Revisar envio do teste: Semana 3 - Atividade Avaliativa

Usuário	LIZIS BIANCA DA SILVA SANTOS
Curso	Algoritmos e Programação de Computadores II - COM120 - Turma 003
Teste	Semana 3 - Atividade Avaliativa
Iniciado	11/05/23 18:13
Enviado	11/05/23 18:24
Data de vencimento	12/05/23 05:00
Status	Completada
Resultado da tentativa	10 em 10 pontos
Tempo decorrido	10 minutos
Instruções	Olá, alunos e alunas! Esta atividade possui múltipla escolha. Para respondê-la: 1. Selecione, com o mouse, a alternativa que você considerar correta; 2. Repare que, ao selecionar uma alternativa, as seleções anteriores são desmarcadas; 3. Após selecionar a resposta correta em todas as questões, vá até o fim da página e clique em "Enviar teste". Pronto! Sua atividade já está registrada no AVA.
Resultados exibidos	Todas as respostas, Respostas enviadas, Respostas corretas, Comentários, Perguntas respondidas incorretamente

Pergunta 1

1,43 em 1,43 pontos

Ao passar um valor x para uma função recursiva para realizar uma soma, o que acontece é que a função vai somar de 1 até o valor x, o que indica que ela está chamando a si mesma, porém, a cada vez, com um argumento diferente.

Assinale a alternativa que representa a função citada de forma generalizada.

Resposta Selecionada: a. f(x) = f(x-1) + x

Respostas:
- a. f(x) = f(x-1) + x
- b. f(x) = f(x-1) - x
- c. f(x) = f(x-1)
- d. f(x) = f(x+1) - x
- e. f(x) = f(x+1) + x

Comentário da resposta: **JUSTIFICATIVA**
A função se torna recursiva quando, dentro dela, começa a chamar ela mesma, mas com outro argumento, ou seja, neste caso foi passado um valor x para a função recursiva, então é somado x com o valor de f(x-1) demonstrando que ela chama ela mesma, só que ao invés de passar o parâmetro x, vai passar o parâmetro x-1. Aí a função que agora recebe x-1 vai somar (x-1) com f(x-2) chamando a si própria de novo só que com o parâmetro x-2. Continua acontecendo o mesmo processo até chegar em f(1) que é igual a 1.

Pergunta 2

1,43 em 1,43 pontos

A recursão é uma função que chama a si própria e vale destacar que o mais importante é saber identificar seu ponto de parada de modo a evitar que ela seja executada infinitamente. Esse ponto de parada é chamado de "caso base".

Com base nas informações apresentadas, identifique se são verdadeiras (V) ou falsas (F) as afirmativas a seguir.

I. ( ) As funções recursivas em Python apresentam grandes benefícios em relação à melhora da eficiência.
II. ( ) Caso base e o caso recursivo são partes que integram toda função recursiva.
III. ( ) O Python usa o recurso de exceção exibindo a fim de evitar que um código seja executado infinitamente.

Assinale a alternativa que apresenta a sequência correta.

Resposta Selecionada: e. F - V - V.

Respostas:
- a. V - V - F.
- b. F - F - V.
- c. V - F - F.
- d. V - F - V.
- e. F - V - V.

Comentário da resposta: **JUSTIFICATIVA**
A afirmativa I é falsa, porque o uso das funções recursivas no Python não provoca benefícios em termos de desempenho, pois os laços permitem resolver o problema com mais eficiência, mas elas conseguem ser mais intuitivas ao programador em virtude de algum problema que possa ser fracionado em problemas menores do mesmo tipo.
A afirmativa II é verdadeira, pois, uma função recursiva compõem-se de, no mínimo, um caso base (função que não chama a si mesma, mas retorna um valor real) e um caso recursivo (produz a recursividade).
A afirmativa III é verdadeira, pois uma exceção (RecursionError: maximum recursion depth exceeded) ocorre devido ao fato de que não foi informado um caso base que avise a função recursiva de quando ela deve parar de funcionar.

Pergunta 3

1,42 em 1,42 pontos

As listas em Python permitem listar informações dentro de uma única variável para que elas sejam utilizadas dentro do código. Uma prática comum ao se trabalhar com listas é a utilização de informações dentro dela, já que uma lista comporta uma estrutura de dados com itens organizados linearmente que podem ser acessados por meio de um índice. Essa tarefa de acesso pode ser facilitada diante de uma ordenação que simplifica o trabalho das informações contidas na lista.

Assinale a alternativa que representa a função cujo objetivo é a ordenação das informações de uma a lista.

Resposta Selecionada: c. sorted()

Respostas:
- a. remove()
- b. list()
- c. sorted()
- d. index()
- e. pop()

Comentário da resposta: **JUSTIFICATIVA**
A função sorted() na linguagem Python classifica uma sequência, sendo ela uma lista ou tupla, retornando como resultado uma lista de elementos classificados sem modificar a sequência original. Diferentemente de index(), que obtém o índice de um item em uma lista, de remove(), que remove um item baseado no seu valor e não na sua posição, de pop(), que remove o último item, mas não o exclui, e de list(), que permite a criação de listas.

Pergunta 4

1,42 em 1,42 pontos

No programas que usam recursão, como no caso do Fibonacci que exemplifica uma sobrecarga de operador de chamada de função, faz-se necessário que quando uma função é chamada de forma repetida fazendo uso das mesmas entradas, o seu resultado seja carregado do cache ao invés de ser recomputado porque isso fará com que recursos da CPU sejam economizados.

Analise as alternativas abaixo e indique qual delas contém a técnica citada no enunciado.

Resposta Selecionada: d. Memoização.

Respostas:
- a. Recorrência.
- b. Função.
- c. Recursão.
- d. Memoização.
- e. Cache.

Comentário da resposta: **JUSTIFICATIVA**
A técnica de memoização oferece ajuda na forma recursiva do algoritmo mantendo um registro dos cálculos já realizados, dessa forma, evita cálculos repetidos e agiliza os programas, ou seja, trata-se de uma técnica de otimização que armazena os resultados de chamadas de funções custosas e retorna o valor armazenado quando a função é chamada novamente, assim, consegue otimizar programas que usam recursão. Em Python a memoização pode ser feita com a ajuda de decoradores de função.

Pergunta 5

1,42 em 1,42 pontos

É necessário o entendimento sobre recursão e iteração para seu uso de forma adequada, pois existem problemas naturalmente recursivos e aqueles definidos em termos recursivos. Uma função pode ser escrita como uma função recursiva sem o uso de iteração e uma função recursiva pode ser descrita por meio de iterações sucessivas.

Com base nas informações apresentadas, identifique se são verdadeiras (V) ou falsas (F) as afirmativas a seguir.

I. ( ) Toda função recursiva deve conter uma condição que estabelece o ponto em que ela deve parar de chamar a si própria.
II. ( ) A recursão usa repetição por meio de comandos e utiliza uma condição de teste que, ao falhar, finaliza a iteração.
III. ( ) A iteração usa repetição por meio de várias chamadas a uma rotina e utiliza o alcance de um caso trivial para ser finalizada.
IV. A recursão entra em *loop* infinito nos casos em que o teste nunca se torna falso.

Assinale a alternativa que apresenta a sequência correta.

Resposta Selecionada: a. V - F - F - V.

Respostas:
- a. V - F - F - V.
- b. V - V - F - F.
- c. V - F - F - F.
- d. F - V - V - F.
- e. F - F - F - V.

Comentário da resposta: **JUSTIFICATIVA**
A afirmativa I é verdadeira, pois toda função recursiva possui uma condição de parada, já que é ela quem determina em que ponto ela não mais chamará a si própria, iniciando o processo de saída das sucessivas chamadas.
A afirmativa II é falsa, pois a iteração usa repetição por meio de comandos e utiliza uma condição de teste que, ao falhar, finaliza a iteração.
A afirmativa III é falsa, pois a recursão usa repetição por meio de várias chamadas a uma rotina e utiliza o alcance de um caso trivial para ser finalizada.
A afirmativa IV é verdadeira, pois a iteração termina quando a condição de teste falha e a recursão termina quando se atinge o caso trivial. As duas podem entrar em *loop* infinito, no caso da iteração se o teste nunca se tornar falso e no caso da recursão se o problema não for reduzido de forma que convirja para o caso trivial.

Pergunta 6

1,44 em 1,44 pontos

A recursividade em Python é apresentada como uma forma para solucionar problemas cujo fundamento é a fragmentação de um problema em subproblemas menores de tal forma que a função para resolver tal contratempo chame a si mesmo até chegar em um problema que tenha uma simplicidade que viabiliza sua resolução de uma forma trivial. Todos os algoritmos recursivos devem obedecer a três leis importantes, apontadas em 1, 2 e 3. Sobre tais leis, avalie as afirmações a seguir, e relacione-as adequadamente aos termos às quais se referem.

1. Primeira lei.
2. Segunda lei.
3. Terceira lei.

I. Deve mudar seu estado para se aproximar do caso básico.
II. Deve chamar a si mesmo, recursivamente.
III. Deve possuir um caso básico.

Assinale a alternativa que correlaciona adequadamente os dois grupos de informação.

Resposta Selecionada: d. 1-III; 2-I; 3-II.

Respostas:
- a. 1-I; 2-III; 3-II.
- b. 1-III; 2-II; 3-I.
- c. 1-I; 2-II; 3-III.
- d. 1-III; 2-I; 3-II.
- e. 1-II; 2-I; 3-III.

Comentário da resposta: **JUSTIFICATIVA**
A sentença I se enquadra no conceito 2, pois a segunda lei diz que, em todo algoritmo recursivo deve ter uma forma de realizar uma mudança de estado que possibilita levar o algoritmo para o caso básico, modificando determinados dados que geralmente representam o problema e que podem ser reduzidos.
A sentença II se enquadra no conceito 3, porque a terceira lei diz que, seguindo a definição da recursão, todo algoritmo recursivo deve fazer uma chamada a si próprio para que a função contida nele resolva o problema chamando a si mesma a fim de que, diante de um grande problema este possa ser fracionado em partes menores para facilitar sua resolução.
A sentença III se enquadra no conceito 1 visto que a primeira lei afirma que a condição que possibilita o algoritmo recursivo parar de recorrer é o caso básico, por isso todo algoritmo deve dispor de um caso básico que tradicionalmente trata-se de um problema que se torna pequeno o suficiente para ser resolvido de forma direta.

Pergunta 7

1,44 em 1,44 pontos

O algoritmo de busca binária considera um vetor ordenado de n elementos para realizar a varredura dos elementos, por isso é possível implementar um algoritmo mais eficiente que utiliza busca sequencial. Adotando o paradigma dividir para conquistar, o problema global é dividido em subproblemas, o que faz com que o espaço de busca se reduza à metade a cada iteração do algoritmo.

Com relação ao algoritmo de busca binária apresentado, avalie as afirmações a seguir.

I. O número máximo de comparações requeridas começa com n/2, n/4, n/6, n/8 e assim sucessivamente.
II. A busca binária é O(nlogn) devido ao logaritmo do número de itens na lista.
III. Se n for um valor pequeno, o custo adicional para ordenar a lista não compensa.
IV. A análise da busca binária considera eliminar metade dos itens que restam a cada comparação.

Está correto que se afirma em:

Resposta Selecionada: b. III e IV, apenas.

Respostas:
- a. I e III, apenas.
- b. III e IV, apenas.
- c. II e IV, apenas.
- d. I, II e IV, apenas.
- e. I e II, apenas.

Comentário da resposta: **JUSTIFICATIVA**
A afirmativa I é incorreta. O número máximo de comparações requeridas para conferir a lista de forma integral é, ao iniciar com n elementos, será em torno de n/2 elementos que sobram após a primeira comparação e após a segunda comparação será em torno de n/4, depois n/8, n/16 e assim sucessivamente.
A afirmativa II também é incorreta. A busca binária é O(logn). Ao fracionar a lista em uma quantidade de vezes suficiente, chega-se a uma lista formada por um só elemento, portanto ele será ou não o elemento procurado e o processo chega ao fim. A quantidade de comparações realizadas ao chegar neste ponto é i, em que n/2^i=1 (n dividido por 2 elevado a i é igual a 1), ao isolar i temos i=log n, portanto o número máximo de comparações é o logaritmo do número de elementos na lista, logo, O(logn).
A afirmativa III é correta. Para valores pequenos de n, o custo adicional de ordenar a lista não compensa, por isso é importante considerar o custo/benefício do trabalho adicional de ordenação para obter benefícios na busca.
A afirmativa IV é correta. Para analisar o algoritmo de busca binária é necessário considerar que cada comparação consome metade dos itens restantes a serem comparados.