Sets and Dictionaries

Dictionary

Na computação, as operações que precisamos realizar para um conjunto ou multiconjunto maisgeralmente estão procurando por um determinado item, adicionando um novo item e excluindo um itemda coleção. Uma estrutura de dados que implementa essas três operações échamou o dicionário.

Sets (Conjuntos)

Um conjunto pode ser descrito como uma coleção não ordenada de itens distintos chamados elementos do conjunto. Um conjunto específico é definido por uma lista explícita de seus elementos ou especificando uma propriedade que todos os elementos do conjunto e somente eles devem satisfazer.As principais operações são: verificar a pertinência de um determinado item em um determinado conjunto,união de dois conjuntos, a intersecção de dois conjuntos.

Implementação - Dictionary

Eles variam de um uso não sofisticado de arrays (classificados ou não) a técnicas muito mais sofisticadas, como hashing e balanced search trees.Uma série de aplicações em computação requerem uma partição dinâmica de alguns n-elemento definido em uma coleção de subconjuntos disjuntos. Depois de inicializado como uma coleção de n subconjuntos de um elemento, a coleção está sujeita a uma sequência de operações de união e busca misturadas. Este problema é chamado de set union problem.

Implementação-Sets

O primeiro considera apenas conjuntos que são subconjuntos de algum grande conjunto U, chamado de universaldefinir. Se o conjunto U tiver n elementos, então qualquer subconjunto S de U pode ser representado por um bitcadeia de tamanho n, chamada de vetor de bits.A segunda e mais comum maneira de representar um conjunto para fins de computação é usar a estrutura da lista para indicar os elementos do conjunto. Esse requisito de exclusividade às vezes é contornado pela introdução de um multiset, ou bag, uma coleção não ordenada de itens que não são necessariamente distintos. Em segundo lugar, um conjunto é uma coleção não ordenada de itens; portanto,alterar a ordem de seus elementos não altera o conjunto. Uma lista, definida como um coleção ordenada de itens, é exatamente o oposto. Pode valer a pena manter a lista em uma ordem ordenada.

Space and Time Trade-Offs

Input enhancement

a ideia é pré-processar a entrada do problema,no todo ou em parte, e armazenar as informações adicionais obtidas para acelerarresolver o problema depois.

-counting methods for sorting
-Boyer-Moore

Dinamic Programing

Esta estratégia baseia-se no registo de soluções para subproblemas sobrepostos de um determinado problema numa tabela a partir da qual é então obtida uma solução para o problema em questão

Prestructuring

O outro tipo de técnica que explora as compensações espaço-tempo simplesmente usa espaço extra para facilitar o acesso mais rápido e/ou mais flexível aos dados. Este nome destaca duas facetas desta variação do trade-off espaço-tempo: algum processamento é feito antes de um problema em questão,é realmente resolvido, mas, ao contrário da variedade de aprimoramento de entrada, lida com acesso estruturando.

-hashing
- B-trees

Dois comentários finais sobre a interação entre tempo e espaço no projeto de algoritmos precisam ser feitos. Primeiro, os dois recursos – tempo e espaço – não têm que competir uns com os outros em todas as situações de projeto. Na verdade, eles podem se alinhar para trazer uma solução algorítmica que minimiza o tempo de execução e o espaço consumido. Em segundo lugar, não se pode discutir trade-offs espaço-tempo sem mencionar oextremamente importante área de compressão de dados.

Hashing

Input enhancement

se escolhermos o tamanho m de uma tabela hash para ser menor que o númerode chaves n, teremos colisões - um fenômeno de duas (ou mais) chaves sendo hash na mesma célula da tabela de hash.Cada esquema de hashdeve ter um mecanismo de resolução de colisão. Este mecanismo é diferente noduas versões principais de hash: hash aberto (colocando as chaves numa linked list)e hashing fechado (não há uso de linked list)

Condições

O tamanho de uma tabela de hash não deve ser excessivamente grande em comparação com o número de chaves, mas deve ser suficiente para não comprometer o tempo de implementação eficiência (veja abaixo).Uma função hash precisa distribuir chaves entre as células da tabela hash como uniformemente possível. (Este requisito torna desejável, para a maioria das aplicações, ter uma função de hash dependente de todos os bits de uma chave, não apenas de alguns deles.)Uma função hash deve ser fácil de calcular.

Hashing é baseado na ideia de distribuir chaves entre um array H[0..m − 1] chamado de tabela hash. A distribuição é feita por computação, porcada uma das chaves, o valor de alguma função predefinida chamada função hash. Esta função atribui um inteiro entre 0 e m − 1, chamado de endereço de hash, para uma chave. Por exemplo, se as chaves forem números inteiros não negativos, uma função de hash pode ser da forma h(K) = K mod m;