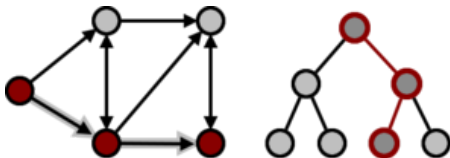


# Desenho e Análise de Algoritmos




Pedro Ribeiro

DCC/FCUP

2021/2022



# Informações Gerais

- Site: <http://www.dcc.fc.up.pt/~pribeiro/aulas/daa2122/>
- Recursos Principais:
  - ▶  **Mooshak:** Submissão de código para problemas e resposta a quizzes
  - ▶  **YouTube**  
**YouTube:** Vídeos (pré-gravados) das aulas teóricas (*do ano passado*)
  - ▶  **slack**  
**Slack:** Esclarecimento de dúvidas em formato *"instant messaging"*

# Obtenção de Frequência

- Não serão registadas presenças (teóricas e práticas)



- Semanalmente, serão feitos **questionários**:
  - ▶ São obrigatórios, mas não contam para nota
  - ▶ Cada um estará online durante uma semana (00:01 de Domingo a 23:59 de Sábado) (depois ficam disponíveis para treino, sem contar para frequência)
  - ▶ São constituídos por perguntas de escolha múltipla
  - ▶ Podem ver os resultados, saber que opção estava correcta e voltar a submeter quantas vezes quiserem
  - ▶ Para obter frequência é necessário ter respondido a 50% **previstos 10 questionários** → **têm de responder a 5**

# Fórmula de Cálculo da Avaliação

- **P:** nota prática, valendo **30%** da nota final, obtida através de 3 componentes:
  - ▶ 2 testes práticos de programação (2.5 valores cada)
  - ▶ resolução de exercícios ao longo do semestre (1 valor).

Nota mínima:  $P \geq 1.5$  (escala da nota: 0 a 6).

- **EN:** nota do exame de época normal, valendo 70% da nota final, obtida através de um exame escrito (presencial) com nota de 0 a 20
- **ER:** na época de recurso será feito um único exame (presencial), valendo 70% da nota final, **não sendo possível repetir ou melhorar a componente prática na época de recurso**

**Classificação da época normal:**  $C = EN * 0.7 + P \geq 9.5$

**Classificação da época de recurso:**  $C = ER * 0.7 + P \geq 9.5$

# Sobre a componente prática

- Poderão usar **C**, **C++** ou **Java**
- **Resolução de exercícios ao longo do semestre**
  - ▶ (pelo menos) 11 aulas com exercícios "pontuáveis"
  - ▶ Cada aula vale 10% (máximo=100%)
  - ▶ Os exercícios estarão disponíveis durante 3 semanas (depois ficam disponíveis para treino, sem contar para avaliação)
  - ▶ Devem reportar no código qualquer ajuda que tenham recebido
- **Testes Práticos**
  - ▶ Serão de 2 horas
  - ▶ Terão objectivos específicos divulgados
  - ▶ Terão acesso a código vosso submetido antes do teste (compensa por isso terem feito os exercícios antes, perceberem bem o que fizeram e terem código organizado)

# Datas dos Testes

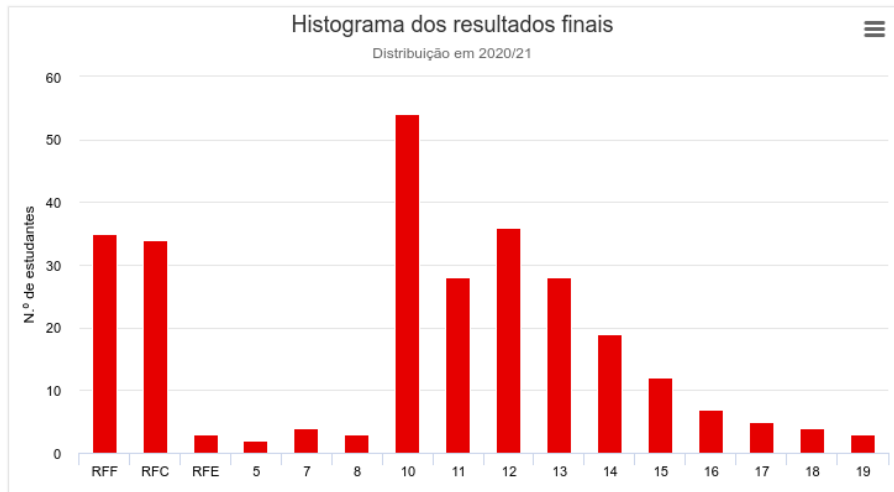
Estas datas são neste momento **provisórias** e apenas para vos dar uma ideia do planeamento:

- **1º teste prático de programação: Final de Abril**  
(complexidade, ordenação/pesq. binária, alg. greedy, prog. dinâmica)
- **2º teste prático de programação: Início de Junho**  
(grafos, dfs, bfs, distâncias, mst)

## Desenho e Análise de Algoritmos nos últimos 5 anos:

- Total de Alunos:  
**158** (16/17), **189** (17/18), **241** (18/19), **270** (19/20), **277** (20/21)
- Alunos aprovados:  
**61** (16/17), **26** (17/18), **58** (18/19), **119** (19/20), **196** (20/21)

Distribuição de notas no ano passado:





## Desenho e Análise de Algoritmos 2021/2022

- Total de Alunos inscritos: **90**
- Nº de inscrições: **64** (1<sup>a</sup>), **14** (2<sup>a</sup>), **7** (3<sup>a</sup>)  
**1** (4<sup>a</sup>), **2** (5<sup>a</sup>), **1** (6<sup>a</sup>), **1** (8<sup>a</sup>)
- Total por curso: **78** (L:CC), **9** (L:M), **2** (L:IACD), **1** (L:F)

# Pré-requisitos

- Conhecimentos de **C/C++** ou **Java**
- Conhecimentos de **algoritmos básicos** (contagem, pesquisa, ordenação, ...)
- Conhecimentos de **estruturas de dados básicas** (arrays, listas, pilhas, filas, ...)
- Preferencialmente ter concluído as unidades curriculares de **"Programação Imperativa"** e **"Estruturas de Dados"** (ou equivalente)

# Objetivos da Unidade Curricular

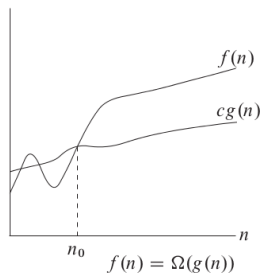
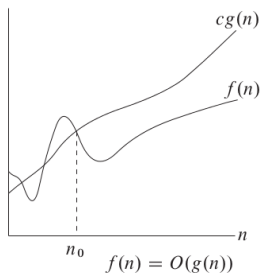
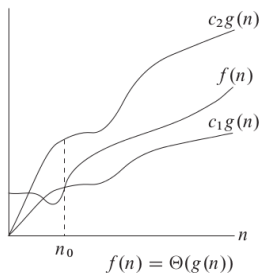
Competência na area de técnicas de **concepção e análise de algoritmos eficientes**:

- Competência na **análise da complexidade de algoritmos** e compreensão de algumas classes de complexidade
- Enriquecimento do conhecimento sobre **modelos genéricos de tipos de problemas e técnicas algorítmicas** a eles associadas.
- **Experiência prática** na aplicação de algoritmos genéricos a problemas concretos.

# Visão Geral do Programa

## Análise assintótica do tempo de execução de algoritmos:

- Notação *Big O* ( $O$ ,  $\Omega$  e  $\Theta$ )
- Análise de programas iterativos e recursivos
- Previsão de tempo de execução

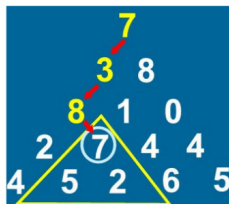
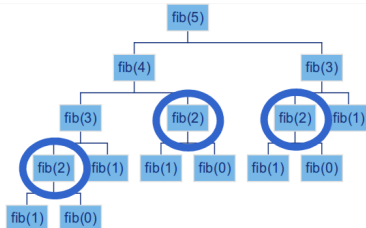


# Visão Geral do Programa

## Técnicas de Desenho de Algoritmos

- Pesquisa exaustiva (*Força Bruta*)
- Dividir para conquistar
- Algoritmos *greedy*
- Programação dinâmica

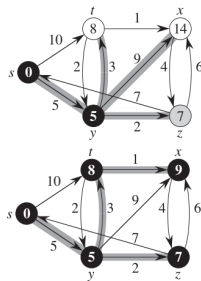
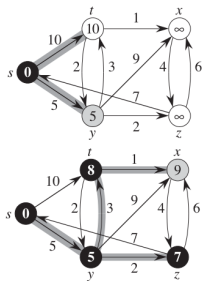
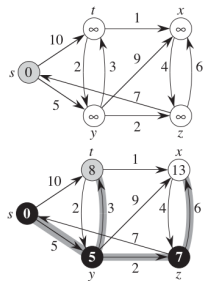
i \ j	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	1	2	3	3	4
2	2	2	2	2	3	4
3	3	3	3	3	3	4
4	4	3	4	4	4	3
5	5	4	4	5	5	4



# Visão Geral do Programa

## Algoritmos de grafos

- Representação de grafos
- Pesquisa em largura e pesquisa em profundidade
- Árvores de cobertura mínima
- Caminhos mínimos
- Redes de fluxo



# Visão Geral do Programa

## Algumas estruturas de dados especializadas

- Filas de prioridade
- Conjuntos disjuntos
- Árvores binárias de pesquisa equilibradas

