Trabalho da M1 de Estruturas de Dados:
Comparação entre Listas

Taryck, Thiago

1. Comparativo

- **Na Tabela 1, calculem e comentem a complexidade das inserções.**

Quando se trata de inserções, os diferentes tipos de lista diferem muito entre si, resultando em diferentes resultados de custo.

Se referindo à inserção no início, as listas encadeadas saem na frente pelo fato de precisarem de apenas um movimento para realizá-la, fazendo sua complexidade ser muito menor que a lista estática O(1). Ainda assim, a lista simplesmente encadeada se destaca pois necessita apenas de um apontamento (para o próximo elemento), enquanto a duplamente encadeada necessita de dois (próximo e anterior). Quanto a lista estática, a complexidade de sua inserção no início é diretamente proporcional à quantidade de elementos que já pertencem ao vetor, pois todos serão deslocados para frente quando necessário a inserção no início O(n).

Se referindo à inserção no fim, também é encontrado algumas diferenças entre as listas: quanto a lista estática genérica, tal se mostra o melhor caso. Por necessitar apenas de um índice para ser realizada a inserção de um item, faz-se melhor entre as “concorrentes” (O(1)). No caso médio se apresenta a duplamente encadeada, pois mesmo sem precisar percorrer toda a lista, ainda sim é necessário criar um novo nó e seus respectivos ponteiros e apontamentos (O(1)). No pior caso, temos a lista simplesmente encadeada, pois quando se trata de inserção no final, além de criar um novo nó, é necessário percorrer todas as posições já inseridas para chegar na última posição (O(n)).

Quanto à inserção por posição, o posto de menos complexa fica para a lista duplamente encadeada. Por contar com dois sentidos de navegação, a busca pela posição se dá pela metade dos elementos já inseridos na lista (O(n/2)), tornando-a a mais rápida forma de inserção por posição. Em seguida, vem a simplesmente encadeada: conta com apenas um sentido de busca, mas a inserção de um novo nó torna-a mais simples que a lista estática (maior complexidade), que necessita de um deslocamento de todos os elementos seguintes àquela posição (Ambas (O(n))).

Table 1. Comparativo entre operações de inserção

	Lista Estática	Lista Encadeada	Lista Duplamente Endadeada
InsereInício	O(n)	O(1)	O(1)
InsereFim	O(1)	O(n)	O(1)
InserePosição	O(n)	O(n)	O(n/2)

- **Na Tabela 2, calculem e comentem a complexidade das remoções.**

Referindo-se à remoção no início, temos a lista simplesmente encadeada como melhor caso: quando é realizada esta remoção na lista encadeada, é necessário apenas um novo apontamento do início da lista e a eliminação daquele dado (O(1)). Logo em seguida, temos a duplamente encadeada como caso médio: sua remoção acontece da mesma forma que a simplesmente encadeada, mas por ter DOIS apontamentos a serem feitos, sai em desvantagem (O(1)). Como pior caso temos a lista estática que, quando necessária a remoção no início, necessita do deslocamento de todos os elementos restantes da lista para uma posição anterior (O(n)).

Quanto à remoção no fim, a melhor opção é a lista estática: por se tratar de um vetor, a remoção no fim se dá apenas por diminuir seu tamanho em -1, assim eliminando o último elemento dessa lista e, quando ser feita uma nova inserção, irá sobrepor o valor que ali estava (O(1)). Quanto ao caso médio, a lista duplamente encadeada ocupa este posto. Mesmo podendo eliminar o último elemento num único momento (da mesma forma que a lista estática), ela tem uma maior complexidade pelo fato de necessitar de novos apontamentos antes da remoção (O(1)). No pior caso, temos a lista simplesmente encadeada pois, para remover o último elemento é necessário percorrer todos os elementos anteriores a ele na lista, além de fazer os apontamentos necessários (O(n)).

Por fim, na remoção por posição, a lista duplamente encadeada é a melhor opção: da mesma maneira que na inserção por posição, o duplo sentido de navegação favorece a duplamente encadeada em relação às outras, fazendo com que sua navegação máxima seja dada pela metade do número de elementos inseridos (O(n/2)). Na lista simplesmente encadeada e na lista estática é necessário percorrer todos os elementos até chegar na posição desejada, mas pelo fato da remoção de um nó ser menos complexa do que a remoção de um elemento de um vetor, a lista simplesmente encadeada tem uma vantagem de complexidade em relação à estática (Ambas (O(n))).

Table 2. Comparativo entre as operações de remoção

	Lista Estática	**Lista Encadeada**	**Lista Duplamente Encadeada**
Remove Início	O(n)	O(1)	O(1)
Remove Fim	O(1)	O(n)	O(1)
Remove Posição	O(n)	O(n)	O(n/2)

Inicialmente, ignorando a complexidade de implementação de cada lista e focando apenas nos índices, é perceptível que diferentemente das formas de inserção e remoção, as funções não dependem da lista que está sendo implementada, e sim da organização inicial que cada uma apresenta. Esta é a primeira forma de comparação entre o Bubblesort e o Quicksort. Visto isso, as fórmulas são iguais independentemente da lista utilizada.

Quanto ao Bubblesort, a escolha de sua equação é simples: como seu funcionamento é dado pela comparação de um dado com todos os outros dados da lista, a sua complexidade é descrita em: $(O(n^2))$, pois se adicionado um novo elemento, terá o dobro de comparações que existiam antes de sua inserção.

Quanto à melhor hipótese (algoritmo ótimo), ela se dá caso as listas já venham ordenadas e não necessitem do Bubble $(O(n))$. Já o caso médio e o pior caso têm a mesma função e são parecidos: $(O(n^2))$. A diferença ocorre que no caso médio a lista fica parcialmente embaralhada, e no pior caso fica totalmente embaralhada. Quando é levado em conta a complexidade de implementação e troca, a lista estática sai na frente por fazer a troca de valores usando apenas índices. Quando nas listas simplesmente encadeadas e duplamente encadeadas, como há mais dados a serem manipulados, tornam-se mais difíceis. A duplamente encadeada tem vantagem pois, caso necessário, pode ser percorrida em duas direções.

Se tratando do Quicksort, o seu funcionamento se dá desta maneira: a lista é dividida repetidamente em partes, e cada parte menor se resolve usando da recursividade. O fato do algoritmo ter diversas divisões leva a sua equação: $(O(n \log n))$. Quando ao melhor caso do Quicksort, ele acontece quando o pivô divide a lista inteira em exatamente duas listas de tamanhos iguais, ou muito parecidos $(O(n \log n))$. O caso médio acontece quando as listas têm tamanhos parecidos mas o pivô está distante da divisão entre cada uma, porém sua função é igual $(O(n \log n))$. O seu pior caso acontece quando a lista está parcialmente ordenada e seu pivô está muito próximo de uma das pontas, o que faz o quicksort usar da mesma equação do bubble $O(n^2)$.

Table 3. Comparativo entre Bubblesort e QuickSort

	Lista Estática	**Lista Encadeada**	**Lista Duplamente Encadeada**
Bubblesort	$O(n^2)$ ou $O(n)$	$O(n^2)$ ou $O(n)$	$O(n^2)$ ou $O(n)$
Quicksort	$O(n \log n)$ ou $O(n^2)$	$O(n \log n)$ ou $O(n^2)$	$O(n \log n)$ ou $O(n^2)$

2. Conclusões

Considerações gerais sobre o trabalho, principalmente sobre o comparativo,

A realização do trabalho trouxe ao grupo a reflexão: independentemente do tipo de estrutura de lista que é usado em determinado algoritmo, todas elas têm uma boa funcionalidade em algum determinado momento ou necessidade. Não existe uma lista certa ou errada para usar, pois cada uma delas têm seus pontos fortes e fracos. Além disso, o trabalho trouxe uma nova proposta de análise dos materiais que já haviam sido desenvolvidos, e comparando entre eles o que diferem entre si.

Porém, o enunciado trouxe o questionamento: *"qual a melhor estrutura de dados para as operações de inserção, remoção e ordenação?".* De uma forma geral se referindo a complexidade, nas operações de inserção e remoção, a lista duplamente encadeada teve uma melhor "média" entre as três listas, pois quando não era a melhor solução para o problema proposto, ela usufruía da mesma equação do melhor colocado e perdeu esse posto por conta da sua difícil implementação de estrutura, ou implementação mais simples de seu concorrente, a fazendo ganhar o posto de Melhor Estrutura para Inserção e Remoção Geral.

Já se tratando dos métodos de ordenação, como já dito no desenvolvimento, não há diferença entre o funcionamento dos dois métodos entre as diferentes tipologias de listas. A única mudança seria a forma de implementar a busca e principalmente a troca de dados de acordo com a estrutura da lista escolhida.