UNIVERSIDADE FEDERAL DO PIAUÍ - CAMPUS SHNB

SISTEMAS DE INFORMAÇÃO

ALGORITMOS E PROGRAMAÇÃO II

Hériclys Sousa Borges

Tiago de Moura Oliveira

Trabalho Prático

1 Introdução

O objetivo desse trabalho prático é calcular o resultado do espalhamento de dados (ou hashing) em uma tabela de 101 compartimentos. As chaves dos compartimentos são strings com comprimento no máximo 15 letras (códigos ASCII 'A', ..., 'z').

As formas de pesquisa vistos até agora buscam informações armazenadas com base na comparação de suas chaves. Para conseguirmos algoritmos eficientes, armazenamos os elementos ordenados e tiramos proveito dessa ordenação. Algoritmos mais eficientes que entra no melhor caso de busca mostrados até o momento demandam esforço computacional O(log n). Digamos que hash seria o caso médio que é possível encontrar a chave em tempo constante. Índices em arranjos ou listas lineares são utilizados para acessar informações que são realizadas no tempo O(1).

A idéia central do Hash é utilizar uma função, aplicada sobre parte da informação (chave), para retornar o índice onde a informação deve ou deveria estar armazenada.Tabela Hash é a estrutura de dados especial cuja armazena as informações desejadas associando chaves como seus índices.

A representação é basicamente vetor + Lista Encadeada na qual o vetor contém ponteiros para as listas que representam as informações. A Função de Hashing é a responsável por gerar um índice a partir de uma determinada chave. Seria bom que a função forneça índices únicos para o conjunto das chaves de entrada possíveis na qual irão ser distribuídas na tabela.

O problema de buscar em uma Lista é que temos que percorrer ela por inteira e isso demora muito: o consumo de tempo é em relação ao número de elementos que ela possui, caso ela tenha muitos elementos o consumo de tempo é enorme. O espaço de busca pode ser muito grande, a Lista pode ter milhões de elementos. O espalhamento será conseguir restringir o espaço de busca, eliminando o maior número possível de elementos sem precisar "olhar" para eles. Nesse caso não precisa manter os elementos em sequência.

Iremos armazenar os elementos do Conjunto espalhados, mas não aleatoriamente e sim com uma certa lógica que facilitará buscá-los. Por isso será usado a Função de Espalhamento deve descobrir qual é a categoria de um determinado elemento. Para isso, ela deve analisar as características chaves do próprio elemento. Por exemplo, na agenda de telefone, a Função deveria ver qual é a primeira letra do nome do contato e devolver o número da página referente àquela letra. A Tabela de Espalhamento deve armazenar as categorias e cada categoria deve ter um índice. Este índice, gerado com a Função de Espalhamento, é usado para recuperar a categoria rapidamente. A Função deve ser determinística, ou seja, toda vez que aplicada a um mesmo

elemento ela deve gerar o mesmo índice, caso contrário procuraríamos um nome na página errada.

2 Solução Proposta

A solução proposta foi um código em C onde se é capaz de inserir, buscar e remover. Na implementação do código foi utilizado uma tabela hash com arquivo, uma lista encadeada, TAD e TED. N inserção, a chave é transformada em um número que vai servir de referência para a posição daquele dado inserido, para quando formos buscar esse dado. A mesma chave será usada para a remoção. Em relação ao tratamento de colisões, As colisões devem ser tratadas usando o método de endereçamento aberto, ou seja, tentar inserir a chave na tabela na primeira posição livre com a seguinte função de dispersão. A partir de 20 ou mais colisões, deve-se assumir que uma operação de inserção não pode ser executada.

A Tabela de Espalhamento pode ser implementada através de uma Lista de Strings. Neste caso, cada posição da Tabela poderá guardar somente uma palavra da língua. Haverá colisões causadas pela Função de Espalhamento. Devemos combinar a Tabela com outra estrutura de dados para contornar as colisões.

Para combinar a Tabela com Listas, poderíamos definir um atributo do tipo Lista de Listas de Strings. Principais causas de colisão : Principais causas: em geral o número N de chaves possíveis é muito maior que o número de entradas disponíveis na tabela. não se pode garantir que as funções de hashing possuam um bom potencial de distribuição (espalhamento). Endereçamento Aberto (Rehash) que vamos utilizar para tratar as colisões.

Precisamos definir quantas posições a Tabela terá, ou seja, quantas Listas secundárias devemos ter, e inicializar cada uma delas. Inicialmente, teremos 26 posições na Tabela, uma para cada letra do alfabeto. Aqui vamos separar as palavras de acordo com a primeira letra delas: essa será nossa Função de Espalhamento. A nossa Função deve observar a primeira letra das palavras e calcular o índice correto na Tabela. Função de Espalhamento deverá receber uma String (palavra) e devolver um int (índice associado a palavra). Para adicionar uma palavra no Conjunto, devemos aplicar a Função de Espalhamento para descobrir em qual posição da Tabela devemos adicionar. Depois, recuperamos a Lista que está nesta posição para guardar a palavra. Podemos evitar a repetição de palavras que causa colisão fazendo uma pequena verificação antes de adicionar uma palavra. Na remoção deve achar a posição da Tabela onde está a Lista na qual a palavra a ser removida estaria. Depois, basta remover a palavra da Lista.

O primeiro problema acontece quando a Função de Espalhamento não "espalha" direito os elementos. É a repetição da mesma palavra com a primeira letra de início.

O segundo problema ocorre quando adicionamos muitas palavras. A Tabela tem tamanho fixo, ou seja, o número de Listas é constante. Ao adicionar 100000palavras, calculando a melhor média possível, cada Lista interna da Tabela terá 100000/26 = 3846.2 elementos. Conforme as Listas vão ficando mais sobrecarregadas pior será o desempenho do sistema, chegando próximo do consumo de tempo linear.

Será feito da seguinte forma: analisar uma dada palavra e gerar um código genérico que chamaremos de **Código de Espalhamento**, e após incumbido de a partir do Código de

Espalhamento calcular um índice válido para a Tabela. Código de Espalhamento faz um calculo sobre **todos** os caracteres que formam a String. Agora a palavra "bola" não terá mais o mesmo resultado da função de hash que a palavra "batida".

Endereçamento aberto ou linear: A partir da posição de colisão, procurar uma posição considerada vaga. Encadeamento: Manter uma lista encadeada de registros de overflow para cada posição no espaço de endereços como foi mencionado acima.

Algoritmo de inserção: - vá ao endereço de k=h(k) - se estiver livre coloque a chave ali - se estiver ocupado, tente a próxima posição até que uma posição desocupada seja encontrada (a próx posição para a última é a posição 0).

3 Análise de Desempenho

O seguinte programa apresenta as funcionalidades de criar a tabela Hash, calcular o Hash, inserir na tabela quando há colisão e quando não há colisão.

No pior caso a complexidade das operações pode ser O(N). Caso em que todos os elementos inseridos colidirem. Hash com endereçamento aberto podem necessitar de redimensionamento. Alto custo para recuperar os elementos da tabela ordenados pela chave. ν Nesse caso, é preciso ordenar a tabela .O pior caso é O(N), sendo N o tamanho da tabela νalor Alto número de colisões.

A função de espalhamento uniforme para um dado conjunto de chaves seria uma que não produzisse nenhum endereço igual no espaço de endereçamento. A pior função seria aquela que, qualquer que fosse a chave, geraria sempre o mesmo endereço . Uma função aceitável é uma que gera poucos sinônimos quer dizer poucos endereços iguais.

Alguns métodos de espalhamento potencialmente melhores que os aleatórios seriam examinar as chaves buscando um certo padrão. Se uma parte das chaves mostrar um padrão, pode-se usar um função que extrai esta parte. Separar partes da chave. Pode-se extrair dígitos de uma parte da chave e somar estes dígitos. Este método destrói eventuais padrões nas chaves originais, mas em algumas circunstâncias pode fazer a separação entre pequeno conjunto das chaves que se espalham naturalmente.

4 Conclusão

O problema seria inserir, remover e usar o método de espalhamento em hash na qual conseguimos resolver com tais procedimentos que foram utilizados :Tabelas de dispersão que são utilizadas para buscar um elemento em ordem constante O(1) . Função de dispersão (função de hash): mapeia uma chave de busca em um índice da tabela. Estratégias para tratamento de colisão: uso da primeira posição consecutiva livre – uso de uma segunda função de dispersão ,uso de listas encadeadas. Todos esses métodos irão calcular o resultado do espalhamento de dados (ou hashing) em uma tabela de 101 compartimentos. As chaves dos compartimentos são strings com comprimento no máximo 15 letras (códigos ASCII ’A’, ..., ’z’). Ignorando espaços em brancos e não permitir colisões. As limitações encontradas seriam tratar as colisões de forma eficiente que não atrapalha-se o desempenho do resto do programa e conseguir obter os resultados necessários. Além das vantagens que é a Alta eficiência na operação de busca νalor. Caso médio é O(1) enquanto o da busca linear é O(N) e a da busca binária é O(log2 N).Tempo de busca é praticamente independente do número de chaves armazenadas na tabela e implementação simples.