Relatório Fase 1 Trabalho EDAII

Luís Maurício - 37722
João Galvão - 37814

2019/2020

1 Estruturas Utilizadas

Para a implementação deste trabalho decidimos optar por utilizar hashtables como a única estrutura utilizada.
Para lidar com colisões parece-nos viável utilizar a forma quadrática mas poderemos facilmente mudar de opinião na segunda fase do trabalho.

A escolha foi motivada por ter de escolher alguma estrutura que permita a procura rápida de um aluno específico de forma a saber se o código identificador já está em uso ou não. Da mesma forma é necesário encontrar rapidamente os dados correspondentes a cada país.

1.1 Hashtable Estudantes

Usaremos uma hashtable em memória secundária para guardar a informação referente aos estudantes. Esta será uma hashtable de tamanho fixo visto que sabemos os número máximo de estudantes que o sistema poderá ter. Sabendo que o número máximo de estudantes é de 10 000 000 escolhemos 20 000 003 como o número de posições para esta hashtable visto que é o primo maior que o dobro de estudantes mais próximo.

Cada posição ocupará 9 Bytes, sendo que é composta de 9 caracteres. Os primeiros 6 caracteres correspondem ao identificador do aluno, os próximos 2 correspondem ao código do país e o último caracter representa o estado em que o aluno se encontra(Activo, Terminado ou Abondono).

Assim sempre que for necessário procurar um aluno basta aceder ao offset (multiplicando o código hash por 9) no ficheiro e ler as 9 posições seguintes.

Table 1: Hashtable para guardar informação dos estudantes

Posição	Identificador	País	Estado
00 000 000	ABC123	PT	A
00 000 001	ABC124	CA	D
...	...	...	...
20 000 003	65VD1U	FR	T

1.2 Hashtable Países

Para evitar estar a percorrer todos os estudantes e contar quantos pertencem a cada país, utilizaremos ainda uma hashtable que servirá para guardar os países e o número de estudantes em cada estado associados a este. Esta hashtable será guardada em memória central para evitar o número de acessos a disco na pesquisa dos dados de cada país, mas para manter a persistência entre utilizações do programa será guardada também em memória secundária.

Cada elemento desta tabela será composto por o código de país (char[2]), e 3 valores inteiros, fazendo com que cada elemento ocupe 14B (2 + 4 + 4 + 4). Um para guardar o número de estudantes que se encontram ativos, o próximo para guardar o número de estudantes que já terminaram o curso e por último os estudantes que abandonaram o ensino superior.

Sabendo que apenas existem 195 países no mundo decidimos que esta tabela também não precisa de rehashing, então escolhemos o número primo superior mais próximo do dobro de 195. Ou seja a tabela terá 397 posições.

Com isto podemos calcular que a tabela ocupará 5.4KiB ou 5.6KB (397 * 14B) em memória principal e consequentemente o mesmo em memória secundária.

Table 2: Hashtable para guardar informação dos países

Posição	Código País	Estudantes Ativos	Estudantes Terminados	Estudantes Desistentes
000	PT	223173	101806	56671
001	US	184802	120306	12182
...	...	...	...	...
396	CA	264941	215149	150364

2 Ficheiros de Dados

Table 3: Exemplo de ficheiro com informação dos estudantes

ABC123	PT	A	ABC124	CA	D	65VD1U	FR	T	
0			1			...			20 000 002

Cada posição da hashtable corresponde a 9 carateres seguidos, que representam os 3 campos seguintes:

ID: Contém o identificador do aluno que tem 6 caracteres.

País: Contém o identificador do país em que o aluno frequenta/ou o ensino superior.

Estado: Representa o estado do aluno em relação ao ensino, neste caso terá 3 possiveis valores, A – Activo, D – Abandono (Letra D pois Desistência é sinónimo), T - Terminado

Table 4: Exemplo de ficheiro com informação dos países

PT	223173	101806	56671	US	184802	120306	12182	..
0				1				...

..	*	*	*	*	CA	264941	215149	150364
...	395				396			

Cada posição da hashtable corresponde à seguinte divisão:

País: Contém o identificador do país que tem 2 caracteres.

Ativo: Contador de estudantes ativos o ensino superior.

Terminado: Contador de estudantes que terminaram o ensino superior.

Abandono: Contador de estudantes que abandonaram o ensino superior.

3 Operações

3.1 Inserir um novo Estudante

Ao inserir o código identificador de um estudante acontecem as seguintes operações:

1. Calcular o hashcode do código do estudante.

2. Calcular o offset da posição no ficheiro dos estudantes desse código.
Caso essa posição esteja ocupada lidar com a colisão como explicado na descrição das estruturas e procurar na próxima possível.

3. Aceder a essa posição do ficheiro e escrever os dados do aluno.
Caso o aluno exista na hashtable será emitido um output com a informação de que esse aluno já existe.

4. Calcular o hashcode do país.

2. Pesquisar esse hashcode na hashtable referente aos países. Caso não exista o país especificado, adicioná-lo à hashtable respetiva.

5. Incrementar o número de estudantes ativos nesse país.

O número de acessos ao disco é 2 no melhor caso, 1 para encontrar a posição correta e 1 para a escrita. Se tiver ocorrido alguma colisão o número de acessos para encontrar a posição irá aumentar até encontrar o identificador correto ou uma posição vazia.
Ambas as estruturas são hashtables e apenas acontece pesquisa e inserção, logo esta ação tem complexidade temporal de $O(n)$ no pior caso e $O(1)$ no melhor.

3.2 Remover um identificador

Para remover um identificador irão ocorrer as seguintes operações:

1. Calcular o hashcode do código do estudante.

2. Calcular o offset da posição no ficheiro dos estudantes desse código.

3. Aceder a essa posição do ficheiro e carregar os dados desse aluno para memória principal.
Caso o aluno não exista na hashtable será emitido um output com a informação de que esse aluno não existe (logo não pode ser removido).

4. Remover as informações encontradas na posição com o código correto, guardando o país e o estado do aluno.

5. Na hashtable dos países decrementar o contador referente ao estado em que o estudante se encontrava.

Serão feitos 2 acessos a disco no melhor caso, 1 para encontrar e 1 para apagar a informação No pior dos casos esta ação tem uma complexidade temporal de $O(n)$ e no melhor caso $O(1)$ referente à pesquisa e inserção (mesmo que vazia) numa hashtable.

3.3 Assinalar que um estudante terminou o curso

Para assinalar que um estudante terminou o curso os passos para a execução são:

1. Calcular o hashcode do código do estudante.
2. Calcular o offset da posição no ficheiro dos estudantes desse código.
3. Aceder a essa posição do ficheiro e carregar os dados desse aluno para memória principal.
   Caso o aluno não exista na hashtable será emitido um output com a informação de que esse aluno não existe, caso o aluno já esteja com o estado de terminado será emitido um output com a indicação de que essa informação já está atualizada.
4. Alterar o estado do aluno para T(Terminado).
5. Substiuir os dados do aluno pelo novos atualizados.
6. Encontrar o país do estudante e incrementar o contador de estudantes Terminados, decrementando o contador referente ao estado anterior do aluno.

A complexidade temporal desta operação pode ser no melhor caso O(1) ou O(n) no pior caso, de acordo com a complexidade de pesquisa e inserção em hashtables.

O número de acessos ao disco é 2 no melhor caso, 1 para encontrar a posição correta e 1 para a escrita. Se tiver ocorrido alguma colisão o número de acessos para encontrar a posição irá aumentar até encontrar o identificador correto ou uma posição vazia.

3.4 Assinalar o abandono de um estudante

Para assinalar que um estudante abandonou o curso os passos para a execução são:

1. Calcular o hashcode do código do estudante.
2. Calcular o offset da posição no ficheiro dos estudantes desse código.
3. Aceder a essa posição do ficheiro e carregar os dados desse aluno para memória principal.
   Caso o aluno não exista na hashtable será emitido um output com a informação de que esse aluno não existe, caso o aluno já esteja com o estado de abandono será emitido um output com a indicação de que essa informação já está atualizada.
4. Alterar o estado do aluno para D(Desistência).
5. Substiuir os dados do aluno pelo novos atualizados.
6. Encontrar o país do estudante e incrementar o contador de estudantes Terminados, decrementando o contador referente ao estado anterior do aluno.

A complexidade temporal desta operação pode ser no melhor caso O(1) ou O(n) no pior caso, de acordo com a complexidade de pesquisa e inserção em hashtables.

O número de acessos ao disco é 2 no melhor caso, 1 para encontrar a posição correta e 1 para a escrita. Se tiver ocorrido alguma colisão o número de acessos para encontrar a posição irá aumentar até encontrar o identificador correto ou uma posição vazia.

3.5 Obter os dados de um país

Para a obtenção dos dados do país os passos para a execução são:

1. Calcular o hashcode do código do país.
2. Pesquisar esse hashcode na hashtable referente aos países.
3. Retornar o código do país e os contadores referentes.

Para a obtenção dos dados, caso não exista o código referente irá ser dado um output indicando que o código pesquisado não se encontra na hashtable, ou seja não existem dados desse país.

A complexidade temporal desta operação pode ser no melhor caso O(1) caso seja encontrado à primeira, ou O(n) no pior caso em que haja colisões.

4 Início e fim da Execução

No inicio do da execução do programa serão abertos 2 ficheiros, o que contém a hashtable referente aos países, e o que contém a hashtable referente aos estudantes, caso não existam, estes serão criados.

Na criação dos ficheiros o ficheiros todas as possiveis posições serão pre-preenchidas com "*", para que o ficheiro já tenha as posições pre-definidas, ou seja o ficheiro referente aos países terá todas as 397 posições preenchidas com "*" e o mesmo irá acontecer nas 20 000 003 posições do ficheiro dos estudantes.

Se o ficheiro referente aos países existir será carregado na memoria principal, onde vai ser trabalhado durante toda a execução do programa, o ficheiro referente aos estudantes ficará aberto até término do programa, para facilitar acessos a estes.

No fim da execução do programa ambos os ficheiros serão fechados.

5 Bibliografia

`https://en.wikipedia.org/wiki/Hash_table`

`https://www.tutorialspoint.com/cprogramming/c_data_types.htm`

`https://www.moodle.uevora.pt/1920/mod/page/view.php?id=34562`

`https://github.com/rjcf18/Trabs_Exerc_Univ/blob/master/BSc/EDA2/TrabalhoEDA2/TrabEDA2Final/relatorioeda2.pdf`