Graduação Presencial Sincron...
Página inicial
Teams
Módulos
Notas
Objetivos
Programa
Biblioteca PUC Minas
Tarefas
Testes
Avaliação CPA
PUC Carreiras

ATIVIDADE OBJETIVA - M32a - Análise de Algoritmos

Entrega Sem prazo **Pontos** 7 **Perguntas** 7 **Limite de tempo** Nenhum **Tentativas permitidas** Sem limite

Detalhes da última tentativa:

Tempo:	2 minutos
Pontuação atual:	7 de 7
Pontuação mantida:	7 de 7

3 tentativas até o momento
Visualizar tentativas anteriores
Tentativas ilimitadas

Fazer o teste novamente
Será mantida sua pontuação mais alta

Instruções

O objetivo desta atividade é proporcionar uma melhor fixação quanto ao conteúdo da unidade 1.
Com isso você será capaz de ter um entendimento das abordagens apresentadas na unidade **ANÁLISE DE ALGORITMOS**.

Instruções:

Clique em "**Fazer o teste**";
Responda todas as questões e depois clique no botão "**Enviar teste**" para finalizar a atividade.

Fazer o teste novamente

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MANTIDO	Tentativa 3	2 minutos	7 de 7
MAIS RECENTE	Tentativa 3	2 minutos	7 de 7
	Tentativa 2	3 minutos	6 de 7
	Tentativa 1	7 minutos	2 de 7

As respostas corretas estão ocultas.

Pontuação desta tentativa: de 7
Enviado 2 Abr em 21:10
Esta tentativa levou 2 minutos.

Pergunta 1

1 / 1 pts

Dentro dos cursos da área de Informática, os estudantes são introduzidos ao estudo de Análise de Algoritmos, um assunto fundamental para compreender a eficiência e o desempenho dos algoritmos utilizados na resolução de problemas computacionais.

Ao analisar algoritmos, é fundamental compreender a relação entre a entrada e o tempo de execução dos mesmos, expresso através da função de complexidade. A função de complexidade de um algoritmo indica como o tempo de execução aumenta em função do tamanho da entrada, fornecendo informações cruciais para escolher a abordagem mais adequada para resolver problemas computacionalmente.

Considerando o contexto acima, assinale a alternativa correta.

- Uma função de complexidade quadrática (O(n^2)) indica que o tempo de execução do algoritmo cresce linearmente com o tamanho da entrada, sendo uma ótima opção para algoritmos com alta eficiência.
- Uma função de complexidade exponencial (O(2^n)) indica que o tempo de execução cresce exponencialmente à medida que o tamanho da entrada aumenta, tornando-se eficiente mesmo para grandes volumes de dados.
- A função de complexidade logarítmica (O(log n)) é aquela em que o tempo de execução cresce rapidamente à medida que o tamanho da entrada aumenta, tendo pouca eficiência para problemas de grande porte.
- **Uma função de complexidade linear (O(n)) implica que o tempo de execução do algoritmo cresce proporcionalmente ao tamanho da entrada, tornando-se uma escolha adequada para grandes conjuntos de dados.**

Alternativa correta. Uma função de complexidade linear (O(n)) implica que o tempo de execução do algoritmo cresce proporcionalmente ao tamanho da entrada. Isso torna esse tipo de função uma escolha adequada para grandes conjuntos de dados, já que o aumento do tempo de execução é previsível e linear.

Pergunta 2

1 / 1 pts

Em uma tarde de sábado, uma estudante de Análise de Complexidade decide reviver seus estudos em preparação para uma importante avaliação. Ela decide iniciar sua revisão pelo tema de complexidade de algoritmos, um assunto que julga particularmente desafiador. Recordando as aulas, ela pondera sobre como a análise de complexidade permite aos desenvolvedores entender e comparar a eficiência dos algoritmos, considerando tanto a perspectiva de tempo quanto de espaço. Ela pensa em como a notação Big O, uma das mais utilizadas na Ciência da Computação, serve como uma ferramenta essencial para representar a taxa de crescimento do tempo de execução de um algoritmo em função do tamanho da entrada. Este conceito de notação Big O a faz refletir sobre as nuances de complexidades lineares, quadráticas e constantes, lembrando que a escolha do algoritmo pode impactar diretamente no desempenho e escalabilidade de um sistema.

Com relação a reflexão da estudante, assinale a opção correta.

- **Um algoritmo com complexidade O(1) é considerado mais eficiente do que um algoritmo com complexidade O(n).**

Correto. Um algoritmo com complexidade O(1) é considerado mais eficiente que um algoritmo com complexidade O(n). O(1) significa que o tempo de execução é constante, independente do tamanho da entrada.

- A notação Big O descreve o melhor caso de tempo de execução de um algoritmo.
- O(n) indica que a complexidade do tempo do algoritmo diminui com o aumento do tamanho da sua entrada.
- A notação Big O pode ser usada para descrever a complexidade de tempo de um algoritmo e não a de espaço.

Pergunta 3

1 / 1 pts

Imagine uma aventura lúdica, fictícia e intrigante pela Floresta da Complexidade.

Reza a lenda que nessa floresta existia uma misteriosa árvore ancestral, conhecida como *'Árvore das Funções'*. Essa árvore possuía galhos que se estendiam até os céus, representando as distintas funções de complexidade dos algoritmos. Cada galho era adornado por belos frutos, simbolizando diferentes algoritmos que podiam ser analisados.

Para desbravar os segredos da Árvore das Funções, os aventureiros (estudantes de complexidade) precisavam decifrar enigmas e desafios matemáticos, que revelavam as características de cada algoritmo e como eles se comportavam diante de diferentes tamanhos de entradas.

Em sua jornada, os aventureiros aprendiam a decifrar a notação **Θ**, que servia como um guia para descrever o comportamento assintótico das funções de complexidade. Em uma clareira da floresta, os aventureiros encontravam um sábio ermitão, que há tempos estudava a relação entre tempo de execução e tamanho das entradas dos algoritmos. O ermitão compartilhou suas sábias palavras: "Caros viajantes do conhecimento, compreender a função de complexidade de um algoritmo é como desvendar os segredos do tempo e espaço que regem a eficiência computacional. Uma vez que vocês entendam a notação **Θ**, poderão discernir entre algoritmos velozes e aqueles que serpenteiam em busca de soluções." Inspirados pelos palavras do ermitão, os estudantes perseveravam em sua exploração, até que, finalmente, chegaram ao topo da Árvore das Funções. Lá, encontraram o fruto dourado da sabedoria, que representava o conhecimento adquirido ao compreender as nuances da análise de complexidade.

Com esse tesouro em mãos, retornaram ao mundo real, preparados para aplicar seus novos conhecimentos em desafios reais de otimização algorítmica.

Considerando as experiências vivenciadas na Floresta da Complexidade, assinale a opção correta.

- A função de complexidade é uma medida para avaliar o tempo de execução de um algoritmo, e quanto maior o valor da função, mais eficiente é o algoritmo.
- O pior caso de um algoritmo é representado pela entrada que resulta no menor tempo de execução, tornando-o mais adequado para a maioria das situações.
- A complexidade de um algoritmo está relacionada apenas com o tamanho da entrada, sem levar em conta o tempo de execução.
- **A notação Θ é utilizada para representar a função de complexidade de um algoritmo, indicando sua tendência de crescimento.**

Correta. A notação **Θ** é utilizada para representar a função de complexidade de um algoritmo e descreve tendência de crescimento do algoritmo, ou seja, como o tempo de execução cresce em relação ao tamanho da entrada. Essa notação é amplamente usada para avaliar a eficiência dos algoritmos.

Pergunta 4

1 / 1 pts

A análise de algoritmos é o estudo do desempenho de algoritmos, especialmente seu tempo de execução e necessidade de espaço. Um componente fundamental deste estudo é a caracterização de algoritmos em termos de seu tempo de execução ou espaço requerido como uma função do tamanho da entrada, conhecida como complexidade de tempo e complexidade de espaço, respectivamente. Além disso, os algoritmos são frequentemente classificados em categorias com base em sua complexidade, tais como algoritmos de tempo polinomial e algoritmos de tempo exponencial.

Com relação ao comentado, avalie as afirmações a seguir.

I. Um algoritmo com complexidade de tempo O(n log n) tem melhor performance que um algoritmo com complexidade de tempo O(n^2) para grandes entradas.

II. A complexidade de espaço de um algoritmo se refere ao tempo que ele leva para executar.

III. Um algoritmo de ordenação por seleção, na melhor das hipóteses, tem complexidade de tempo O(n log n).

É correto o que se afirma em

- I e II apenas.
- **I, apenas.**
- I e III, apenas.
- II, apenas.

I. Esta afirmação é verdadeira. O(n log n) tem melhor performance que O(n^2) para grandes entradas. À medida que o se torna grande, o tempo de execução do algoritmo de O(n log n) cresce muito mais devagar do que o do algoritmo O(n^2), tornando-o mais eficiente para grandes entradas.

II. Esta afirmação é falsa. A complexidade de espaço de um algoritmo se refere à quantidade de memória que ele necessita para executar, e não ao tempo que ele leva para executar. A quantidade de memória necessária pode ser em termos do tamanho da entrada, mas também pode ser em termos de variáveis auxiliares, estruturas de dados, etc.

III. Esta afirmação é falsa. O algoritmo de ordenação por seleção tem complexidade de tempo O(n^2) mesmo na melhor das hipóteses. Ele opera encontrando o menor (ou maior, dependendo da ordem de classificação) elemento da lista e trocando-o com o primeiro elemento não classificado. Este processo é repetido para o restante dos elementos, até que toda a lista esteja classificada. Cada passo envolve uma varredura completa dos elementos não classificados, resultando em uma complexidade de tempo quadrática.

Pergunta 5

1 / 1 pts

O melhor caso do algoritmo Quicksort ocorre quando a partição divide o array de maneira equilibrada, isto é, quando o pivô escolhido divide o array em duas partes aproximadamente iguais. Assim, cada partição subsequente continuará a dividir os dados restantes igualmente, resultando em um array equilibrado, semelhante a uma árvore binária balanceada. Este cenário é ideal porque maximiza a eficiência do algoritmo, mantendo a profundidade da pilha de chamada recursiva ao mínimo e permitindo que o Quicksort opere na sua eficiência máxima de Θ(n log n).

Contudo, na prática, para trabalhar próximo do melhor caso é importante escolher um bom pivô, como o uso do método "mediana de três", que consiste em pegar a mediana entre o primeiro, o último e o elemento central do array.

Sobre Quicksort, avalie as afirmações a seguir.

I. O melhor caso do Quicksort é O(n).

II. O melhor caso do Quicksort é O(n^).

Assinale a alternativa correta.

- As afirmações I e II são falsas.
- **A afirmação I é falsa e a II é verdadeira.**
- As afirmações I e II são verdadeiras.
- A afirmação I é verdadeira e a II é falsa.

A afirmação I é verdadeira porque se o melhor caso do Quicksort é Θ(n log n), então ele é O(n log n), ele também é O(n).

A afirmação II é verdadeira porque se o melhor caso do Quicksort é Θ(n log n), então ele é O(n log n) e se ele é O(n log n), ele também é O(n^).

Pergunta 6

1 / 1 pts

A Informática, em sua vasta abrangência, lida com uma gama de problemas que exigem resoluções eficientes. A fim de analisar a eficiência dessas soluções, faz-se uso de notações especiais como O, Omega e Theta, que descrevem o comportamento de algoritmos em termos de tempo e espaço. Estas notações, em sua essência, ajudam a avaliar os limites superior e inferior do desempenho do algoritmo em relação ao tamanho da entrada.

Com relação ao comentado, avalie as afirmações a seguir.

I. A notação O (Big O) representa o limite superior do tempo de execução de um algoritmo.

II. A notação Omega é usada para descrever o limite inferior do tempo de execução de um algoritmo.

III. A notação Theta é usada quando o tempo de execução de um algoritmo é exatamente o mesmo para todos os tamanhos de entrada.

É correto o que se afirma em

- I e II, apenas.
- I e III, apenas.
- II e III, apenas.
- I, apenas.

I. Esta afirmação é verdadeira. A notação Big O é usada para descrever o limite superior de tempo que um algoritmo pode levar para executar, logo, se um algoritmo é O(f(n)), ele também será O(g(n)) para toda função g(n) tal que "g(n) é maior que f(n)"

II. Esta afirmação é verdadeira. A notação Omega é usada para descrever o limite inferior de tempo que um algoritmo pode levar para executar, logo, se um algoritmo é Ω(f(n)), ele também será Ω(g(n)) para toda função g(n) tal que "g(n) é menor que f(n)"

III. Esta afirmação é falsa. A notação Theta não é usada quando o tempo de execução de um algoritmo é exatamente o mesmo para todos os tamanhos de entrada. Ao contrário, a notação Theta é usada para denotar um tempo de execução que é limitado tanto superiormente quanto inferiormente por duas funções constantes do tamanho de entrada, ou seja, quando o tempo de execução de um algoritmo é o mesmo, independentemente do tamanho da entrada.

Pergunta 7

1 / 1 pts

Os diversos desafios no mundo dos algoritmos se apresentam quando se busca a resolução eficiente de problemas. Diante de uma variedade de soluções possíveis, a escolha de uma estratégia ideal de resolução pode se tornar uma tarefa complexa. Para auxiliar nesse processo, faz-se uso de mecanismos de análise que possibilitem a comparação entre diferentes algoritmos, como a notação Theta (**Θ**). Essa notação se destaca por sua capacidade de expressar o tempo de execução de um algoritmo em termos de um limite superior e um limite inferior que crescem no mesmo ritmo para grandes entradas.

Considerando o contexto acima, avalie as afirmações a seguir.

I. A notação Theta é utilizada para representar o limite superior estrito no desempenho de um algoritmo.

II. A notação Theta também é usada para descrever o limite inferior estrito do tempo de execução de um algoritmo.

III. Quando um algoritmo é dito estar em Theta de uma função, significa que seu tempo de execução é diretamente proporcional ao tamanho da entrada.

É correto o que se afirma em

- III, apenas.
- I e III apenas.
- II e III apenas.
- I e II apenas.

I. Esta afirmação é falsa. A notação Theta não representa o limite superior estrito no desempenho de um algoritmo. Este papel é desempenhado pela notação Big O.

II. Esta afirmação é falsa. A notação Theta não é usada para descrever o limite inferior estrito do tempo de execução de um algoritmo. Este papel é desempenhado pela notação Omega.

III. Esta afirmação é verdadeira. Quando um algoritmo é dito estar em Theta de uma função, isto implica que o tempo de execução do algoritmo é diretamente proporcional ao tamanho da entrada para grandes entradas, ou seja, existe um limite superior e um limite inferior que crescem no mesmo ritmo que a função para tais entradas.

Pontuação do teste: de 7