ipluso INSTITUTO POLITÉCNICO DA LUSOFONIA

DWDM/AICD/EIA

1º semestre 2022/2023

Prof.: Carlos Limão

Trabalho Prático 5

Listas Ligadas

Introdução

As listas ligadas fornecem uma alternativa às sequências baseadas em *arrays* (tal como as listas em Python). Quer os *arrays*, quer as listas mantêm os elementos numa determinada ordem, embora usando uma abordagem bastante diferente. Enquanto um *array* usa uma representação unificada, baseada numa grande zona de memória capaz de armazenar muitos elementos (ou referências para elementos), a lista ligada utiliza uma representação mais distribuída dos dados, em que um objeto relativamente simples, designado por **nó**, é reservado independentemente para representar cada elemento. Cada nó armazena para além do elemento (ou referência para o elemento) uma ou mais referências para nós vizinhos de modo a representar a ordem linear da sequência.

Objetivos

Vimos que os *arrays* possuem três desvantagens assinaláveis:

- A dimensão de um *array* tem de ser, em geral, majorada, pelo que pode exceder o número de elementos que se pretendem armazenar.
- A necessidade de copiar todo o conteúdo do *array* sempre que a sua dimensão aumenta pode ser inaceitável nalguns casos.
- As inserções e remoções de elementos em posições interiores são demoradas.

Os objetivos deste trabalho consistem em demonstrar as vantagens e desvantagens das listas ligadas quando comparadas com os *arrays*. Como veremos, as listas ligadas permitem evitar as três desvantagens anteriormente apontadas aos *arrays*.

Preparação

Todos os exercícios deste trabalho serão colocados dentro de um único projeto.

Passo 1 - Criação de um projeto

Comece por criar um novo projeto no PyCharm chamado ListasLigadas.

Passo 2 - Adicionar ficheiros Python ao projeto

Neste projeto vamos começar por adicionar um ficheiro por cada um dos algoritmos a implementar:

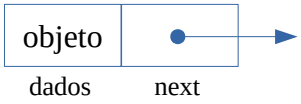
- Um ficheiro Python chamado ***sllist.py*** (*Singly Linked List*).
- Um ficheiro Python chamado ***dllist.py*** (*Doubly Linked List*).

Em cada um destes ficheiros deverá escrever **uma classe** que implementa cada uma das listas.

Juntamente com este enunciado será fornecido um ficheiro com o **esqueleto de uma classe** que servirá de base às duas classes que deverá escrever, bem como um **programa para as testar**.

Exercício 1 - Lista Simplesmente Ligada

Uma Lista Simplesmente Ligada (LSL) é um conjunto de nós que formam, coletivamente, uma sequência. Cada nó armazena (uma referência para) um objeto, que é um elemento da sequência, e também uma referência para o próximo nó da lista.



Associada a cada lista existem referências para o início (ou cabeça) da lista (*head*), e fim (ou cauda) da lista (*tail*). O último elemento usa como referência do próximo elemento (*next*) um valor especial que indica que não existe próximo elemento.

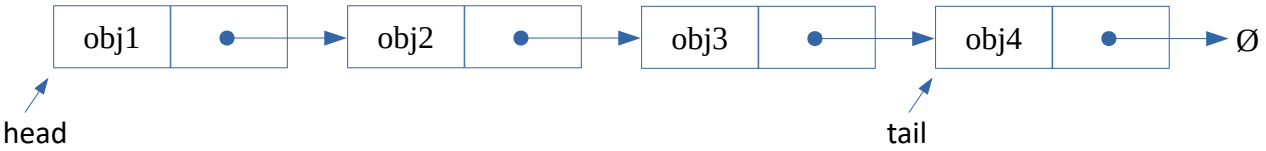


Tabela 1: Operações típicas sobre LSL

Operação	Algoritmo	Complexidade
Inserir no Início	1. Criar nó com elemento a inserir 2. Novo nó tem next = atual cabeça 3. Cabeça passa a ser novo nó 4. Atualizar dimensão da lista	O(1)
Inserir no Fim	1. Criar nó com elemento a inserir 2. Novo nó tem next = Ø 3. Elemento seguinte à cauda passa a ser novo nó 4. A cauda passa a ser o novo nó 5. Atualizar dimensão da lista	O(1)
Inserir Antes de um Nó	1. Clonar nó corrente 2. Modificar nó corrente com dados do novo. 3. Inserir clone depois do nó corrente	O(1)*
Inserir Depois de um Nó	Semelhante a inserir no fim	O(1)
Remover do Início	1. Se a lista está vazia, remover é um erro (exceção) 2. A nova cabeça é o nó a seguir à cabeça 3. Atualizar dimensão da lista	O(1)
Remover do Fim	1. Percorrer lista até ao nó antes do último 2. Remover como no início	O(n)
Remover Nó	1. Copiar próximo nó para o nó especificado. 2. Remover nó seguinte.	O(1)*/O(n)

Passo 3 - Escrita dos programas

No ficheiro ***sllist.py*** deverá escrever a classe que implementa uma lista simplesmente ligada.

Para isso deverá usar como base o ficheiro fornecido com este enunciado. Esse ficheiro inclui uma descrição de todas as funções (métodos) que deve escrever.

Passo 4 - Execução do programa

O ficheiro fornecido com este enunciado inclui também um programa de teste que deverá utilizar para testar a sua implementação da LSL.

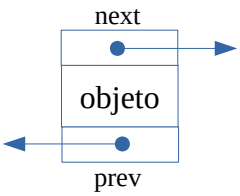
Tenha em atenção que o programa de teste fornecido pode não ser suficiente para testar todos os casos.

Pergunta 1: Na submissão do trabalho, deve incluir, num ficheiro com extensão .zip, um diretório **exercício1** contendo todos os módulos Python que escreveu para este exercício.

> ⚠ Como sempre, deve ter em atenção que todos os ficheiros escritos por si devem inclui um cabeçalho com as características indicadas no enunciado do 1º trabalho prático.
>
> Todas as funções deverão também incluir uma *docstring* que descreva o objetivo da função, os parâmetros de entrada e os valores de saída.

Exercício 2 - Lista Duplamente Ligada

Uma Lista Duplamente Ligada (LDL) acrescenta , em relação a uma LSL, uma nova referência para o nó anterior na sequência.



As operações suportadas são as mesmas das LSL. No entanto, apagar no fim e fazer operações com o nó anterior são agora possíveis em O(1).

Passo 5 - Escrita dos programas

No ficheiro ***sdlist.py*** deverá escrever a classe que implementa uma lista duplamente ligada.

Para isso deverá usar como base o ficheiro fornecido com este enunciado. Esse ficheiro inclui uma descrição de todas as funções (métodos) que deve escrever.

Tenha em atenção que o referido ficheiro tem de ter o nome da classe modificado.

> ⚠ Como sempre, deve ter em atenção que todos os ficheiros escritos por si devem inclui um cabeçalho com as características indicadas no enunciado do 1º trabalho prático.

Passo 6 - Execução do programa

O ficheiro fornecido com este enunciado inclui também um programa de teste que deverá utilizar para testar a sua implementação da LDL.

Tenha em atenção que o programa de teste fornecido pode não ser suficiente para testar todos os casos.

Pergunta 2: Na submissão do trabalho, deve incluir, num ficheiro com extensão .zip, um diretório **exercício2** contendo todos os módulos Python que escreveu para este exercício.

⚠ Deve informe o professor, caso detete alguns erros no ficheiro que lhe foi fornecido como ponto de partida para a realização dos exercícios deste trabalho.

Conclusão do Trabalho

A entrega do trabalho faz-se submetendo, no Moodle, um ficheiro **PDF**, devidamente identificado (aluno, curso, disciplina, trabalho, data), com as **respostas** às perguntas colocadas, eventualmente justificadas com cópias dos ecrãs.

Caso tal seja solicitado, deve submeter também os ficheiros que contêm os eventuais programas que tenham sido pedidos no enunciado. Neste caso os ficheiros devem ter nomes sugestivos, que indiquem claramente a que se referem (p.e.: algoritmoX.py ou exercícioY.py admitindo que são ficheiros com programas em Python).

Notas:

1. Não se limite a usar imagens do ecrã inteiro e esperar que o professor encontre nelas a resposta correta ou a justificação para a sua resposta!
2. Use uma imagem **indicando claramente** (sublinhado, círculo, etc.) a **resposta** ou a **justificação** para uma resposta que deu previamente, consoante o caso.
3. Por outro lado, recortes muito pequenos podem não permitir perceber a que contexto se referem. Os recortes devem incluir o contexto suficiente para que se perceba a que se referem e de onde foram retirados.

Dica: Pode usar a **Ferramenta de Recorte** (**Snip & Sketch** em Inglês) para cortar facilmente zonas do ecrã que queira usar nas suas respostas.