Universidade do Minho
Escola de Engenharia

Relatório de Laboratórios de Informática III

Guião 2

2ºano - 1ºsemestre

Trabalho realizado por:
Joana Branco (A96584)
Joana Pereira (A97588)
Marta Sá (A97158)

Braga, 13 de dezembro de 2021

Introdução

Este guião visa consolidar alguns conhecimentos desenvolvidos no guião anterior mas, também, como a aplicação de novos. Neste guião desenvolve-se o conceito de estruturas de armazenamento de dados e, cada exercício está relacionado a uma query com um determinado input.

Query 1

A estratégia adotada para a primeira query passou pela comparação da string inserida pelo utilizador com os tipos de usuários possíveis (User, Bot e Organization) e, ir contando os mesmos. A Hash Table criada armazena todos os parâmetros do ficheiro dos usuários e, à medida que é lido este ficheiro, vai adicionando um outro usuário na Hash Table que vai ser identificativo por uma chave.

Esta função tem um bom desempenho mas quanto ao custo computacional, a Hash Table armazena bastantes dados e estruturas de dados o que faz com que seja necessário mais espaço.

Query 2

Esta query tem uma implementação semelhante à anterior mas são necessários dois valores de ficheiros diferentes, para calcular o total de colaboradores e de repositórios. Ambos são calculados de forma cíclica em que há a sua inicialização mas também a inserção de cada um dos parâmetros numa Hash Table. Esta foi desenvolvida como auxílio para armazenar todos os dados correspondentes aos repositórios e, numa outra, os dados dos commits.

No desempenho, esta função é mais lenta comparativamente à anterior pois são feitas diversas chamadas para o output pedido e é efetuado o acesso a duas funções distintas de ficheiros diferentes.

Query 3

O processo para chegar ao output pedido contou com a chamada da função principal que lê o ficheiro dos commits. Recorreu-se a uma função auxiliar que confirma se a linha do ficheiro dos usuários já foi acedida ou não, este valor está é dissimulado na Hash Table relativa aos usuários com os outros parâmetros .

Esta query tem um processo mais demorado pois executa funções necessárias que se encontram noutros ficheiros. A Hash Table também influencia no custo computacional pois é gasto mais armazenamento.

Query 4

O desenvolvimento desta query é bastante semelhante ao anterior em que se calcula o total dos commits pela função principal do ficheiro. A Hash Table contém todos os parâmetros relacionados aos commits incluindo um adicional que verifica se a linha já foi acedida ou não. Para além disto, o total de usuários é feito em conjunção com os três arrays que diferenciam e calculam o tipo de usuário existente.

Esta função tem um bom desempenho e o custo computacional não é tão elevado pois à medida que se vai calculando os tipos de usuários da HashTable, o valor do mesmo é guardado num array.

Query 5

A estratégia adotada para a quinta query passou pela construção de uma árvore para cada usuário com todos os commits que o mesmo fez, que são armazenados na Hash Table correspondente. Primeiramente converteu-se todas as datas num número inteiro, sendo mais fácil assim a comparação entre números inteiros e, adicionou-se esta mesma data ao seu respetivo usuário na árvore dos usuários. Assim, foi criada uma árvore ordenada com os nodos relativos às datas em que foram feitos os commits. Dado um intervalo de tempo no input, conseguiu-se localizar e contar o total de commits na árvore armazenando estes valores de forma cíclica numa nova estrutura de dados, listas ligadas, de maneira a obter os top N pedidos. Nesta query foram sentidas, inicialmente, algumas dificuldades mas estas mesmas foram ultrapassadas conforme se foi definindo o plano de implementação.

Comparativamente às anteriores, a query cinco tem mais custo computacional pois recorre à Hash Table, para relacionar os commits com o seu determinado usuário, e a uma árvore binária de procura em que os elementos dos nodos da esquerda são sempre inferiores aos da direita. Também recorre à inserção de listas ligadas, o que torna o código muito mais eficiente. Tem um bom desempenho apesar da solução apresentada ser bastante baseada na recursividade.

Antes de adicionar a condição de mostrar a solução num outro ficheiro, o output era corretamente o esperado, não tendo havido nenhuma alteração no código apresentado, tal como apresentado no anexo (figura 1). Tentou-se ultrapassar esta dificuldade mas, não tínhamos ideia de como modificar tal coisa.

Query 6

O processo para chegar ao output pedido contou com a comparação da linguagem da tabela correspondente a cada um dos repositórios com a inserida no input. Nesta comparação usou-se um caso especial que não tem em atenção se o input é em maiúsculas ou minúsculas. Foi usada a árvore anteriormente definida na quinta query, associando os usuários aos seus respetivos commits, sendo o objetivo aqui diferente. Pretendeu-se contar o número de commits efetuados, ou seja, os nodos da árvore binária de procura. As listas ligadas foram usadas de forma cíclica de modo a obter uma estrutura de dados que engloba todos os parâmetros pedidos para o output.

Esta última implementação torna o código muito mais eficiente e, sendo mais complexo, ocupa bastante armazenamento.

A condição de mostrar a solução num outro ficheiro implicou a mesma limitação no output, não tendo havido, novamente, nenhuma alteração no código. Esta dificuldade está apresentada no anexo (figura 2).

Query 7

Esta query começa por converter a data do input para um número inteiro, tal como visto na quinta query. É verificado na Hash Table dos repositórios o valor do parâmetro que se adicionou para analisar se a linha do ficheiro foi acedida ou não e se um dado repositório efetuou algum commit depois da data do input. Esta segunda parte é feita a partir da informação inserida na Hash Table em que se compara se a data do commit guardado na tabela é menor ou igual ao input.

O custo computacional não é muito elevado pois só foi necessário recorrer à Hash Table que armazena todos os dados relativos aos repositórios.

Query 8

A oitava query tem um processo inicial semelhante à sétima pois também converte a data do input para um número inteiro de modo a ser possível a ordenação das datas dos ficheiros. É verificado na Hash Table dos repositórios a data de criação do mesmo e se esta é maior ou não que a data do input. Nesta comparação usou-se um caso especial que não tem em atenção se o input é em maiúsculas ou minúsculas. Foi criada uma estrutura de dados com base numa lista ligada onde foi armazenado as linguagens presentes em cada um dos repositórios e as suas respetivas ocorrências e, posteriormente, estas serão ordenadas por ordem de utilização por uma outra estrutura de dados baseada no mesmo conceito.

Esta função tem um bom desempenho mas a nível de custo computacional já não, pois há uma grande ocupação de armazenamento pelo facto de não se saber ao certo quanto dele vai ser necessário.

A estratégia inicial era a criação de arrays para a ordenação das linguagens, que é um processo muito mais simples de implementação, mas, apesar da dificuldade, optou-se pela recorrência à inserção de listas ligadas, o que torna o código muito mais eficiente.

Query 9

O processo para chegar ao output pedido passou pelo acesso à Hash Table representativa dos usuários e, também, dos commits. Para além disto, foi criado um novo array, inicializado completamente a zero e de tamanho da tabela dos usuários. Esta nova implementação contém o número de commits e o usuário do mesmo. Estes commits estão associados a um repositório e é verificado se o usuário criador do repositório é amigo do usuário que fez commits no mesmo. Se esta situação se concretizar, o array é incrementado na posição onde o mesmo usuário que fez o commit se encontra na Hash Table relativa aos usuários.

Existe um enorme custo computacional a nível de armazenamento pois o array vai ter um grande comprimento.

Esta função é muito pouco eficiente pois tivemos algumas limitações em implementar algumas funções e em que estas resultem na solução esperada. Tendo em conta a segunda parte, tentou-se resolver os erros obtidos. Como o output era constituído apenas por um usuário, é provável que o problema se encontre na função que armazena os dados numa lista ligada (insereLL) mas não conseguimos chegar a nenhuma conclusão.

Esta estratégia não era de todo o nosso plano mas, de acordo com situações excecionais não nos foi possível executar de uma outra maneira.

Query 10

O desenvolvimento desta query começou pela criação de uma nova estrutura de armazenamento de dados onde foi guardado o tamanho das mensagens relativas aos commits de um determinado usuário, o seu id e login. A função que adiciona as mensagens à estrutura de dados foi chamada no parseCommit. Também contou com o auxílio da Hash Table relativa aos repositórios, sítio onde são impressas as mensagens de commit com tamanho maior. Adicionalmente foram adicionados na lista ligada novos parâmetros relativos à mensagem de commit, ou seja, o seu tamanho, id e login. Para se poder visualizar o output pedido, é percorrida a estrutura de dados associada às mensagens, o parâmetro que foi criado na tabela dos repositórios.

No desempenho, esta função é eficiente mas, apesar disto, existiram algumas limitações a corresponder ao resultado esperado pelo guião.

Para além disso, é importante realçar que, esta query não foi implementada corretamente depois de condicionarmos a que o output seja visualizado num outro ficheiro. Tal como se apresenta no anexo (figura 3), é possível ter uma ideia do resultado esperado apesar de estar mal.

Conclusão

Este guião permitiu desenvolver diferentes capacidades relacionadas com vários algoritmos e com diferentes estruturas de armazenamento de dados. Apesar de nem todas as querys terem resultado no output esperado conseguiu-se identificar em que situações o código implementado poderia ser melhor e mais eficiente. Tal como exemplo, a query nove, o código não é eficiente mas, devido a limitações extraordinárias (como o tempo para fazer este guião) não nos foi possível desenvolver de uma outra forma pois iria tomar bastante tempo de trabalho.

Para além disto, depois de adicionarmos ao código a restrição de que o output das query deveria ser escrita num outro ficheiro verificou-se que a solução dava diferente e errado comparativamente à situação inicial, onde o resultado das query era impresso apenas no terminal.

Contudo, o trabalho foi, em maior parte, satisfatoriamente executado.

ANEXO:

```
Query (0 para sair): 5

 N: 3

 Data inicial (AAAA-MM-DD): 2014-01-01

 Data Final (AAAA-MM-DD): 2016-01-01

id:8889629; login:ricbarraes; commits:30
id:3796452; login:fboyrive; commits:30
id:9815428; login:mrmikeeSC99; commits:30
```

```
1  id:14315249; login:DEFEC8ED; commits:1
2  id:16506209; login:dreamerBo; commits:1
3  id:16506578; login:rey-brujah-1980; commits:1
```

Figura 1: Query 5 (antes e depois)

```
Query (0 para sair): 6

 N: 3

 Linguagem: java

id:5583170; login:andremota; commits:30
id:5609717; login:xWildFirex; commits:30
id:6687207; login:vamsimocherla; commits:30
```

```
id:5583170; login:andremota; commits:60
id:5609717; login:xWildFirex; commits:60
id:6687207; login:vamsimocherla; commits:60
```

Figura 2: Query 6 (antes e depois)

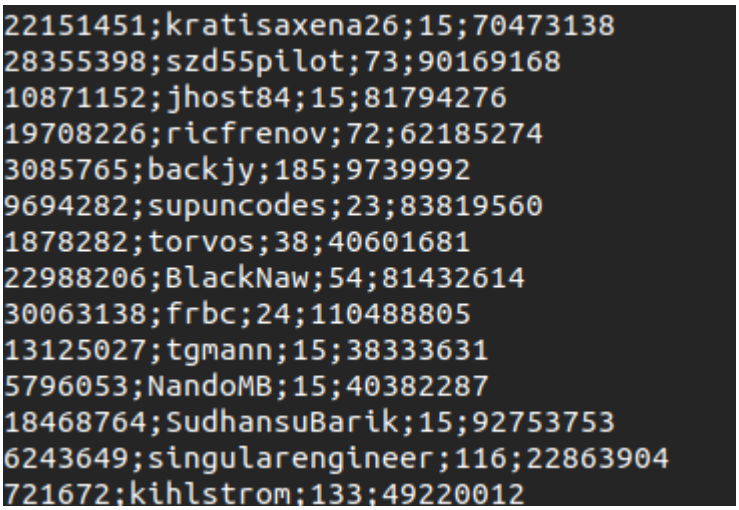

Figura 3: Query 10 (antes)