Rick Bar

Luis Henrique Soares Monteiro 2018054583

luismonteiro30@hotmail.com

Abstract. *Dado um número de recipientes disponíveis e uma quantia a ser medida, o programa retorna o menor número de operações necessárias envolvendo os recipientes passados para alcançar a medida. Ele funciona utilizando estruturas de dados de listas. Basicamente, sempre que um recipiente é passado, ele é adicionado à uma lista principal, responsável por armazenar as medidas disponíveis. Quando é solicitado que se faça um medição, é utilizada uma lista adicional chamada operações, que faz e armazena todas as operações possíveis envolvendo os recipientes disponíveis (ignorando as que resultarem em/ou menos de zero ou em elementos repetidos) até que alguma cumpra a medida desejada, e retorna o número de operações feitas para chegar naquela medida.*

Resumo. *Programa desenvolvido com o objetivo de praticar e entender o funcionamento de estruturas de dados, partindo do próprio desenvolvimento das mesmas*

1. Implementação

Inicialmente foi construída a estrutura de dados célula, que possui um valor identificador chamado chave, um valor inteiro chamado operações, que é responsável por armazenar o número de operações feitas para chegar naquela chave (valor default é 1) e um apontador para outra célula chamado próximo.

Lista: Estrutura que agrupa células, é constituída de dois apontadores para célula, sendo um referente ao primeiro elemento da lista e outro ao último, e ambos são utilizados para todas as operações de manipulação e acesso a estrutura.

Lista::existe(elemento): Método booleano que recebe uma chave por valor e verifica, iterando sobre todos os elementos da lista, se aquela chave pertence à algum elemento. Em caso positivo retorna 1, no caso de elementos negativos serem passados retorna -1, pois as listas não podem possuir elementos negativos e em caso de o elemento não existir na lista retorna 0.

Lista::Inserir(elemento,operação): Método de inserção, recebe dois inteiros, um referente à chave do elemento e outro ao número de operações feitas para chegar naquele elemento (o inteiro operações só é diferente de 1 na lista auxiliar, usada para fazer as operações de medição). Basicamente, antes de inserir um elemento, é utilizado o método existe() para garantir que elementos repetidos não sejam inseridos. No caso de o elemento não existir na lista, é criada uma célula auxiliar que recebe o valor do elemento e o número de operações a serem adicionados. Depois, uma cláusula if verifica se a cabeça da lista está vazia, e adiciona a célula àquela posição em caso positivo. Se não, o ponteiro para o próximo elemento do último da lista passa a apontar para o elemento inserido e o último elemento passa a ser a nova célula.

Lista::RemoveInput(elemento): Método de remoção, inicialmente cria uma célula auxiliar que recebe um ponteiro para a cabeça da lista, depois confere se o elemento passado como valor é equivalente à chave da cabeça da lista. Em caso positivo, o elemento seguinte a cabeça passa a ser a cabeça e o anterior é removido.

Se o elemento passado por valor não for equivalente a chave da cabeça, o método cria outro ponteiro para célula referente à terceira célula da lista. Em seguida, é feito um loop que roda enquanto o terceiro elemento em relação ao elemento atual for diferente de nulo. Enquanto o loop acontece, é verificado se a chave do segundo elemento é equivalente ao inteiro recebido por valor. Enquanto essa condição for falsa, o ponteiro atual é atualizado em uma posição. Se o while chegar a um ponto em que a terceira célula em relação a atual é nula, quer dizer que a segunda célula em relação a atual é a última, e então é verificado se o elemento é equivalente a chave dessa célula. Em caso positivo ela é removida, em caso negativo nada acontece e o elemento não é removido pois não existe célula com chave equivalente a ele na lista.

Lista::Print(): Método criado para printar todas as chaves dos elementos existentes em uma lista. Utilizado durante a criação do programa para efetuar testes. Basicamente é criado um ponteiro auxiliar para célula que vai apontar para a cabeça da lista, depois um loop rodando sobre a condição de o ponteiro para célula não ser nulo printa todas as chaves referentes a célula atual e em seguida atualiza a célula em uma posição.

Lista::size(): Método utilizado para medir o número de elementos existente em uma lista. Nele, é criado um inteiro referente ao contador que é inicializado como zero. Em seguida, uma célula auxiliar referente a cabeça da lista, e enquanto a célula for diferente de nulo, o contador é somado em 1 unidade e o apontador para célula é atualizado. Ao final da execução do loop, o programa retorna o valor somado para o contador.

Lista::LimparLista(): Método criado para zerar uma lista. Utiliza um inteiro auxiliar chamado tamanho, que é equivalente ao número de elementos da lista. Depois, um loop que varia do primeiro ao último elemento da lista chama a função RemoveInput passando a chave do elemento referente à atual cabeça da lista.

Lista::CopiaLista(exemplo): Método utilizado para copiar as chaves de elementos de uma lista. Foi utilizado durante a criação do programa para verificar se a inserção e remoção de recipientes iniciais estava sendo feita corretamente. Nesta função, são criados dois apontadores para células auxiliares. A primeira aponta para a cabeça da lista e a segunda para o elemento seguinte. Em seguida, um loop é executado enquanto a célula seguinte a principal for diferente de nula, e todos os elementos referentes a célula atual são inseridos na nova lista , atualizando a cada execução o ponteiro para as células principais e seguintes da lista.

Bar(recipientes,volume,operacoes): Parte responsável pelo cálculo do número de operações necessárias para se obter uma medição. Recebe um ponteiro para a lista

referente aos recipientes disponíveis, um inteiro referente ao volume que se deseja medir e um ponteiro para a lista de operações que será incrementada a medida que o método é executado.

Inicialmente são criados dois inteiros, NumeroElementos referente ao número de recipientes existentes e contador que é instaciado com o valor 1. Em seguida, uma cláusula if verifica se o volume que se deseja medir é igual a 0. Em caso positivo a função termina a exeução e retorna 0.

Em caso negativo, são criados dois apontadores para células auxiliares, e um deles recebe o endereço do primeiro elemento da lista de recipientes. Em seguida, um loop que varia de 1 até o número de elementos da lista é executado alternando o ponteiro para célula atual e verificando se o volume que se deseja medir já existe na lista de recipientes. Em caso positivo, termina a execução e retorna 1. Em caso negativo, insere a chave do elemento na lista de operações, com número de operações igual a 1 e atualiza a posição do ponteiro para célula. Se o loop chegar ao fim sem retornar nada, quer dizer que o número de operações até agora executadas não foi suficiente para fazer a medição de volume, iniciando um loop "infinito" que é executado enquanto o programa não encontrar uma medição equivalente ao volume desejado.

Nesse loop, os dois ponteiros para célula são resetados, apontando novamente para a cabeça da lista de recipientes, e um outro loop é criado, com a condição de o contador (inicialmente 1) ser menor ou igual ao número de elementos da lista de recipientes. Dentro do loop, uma cláusula if verifica se a soma de um entre todas possiveis combinações de dois elementos é equivalente ao volume que desejamos medir, em caso positivo a função retorna o número de operações necessárias para chegar em tal valor e a execução dos loops termina. Em caso negativo, uma cláusula else if verifica se a soma das chaves citadas acima já existe na lista de operações, e em caso de não existir, adiciona o valor das mesmas, seguido da soma de suas operações.

Após isso, outra cláusula if executa a mesma operação acima, dessa vez realizando a subtração de dois elementos e terminando a execução caso alguma das operações seja equivalente a medida desejada. Se ,com isso, ainda não for possível encontrar a medida, o valor do apontador para célula da lista de recipientes e o contador são atualizados em uma unidade, para novamente executar o loop. Se o loop chegar ao número máximo de execuções, ou seja, se o contador passar a ser maior que o número de elementos da lista de recipientes, sua execução é terminada e passamos para outra cláusula if, que verifica se o elemento seguinte da lista de operações é nulo. Em caso positivo, esse apontador volta a apontar para a cabeça da lista de operações. Em caso negativo, esse apontador é atualizado em uma posição e o primeiro loop (que acontece enquanto uma medida não for encontrada) é executado novamente.

Função main(): Semelhante à função disponibilizada inicialmente. São criados dois apontadores para as listas que serão utilizadas por todo o programa, uma referente aos recipientes, e outra referente às operações. Depois, é executado o while que verifica os inputs recebidos enquanto 'EOF' não for passado pelo usuário. Se o input for um número seguido do caractere 'i', é chamado o método de inserção de elemento pela lista de recipientes. Se for passado um número seguido do caractere 'r', a lista de recipientes chama o método de remover inputs, passando o número inteiro por valor. No caso de o

input ser um inteiro seguido do caractere ‘p’, a lista de operações chama o método para limpar uma lista, e é printado o valor retornado pela função bar() (responsável pelo cálculo do número de medições de um volume). Quando o loop de entrada é terminado, a lista de recipientes chama o método de limpeza, assim como a lista de operações, e os apontadores para as duas listas são deletados, terminando assim a execução do programa.

2. Compilação e execução

Para compilar o programa por um terminal linux, basta abrir o terminal, acessar a pasta referente ao programa (luis_monteiro) e digitar make. Para executar basta digitar ./tp1 depois de efetuar a compilação. No caso de executar a bateria de testes, basta digitar make test e o resto será feito automaticamente.

3. Análise de complexidade

No melhor caso (caso de a medida desejada ser 0) a complexidade do programa é O(1), pois só é feita uma chamada e é retornado 1.

No pior caso, quando a medida é alcançada somando n vezes o último elemento da lista e a lista é passada ordenadamente, a complexidade é 3(2n-1)+3i - 1 sendo ‘n’ o número de recipientes iniciais e ‘i’ o número da iteração feita sobre a lista de operações no momento em que a medida desejada é alcançada(i é inicializado com valor 0). Assintoticamente a complexidade no pior caso é O(nn) sendo ‘n’ o número de vezes que o segundo loop de cálculo da medida é executado.

4. Conclusão

O desenvolvimento do trabalho foi satisfatório e tranquilo num aspecto geral, enfrentei mais dificuldades especificamente com a função de remoção de recipientes no caso de uma única entrada (quando a lista só possuía dois elementos e eu precisava remover o primeiro) mas que puderam ser resolvidos com algumas mudanças no método inicial de remoção de células da lista(passei a iterar de 3 em 3 elementos, ao invés de 2 em 2).

5. Bibliografia

Neste trabalho foram utilizados como fontes de consulta os sites geekforces , codementor.io e stack overflow.