1. Estruturas de dados

a) Hashtable:

Decidimos utilizar uma hashtable para guardar o identificador de um estudante, código de um país e estado do aluno (Ativo(A), Terminou (E) e abandonou(L)). Decidimos escolher uma dimensão de 16 500 000 posições para obtermos um índice de ocupação máxima de 0.65 e para colocarmos os 10 000 000 estudantes. A hashtable ficará em memoria secundária e ocupará 188,82 MB, o que é suficiente para a memoria que o professor disponibilizou em memoria secundaria.

Utilizamos a hashtable porque sabendo que é preciso guardar um grande número de dados e sendo esta uma estrutura de dados especial porque nos permite a partir de uma chave simples (hashcode) fazer uma pesquisa mais rápida para obtermos o que procuramos.

b) Array Structs:

Para guardar as informações sobre a o número total de estudantes que frequentaram ou frequentam o ensino superior nesse país, o número de estudantes ativos, o número de estudantes que completaram o curso e o número de estudantes que abandonaram o ensino, utilizamos um array de structs que terá como elementos o código do pais, e as informações dos números dos estudantes com as referidas condições a cima escrito. Terá como memória ocupada 6,27 MB, uma dimensão de 676 posições (devido a seres 2 chars no código do país e 26 letras do alfabeto, então irá ser 26*26) e estará em memória secundária.

Utilizamos o array struct para guardar as informações sobre os estudantes de cada país.

c) Trie:

Utilizamos a trie para a memoria disponibilizada pelo professor a trie requer menos espaço quando contém um grande número de cadeias curtas, o que se verifica nesta situação, porque os nós iniciais das chaves são comuns. Isto torna a pesquisa mais rápida. Não podíamos colocar uma hashtable na memória central porque seria impensável fazer uma hashtable com a memória e com o grande número de dados disponibilizados, daí termos pensado na trie. A sua localização é em memoria central, a sua dimensão máxima são os 10 000 000 de estudantes.

As razões:

O ponto principal para a escolha destas estruturas de dados foi a rapidez com que se podia processar a pesquisa dos dados.

2. Conteúdo dos Ficheiros

O ficheiro vai servir para mantermos a informação do sistema de maneira persistente.

O mesmo contém uma Hashtable, que manterá todos os identificadores dos estudantes guardados, bem como os códigos dos países e os estados de atividade dos mesmos, e um Array, que manterá todos os códigos dos países e todos os dados referentes a esse país (números de estudantes).

O conteúdo do ficheiro consiste na Hashtable em si (com as 16 500 000 posições) e um Array (com as 676 posições). Pode ver se em seguida um item da Hashtable:

Posição na Hashtable	char id_estudante
	char id_país
	char estado

E um item do Array:

Posição no Array	char id_país
	char total_e
	char total_a
	char total_t
	char total_f

A Hastable tem 3 campos com valores de tipo char:

1. id_estudante -> Contém o identificador único que vai identificar o estudante;
2. id_país -> Contém o identificador do país do estudante;
3. estado -> Indica o estado de atividade do estudante (ativo, terminado, abandonado);

O Array tem 5 campos com valores do tipo char:

1. id_país -> Indica o país;
2. total_e -> Indica o número total de estudantes naquele país;
3. total_a -> Indica o número total de estudantes ativos naquele país;
4. total_t -> Indica o número total de estudantes que terminaram o curso naquele país;
5. total_f -> Indica o número total de estudantes que abandonaram o curso naquele país;

Como cada char ocupa 1 *byte*, cada item da Hashtable terá 12 *bytes,*7 bytes do identificador do estudante mais carater nulo,3 do código do país mais carater nulo, e 2 do estado do estudante mais carater nulo. O mesmo acontece com o item do Array, cada item terá 19 *bytes,*4 bytes por cada int, por isso são 4 ints*4 bytes que dá 16 mais 3 bytes do código do país mais carater nulo .

Sabendo que a Hashtable tem 16 500 000 posições, a Hashtable ocupará 198 000 000 *bytes*,o que corresponde a 188,83 MB. E o Array terá 676 posições, o que ocupará 12844 *bytes*, correspondendo a 6,27 MB.

3. Operações

- Introduzir um novo estudante

A inserção de um novo estudante passa por introduzir o identificador do estudante e o código do país, obtidos do standard input, na hashtable do ficheiro. Os passos para a execução desta operação são:

1º. São obtidos o identificador do estudante e o código do país do standard input.

2º. Coloca-se o identificador do estudante e o código do país numa posição da hashtable do ficheiro por meio do hashcode, caso estes não existam. É atualizado o estado do estudante para estudante no ativo.
3º. Coloca-se o código do país no Array e atualiza-se o número total de estudantes nesse país bem como o número total de estudantes no ativo desse mesmo país.
4º. Coloca-se na trie o identificador do estudante para posterior uso (no caso de ser preciso utilizar o mesmo identificador do estudante para outra operação não se ter de ir procurar em memória secundária).

Ao colocar o identificador do estudante e o código do país poderemos ter uma colisão na hashtable, isto é, ao tentarmos inserir naquela posição, a mesma já estar ocupada, o que leva a ser necessário fazer o rehash dos atributos.

Ao tentar introduzir o mesmo identificador do estudante e o mesmo código do país, em vez de se ir procurar na hashtable do ficheiro se esse identificador já existe, procura-se na trie, onde estarão todos os identificadores que foram utilizados na execução do programa, porque procurar em memória secundária torna-se mais dispendioso em termos temporais do que procurar em memória central. Tudo isso para garantir que os acessos à memória secundária sejam os menores possíveis.

A complexidade temporal é O(1), uma vez que os acessos ao disco, os passos da execução são executados em tempo constante, e nada dependem das dimensões das estruturas utilizadas.

- <u>Remover um identificador</u>

Remover um identificador de um estudante passa por remover o identificador, que foi recebido do standard input, da hashtable do ficheiro. Os passos para a execução são os seguintes:

1º. É obtido o identificador a remover do standard input.
2º. Procura-se o identificador na trie, para garantir a sua existência na hashtable do ficheiro, e se existir retira-se da trie.
3º. Procura-se na hashtable do ficheiro e remove-se o identificador, bem como o código do país e o estado do estudante.
4º. Procura-se o código do país no Array, que guarda os países, e retira-se 1 ao número de estudantes total daquele país, bem como 1 ao número de estudantes no ativo.

Ao tentar procurar o identificador na trie, se este não existir na estrutura, não vale a pena aceder ao disco, pois também não estará na hastable. Porém, se for um código ainda não utilizado na execução (não vai estar na trie), procura-se na hashtable.

A complexidade temporal é O(1), uma vez que os acessos ao disco, os passos da execução são executados em tempo constante, e nada dependem das dimensões das estruturas utilizadas.

- <u>Assinalar que um estudante terminou o curso</u>

Assinalar que um estudante terminou o curso passa por receber do standard input o identificador do estudante para se atualizar o estado do mesmo. Esta operação passa pelos seguintes passos:

1º. Obter do standard input o identificador do estudante.

2º. Procurar na trie a existência desse identificador.
3º. Procura-se na hastable, se existir atualiza-se o estado do estudante para estudante que terminou o curso.
4º. Procura-se o código do país do estudante no Array para se atualizar o número de estudantes que está no ativo para -1 e o número de estudantes que acabou o curso para +1.

Na primeira execução do programa, pode não existir nada na trie, pelo que se deve logo procurar na hastable, e assim atualizar o estado do estudante e obter o código do país para se atualizar os números no Array.

A complexidade temporal é O(1), uma vez que os acessos ao disco, os passos da execução são executados em tempo constante, e nada dependem das dimensões das estruturas utilizadas.

- <u>Assinalar que um estudante abandonou o curso</u>

Assinalar que um estudante abandonou o curso passa por receber do standard input o identificador do estudante para se atualizar o estado do mesmo. Esta operação passa pelos seguintes passos:

1º. Obter do standard input o identificador do estudante.
2º. Procurar na trie a existência desse identificador.
3º. Procura-se na hastable, se existir atualiza-se o estado do estudante para estudante que abandonou o curso.
4º. Procura-se o código do país do estudante no Array para se atualizar o número de estudantes que está no ativo para -1 e o número de estudantes que abandonou o curso para +1.

Na primeira execução do programa, pode não existir nada na trie, pelo que se deve logo procurar na hastable, e assim atualizar o estado do estudante e obter o código do país para se atualizar os números no Array.

A complexidade temporal é O(1), uma vez que os acessos ao disco, os passos da execução são executados em tempo constante, e nada dependem das dimensões das estruturas utilizadas.

- <u>Obter os dados de um país</u>

Mostra o número total de estudantes que frequentaram ou frequentam o ensino superior nesse país, o número de estudantes ativos, o número de estudantes que completaram o curso e o número de estudantes que abandonaram o ensino recebendo do standard input o código de um país. Os passos para esta operação são:

1º. Obter do standard input o código de um país.
2º. Procurar na Array struct o respetivo código desse país.
3º. Fazer print dos elementos que estão na array struct, ou seja, mostrar os dados todos desse país.

Se o programa não encontrar o código desse país irá aparecer uma mensagem de erro.

A complexidade temporal é O(1), uma vez que os acessos ao disco, os passos da execução são executados em tempo constante, e nada dependem das dimensões das estruturas utilizadas.

4. Inicio e fim da execução do programa

No início da execução do programa a trie que serve de memória central estará vazia, e será criado um ficheiro, se o mesmo não existir já.

Todas as operações no início do programa acederão ao disco, sendo que a partir daí estará tudo em memória central, a não ser que se insira algum estudante que não está na trie. Em qualquer dos casos, tudo será depois guardado em disco.

5. Bibliografia e Webgrafia

a. https://pt.wikipedia.org/wiki/Tabela_de_dispers%C3%A3o
b. https://pt.wikipedia.org/wiki/Trie