UNIVERSIDADE
FEDERAL DO CARIRI

UNIVERSIDADE FEDERAL DO CARIRI
CENTRO DE CIÊNCIAS E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Projeto de Implementação – Parte II

Professor: Carlos Vinícius Gomes Costa Lima

Wanderson Faustino Patricio - MAT: 2022005052
Francisco Anderson Maciel Cruz - MAT: 2022005876

Juazeiro do Norte, 22 de outubro de 2023

1 Apresentação

Com o intuito de identificar quando o portal das naves irá se expandir foi criado um sistema de inserção para os compartimentos.

Foi identificado que o portal se expande sempre que todas as permutações de uma sequência de recursos atravessam o portal. Com isso em mente, foi criado um sistema de tabela de dispersão para analisar em tempo aproxiadamente constante o momento em que todas as permutações de um conjunto de recursos atravessaram.

2 Apresentação da implementação das estruturas

2.1 Struct Resource

Esta estrutura representa um recurso, contendo o nome do recurso (armazenado como uma string de até 50 caracteres) e um identificador (ID) inteiro. Membros: nome: Uma string que armazena o nome do recurso. id: Um inteiro que representa o identificador do recurso.

2.2 Struct Compartment

Esta estrutura representa um compartimento, que é composto por uma matriz de recursos (recursos) com um tamanho máximo definido por MAX-RESOURCES. Cada compartimento contém vários recursos.

2.3 Struct Table

Esta estrutura representa uma tabela que armazena compartimentos. A tabela tem uma estrutura bidimensional, com um tamanho máximo de MAX_SIZE para as linhas e MAX_PERMUTATIONS para as colunas. Além disso, a estrutura inclui uma matriz para os compartimentos (tabela), uma matriz para rastrear a quantidade de compartimentos em cada linha (qtds), um contador para o número total de inserções (contador) e um contador de aberturas (aberturas).

3 Apresentação da implementação das funções

3.1 Resource createResource(int *id*)

Essa função cria um recurso a partir de seu id. Foram predefinidos os ids existentes, pois há apenas 6 recursos disponíveis.

1 - Minérios

2 - combustivel

3 - remedios

4 - comida

5 - armamento

6 - agua

A função retorna um compartimento com o id correto e o nome do recurso.

3.2 void infoResource(Resource *r*)

Essa função imprime na tela as informações referentes ao recurso, informando seu nome e seu id.

3.3 Compartment createCompartment(int $ids[]$)

A partir de uma lista de ids são criados os recursos correspondentes a cada id da lista. Os recursos criados são inseridos em um novo compartimento. O compartimento criado com os recursos é então retornado.

3.4 void infoCompartment(Compartment c)

Essa função imprime na tela as informações de cada um dos recursos no compartimento na ordem em que estão armazenados nele.

3.5 Table$*$ createTable()

Essa função aloca na memória o espaço necessário para uma tabela de compartimentos e, caso alocada com sucesso, retorna um ponteiro para o endereço de memória correspondente.

3.6 void deleteTable(Table* t)

Essa função libera o espaço de memória previamente reservado para comportar a tabela t.

3.7 void infoTable(Table* t)

Essa função retorna as informações da tabela, dizendo quais linhas estão sendo ocupadas e quantos compartimentos essas linhas estão comportando atualmente.

3.8 void insertCompartment(Table* t, Compartment c)

Essa função é a principal do sistema.

Para inserir um novo compartimento na tabela inicialmente vemos qual é a linha correspondente em que este deverá ser inserido através de sua função de hash.

Após a verificação da posição é analisado se a combinação de recursos a ser inserido é uma permutação que ainda não está na linha, através da função isNewPermutation. Caso seja uma nova permutação o compartimento é inserido no final da linha, e o contador da linha aumenta.

Finalmente, se o contador da linha é igual ao total de permutações, é informado que o portal se abrirá, a linha é então zerada e permitido inserir novamente todas as permutações, e informado ao usuário quantas inserções ocorreram desde a última abertura.

3.9 int isEquals(Compartment a, Compartment b)

Esta função verifica se dois compartimentos são iguais, comparando os identificadores dos recursos em cada compartimento. Se todos os identificadores forem iguais, a função retorna 1; caso contrário, retorna 0.

3.10 int concatenateIntegers(int numbers[ ], int n)

Esta função recebe uma matriz de inteiros numbers e seu tamanho n, e a função concatena os inteiros da matriz para formar um único número inteiro. Ela é usada para criar um valor hash a partir dos identificadores de recursos em um compartimento.

3.11 void selectionSort(int arr[ ], int n)

Esta função realiza a classificação de seleção em uma matriz de inteiros arr com n elementos. Ela organiza os elementos em ordem crescente, o que é usado na função h para calcular o valor hash do compartimento.

3.12 int h(Compartment c)

Esta função calcula um valor hash único para um compartimento c. Ela utiliza os identificadores dos recursos no compartimento após ordená-los. O valor hash é usado para determinar a linha na tabela onde o compartimento será inserido.

3.13 int verificaAbertura(Table* t, int linha)

Esta função verifica se um compartimento em uma linha específica da tabela atingiu o limite máximo de permutações (MAX_PERMUTATIONS). Se a linha estiver cheia, a função retorna 1; caso contrário, retorna 0.

3.14 void resetar(Table* t, int linha)

Esta função redefine a quantidade de compartimentos em uma linha específica da tabela para zero e reinicia o contador de inserções. Isso é usado quando a linha atinge o limite de permutações e a tabela é "aberta".

3.15 int isNewPermutation(Table* t, int linha, Compartment c)

Esta função verifica se um compartimento c é uma nova permutação (ou seja, ainda não existe na linha da tabela). Se for uma nova permutação, a função retorna 1; caso contrário, retorna 0. É usada para evitar a inserção de compartimentos duplicados na mesma linha.