Avaliando Aprendizad

Teste seu conhecimento acumulado

Disc.: **ESTRUTURA DE DADOS**

Acertos: **1,8** **de 2,0** **14/11/2023**

1ª Questão /

Acerto: **0,2** / **0,2**

O método de ordenação da bolha, ou Bubblesort (BS) tem complexidade de pior caso O(n2) e melhor caso O(n). Suponha que exista um algoritmo de ordenação MS que tem complexidade de melhor caso O(nlog n) e de pior caso O(nlog n). Podemos afirmar que:

- ☐ MS e BS são igualmente eficientes em ordenar elementos, independente da entrada ou seu tamanho.
- ☒ ✔ Para um grande conjunto de entradas variadas de tamanho grande, MS executará em menos tempo que BS, em média.
- ☐ Para um grande conjunto de entradas variadas de tamanho grande, BS executará em menos tempo que MS, em média.
- ☐ Para uma única entrada de tamanho grande, BS executará em menos tempo que MS.
- ☐ Para uma única entrada de tamanho grande, MS executará em menos tempo que BS.

Respondido em 14/11/2023 16:44:30

Explicação:

Pela natureza do tratamento de complexidade de algoritmos e o uso da notação O, a única afirmativa verdadeira é a de que em média, com um grande número de entradas distintas e de tamanho grande, MS executará mais rápido que BS pois sua complexidade assintótica O(nlogn) é melhor que a de BS $O(n^2)$.

A afirmação inversa está errada pelo mesmo argumento, O(nlogn) é melhor que $O(n^2)$.

As afirmações que tratam de uma única entrada são falsas, pois você sempre pode escolher uma entrada que seja de melhor caso para um dos algoritmos e seja ruim para o outro.

Por fim, a afirmação de que ambos são igualmente eficientes é desmentida pelo pior caso.

2ª Questão /

Acerto: **0,0** / **0,2**

Suponha que você está implementando um programa que precisa armazenar dados ordenados em uma estrutura para serem tratados posteriormente, na ordem em que foram recebidos. Haverá uma grande quantidade de recebimentos e tratamento de dados, mas o tamanho esperado da estrutura não deve variar muito. Qual tipo de estrutura de dado é a melhor nessa situação?

- [ ] Lista duplamente encadeada.
- [ ] Lista em alocação contígua.
- [ ] Lista simplesmente encadeada.
- [ ] ✔ Fila.
- [x] ✘ Pilha.

Respondido em 14/11/2023 16:41:58

Explicação:

A fila permite o tratamento de nós usando a política requerida, FIFO ¿ ¿first in first out¿ -. Além disso, as operações de inserção e remoção são O(1), ou seja, de complexidade constante, a melhor possível. Isso condiz com o requisito de que haverá muitas operações desse tipo. Por fim, o fato de a estrutura não variar muito em tamanho permite o uso de uma alocação contígua e otimizada para a fila usando lógica circular e variáveis para o início e final da fila. A pilha não obedece a lógica FIFO e as listas tem complexidade de inserção e remoção O(n) sendo muito piores que a fila, principalmente quando o número desses tipos de operação é grande.

3ª Questão /

Acerto: **0,2** / **0,2**

Seja a seguinte função em Python para percurso em uma árvore binária implementada em Python. Marque a opção correta de qual percurso em árvores se trata essa função:

```
def Visita(raiz):
    if raiz:
        print(raiz.chave),
        Visita(raiz.esquerda)
        Visita(raiz.direita)
```

- [x] ✔ Percurso Pré-ordem
- [ ] Percurso Anti simétrico
- [ ] Percurso raiz-folha
- [ ] Percurso Em-ordem
- [ ] Percurso Pós-ordem

Respondido em 14/11/2023 16:52:47

Explicação:

Para realizar o percurso em pré-ordem, são necessários três acessos ao nó. No caso da pré-ordem, no primeiro, executamos a visita (print(raiz.chave)), no segundo, chamamos recursivamente o algoritmo para a sub-árvores esquerda (Visita(raiz.esquerda)) e, no terceiro, ocorre a chamada do percurso em pré-ordem do ramo direito (Visita(raiz.direita)).

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 do 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).

- [x] I-V, II-V, III-V, IV-V, V-F.
- [ ] I-V, II-V, III-F, IV-V, V-F.
- [ ] I-F, II-F, III-F, IV-V, V-V.
- [ ] I-V, II-F, III-F, IV-V, V-V.
- [ ] I-F, II-F, III-V, IV-F, V-F.

Respondido em 14/11/2023 16:43:29

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**

Matrizes podem ser implementadas em Python utilizando a biblioteca numpy, trazendo diversas funções já implementadas. Dentre os pares de função com sua funcionalidade a seguir, qual é o correto?

- [ ] matriz.min() retorna o valor médio da matriz.
- [ ] matriz.max() retorna o desvio padrão da matriz.
- [ ] matriz.std() retorna a variância da matriz.
- [ ] matriz.mean() retorna o valor mínimo da matriz.
- [x] matriz.sum() retorna a soma dos elementos da matriz.

Respondido em 14/11/2023 16:40:44

Explicação:

Dentre os pares apresentados, o único correto é o da função sum() que é a soma dos elementos. std() e mean() são funções estatísticas que retornam o desvio padrão e a média respectivamente. max() retorna o elemento de maior valor e min(), por sua vez, retorna o elemento de menor valor.

6ª Questão / Acerto: **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(1), O(n) e O(n).
- ☐ O(log n), O(n) e O(n).
- ☐ O(n), O(1) e O(n).
- ☐ O(n), O(n) e O(1).
- ☒ ✔ O(n), O(n) e O(n).

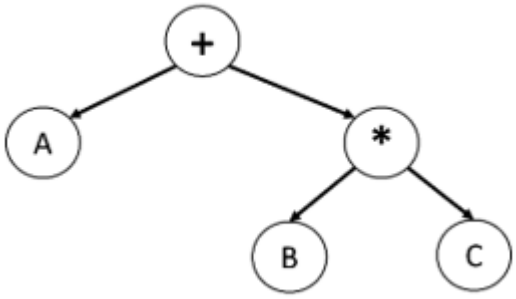
Respondido em 14/11/2023 16:48:43

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).

7ª Questão / Acerto: **0,2 / 0,2**

Seja a seguinte árvore de expressões aritméticas abaixo.



O resultado da visita em prefixo dessa árvore é:

- ☐ A + B * C
- ☒ ✔ + A * B C
- ☐ A * B + C
- ☐ A + (B * C)
- ☐ A + * B C

Respondido em 14/11/2023 16:53:27

Explicação:

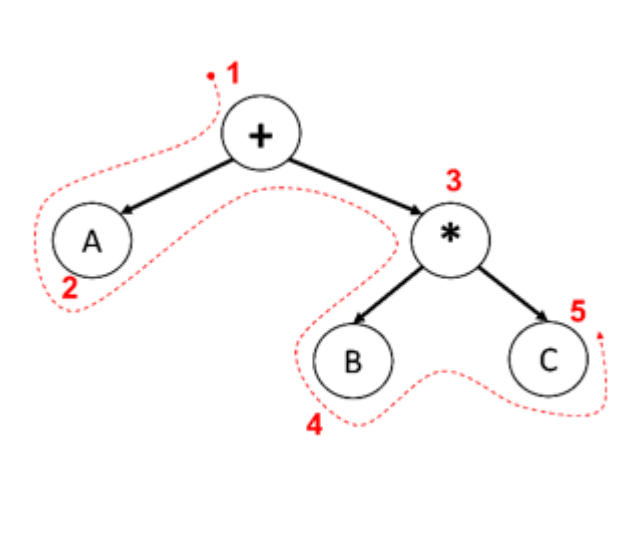
O percurso em prefixo é definido recursivamente. A partir da raiz *r* da árvore de expressões aritméticas *T*, percorre-se a árvore de da seguinte forma:

1 - visita-se a raiz;

2 - percorre-se a sub-árvores esquerda de *T,* em prefixo e

3 - percorre-se a sub-árvores direita de *T*, em prefixo.

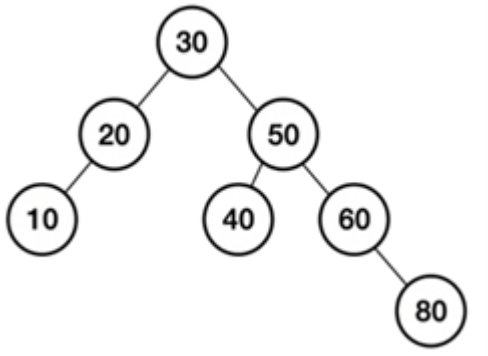
O resultado da visita é representado abaixo:



8ª Questão /

Acerto: **0,2** / **0,2**

Seja a seguinte árvore AVL abaixo. Com a inserção da chave 90, marque a opção que indica exatamente o que acontecerá com a árvore resultante após essa inserção:

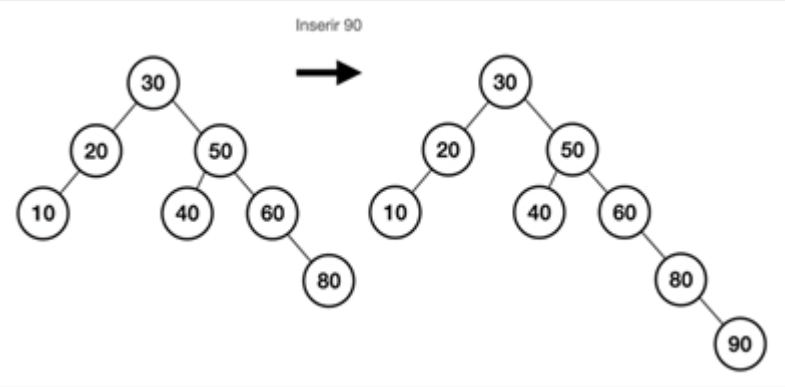


- [x] A árvore resultante irá desbalancear à esquerda do nó de chave 60.
- [ ] A árvore resultante irá manter o balanceamento geral da árvore.
- [ ] A árvore resultante irá desbalancear à direita do nó de chave 40.
- [ ] A árvore resultante irá desbalancear à esquerda do nó de chave 10.
- [ ] A árvore resultante irá desbalancear à direita do nó de chave 80.

Respondido em 14/11/2023 16:47:21

Explicação:

Ao inserir o nó de chave 90, ele é maior que o nó 80, sendo assim, inserido ao lado direito de 80, causando desbalanceamento do nó 60 que tem altura da subárvore direita 2 e esquerda 0.



9ª Questão /

Acerto: **0,2** / **0,2**

O método de ordenação da bolha, ou Bubblesort tem como melhor caso a entrada já ordenada, que resulta em complexidade O(n). Como seu pior caso, a entrada em ordem invertida, resultando em complexidade $O(n^2)$. Baseado nessas duas afirmações, podemos afirmar que a sua complexidade de caso médio é:

- ☐ O(nlog n)
- ☐ O(1)
- ☒ ✔ $O(n^2)$
- ☐ O(log n)
- ☐ O(n)

Respondido em 14/11/2023 16:41:15

Explicação:

Pelas características da notação O, a única afirmação que podemos extrair é que o caso médio é melhor ou igual ao pior caso. Portanto, é possível afirmar que o caso médio é $O(n^2)$, ou qualquer função assintoticamente superior a n^2, como $n^2 \log n$, n^3, 2^n etc.. Como dentre essas a única opção disponível é $O(n^2)$ essa é a resposta correta.

Podemos descartar O(1) e O(log n) por serem melhores que o melhor caso, o que contradiz a afirmativa do melhor caso.

Os casos O(n) e O(nlog n) seriam possíveis teoricamente para a complexidade média de um algoritmo qualquer que seja O(n) no melhor caso e $O(n^2)$ no pior caso, mas não é possível afirmar nenhuma das duas com as informações dadas.

De fato, o caso médio do Bubblesort é $O(n^2)$.

10ª 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, 33.
- [x] Não ordenada: 40, 42; ordenada: 32, 42.
- [ ] Não ordenada: 36, 37; ordenada: 32, 37.
- [ ] Não ordenada: 32, 34; ordenada: 32, 34.
- [ ] Não ordenada: 40, 42; ordenada: 40, 42.

Respondido em 14/11/2023 16:49:34

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.