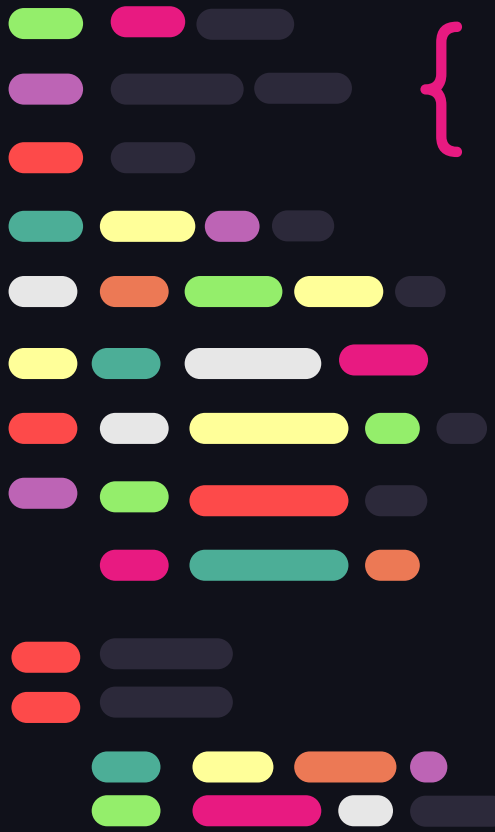
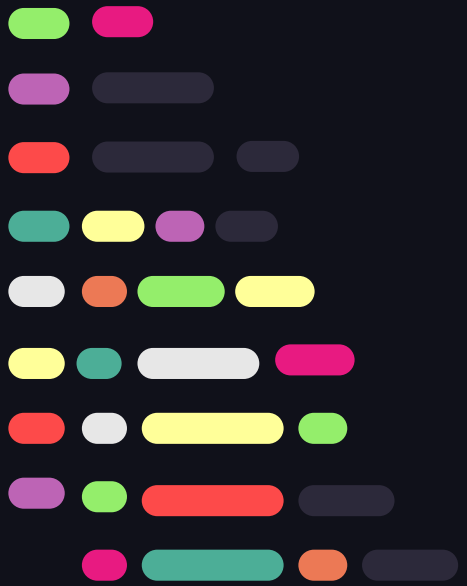


Algorithms and Data Structures 2023/24 L.EIC – 2nd Year Project 1 L.EIC Schedules

G135 - André de Sousa, Álvaro Pacheco e Rodrigo de Sousa

Table of contents



01 { ..

Objetivos

O que era suposto fazermos?

} ..

Statement of Work (SoW)

Neste projeto era nos pedido para desenvolver um sistema que permitisse ao utilizador **visualizar**, **editar** e **pesquisar** horários de **alunos**, bem como de **turmas** e **UCs.**

Desenvolver um **Sistema de Gestão de Horários**, utilizando estruturas de dados **lineares** e **hierárquicas**, consoante as necessidades.

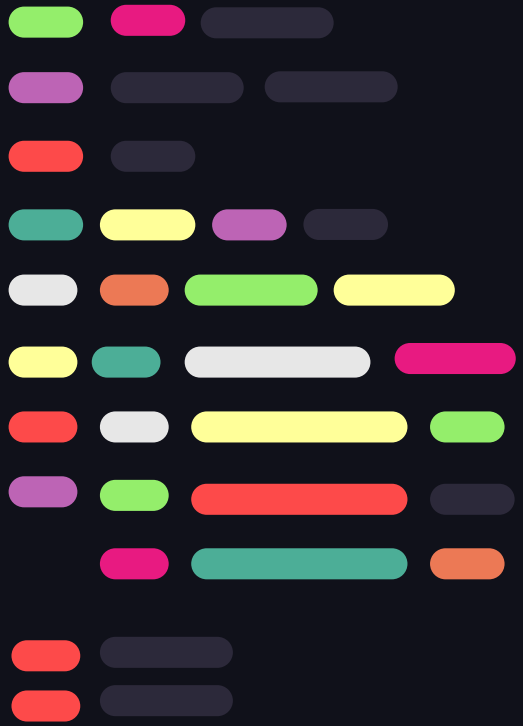
02 { ..

Implementação

O que fizemos?

} ..

Passagem de Dados



A **Passagem de Dados** dos ficheiros para o nosso programa tem de ser:

- Consistente
- Rápida
- Simples

Por isso, optamos por organizar essa informação em **vetores** de **objetos**, tanto **estudantes**, como **turmas** e **UCs**.

Classes

Student

Cada estudante tem como atributos o seu número de estudante, nome, lista de UCs inscritas e horário.

UC

Cada UC está associada ao seu código e a uma turma, bem como a capacidade máxima de estudantes.

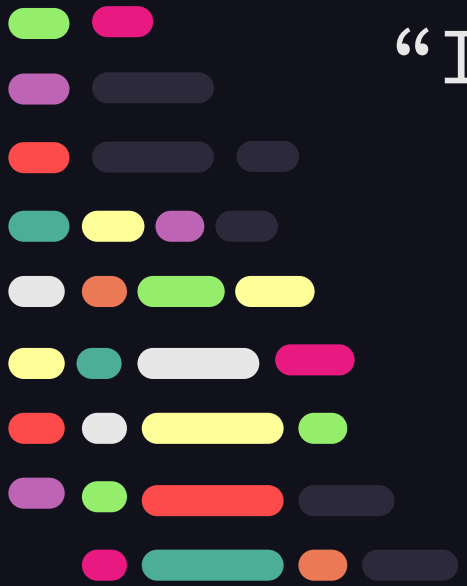
Classes

Classes

É um **bloco do horário**, tendo uma UC associada, o seu dia da semana, hora de começo, duração e tipo.

Schedule

Um horário é uma **coleção** de “Classes” ou seja blocos e pode ser criado para estudantes ou turmas.



“If the users don't control the program, the program controls the users.”

— Richard Stallman

Interface

Estudante

Disponibilizando o número de estudante, podemos:

- Ver UCs inscritas
- Ver Horário
- Inscrever numa UC
- Remover uma UC
- Trocar de Turma

Turma

Com o código da turma é possível:

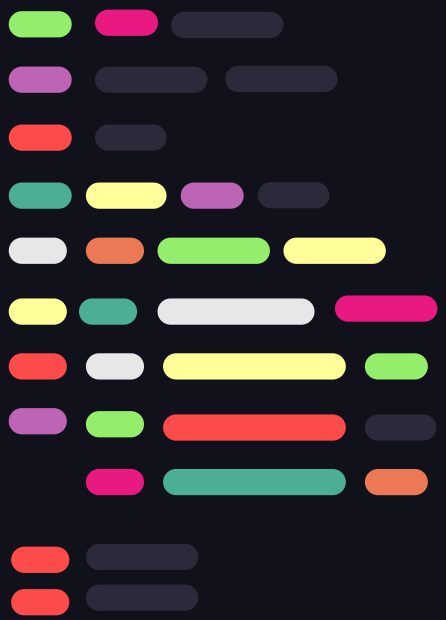
- Ver UCs
- Ver Horário
- Ver Estudantes

UC

Nas UCs conseguimos:

- Ver estudantes inscritos
- Ver o número de estudantes
- Ver turmas com esta UC

Interface



Lista Ordenada - Estudantes

Utilizando uma **queue**, apresenta os estudantes por número.

Lista Ordenada - UCs

Utilizando uma **queue**, apresenta as UCs por ocupação.

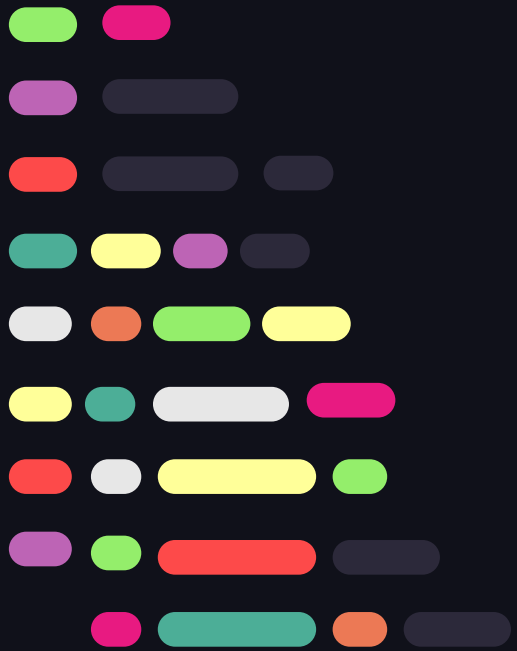
Ação mais recente

Mostra a ação mais recente e remove, utilizando uma **stack**.

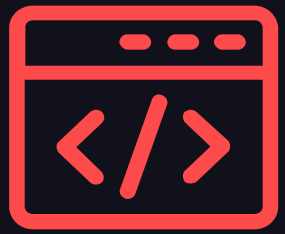
Créditos

Mostra os autores deste programa: Nós **:)**

Demonstração



Hora de ver em **funcionamento**!



Estrutura de Dados

VECTOR

Usado pela sua **simplicidade** como explicado anteriormente.

LIST

Usado para **guardar** as UCs dos estudantes.

STACK

Usado para guardar o **histórico** de ações.

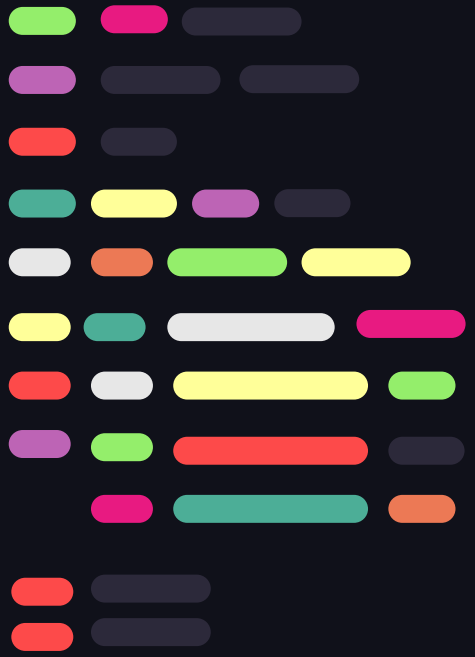
QUEUE

Usado como **fila de impressão** de elementos ordenados.

MAP

Usado para **traduzir** de inglês para português.

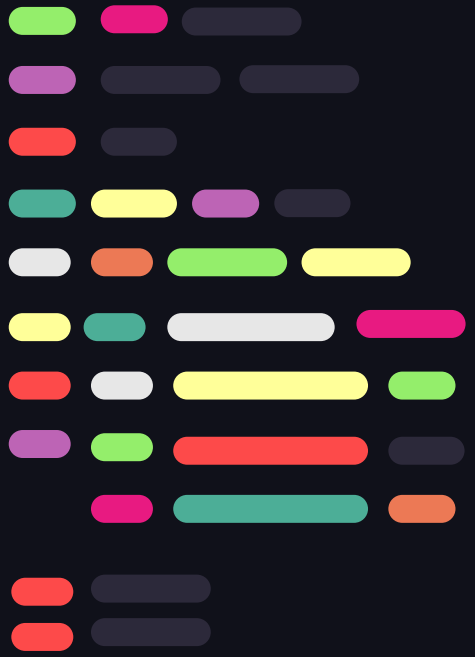
Time Complexity



Cada operação tem uma complexidade diferente, mas a maioria é **linear**.

- Passagem de dados:
  - Todas as ações de leitura e passagem de dados são O(n);
- Geração de Horários:
  - Criar um horário para um estudante requer a leitura da lista de UCs e procurar no vetor de Classes o correspondente. $O(n^2)$;
  - Para uma turma, é apenas um vetor. O(n);

Time Complexity



Cada operação tem uma complexidade diferente, mas a maioria é **linear**.

- Troca/Adição de Turma:
  - Usa a pesquisa da UC/Turma no vetor. O(n);
- Pesquisa de Estudante/Turma/UC:
  - Da mesma forma, é a pesquisa no vetor. O(n);
- Sorting and Displaying:
  - Com o uso de multimap para o sort, a complexidade é O(nlog(n)), mas o display é O(n).

03 { ..

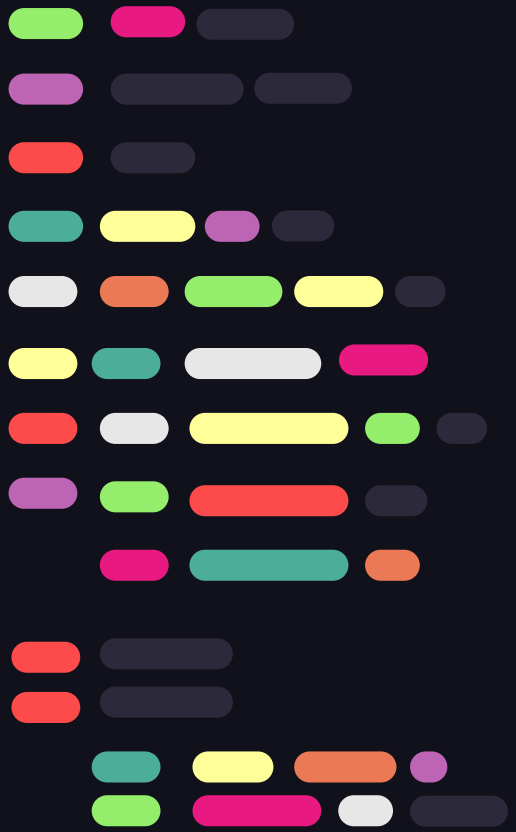
Notas Finais

O que aprendemos?

} ..

Conclusões

Para cada escolha de Estrutura de Dados houve um balanço de prós e contras de optimização de código até chegar às atuais. O uso de sets revelou-se desfavorável na nossa implementação devido ao aumento da complexidade para as outras ações que não fossem de pesquisa.



Thanks!

< Perguntas? >