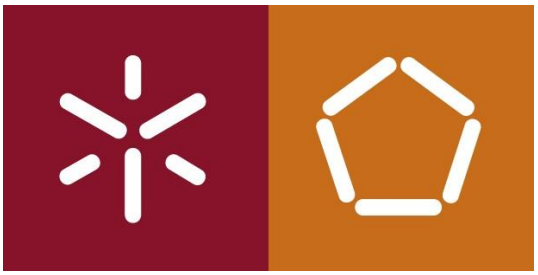
Universidade do Minho

Mestrado Integrado em Engenharia Informática



Stack Overflow - Relatório

Laboratórios de Informática III

Grupo 032

Trabalho realizado por:	*Número*
Ana Filipa Pereira	A81712
Helena Martins	A82500
Pedro Medeiros	A80580

5 de Maio de 2018

Conteúdo

1 Introdução

No âmbito da unidade curricular de Laboratórios de Informática III do 2º ano do Mestrado Integrado em Engenharia Informática, foi proposto pelos docentes o desenvolvimento de um projeto em linguagem C capaz de processar ficheiros XML que armazenam as várias informações utilizadas pelo Stack Overflow, sendo assim possível responder a um conjunto de interrogações na forma mais eficiente. Outro objetivo deste projeto será a consolidação de conhecimentos adquiridos em outras unidades curriculares sendo as mais relevantes Programação Imperativa e Algoritmos e Complexidade. O grande desafio deste projeto será a programação em larga escala, devido à elevada quantidade de dados e assim, com maior complexidade a nível algorítmico e de estruturas.

2 Descrição das Estruturas

2.1 Ficheiros

Os ficheiros usados para obtermos a informação estavam contidos em dois backups de diferentes comunidades, disponibilizados pelos docentes da UC. Cada backup era constituido por 8 ficheiros no formato XML, o que obrigou à realização de um Parser. No entanto, para conseguirmos responder às diversas interrogações apenas foi necessário utilizar os ficheiros Tags.xml,Users.xml e Posts.xml. Para a realização do trabalho também usamos a biblioteca do GLib de modo a facilitar o uso das várias estruturas.

2.2 Estrutura

As estruturas principais neste trabalho são o parser, a tabela de Hash dos utilizadores e as AVLs dos posts.

2.2.1 Parser

Como os ficheiros fornecidos se encontravam no formato XML, era necessário haver passagem para as estruturas em questão. Deste modo, foi usada a libraria Libxml2 para fazer parsing desses mesmos ficheiros.

2.2.2 Tabela de Hash

Inicialmente, foi pensada a estrutura que iria guardar informações dos Users que seria uma tabela de hash que armazena os utilizadores(os IDs). Esta estrutura armazena também a reputação de cada utilizador e dois apontadores para o nome e a biografia. Uma das vantagens da tabela de Hash é a eficiência e rapidez na procura, daí ter sido escolhida como estrutura.

2.2.3 AVLs

Neste trabalho também decidimos utilizar duas AVLs de forma a armazenar os posts. Uma que armazena os posts ordenando-os pelos seus identificadores (Posts_id) e outra que armazena os posts ordenando-os pela sua data de criação (Posts_data). A razão pela qual nos levou a utilizar estas estruturas foi o facto de o tempo de inserção na estrutura ser constante, ou seja, não ser necessário realocar nova memória para toda a árvore, uma vez que a memória é alocada sempre que se pretende inserir um novo elemento. E também pelo facto da procura nas AVL ser bastante rápida.

3 Interrogações

Antes de se começar a fazer as interrogações, é feita uma inicialização de algumas estruturas, isto é, o alocamento de memória para o TCD_community.

```
TAD_community init(){

  TAD_community com = malloc(sizeof(struct TCD_community));

  GTree* treeid = g_tree_new((GCompareFunc)&compareID);
  GTree* treedata = g_tree_new((GCompareFunc)&data_ord);
  GHashTable* u = g_hash_table_new_full(g_direct_hash, g_direct_equal, free,(GDestroyNotify) freeUsers);

  com->postsbyid = treeid;
  com->postsbydata = treedata;
  com->users = u;

  return com;
}
```

Depois da inicialização das estruturas, é necessário carregar a informação presente nos backups para as estruturas. Para tal, é necessário retirar o formato XML dos ficheiros e carregar os dados que lá se encontram para as estruturas adequadas.

```
TAD_community load(TAD_community com, char* dump_path){

  char *docname1 = "Posts.xml";
  char *docname2 = "Users.xml";

  parseDoc(strcat(dump_path, docname2), strcat(dump_path, docname1), com->users, com->postsbyid,com->postsbydata);

  return com;
}
```

Por fim, quando as interroações acabam, é necessária a libertação do espaço em memória ocupado pelas estruturas.

```
TAD_community clean(TAD_community com){

  if(com != NULL){
    freePostsD(com->postsbydata);
    freePostsID(com->postsbyid);
    freeUsers(com->users);
    free(com);
    com = NULL;
  }
  return com;
}
```

3.1 Interrogação 1

Esta interrogação procura o título do post e o nome de utilizador do autor, dado o identificador de um post. Caso o post for uma resposta, devem ser retornados o título e utilizador da pergunta correspondente.

Estratégia: Ir à AVL ordenada pelo ID do post e ver se é pergunta ou resposta. Caso seja pergunta, guarda o título do post e vai à tabela de hash dos users buscar o nome do utilizador. Se for resposta, vai buscar na AVL dos IDs o post/pergunta correspondente(ParentID), fazendo o mesmo para a pergunta.

3.2 Interrogação 2

Esta interrogação procura o top N de utilizadores com maior número de posts de sempre (tanto perguntas como respostas).

Estratégia: Percorremos a estrutura arrayposts (que foi posteriormente retirada devido a erros) e para cada utilizador contamos o número de posts adicionando a um array o ID e a respetiva contagem de posts. Para tal foi usada uma estrutura auxiliar chamada ArrayTop, que é um array de tamanho N que contém um ID e uma contagem, ordenado da maior para a menor.

3.3 Interrogação 3

A interrogação número 3 pretende obter o número total de posts, durante um determinado intervalo de tempo, identificando perguntas e respostas separadamente.

Estratégia: Fez-se lookup da data de início na AVL dos posts ordenados pela data e a partir disso, fez-se uma travessia utilizando uma estrutura auxiliar, UserDataPar, que guarda a data de início, de fim e um par de longs que servem como contadores das perguntas e respostas. Para cada post verificou-se se era pergunta ou resposta e incrementou-se os respetivos contadores. Quando a data final fosse encontrada, parava a travessia.

3.4 Interrogação 4

Esta interrogação procura todas as perguntas que contenham uma determinada tag, dado um intervalo de tempo arbitrário. Deve ser retornada uma lista com os IDs das perguntas ordenadas em cronologia inversa.

Estratégia: Foi feito lookup da data de início da AVL dos posts ordenados pela data e a partir daí fez-se uma travessia utilizando uma estrutura auxiliar, UserDataTag, que guarda as datas de início e fim, a tag e uma lista. Na travessia verificou-se se uma tag pertence à lista de tags do post e se isso acontecer adiciona à lista o ID da pergunta.

3.5 Interrogação 5

Nesta interrogação, dado o ID de um utilizador, pretende-se obter a informação do seu perfil e os IDs dos seus 10 últimos posts (perguntas ou respostas), ordenadas por cronologia inversa.

Estratégia: Foi feito lookup na tabela de hash do utilizador e foi-se buscar a sua biografia. De seguida foi-se ao array de posts buscar a sua lista de posts aos quais se retiram os 10 primeiros, ou seja, os 10 posts mais recentes.

3.6 Interrogação 6

A interrogação número 6 procura os IDs das N respostas com mais votos, em ordem decrescente do número de votos, num dado intervalo de tempo arbitrário.

Estratégia: Fez-se lookup da data de início na AVL dos posts ordenados pela data e a partir disso fez-se uma travessia utilizando uma estrutura auxiliar, UserDataTop, que guarda as datas de início e fim e um ArrayTop. Na travessia é adicionado ao ArrayTop o ID do post e o seu score caso este fosse uma resposta.

3.7 Interrogação 7

Nesta interrogação pretende-se devolver os IDs das N perguntas com mais respostas, em ordem decrescente do número de respostas, dado um intervalo de tempo arbitrário.

Estratégia: Foi usada a mesma técnica usada na interrogação 3 para saber o número de perguntas entre as datas e criou-se um ArrayTop com esse tamanho. Fez-se uma travessia entre essas datas para se inserir as perguntas no ArrayTop com a contagem 0. Foi feita outra travessia para atualizar a contagem do número de respostas de cada pergunta. No fim, ordenou-se o array e foi-se buscar os N primeiros IDs.

3.8 Interrogação 8

A interrogação número 8 pede o retorno de uma lista com os IDs de N perguntas cujos títulos contenham uma dada palavra, ordenados por cronologia inversa.

Estratégia: Utilizou-se uma estrutura auxiliar, UserDataTitle, que contém a palavra e uma lista que guarda os IDs dos títulos que contêm a palavra. Fez-se uma travessia da AVL, ordenada pela data, e caso a palavra pertença ao título do post inserimos o seu ID na lista. Depois de acabada a travessia, vai-se buscar os N primeiros IDs da lista.

3.9 Interrogação 9

Esta interrogação procura, dados os IDs de dois utilizadores, as últimas N perguntas(perguntas e respostas) em que participaram os dois utilizadores.

Estratégia: Utilizou-se uma estrutura auxiliar parecida com o ArrayTop, chamada ArrayData, que em vez de estar ordenada pela contagem, está ordenada pela data. Percorreu-se a lista de posts do utilizador 1 (no array de posts), e para cada post fez-se lookup na AVL ordenada pelos IDs. Caso fosse uma resposta, guardava-se o ID do pai/ da pergunta correspondente numa lista e vendo depois se esse ID está contido na lista de posts do utilizador 2. Se tal se verificasse, era adicionado ao ArrayData, fazendo-se primeiro lookup do post para buscar a sua data. Fez-se o mesmo para todos os posts do utilizador 2, para verificar a situação contrária, ou seja, que o utilizador 2 tenha respondido

à mesma pergunta que o utilizador 1. Depois, para verificarmos se os dois utilizadores responderam à mesma pergunta, verificou-se nas listas onde guardamos os IDs dos pais, se existem IDs em comum.

3.10 Interrogação 10

Nesta interrogação é pedido a melhor resposta dado o ID de um pergunta, usando-se a seguinte função de média : (Scr * 0.45) + (Rep * 0.25) + (Vot * 0.2) + (Comt * 0.1) onde Scr é o score da resposta, Rep é a reputação do utilizador, Vot é o número de votos recebidos pela resposta e Comt é o número de comentários recebidos pela resposta.

Estratégia: Fizemos lookup na AVL ordenada pelos IDs da pergunta. Aí buscamos o ID do utilizador que é procurado na tabela de hash para se conseguir buscar a sua reputação. Fez-se lookup da pergunta e buscamos a sua data da qual se fez lookup na AVL. Fizemos uma travessia utilizando um estrutura auxiliar, UserDataRes, que guarda a data, o ID e o número de respostas da pergunta, a reputação e uma célula com o ID do melhor e o seu quociente. Na travessia, quando é encontrada uma resposta da pergunta, calcula-se o seu valor e se for maior que o melhor até agora encontrado, substitui-se na estrutura auxiliar, decrementando o número de respostas cada vez que se encontra uma resposta. A travessia pára quando o número de respostas for 0.

3.11 Interrogação 11

A interrogação 11 pede os identificadores das N tags mais usadas pelos N utilizadores com melhor reputação, dado um intervalo arbitrário de tempo.

Estratégia: Esta interrogação não foi respondida devida a várias falhas nas estruturas.

4 Conclusão

A grande dificuldade na realização deste trabalho foi o facto do volume de dados analisados ser bastante elevado. Devido a vários erros na estrutura, não conseguimos resolver a interrogação 11 e obtivemos várias falhas nas interrogações 2, 5 e 9. Outro erro que cometemos ao realizar este trabalho foi o facto de, como tivemos alguns problemas no início com o load a partir dos ficheiros XML e não estavamos a conseguir resolvê-los, decidimos avançar para a resolução das interrogações. Isto levou a que não conseguissemos testar as interrogações atempadamente. Também tivemos a necessidade de aprender como funciona a libraria XML de modo a conseguirmos efetuar o parse dos documentos e guardas as informações úteis nas estruturas referidas acima. Outra grande dificuldade foi o facto de termos usado a biblioteca do GLib, uma vez que tornava o modo de resolução dos problemas um pouco restrito, daí termos optado pelo uso de várias estruturas auxiliares nas travessias. Todo o processo de realização deste trabalho implicou a consolidação de matéria de outras unidades curriculares nomeadamente Algoritmos e Complexidade e serviu para ganhar uma maior experiência na realização de trabalhos em equipa.