Aula prática 10

Esta aula tem como objetivo rever algumas das estruturas de dados apresentadas anteriormente, nomeadamente: árvore AVL, filas de prioridade (heaps) e grafos.

1. Tendo por base as bibliotecas de estruturas de dados fornecidas na pasta **prob1**, implemente as funcionalidades pedidas no ficheiro **prob1.c**. Sempre que conveniente utilize as funções disponíveis na estrutura árvore AVL.

   Implemente a função `avl_maiorstring` para uma **árvore AVL** (definida pelo nó raiz) que devolve a maior *string* guardada nos nós da árvore.

   ```
   char* avl_maiorstring(no_avl *no)
   ```

   O parâmetro da função é o apontador para o nó raiz da árvore e o retorno é a *string* de maior comprimento. Por comprimento entende-se o número total de caracteres.

   Indique ainda num comentário no início do código da função qual a complexidade do algoritmo que implementou.

   Depois de implementada a função, o programa deverá apresentar:

   ```
   Maior string: republica centro-africana
   ```

2 Tendo por base as bibliotecas de estruturas de dados fornecidas na pasta **prob2**, implemente as funcionalidades pedidas no ficheiro **prob2.c**. Sempre que conveniente utilize as funções disponíveis nas estruturas heap e vetor.

   Implemente a função `heap_ordena` que cria um novo vetor ordenado seguindo uma ordem decrescente, usando uma **heap** auxiliar. A prioridade associada a cada *string* é calculada com base nos dois primeiros caracteres; considere o seguinte exemplo para determinar a prioridade de uma *string* `str`: `prioridade = (str[0] << 8) + str[1];`

   ```
   vetor* heap_ordena(vetor *v)
   ```

   O parâmetro da função é o vetor contendo as *strings* a ordenar. A função deve retornar um novo vetor com as *strings* ordenadas se for bem sucedida ou NULL em caso contrário, incluindo erro nos parâmetros.

   Depois de implementada a função, o programa deverá apresentar:

   ```
   yunkai

   wickendon

   winterfell

   ...
   ```

```
astapor

ashford

ar noy
```

3 Tendo por base as bibliotecas de estruturas de dados fornecidas na pasta **prob3**, implemente as funcionalidades pedidas no ficheiro **prob3.c**. Sempre que conveniente utilize as funções disponíveis nas estruturas grafo e árvore avl.

Uma rede social é mapeada num grafo não direcionado, onde um vértice representa o id de um utilizador. Implemente a função:

```
arvore_avl* sugestao_amizade(grafo *g, int u)
```

A função sugere ao utilizador `u` um conjunto de amizades. Este conjunto é constituído pelos amigos dos amigos do utilizador `u` que ainda não são amigos deste. A função retorna as sugestões numa árvore avl. Em caso de erro (grafo não existe ou utilizador `u` superior a tamanho do grafo), a função retorna NULL.

Notas: Na implementação do `grafo` foi incluída a função `grafo_adjacentes(grafo *g, int v, int* n)`. A implementação da árvore avl, `avl.h/.c`, foi alterada para guardar informação do tipo `int`.

Depois de implementada a função, o programa deverá apresentar:

```
Sugestões de amizade para '2': 6 7


Utilizador '12' ou rede nao existe
```