Comentários Estruturação Projeto

1. Motivação da Reformulação da Estrutura do TAD Servidor

Para a parte I do projeto, foi criado um TAD SITE, do qual era possível inserir e extrair informações (código, nome, relevância, link, palavras-chave...) de determinado site. O conjunto de sites era gerenciado pelo TAD SERVIDOR, implementado como uma lista sequencial de ponteiros para SITE, alocada dinamicamente.

No início da execução do programa cliente, no escopo da função de criação do servidor, era alocada memória para os 10000 sites possíveis, sendo todos considerados inicialmente inativos. Cada site era inserido no servidor na posição equivalente à do seu código pois, dessa forma, a operação de busca por um site no servidor (até então a operação mais frequente) possuía custo operacional constante $O(1)$, visto que, dado um código, para obter um site bastava checar se havia um site ativo na mesma posição da lista de sites e assim retorná-lo.

Entretanto, a implementação anterior possuía uma alta complexidade de memória pois, mesmo que a quantidade de sites ativos no servidor fosse pequena (10, 100, 1000), o TAD SERVIDOR alocava sempre espaço para 10000 sites com suas informações, eventualmente desperdiçando memória.

Para resolver esse problema, uma nova estrutura foi utilizada no TAD Servidor no lugar da lista sequencial: uma Árvore AVL. Essa árvore foi escolhida pois, com a mesma, a memória alocada para o armazenamento dos sites é proporcional à quantidade de sites inseridos, evitando possíveis desperdícios, e a busca por um site nessa estrutura possui complexidade satisfatória da $O(\log(n))$.

Essa complexidade vale também para as operações de inserir site, remover site, inserir palavra-chave e atualizar relevância.

2. Novos TADs

Tendo em vista as novas operações (buscar sites e sugerir sites), foram criados os TADs PalavrasChave e BufferSite.

O primeiro é implementado como uma Trie, em que cada caminho é percorrido dado uma palavra-chave. Um ponteiro para um vetor dinâmico de inteiros, representando as chaves dos sites que estão relacionados à palavra-chave inserida, é alocado no fim de cada nó da Trie.

O segundo é implementado como uma Heap que é alocada quando as funções de buscar e de sugerir sites são chamadas. São alocados nessa Heap todos os sites cujos códigos estiverem presentes no array de códigos retornado pela busca na Trie.

3. Operação de Busca por Site

Para imprimir os sites relacionados à uma palavra-chave, é buscada a palavra-chave em questão na Trie de palavras-chave e, a partir da lista de códigos dos sites que possuem essa palavra-chave, obtemos os sites buscando-os no servidor (AVL principal). Após isso, os mesmos sites são ordenados por relevância e imprimidos na saída padrão.

Usando o algoritmo Heap Sort para ordenação, e denotando por S o número de sites que têm a palavra-chave buscada e por n o número de sites presente no servidor, temos, no total, a seguinte complexidade:

$$\begin{aligned} &\text{custoBuscaTrie} + S \,.\, \text{custoBuscaAvl} + \text{custoOrdenacao} \\ &= 1 + S \,.\, \log(n) + S \,.\, \log(S) \\ &= O(n \,.\, \log(n)) \end{aligned}$$

OBS: no pior caso, todos os S sites possuem a palavra-chave buscada $(S = n)$.

4. Operação de Sugestão de Sites

Para imprimir os sites sugeridos em relação à uma palavra-chave, é buscada a palavra-chave fornecida entre as palavras de todos os sites na Trie de palavras-chave. Para todos os códigos retornados pela busca na Trie, são buscados os sites no servidor (AVL principal) e são buscados também os sites que contêm alguma de suas palavras-chave. Todos os sites cujos códigos foram retornados são inseridos na Heap de sites e passados para um vetor ordenado por relevância pelo algoritmo Heap Sort. Os 5 sites de maior relevância são imprimidos na saída padrão.

Denotando por S a quantidade de sites que contêm a palavra-chave passada por parâmetro, P a quantidade média de palavras em cada site, n o número de sites inseridos no servidor, a complexidade da operação é dada por:

$$\begin{gathered} \text{custoBuscaTrie} + S \,.\, \text{custoBuscaAvl} + S \,.\, P \,.\, \text{custoBuscaTrie} + \\ S \,.\, P \,.\, \text{custBuscaAVL} + \text{custoOrdenacao} \\ = 1 + S \,.\, \log(n) + S \,.\, 10 \,.\, 1 + S \,.\, 10 \,.\, \log(S.10) \\ = O(n \,.\, \log(n)) \end{gathered}$$

OBS: no pior caso, todos os S sites possuem a palavra-chave buscada (ou seja, $S = n$) além de outras 9 palavras-chave.