Atividade de Revisão

Anápolis, 31 de outubro de 2024

Nome da disciplina: Estrutura de Dados

Alunos: Matheus Marques Portela

1) **1:** Listas encadeadas são estruturas compostas por nós e é uma representação de uma sequência de objetos armazenadas na memoria ram. Cada elemento é armazenado em uma célula da lista, a estrutura permite uma alocação dinâmica da memória. A principal vantagem é a flexibilidade que ela pode oferecer, além do ganho de desempenho em termos de velocidade nas inclusões e remoções de elementos.
   2: A principal diferença entre a lista simplesmente e a duplamente encadeado é a quantidade de ponteiros, na lista simples possui somente um ponteiro que aponta para o próximo elemento da lista, já na lista duplamente ela possui dois ponteiros, um que aponta para o elemento anterior e outro que aponta para o próximo elemento da lista. Essa função possibilita uma navegação dos dois lados e facilita o acesso aos nós.
   3: A diferença das listas circulares para as não circulares é que em listas não circulares o ultimo ponteiro aponta para um valor NULL, já em listas circulares o ultimo ponteiro aponta para o primeiro novamente fazendo um círculo. Um exemplo vantajoso de se usar lista circulares seria em aplicações que precisam de looping, em jogos por exemplo pode se utilizar em navegações que precisam ser cíclicas.

2) **1:** Para realizar a inserção de um novo elemento em uma lista encadeada, primeiro devemos criar um novo nó e alocarmos ele na memória, depois apontamos o ponteiro do novo nó para o atual que está na cabeça e por fim atualizamos a cabeça para que aponte ao novo nó.
   2: Para realizar a operação de remoção de um nó específico, o ponteiro próximo do nó anterior ao nó a ser removido deve apontar para o próximo nó deste. Em seguida, o ponteiro anterior do próximo nó deve apontar para o nó anterior ao que foi removido. O impacto principal seria na reorganização dos ponteiros adjacentes para manter a estrutura.
   3: Em listas circulares para inserirmos um novo nó na lista no final dela, primeiro apontamos o próximo do nó anterior que apontava para o cabeça e ele começa apontar para o novo nó e o novo nó irá apontar para a cabeça da lista.

3) **1:** Uma vantagem da lista simplesmente encadeada é a sua maior facilidade de navegação, além de menor uso de memoria e a simplicidade na implementação, já as suas desvantagens seriam na navegação reversa, outra desvantagem seria para acessar os elementos, pois é preciso percorrer todos os elementos anteriores e o seu acesso pode ser mais lento. Um cenário que pode ser utilizado seria em listas que percorrer em somente uma direção, por exemplo buffers de leitura.
   A vantagem da lista duplamente encadeada é a navegação em ambas as direções, a inserção e remoção de nós é mais eficiente e possui uma maior flexibilidade em certas operações. Já a desvantagem seria seu uso maior de memoria e sua implementação é mais complexa. Um cenário

que é aplicável listas duplamente seria em casos que precisamos navegar pra frente ou para trás, um exemplo seria histórico de navegação.
2: Em listas circular o ultimo nó aponta para o primeiro da lista formando um círculo, já listas lineares os últimos ponteiros apontam para NULL. Outro exemplo de diferença é em sua eficiência em inserção e remoção de nós, em listas circulares inserir ou remover do ultimo nó para o primeiro é mais eficiente quando comparamos a listas lineares, pois nelas precisamos navegar toda a lista para inserir no final. E em relação a implementação as listas lineares são mais simples pois o ultimo nó aponta para NULL, já as circulares elas apontam para o primeiro nó fazendo um looping.
Um exemplo de aplicação de lista circular seria em jogos de turno, aonde a partida é cíclica, já em listas lineares um exemplo seria em aplicações de navegação sequencial, como itens de coleção ou arquivos de linha em linha.
3: Caso se perca o ponteiro cabeça de uma lista duplamente encadeada circular ela pode se tornar uma lista inacessível e causar perda de dados. A prevenção deste problema pode ser feita de referencias ou estruturas de dados auxiliares e caso o ponteiro seja perdido a recuperação dele pode exigir reiniciar a lista ou percorrer os nós acessíveis.

4) **1:** O código implementa uma lista simplesmente encadeada, cada nó da lista possui apenas um ponteiro que aponta para o próximo elemento. A função inserir_inicio insere um novo nó no início da lista, ajustando o ponteiro cabeça para que ele aponte para o novo nó, sem tornar a lista circular ou permitir navegação reversa.

2:
```
typedef struct No {
        int valor;
        struct No* anterior;
        struct No* proximo;
} No;
void remover_final(No** cabeca) {
        if (*cabeca == NULL) {
                return;
        }
        if ((*cabeca)->proximo == NULL) {
                free(*cabeca);
                *cabeca = NULL;
        return;
        }
        No* atual = *cabeca;
                while (atual->proximo != NULL) {
                atual = atual->proximo;
        }

        atual->anterior->proximo = NULL;
        free(atual);

}
```

3: Para verificar se todos os nós foram acessados, primeiro identificamos a cabeça e iniciamos para percorrer a lista, depois seguimos em qual sentido queremos que a lista faça a varredura e a cada iteração avança para o próximo nó, a sua condição de parada é quando volta para a cabeça.

5) Segue em anexo o código conforme o solicitado e segue um print do código em execução.