

Disc.: **ESTRUTURA DE DADOS**Acertos: **2,0** de 2,0

12/09/2023

1ª QuestãoAcerto: **0,2 / 0,2**

A complexidade computacional é uma abstração para facilitar a comparação de algoritmos de forma independente do ambiente de execução e de variações na sua entrada. As complexidades podem ser representadas pelo número de operações requeridas. Dentre as seguintes complexidades de pior caso, representadas pelo seu número de operações, qual é a melhor? (menos operações)

- ☐ $2n^2$
- ☐ $n \log n + 500$
- ☐ $(n \log n)/2$
- ☒ $100n + 5 \log n$
- ☐ 2^n

Respondido em 12/09/2023 02:07:54

Explicação:

O conceito de complexidade é assintótico, ou seja, o que importa é quando o tamanho da entrada n cresce arbitrariamente. Por isso, os termos dominantes de cada resposta são os únicos relevantes. $500n$ é assintoticamente menor que n^2 , por exemplo, pois para n acima de 500 o quadrado de n será maior que $500n$. Dessa forma podemos ordenar em forma crescente de complexidade os termos dominantes das respostas: n , $n \log n$, n^2 , 2^n . O menor deles é n , logo a resposta correta é $100n + 5 \log n$.

2ª QuestãoAcerto: **0,2 / 0,2**

Uma Lista pode ser implementada de forma contígua ou encadeada. No caso de uma lista ordenada implementada de forma encadeada, as complexidades de pior caso de busca, inserção e remoção são respectivamente:

- ☒ $O(n)$, $O(n)$ e $O(n)$.
- ☐ $O(n)$, $O(1)$ e $O(n)$.
- ☐ $O(1)$, $O(n)$ e $O(n)$.
- ☐ $O(\log n)$, $O(n)$ e $O(n)$.
- ☐ $O(n)$, $O(n)$ e $O(1)$.

Respondido em 12/09/2023 02:08:56

Explicação:

A busca é $O(n)$ pois no pior caso você terá que percorrer toda a lista sequencialmente até encontrar o último elemento. Já a inserção, no seu pior caso, colocará um elemento no final da lista, uma operação simples, mas a busca para achar a posição correta já é $O(n)$. A remoção de qualquer nó também é uma operação de custo constante, bastando reapontar um ponteiro, mas a busca pelo nó a ser removido também é $O(n)$, o que faz a operação de remoção também possuir complexidade $O(n)$.



3ª Questão

Acerto: 0,2 / 0,2

As árvores de busca são estruturas de dados que armazenam elementos de forma hierárquica, permitindo uma busca eficiente em grandes conjuntos de dados. Marque a opção **correta** acerca das estruturas de dados Árvores e Árvores Binárias:

- ☒ Ao acessar uma árvore, deve-se acessar pela referência a sua raiz.
- ☐ As folhas estão sempre no nível 1 da árvore.
- ☐ Os nós de uma árvore que possuem grau zero são chamados de raiz.
- ☐ Nas Árvores Binárias de Busca cada nó deve ter exatamente 2 filhos.
- ☐ A raiz está no maior nível da árvore.

Respondido em 12/09/2023 02:10:11

Explicação:

A forma comum de representar uma árvore em memória é utilizando alocação dinâmica. Não representamos a árvore como um todo, mas sim uma **referência para sua raiz** que guarda a chave (dado) e uma referência para a raiz das sub-árvores esquerda e direita. Um nó pode ter 0, 1 ou 2 filhos. As folhas podem estar em qualquer nível. A raiz pode ter grau zero quando é raiz e folha simultaneamente. A raiz está sempre no nível 1.



4ª Questão

Acerto: 0,2 / 0,2

As árvores AVL constituem uma importante estrutura de dados que disponibilizam operações de busca, inserção e remoção. Classifique como verdadeiro ou falso as afirmativas abaixo:

- I - As árvores de Fibonacci são as árvores de altura máxima h com número mínimo de nós n e altura proporcional a $\log n$.
- II - As árvores completas são árvores AVL.
- III - É possível construir uma topologia de uma árvore AVL que não seja nem completa nem de Fibonacci com altura proporcional a $\log n$.
- IV - Uma vez que a altura das árvores AVL é proporcional a $\log n$, podemos garantir que a busca ocorre numa complexidade de $O(\log n)$.
- V - Na remoção, pode ser necessário realizar todas as rotações, no pior caso, do pai de uma folha que está sendo removida até a raiz. Por esta razão, a complexidade da remoção é maior que $O(\log n)$.

- ☐ I-F, II-F, III-V, IV-F, V-F.
- ☐ I-F, II-F, III-F, IV-V, V-V.
- ☐ I-V, II-F, III-F, IV-V, V-V.
- ☐ I-V, II-V, III-F, IV-V, V-F.
- ☒ I-V, II-V, III-V, IV-V, V-F.

Respondido em 12/09/2023 02:11:39

Explicação:

Nem sempre é necessário realizar todas as operações, visto que a remoção pode eliminar uma folha e não causar desbalanceamento na árvore.



5ª Questão

Acerto: 0,2 / 0,2

Um vetor está armazenado em memória no endereço-base 24. Considerando que uma palavra em memória ocupa 1 byte, e esse vetor é constituído por elementos que ocupam 4 palavras, qual é o endereço de memória ocupado pelos elementos de índices 2 e 50 respectivamente? .

- ☐ 26 e 74
- ☐ 28 e 224
- ☐ 28 e 220
- ☐ 32 e 220
- ☒ 32 e 224

Respondido em 12/09/2023 02:13:16

Explicação:

Para calcular o endereço absoluto em memória você deve utilizar a fórmula $A = B + t * i$. No caso B é o endereço base (24), t é o tamanho de cada elemento (4) e i é o índice do elemento (2 e 50). Aplicando a fórmula temos $32 = 24 + 2 * 4$ e $224 = 24 + 50 * 4$



6ª Questão

Acerto: 0,2 / 0,2

Uma lista L em alocação contígua está armazenada em memória no endereço 32. L possui elementos de 2 bytes cada e no momento contém [10, 20, 30, 40]. Os elementos 5 e 50 serão inseridos em sequência. Em que endereços eles serão inseridos, respectivamente, caso a lista não seja ordenada, e caso a lista seja ordenada?

- ☐ Não ordenada: 36, 37; ordenada: 32, 37.
- ☒ Não ordenada: 40, 42; ordenada: 32, 42.
- ☐ Não ordenada: 32, 34; ordenada: 32, 34.
- ☐ Não ordenada: 36, 37; ordenada: 32, 33.
- ☐ Não ordenada: 40, 42; ordenada: 40, 42.

Respondido em 12/09/2023 02:13:58

Explicação:

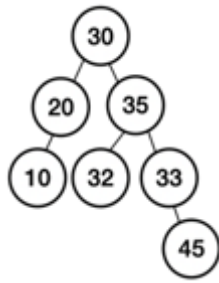
A inserção na lista não ordenada ocorre ao final da lista, o 5o elemento será inserido na posição L[4] ou seja endereço $32 + 4 * 2 = 40$. O elemento seguinte L[5] será inserido no endereço $32 + 5 * 2 = 42$. Já no caso ordenado, o primeiro elemento deverá ser inserido na primeira posição L[0], endereço 32. Todos os demais elementos serão deslocados uma posição. O segundo elemento será inserido ao final da lista em L[4]. Ou seja, endereço $32 + 4 * 2 = 42$ (levando em conta o deslocamento). Solução é, portanto: 40,42, 32,42.



7ª Questão

Acerto: 0,2 / 0,2

Seja a seguinte árvore binária. Analise as afirmativas e marque a correta.



- ☐ A árvore acima possui 4 nós folhas.
- ☒ A figura ilustra uma árvore binária de busca porque para todos os nós vale a seguinte propriedade: os nós contidos na subárvore esquerda são menores que a raiz e os contidos na subárvore direita são maiores.
- ☐ Ao buscar pelo nó 45 na árvore acima, um algoritmo de busca em Python irá realizar sempre $O(n)$ passos. Isto ocorre uma vez que é necessário analisar todos os nós da árvore para encontrar o nó 45.
- ☐ É possível inserir mais um nó filho ao nó 30.
- ☐ Supondo que a árvore em questão é uma árvore binária de busca, a inserção de um novo nó com chave 47 pode ser feita em qualquer subárvore vazia.

Respondido em 12/09/2023 02:16:14

Explicação:

A árvore é uma árvore binária de busca porque dado um nó qualquer da árvore, os nós contidos na subárvore esquerda são menores que a raiz e os a direita, maiores que a raiz. A busca só seria executada em $O(n)$ caso a árvore fosse uma árvore zig zag. 47 só poderia ser inserido à direita de 45. A árvore tem 3 folhas. Árvores binárias só podem ter nós com grau 2.



8ª Questão

Acerto: 0,2 / 0,2

Árvores de busca são organizadas de forma hierárquica, onde cada nó é um objeto que contém informação e cada nó filho é uma subdivisão da informação contida no nó pai. Sobre árvores binárias de busca, marque a opção correta.

- ☐ A melhor árvore binária que podemos construir tem altura proporcional a n , onde n é a quantidade de nós da árvore.
- ☒ Seja T uma árvore binária completa com $n > 0$ nós. Então T possui altura mínima e $h = 1 + \lceil \log n \rceil$.
- ☐ As operações de busca, inserção e remoção são exatamente $O(n \log n)$.
- ☐ Em toda árvore binária de busca as inserções correm em $O(\log n)$ e as remoções em $O(n)$, independentemente da altura da árvore.
- ☐ Ao se construir uma árvore binária de busca com a sequência 8-13-10-2-5-15-4-7 o número de folhas raiz é igual a 3.

Respondido em 12/09/2023 02:20:04

Explicação:

As operações de remoção e inserção tem complexidade assintótica proporcional à altura da árvore, assim, se a árvore é balanceada ocorrerá em $O(\log n)$. Toda árvore de busca e que está balanceada tem altura proporcional a $\log n$. Logo, a complexidade das operações busca, inserção e remoção são exatamente $O(\log n)$. A melhor árvore binária de busca é a balanceada. Temos 4 folhas.



9ª Questão

Acerto: 0,2 / 0,2



Em Python é possível implementar um array utilizando o tipo padrão list. Essa implementação permite o uso das seguintes funções para inserir e remover um elemento, respectivamente:

- ☐ insert, delete/pop.
- ☒ append, remove/pop.
- ☐ append, pop/delete.
- ☐ impose, remove/destroy.
- ☐ insert, remove/destroy.

Respondido em 12/09/2023 02:21:20

Explicação:

Em Python a função append insere um elemento ao final da lista. As funções "remove" e "pop" podem remover um elemento, de maneiras diferentes. Remove tira um elemento conhecido usando o seu conteúdo, já pop remove um elemento usando seu índice, ou seja, a sua posição na lista.



10ª Questão

Acerto: 0,2 / 0,2

Uma Lista pode ser implementada de forma contígua ou encadeada. No caso de uma lista implementada de forma contígua, as complexidades de pior caso de busca, inserção e remoção são respectivamente:

- ☐ $O(1)$, $O(n)$ e $O(n)$.
- ☐ $O(n)$, $O(n)$ e $O(1)$.
- ☒ $O(n)$, $O(n)$ e $O(n)$.
- ☐ $O(\log n)$, $O(n)$ e $O(n)$.
- ☐ $O(n)$, $O(1)$ e $O(n)$.

Respondido em 12/09/2023 02:22:51

Explicação:

A busca é $O(n)$ pois no pior caso você terá que percorrer toda a lista sequencialmente até encontrar o último elemento. Já a inserção, no seu pior caso, colocará um elemento no início da lista, obrigando todos os demais a serem deslocados uma posição, levando $O(n)$ operações.

A remoção também tem nesse seu pior caso, remover o primeiro elemento, o que obrigará todos os demais a serem puxados para uma posição levando tempo $O(n)$.

Esse custo pode ser diminuído caso você implemente uma variável que indique o início da lista, mesmo assim, ao remover um elemento exatamente do meio da lista, você precisará mover $n/2$ elementos, o que ainda é um custo linear.