

**TeSP - Tecnologias e Programação de Sistemas de Informação**

Técnico Superior Profissional

Plano: Despacho n.º 12805/2021 - 29/12/2021

**Ficha da Unidade Curricular: Algoritmos Computacionais**

ECTS: 2; Horas - Totais: 54.0, Contacto e Tipologia, TP:35.0;

Ano | Semestre: 1 | A

Tipo: Obrigatória; Interação: Presencial; Código: 602430

Área de educação e formação: Matemática

**Docente Responsável**

Maria Manuela Morgado Fernandes Oliveira

Assistente 2º Triénio

**Docente(s)**

Maria Manuela Morgado Fernandes Oliveira

Assistente 2º Triénio

**Objetivos de Aprendizagem**

Dotar os formandos de capacidade analítica que lhes permita construir, desenvolver e adaptar algoritmos de modo a serem capazes de conceber soluções lógicas para problemas surgidos nos mais diversos âmbitos, nomeadamente na construção de software, trabalhando individualmente ou em equipa.

**Objetivos de Aprendizagem (detalhado)**

Conhecer e dominar aspectos introdutórios das ciências da computação, nomeadamente armazenamento de dados, estruturas de dados e estruturas de controle, funções, mas também conhecer algoritmos nas áreas de Ordenação e Pesquisa, Grafos e Digrafos. Ter percepção das aplicações, já existentes, de algoritmos clássicos essenciais no desenvolvimento de software para as mais diversas áreas. Garantir o acesso à educação inclusiva, de qualidade e equitativa e, promover oportunidades de aprendizagem ao longo da vida para todos. (Objetivo de Desenvolvimento Sustentável n.º 4, conforme Agenda 2030 para o Desenvolvimento Sustentável, adotada pela Assembleia Geral das Nações Unidas em setembro de 2015)

**Conteúdos Programáticos**

Aspectos introdutórios. Funções. Algoritmos de ordenação e de pesquisa. Pesquisa linear e pesquisa binária. Algoritmos sobre Grafos e Digrafos. Extensões: noções básicas sobre heurísticas.

### **Conteúdos Programáticos (detalhado)**

Aspectos introdutórios: Breve introdução ao Python em ambiente Windows; dados, estruturas de dados e estruturas de controle; vectores e matrizes; funções recursivas e não recursivas. Algoritmos de ordenação e de pesquisa: Bubblesort, ordenação por seleção, ordenação por inserção, Shellsort, Mergesort e Quicksort. Pesquisa linear e pesquisa binária. Algoritmos sobre Grafos: definição e propriedades fundamentais dos grafos; estruturas de dados para representação, armazenagem e manipulação de grafos; construção de caminhos e ciclos em grafos; grafos conexos; árvores; extensão aos digrafos e a redes. Aplicações: algoritmo DFS para a construção de uma árvore geradora de um grafo conexo; algoritmo para a construção de um ciclo euleriano; o problema da determinação de uma árvore geradora de custo mínimo: algoritmos de Kruskal e de Prim; o problema da determinação de um caminho de custo mínimo numa rede: algoritmos de Dijkstra e de Floyd-Marshall; Problema do fluxo máximo numa rede: algoritmo de Ford-Fulkerson. Extensões: noções básicas sobre heurísticas; aplicação ao problema do caixeiro-viajante.

### **Metodologias de avaliação**

Avaliação Contínua: Realização de uma prova (escrita e computacional) , realização de trabalhos em sala de aula e um projecto final. O projecto é um trabalho computacional e vale 20 valores. A prova é cotada de 0 a 20 valores. Os trabalhos estão cotados de 0 a 20 valores. A nota final será dada pela média ponderada dos três itens anteriores, 30% para a prova escrita, 50% para o projeto final e 20% para os trabalhos em sala de aula.

Avaliação por Exame: Os alunos admitidos a exame ou os aprovados que pretendam melhorar a sua nota, podem fazer um exame em época normal. Esse exame divide-se em duas vertentes: escrita e computacional e abrange toda a matéria leccionada. Os alunos que reprovarem na época normal ou que pretendem melhorar nota podem ainda submeter-se a um exame de recurso, nos moldes descritos. As notas dos trabalhos poderão ou não transitar para exame, conforme escolha do aluno. Dando-se ainda a hipótese de melhorar, uma única vez, os trabalhos até à data do exame, mantendo a nota da prova.

### **Software utilizado em aula**

Anaconda, recorrendo ao Synder para desenvolver algoritmos em Python, em ambiente Windows.

### **Estágio**

Não aplicável.

### **Bibliografia recomendada**

- Aho, Hopcroft, Ullman, A. (1974). *The Design and Analysis of Computer Algorithms*. (pp. 1-470). Primeira, Addison-Wesley. Massachusetts
- Fernandes, M. (0). *Apontamentos da disciplina*. Acedido em 5 de março de 2025 em [www.e-learning.ipt.pt](http://www.e-learning.ipt.pt)
- Gaia, F. (2022). *Grafos em Python*. Acedido em 12 de setembro de 2025 em <https://www.kaggle.com/code/flaviagg/grafos-em-python>
- MARTINS, R. e NASCIMENTO, J. e PAIVA, F. e SOUZA, G. (2021). *INTRODUÇÃO A PYTHON COM APLICAÇÕES DE SISTEMAS OPERACIONAIS*. Acedido em 12 de setembro de 2025 em <https://memoria.ifrn.edu.br/bitstream/handle>
- Rosen, K. (2012). *Discrete Mathematics and its Applications*. (pp. 641-803). Sétima edição, McGraw-Hill. New York
- Wirth, N. (1976). *Algorithms + Data Structures = Programs*. (pp. 1-212). Primeira, Prentice-Hall. New Jersey

### **Coerência dos conteúdos programáticos com os objetivos**

A recolha, organização, armazenamento e pesquisa de dados são temas da maior importância, justificando pois a abordagem dos temas: estruturas de dados, algoritmos de ordenação e pesquisa. Na era das redes sociais, redes de telecomunicações, das grandes redes de distribuição e recolha de materiais e/ou produtos, das enormes rotas comerciais, de transportes ou eléctricas, é essencial perceber que é possível representar estas estruturas através de grafos e digrafos. Para além disso, é incontornável a necessidade de entender e saber implementar, algoritmos como o BFS, DFS, Kruskal, Prim, Dijkstra de Floyd-Marshall ou Ford-Fulkerson, uma vez que estes dão frequentemente resposta a problemas reais. Para terminar, é preciso ter a noção, que embora a evolução dos computadores e das ciências da computação ser abismal, há ainda problemas cuja resolução exacta é pouco provável em tempo útil tornando a utilização de heurísticas crucial para obtenção de uma solução, embora sem garantia que seja a melhor possível. Desta forma, é introduzido o conceito de heurística e é explorado o caso do Problema do Caixeiro Viajante.

### **Metodologias de ensino**

As aulas decorrem predominantemente em ambiente computacional recorrendo à linguagem de programação Python. Propõe-se o reconhecimento da aplicabilidade de alguns algoritmos clássicos, a sua compreensão teórica e procede-se à sua implementação.

### **Coerência das metodologias de ensino com os objetivos**

A metodologia de ensino reforça o conhecimento, compreensão e capacidade de implementação de algoritmos. Contribui ainda, para a flexibilização do raciocínio e desenvolvimento da capacidade de concepção de soluções lógicas na resolução de problemas surgidos no âmbito da informática, e mesmo noutros âmbitos, como por exemplo, a recolha de lixo, definição de rotas de distribuição, instalação de redes de esgotos, definição de redes de telecomunicações, etc. A discussão fomentada em sala de aula permite desenvolver um sentido crítico e espírito de equipa, tão necessário a profissionais, e mesmo académicos, nas áreas da computação. Esta prática e o conhecimento teórico adquirido ajudará na construção de software, de forma individual ou integrada em equipas.

### **Língua de ensino**

Português

**Pré-requisitos**

Não aplicável.

**Programas Opcionais recomendados**

Não aplicável.

**Observações**

ODS: 4 e 5.

---

**Docente responsável**

---