to build software and hardware systems that augment human and machine intelligence.

- Picture, a probabilistic language being developed in collaboration with Microsoft, lets users solve hard computer vision problems such as inferring 3D models of faces, human bodies and novel generic objects from single images by writing short (<50 line) computer graphics programs that generate and render random scenes.

- Video of Vikash's MIT Media Lab talk on Probabilistic Programming for Augmented Intelligence
  - https://www.media.mit.edu/video/view/mansinghka-2016-03-15

Word predictor

- Filme Youtube:
  - https://youtu.be/5Mp_10tPcCU

Mais exemplos de aplicação …

- Filtrar emails com SPAM
- Máquinas de estados probabilísticas
- Parsers para análise sintáctica
- Information Retrieval
- Reconhecimento de padrões
- Reconhecimento de fala [Interacção Multimodal]
- Inteligência Artificial
  - Ex: planeamento nos robôs de Futebol robótico

MATLAB

- **Instalação MATLAB:**
  https://www.mathworks.com/academia/tah-portal/universidade-de-aveiro-40766421.html
  Use as suas credenciais de Utilizador Universal
- Ajuda:
  https://www.mathworks.com/support/contact_us.html?s_tid=tah_po_helpbutton_ua.pt
- Aprenda MATLAB em duas horas:
  Curso online MATLAB Onramp

  Mais informações sobre este e outro software disponível:

  http://www.ua.pt/stic/page/16014

Um bom semestre

A equipa docente de MPEI