

Trabalho realizado por:

Sebastião Manuel Inácio Rosalino, n.º 98437, turma CDA1

Trabalho 5 – EDA

Questão 1.

Objetivo: Indicar um exemplo em que seja viável usar uma Lista Duplamente Ligada

Listas ligadas são constituídas por nós, onde cada nó contém uma referência (apontador) para o próximo e anterior nós na lista. Para além disso, cada nó contém um determinado elemento (*data*).

Uma lista duplamente ligada é assim denominada pela forma como as unidades (nós) que a compõem são ligadas. Ao contrário das listas simples, existem dois caminhos possíveis para percorrer a lista. Pode percorrer-se a lista duplamente ligada em sentido normal e em sentido inverso. Isto acontece porque os elementos (nós) da lista possuem dois ponteiros, um que aponta para o próximo nó e outro que aponta para o nó anterior.

As listas duplamente ligadas têm como principal vantagem a possibilidade de realizar inserções e exclusões em posições arbitrárias dentro da lista, realizando o processo da forma muito eficiente (a maioria das operações é de ordem $O(1)$).

Exemplo de aplicação de uma lista duplamente ligada:

- **Gestão do processo de transplantes de órgãos no Serviço Nacional de Saúde (SNS)**

Explicação do processo:

1) O que é um transplante?

Um transplante, ou transplantação, é a transferência de células, tecidos ou órgãos vivos de uma pessoa (o dador) para outra (o recetor) ou de uma parte do corpo para outra (por exemplo, os enxertos de pele) com a finalidade de repor uma função perdida.

2) Quem pode ser transplantado?

Podem ser transplantados os doentes que, segundo um relatório médico, sofram uma lesão irreversível num dos seus órgãos, sem que outro tipo de tratamento médico se adeque. Neste tipo de casos, o transplante é a única solução possível para evitar a morte ou para melhorar a qualidade de vida.

3) Qual é o período de espera para transplante?

O período de espera varia consoante a disponibilidade de órgãos capazes de serem transplantados. Por esse motivo, este processo pode ser demorado. Quando um órgão fica disponível, o doente é contactado para que num espaço de tempo muito reduzido a intervenção se concretize, uma vez que os órgãos geralmente não sobrevivem muito tempo fora do corpo humano. Se o doente não estiver contactável perde a vez para outro.

3) Como funcionam as listas de espera e quais são os critérios de distribuição dos órgãos?

De modo a garantir os princípios de igualdade e equidade, os critérios são definidos tendo em conta dois aspetos elementares: critérios regionais e critérios clínicos.

Os critérios regionais possibilitam que os órgãos de dadores de uma determinada região sejam transplantados na mesma região, para diminuir ao máximo o tempo de isquémia (tempo máximo que pode decorrer entre a colheita do órgão e o seu transplante no recetor).

Os critérios clínicos determinam a compatibilidade entre dador/recetor e a gravidade do doente.

A urgência/emergência do transplante, tendo em conta o estado de saúde do doente, caracteriza-se como um critério preferencial perante o critério regional.

A equipa de transplante decide, consultando a lista de espera, qual o doente mais indicado para receber o órgão, seguindo critérios clínicos, como a compatibilidade do grupo sanguíneo, características antropométricas e gravidade do doente.

Mais informação disponível em:

<https://www.sns24.gov.pt/tema/dadiva-e-transplante/transplante-de-orgaos/#sec-0>

A lista duplamente ligada para gestão do processo de transplantes servirá os seguintes objetivos:

- Gerir uma base de dados global (lista) de utentes em espera para transplantação de órgãos, com informação nacional e regional;
- Cada nó da lista, para além dos apontadores (um para o nó anterior e outro para o nó posterior), guarda um dado, que se traduz no número de SNS do utente em espera para o transplante, ao qual está associada informação complementar de natureza clínica (por exemplo: órgão a transplantar; nível de urgência; etc.);
- Esta lista duplamente ligada permitirá que, em função do grau de criticidade e urgência dos utentes a transplantar, novos utentes possam ser inseridos em qualquer posição da lista, aumentando o tamanho e diminuindo a lista estritamente em função dessa necessidade;
- Considerando que “a equipa de transplante decide, consultando a lista de espera, qual o doente mais indicado para receber o órgão, seguindo critérios clínicos, como a compatibilidade do grupo sanguíneo, características antropométricas e gravidade do doente”, qualquer utente em espera pode ser retirado da lista para receber um transplante, independentemente da sua posição. Do mesmo modo, qualquer utente em espera pode ver alterada a sua posição na lista em função da evolução da sua situação clínica (inclusive sair da lista por óbito). A movimentação de um utente na lista altera todas as posições relativas dos restantes utentes.

Para além destas funcionalidades, a lista disponibilizará as seguintes funcionalidades adicionais:

- Mostrar o conteúdo da lista;
- Localizar utentes na lista (n.º SNS);
- Mostrar o número de utentes em espera;
- Reordenar prioridades entre utentes;
- Indicar se a lista está vazia;
- Esvaziar a lista.

Questão 2.

Objetivo: Pretende-se uma **implementação eficiente** de uma **lista ligada estritamente ordenada por ordem crescente** que obedeça à seguinte interface.

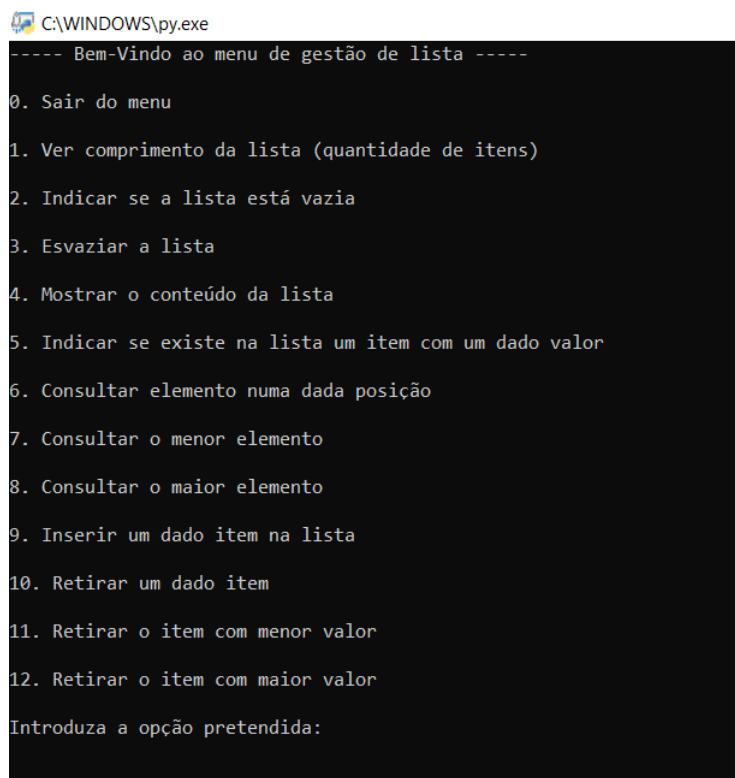
- *comprimento ou tamanho: quantos itens estão na lista*
- *indicar se a lista está vazia – vazia()*
- *esvaziar a lista – limpar()*
- *mostrar o conteúdo da lista*
- *indicar se existe na lista um item com um dado valor – existe(item)*
- *consultar elemento na posição p – ver(p)*
- *consultar o menor elemento – ver_min()*
- *consultar o maior elemento – ver_max()*
- *inserir um dado item na lista - ins(item)*
- *retirar item - rem(item)*
- *retirar o item com menor valor – rem_min()*
- *retirar o item com maior valor – rem_max()*

A implementação em linguagem Python está disponível no ficheiro:

Trabalho_5_EDA_Sebastião_Rosalino_98437.py

Todo o código está devidamente documentado.

Ao ser executado o programa oferece o seguinte menu:



```
C:\WINDOWS\py.exe
----- Bem-Vindo ao menu de gestão de lista -----
0. Sair do menu
1. Ver comprimento da lista (quantidade de itens)
2. Indicar se a lista está vazia
3. Esvaziar a lista
4. Mostrar o conteúdo da lista
5. Indicar se existe na lista um item com um dado valor
6. Consultar elemento numa dada posição
7. Consultar o menor elemento
8. Consultar o maior elemento
9. Inserir um dado item na lista
10. Retirar um dado item
11. Retirar o item com menor valor
12. Retirar o item com maior valor
Introduza a opção pretendida:
```

Foram, ainda, criados testes de execução simples a todas as capacidades da lista, usando:

```
if __name__ == "__main__"
```

Questão 3.

Objetivo: Pretende-se uma implementação de lista ligada ordenada por ordem crescente lata. Indicar as alterações na implementação relativamente à questão 2) de modo a tornar a inserção de um novo item o mais eficiente possível.

Procedeu-se à implementação em Python desta nova variante da lista (ordenada por ordem crescente lata), estando o código disponível no ficheiro:

Trabalho_5_EDA_Sebastião_Rosalino_98437.py (a partir da linha 299)

Alterações efetuadas ao código da questão 2:

- O método **ins** permite agora a inserção de elementos repetidos, já que a lista segue uma ordenação crescente lata. Foi usado o seguinte código para este efeito:

```
def ins(self, item): # Método para inserir um dado item na lista, neste caso, já pode ser um item repetido
    current = self._head # A variável current é atribuída ao primeiro nó da lista
    anterior = None # A variável anterior é inicialmente atribuída a None
    stop = False # A indicação de paragem começa a False
    self._size += 1 # Caso este elemento seja único, significa que vai ser inserido, incrementando o tamanho em 1
    while current is not None and not stop: # Enquanto existam elementos na lista e ainda não for altura de parar
        if current.value > item: # Caso o valor do nó em análise for superior ao item a colocar
            stop = True # Significa que esta lista tinha apenas 1 elemento, portanto a paragem passa a True
        else: # Se o valor do nó em análise for inferior ao item a colocar
            anterior = current # O anterior passa a ser o nó em análise naquele momento
            current = current.next # E o current avança na navegação da lista
    new_node = Node(item) # Quando se sai do ciclo, é então criado um nó com o item dado em input
    if anterior is None: # Se o anterior for None
        new_node.next = self._head # O nó a inserir passará a ser a cabeça da lista
        self._head = new_node
    else: # Se o anterior não for None
        new_node.next = current # O apontador next do novo nó passará a apontar para o current
        anterior.next = new_node # E, por fim, o apontador next do nó anterior passará a apontar para o novo nó
```

- O método **rem** remove agora todos os elementos iguais ao pedido no input. Foi usado o seguinte código para este efeito:

```
def rem(self, item): # Método para retirar um determinado item na sua totalidade, ou seja incluindo repetidos
    current = self._head # A variável current é atribuída ao primeiro nó (self._head)
    ant = current.prev # A variável ant é atribuída ao nó anterior
    encontrou = False # A variável encontrou é iniciada a False
    while current is not None: # Enquanto existam elementos
        if current.value == item:
            encontrou = True # Se for encontrado, encontrou passa a True
            if ant is None: # Se for encontrado, e o anterior for None é porque era o primeiro
                self._head = current.next # Então a cabeça passa a ser o próximo nó
            else: # Se o anterior não for None
                ant.next = current.next # O apontador do anterior passa a apontar para próximo nó
            self._size -= 1 # O tamanho da lista é reduzido em 1
            current = self._head # Sempre que é encontrado um item é preciso reiniciar a variável current e ant
            ant = current.prev
        else:
            ant = current # Caso contrário, o nó anterior passa a ser o nó que foi analisado
            current = current.next # E o novo nó é o próximo àquele que foi analisado
    if encontrou:
        print("Elemento retirado com sucesso.") # Mensagem informativa
    else:
        print("A lista não contém o elemento que procura.") # O utilizador é informado que o elemento não existe
```