Laboratórios de Informática III 2016/2017
Trabalho C
Grupo 48

Cesário Miguel Pereira Perneta A73883
João Miguel Freitas Palmeira A73864
José Miguel Silva Dias A78494

30/04/2017

Conteúdo

1 Módulo BaseBTree

1.1 Definição da Estrutura

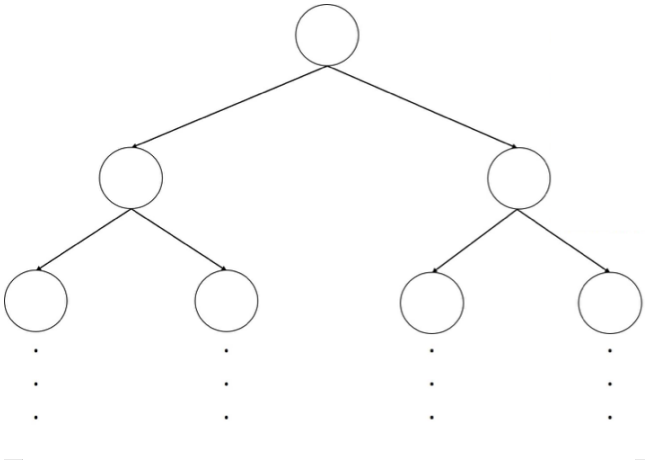

Figura 1: Estrutura dos artigos e das revisões

No ficheiro .c é definida a estrutura de dados dos artigos e das revisões (usando o construtor struct). A estrutura de ambos é igual, pois tanto os artigos como as revisões são definidos por uma struct ABP onde cada nodo é uma struct nodo. Cada nodo da ABP contém diferentes elementos que o permitem definir, tais como um apontador para um char "Titulo", outro para "Autor e outro para "TimeStamp", e vários longs para o id do artigo ou revisão ("id"), para os artigos duplicados ("dups"), para o id do autor ("idAutor"), para o número de caractéres ("nchars"), para o número de palavras ("nwords") e para o número de revisões ("nrevisoes"). Como partilham as mesmas estruturas, cada artigo tem um apontador "revisoes"que aponta para a árvore de revisões dessse mesmo artigo. No ficheiro .h apenas é escrito o nome da ABP e do nodo através de dois typedef, ou seja, o utilizador final apenas sabe o nome do tipo de dados que representa a ABP e o nodo, mas desconhece as suas estruturas internas.

```
struct nodo{
    char* Titulo;
    char* Autor;
    char* Timestamp;
    long dups;
    long id;
    long IDautor;
    long nchar;
    long nwords; // em revisões
    long nrevisoes;

    struct ABP *revisoes;
};

struct ABP{
    struct nodo *info;

    int height;
    struct ABP *ptEsq;
    struct ABP *ptDir;
};
```

Figura 2: Struct ABP e Struct nodo

1.2 API

O encapsulamento de dados é garantido ao longo da API tendo como base o facto de que, sempre que é requerida a geração de um resultado, o espaço que este resultado requer (e o seu preenchimento), seja ele uma lista, uma String ou um inteiro, é alocado dentro da própria função. No entanto, e visto que se trata de uma estrutura de dados de tamanho considerável, a inserção de um artigo ou de uma revisão na respetiva árvore é feita diretamente na árvore passada como argumento, caso contrário iria ter um efeito muito negativo na performance do programa (devido a sucessivas copias de uma estrutura de dados grande).

- Nodo ABP_nodeInit
  Função que inicializa o nodo.

- ABP ABP_init
  Função que inicializa a estrutura, atribuindo a todos os valores -1 ou NULL. A memória é alocada na própria função.

- int ABP_hasLeft

  Função que verifica se no nodo atual da árvore existe árvore há esquerda.

- int ABP_hasRight

  Função que verifica se no nodo atual da árvore existe árvore há direita.

- ABP ABP_left

  Função que devolve o nodo á esquerda do nodo atual.

- ABP ABP_right

  Função que devolve o nodo á direita do nodo atual.

- int ABP_hasNode

  Função que verifica se no local atual existe nodo além de existir árvore.

- Nodo ABP_getInfo

  Função que devolve o nodo com toda a informação do artigo, ou revisão caso a seja uma das árvores de revisão de um artigo, com o id passado como argumento

- Funções Get's

  Funções que permitem ir à estrutura e retornar qualquer tipo de informação relativa ao artigo ou à revisão em questão: ABP ABP_getRevisoes; char* ABP_getTitulo; char* ABP_getAutor; char* ABP_getTimestamp; long ABP_getID; long ABP_getDups; long ABP_getIDAutor; long ABP_getNRevisoes; long ABP_IDgetNChar; long ABP_getNChar; long ABP_IDgetNWords; long ABP_getNWords;

- ABP ABP_rotateLeft

  Função que balanceia a árvore.

- ABP ABP_rotateRight

  Função que balanceia a árvore.

- void ABP_destroi

  Função que faz free da árvore em questão.

- ABP ABP_insertRev

  Função que insere todos os elementos de cada revisão (fornecidos pelo parser) de um determinado artigo numa árvore de revisões.

- ABP ABP_insertRevAUX

  Função auxiliar da função insertRev.

- ABP ABP_insert

  Função que insere todos os elementos de cada artigo (fornecidos pelo parser) numa árvore de artigos.

2 Módulo Colaborator

2.1 Definição da Estrutura

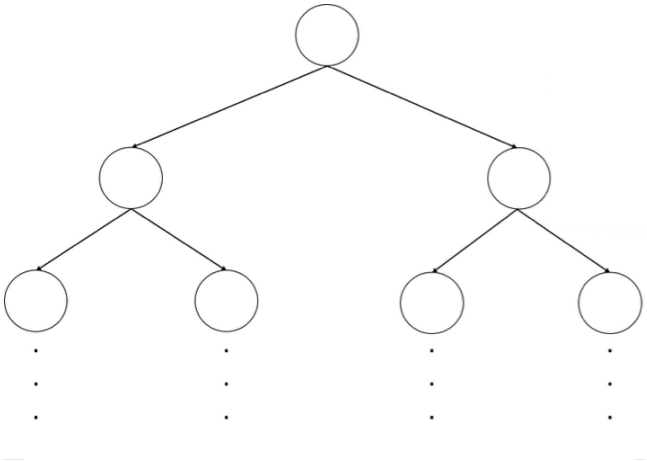

Figura 3: Estrutura dos colaboradores

Tal como na BaseBtree, no ficheiro .c é definida a estrutura de dados dos colaboradores (usando o construtor struct). Foi criado este módulo com o intuito de facilitar a realização de algumas interrogações. Cada nodo da ABP contém diferentes elementos que o permitem definir, tais como um apontador para um char "Autor", um long para o id do autor ("iDAutor") e um inteiro para o número de contribuições desse autor ("nContribuicoes"). No ficheiro .h apenas é escrito o nome da ABP_C através de um typedef, ou seja, o utilizador final apenas sabe o nome do tipo de dados que representa a ABP_C, mas desconhece a sua estrutura interna.

```
struct ABP_C{
    char* Autor;
    long IDautor;
    int nContribuicoes;


    int height;
    struct ABP_C *ptEsq;
    struct ABP_C *ptDir;
};
```

Figura 4: Estrutura dos colaboradores

2.2 API

Mais uma vez, o encapsulamento de dados é garantido ao longo da API tendo como base o facto de que, sempre que é requerida a geração de um resultado, o espaço que este resultado requer (e o seu preenchimento), seja ele uma lista, uma String ou um inteiro, é alocado dentro da própria função. No entanto, e visto que se trata de uma estrutura de dados de tamanho considerável, a inserção de um colaborador na respetiva árvore é feita diretamente na árvore passada como argumento, caso contrário iria ter um efeito muito negativo na performance do programa (devido a sucessivas copias de uma estrutura de dados grande).

- ABP_C ABP_C_init

  Função que inicializa a estrutura, atribuindo a todos os valores -1 ou NULL. A memória é alocada na própria função.

- int ABP_C_hasLeft

  Função que verifica se no nodo atual da árvore existe árvore há esquerda.

- int ABP_C_hasRight

  Função que verifica se no nodo atual da árvore existe árvore há direita.

- ABP_C ABP_C_left

  Função que devolve o nodo á esquerda do nodo atual.

- ABP_C ABP_C_right

  Função que devolve o nodo á direita do nodo atual.

- Funções Get's

  Funções que permitem ir à estrutura e retornar qualquer tipo de informação relativa ao colaborador em questão: char* ABP_C_IDgetAutor; int ABP_C_IDgetNCont; char* ABP_C_getAutor; int ABP_C_getNCont; long ABP_C_getIDAutor;

- ABP_C ABP_C_rotateLeft
  Função que balanceia a árvore.

- ABP_C ABP_C_rotateRight
  Função que balanceia a árvore.

- void ABP_C_destroi
  Função que faz free da árvore em questão.

- int ABP_C_getBalance
  Função que verifica se a árvore se encontra balaceada.

- ABP_C ABP_C_insert
  Função que insere todos os elementos de cada colaborador (fornecidos pelo parser) numa árvore do colaborador.

3 Módulo Parser

3.1 Definição da Estrutura

Este módulo tem como objetivo fazer o parsing dos ficheiros xml e, à medida que o parsing é feito, são invocadas as funções de inserir dos módulos anteriores de modo a inserir os artigos, as revisões e os colaboradores nas respetivas árvores. São ainda criadas duas funções, nwords (conta o número de palavras de cada texto) e nchar (conta o número de caractéres de cada texto), de modo a facilitar, mais tarde, a realização de algumas interrogações.

3.2 API

- char* parsePageTitle

  Função que percorre as tag's do ficheiro xml até encontrar uma tag "page"e posteriormente uma tag "title"do qual guarda o título.

- char* parsePageId

  Função que percorre as tag's do ficheiro xml até encontrar uma tag "page"e posteriormente uma tag "id"do qual guarda o id do artigo.

- char* parsePageRevisionId

  Função que percorre as tag's do ficheiro xml até encontrar uma tag "page", posteriormente uma tag "revision"e por último uma tag "id"do qual guarda o id da revisão.

- char* parsePageRevisionTime

  Função que percorre as tag's do ficheiro xml até encontrar uma tag "page", posteriormente uma tag "revision"e por último uma tag "timestamp"do qual guarda a data da revisão.

- char* parsePageRevisionColaboratorUsername

Função que percorre as tag's do ficheiro xml até encontrar uma tag "page", posteriormente uma tag "revision", posteriormente uma tag "contributor"e por último uma tag "username"do qual guarda o nome do contribuidor.

- char* parsePageRevisionColaboratorID

  Função que percorre as tag's do ficheiro xml até encontrar uma tag "page", posteriormente uma tag "revision", posteriormente uma tag "contributor"e por último uma tag "id"do qual guarda o id do contribuidor.

- long nwords

  Função que devolve o long com o número de palavras de cada revisão.

- long nchar

  Função que devolve o long com o número de caractéres de cada revisão.

- char* parsePageRevisionText

  Função que percorre as tag's do ficheiro xml até encontrar uma tag "page"e posteriormente uma tag "text"do qual guarda o texto da revisão.

- void parseDoc

  Função que recebe os vários snapshots bem como as estruturas onde vai inserir os dados e percorre as funções anteriores em ciclos acabando por inserir nas três árvores diferentes os diferentes dados.

4 Módulo Interface e módulo AuxInterface

4.1 Definição da Estrutura

O módulo Interface foi concebido pelo ficheiro .h fornecido pelos docentes onde estão contidas todas as interrogações do problema proposto. Já o módulo AuxInterface foi elaborado com o objetivo de criar funções auxiliares das funções da interface que recebessem estruturas mais amigáveis para a resolução dos problemas.

4.2 API

- long all_articles e long AUX_all_articles
  Função que retorna todos os artigos encontrados nos backups analisados.

- long unique_articles e long AUX_unique_articles
  Função que retorna todos os artigos únicos nos backups analisados.

- long all_revisions e long AUX_all_revisions
  Função que retorna quantas revisões foram efetuadas nos backups analisados.

- void top_10_contributors e void AUX_top_10_contributors
  Função que devolve um array com os identificadores dos 10 autores que contribuíram para um maior número de versões de artigos.

- char* contributor_name e char* AUX_contributor_name
  Função que devolve o nome do autor com um determinado identificador.

- void top_20_largest_articles e void AUX_top_20_largest_articles

Função que devolve um array com os identificadores dos 20 artigos que possuem textos com um maior tamanho em bytes.

- char* article_title e char* AUX_article_title

  Função que devolve o título do artigo com um determinado identificador.

- void top_N_articles_with_more_words e void AUX_top_N_articles_with_more_words

  Função que devolve um array com os identificadores dos N artigos que possuem textos com um maior número de palavras.

- char** titles_with_prefix e char** AUX_titles_with_prefix

  Função que devolve um array de títulos de artigos que começam com um prefixo passado como argumento da interrogação.

- char* article_timestamp e char* AUX_article_timestamp

  Função que devolve o timestamp para uma certa revisão de um artigo.

5 Grafo de Dependências

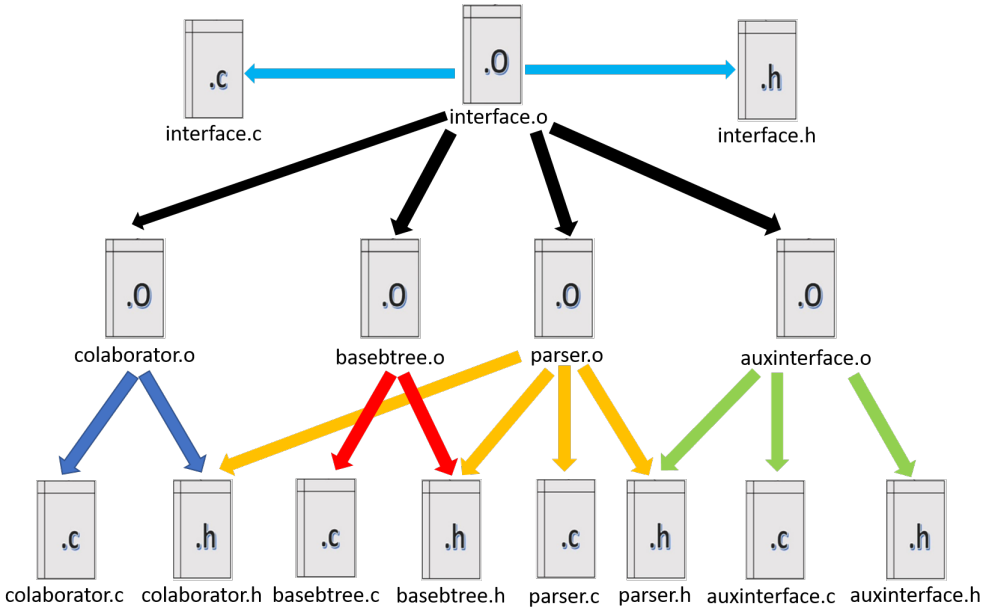

Figura 5: Makefile

6 Conclusão

O projeto foi realizado de modo responder a todos os problemas propostos no enunciado, bem como objetivos básicos do trabalho. Foi respeitado o encapsulamento de modo a proteger os dados, e utilizadas estruturas de dados abstratas no âmbito da reutilização de código.