ATIVIDADE OBJETIVA - M32b - Projeto de Algoritmos e Teoria da Complexidade

Entrega Sem prazo **Pontos** 8 **Perguntas** 8 **Limite de tempo** Nenhum **Tentativas permitidas** Sem limite

Detalhes da última tentativa:

Tempo:	1 minuto
Pontuação atual:	8 de 8
Pontuação mantida:	8 de 8

4 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 2.
Com isso você será capaz de ter um entendimento das abordagens apresentadas na unidade **PROJETO DE ALGORITMOS E TEORIA DA COMPLEXIDADE**.

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 4	1 minuto	8 de 8
MAIS RECENTE	Tentativa 4	1 minuto	8 de 8
	Tentativa 3	5 minutos	7 de 8
	Tentativa 2	4 minutos	6 de 8
	Tentativa 1	Menos de 1 minuto	3 de 8

As respostas corretas estão ocultas.

Pontuação desta tentativa: de 8
Enviado 2 dez em 21:26
Esta tentativa levou 1 minuto.

Pergunta 1

1 / 1 pts

Na busca por soluções eficientes no universo da Informática, a compreensão e aplicação de técnicas robustas de Projeto de Algoritmos são essenciais. Dentre essas técnicas, a Backtracking e a Programação Dinâmica destacam-se como abordagens fundamentais para a solução de problemas complexos. Backtracking é uma estratégia que tenta resolver um problema gerando soluções candidata progressivamente e abandonando uma candidata assim que se torna óbvio que a candidata não pode ser possivelmente estendida para uma solução válida. Por outro lado, a Programação Dinâmica é uma abordagem que resolve problemas dividindo-os em subproblemas menores e mais gerenciáveis, armazenando os resultados dos subproblemas para evitar trabalho redundante.

Com relação ao comentado, assinale a opção correta.

- O Backtracking envolve a geração e o teste de todas as possíveis soluções candidatas para um problema.

  Correto. O Backtracking envolve a geração progressiva de soluções candidatas para um problema e o abandono de uma candidata assim que se torna claro que ela não pode ser estendida para uma solução válida.

- O Backtracking e a Programação Dinâmica são técnicas intercambiáveis e podem ser usadas indistintamente para resolver os mesmos problemas.
- A Programação Dinâmica é ineficaz para problemas que envolvem a tomada de decisões sequenciais.
- A Programação Dinâmica é uma abordagem ineficiente que aumenta a complexidade do tempo de execução.

Pergunta 2

1 / 1 pts

Dentro do universo da Informática, os desafios são constantes e a demanda por soluções eficientes e rápidas é uma realidade inescapável. Profissionais da área devem estar preparados para enfrentar problemas de diferentes naturezas e magnitudes, e para isso, possuírem um amplo repertório de técnicas e estratégias é crucial. A estratégia de Dividir e Conquistar, por exemplo, é uma abordagem poderosa que quebra um problema em subproblemas menores e mais gerenciáveis, resolve esses subproblemas independentemente e, por fim, combina suas soluções para resolver o problema original. Esta técnica é amplamente utilizada em algoritmos de ordenação e busca, como QuickSort e MergeSort, além de outras situações em que a quebra de problemas em partes menores pode facilitar a obtenção de soluções.

Considerando o contexto acima, assinale a alternativa correta.

- Na estratégia de Dividir e Conquistar, os subproblemas são sempre resolvidos em paralelo.
- A estratégia de Dividir e Conquistar é eficaz apenas para problemas de ordenação.
- A estratégia de Dividir e Conquistar é impraticável quando o problema não pode ser quebrado em partes de tamanhos iguais.
- A estratégia de Dividir e Conquistar envolve dividir o problema em subproblemas menores, resolver esses subproblemas e combinar suas soluções.

  Correto. A estratégia de Dividir e Conquistar envolve dividir o problema em subproblemas menores, resolver esses subproblemas de forma independente e, por fim, combinar suas soluções para resolver o problema original.

Pergunta 3

1 / 1 pts

Em muitas situações práticas da área de Informática, um profissional é desafiado a construir soluções eficientes para problemas complexos. O mundo digital, no qual vivemos atualmente, demanda respostas rápidas e precisas. No corne dessas soluções, frequentemente, residem algoritmos inteligentes projetados para lidar com essas complexidades. Dentre as diversas abordagens para o projeto de algoritmos, a estratégia guiosa é uma técnica proeminente. Essa estratégia opera por meio de uma lógica de otimização contínua, realizando, a cada passo, a escolha que parece ser a melhor naquele momento, com a expectativa de que essas decisões locais resultem em uma solução globalmente ótima. Embora seja uma abordagem poderosa, seu uso requer entendimento aprofundado e atenção, pois nem todos os problemas podem ser resolvidos de forma eficiente por meio de estratégias guiosas.

Com relação ao comentado, assinale a opção correta.

- A estratégia guiosa é útil apenas quando há uma solução única para o problema.
- A estratégia guiosa é uma variação da pesquisa exaustiva, eliminando soluções desnecessárias
- Na estratégia guiosa, a escolha feita em cada etapa depende das escolhas feitas nas etapas anteriores.
- A estratégia guiosa, em certos casos, pode conduzir a uma solução ótima global fazendo a escolha ótima local a cada etapa.

  Esta é a afirmação correta. A estratégia guiosa, em alguns casos, pode levar a uma solução ótima global fazendo a escolha ótima local em cada etapa.

Pergunta 4

1 / 1 pts

Em diversas situações no campo da Informática, especialmente na resolução de problemas de alta complexidade, é crucial selecionar a abordagem algorítmica adequada. Uma dessas abordagens é a técnica de Força Bruta, que busca solucionar problemas através da geração de todas as possíveis soluções, testando cada uma delas até encontrar a solução desejada. A técnica de Força Bruta pode ser simples e direta, mas também pode ser ineficiente para problemas de maior escala devido à grande quantidade de tempo computacional que pode consumir.

Com relação ao uso da técnica de Força Bruta, assinale a opção correta.

- A Força Bruta não é adequada para problemas que exigem soluções otimizadas, pois essa técnica busca apenas soluções viáveis.
- A técnica de Força Bruta é uma abordagem de baixo para cima, onde as soluções dos subproblemas são combinadas para formar a solução do problema maior.
- A Força Bruta é sempre a técnica mais eficiente para solucionar problemas complexos devido à sua simplicidade.
- A Força Bruta é uma técnica que pode ser útil quando o espaço de soluções é pequeno e a geração de todas as possíveis soluções é viável.

  Correto. A Força Bruta pode ser útil quando o espaço de soluções é pequeno e a geração de todas as possíveis soluções é viável, embora possa ser ineficiente para problemas de maior escala.

Pergunta 5

1 / 1 pts

Em cenários desafiadores na área de Informática, especialmente quando confrontados com problemas complexos, uma série de técnicas de projeto de algoritmos pode ser empregada. Entre elas, encontramos a Força Bruta, que investiga exaustivamente todas as soluções possíveis; a Divisão e Conquista, que divide o problema em partes menores e mais gerenciáveis; as Estratégias Guiosas, que selecionam a melhor opção a cada etapa na expectativa de chegar à melhor solução geral; e a Programação Dinâmica, que resolve o problema ao resolver subproblemas sobrepostos de maneira eficiente. Cada uma dessas estratégias tem seus próprios méritos e é mais adequada a certos tipos de problemas do que outras.

Considerando o contexto acima, assinale a opção correta.

- A técnica de Divisão e Conquista é particularmente eficaz quando aplicada a problemas que podem ser divididos em subproblemas independentes.

  Correto. A técnica de Divisão e Conquista é particularmente eficaz quando aplicada a problemas que podem ser divididos em subproblemas independentes, que são resolvidos individualmente e depois combinados para formar a solução do problema original.

- A estratégia guiosa garante a obtenção de uma solução ótima global.
- A técnica de Força Bruta é sempre a mais eficiente em termos de tempo de execução, pois verifica todas as possíveis soluções.
- Todas as técnicas de projeto de algoritmos são universalmente aplicáveis a todos os tipos de problemas computacionais.

Pergunta 6

1 / 1 pts

Em muitos cenários na área de Informática, é essencial resolver problemas complexos de forma eficiente. Quatro estratégias de resolução de problemas que se destacam são a Divisão e Conquista, as Estratégias Guiosas, o Backtracking e a Programação Dinâmica. A Divisão e Conquista é uma abordagem que resolve problemas dividindo-os em subproblemas menores que são mais simples de resolver, para depois combinar as soluções desses subproblemas para formar a solução do problema original. Já as Estratégias Guiosas abordam problemas selecionando a opção mais promissora disponível a cada passo com a esperança de encontrar a solução ideal. O Backtracking resolve problemas explorando todas as possíveis soluções e abandonando as opções que claramente não levam a uma solução. Por fim, a Programação Dinâmica é uma técnica que resolve problemas complexos dividindo-os em subproblemas menores e resolvendo cada um desses subproblemas apenas uma vez, armazenando os resultados para uso futuro.

Ao lidar com um problema de otimização que envolve uma série de decisões inter-relacionadas, a técnica mais adequada a ser adotada

- é a Divisão e Conquista, pois esta técnica permite dividir o problema em partes independentes que podem ser resolvidas simultaneamente.
- é o Backtracking, pois esta técnica garante a busca completa do espaço de soluções.
- são as Estratégias Guiosas, pois essas estratégias são simples de implementar e exigem menos recursos computacionais.
- é a Programação Dinâmica, pois esta técnica evita a resolução repetida de subproblemas, economizando tempo computacional.

  Correto. A Programação Dinâmica é uma técnica adequada para problemas de otimização que envolvem uma série de decisões inter-relacionadas, pois evita a resolução repetida de subproblemas, economizando tempo computacional.

Pergunta 7

1 / 1 pts

A análise da complexidade computacional de algoritmos e processos é uma competência crucial na área da Computação. É por meio desta lente de avaliação que podemos fazer escolhas informadas e determinar estratégias eficazes de resolução de problemas. Isso é possível graças à Teoria da Complexidade, um componente-chave que nos permite discernir a viabilidade de soluções algorítmicas dentro de limitações específicas de tempo e espaço. Tal teoria, rigorosa e matemática, segmenta problemas em diferentes classes (P, NP, NP-Completo e NP-Difícil) e fornece uma estrutura para entender as fronteiras teóricas do que é computacionalmente possível. É uma ferramenta indispensável para os cientistas da computação, equipando-os com a capacidade de navegarem os desafios do campo da Informática.

Com base nesta contextualização, veja as seguintes declarações e identifique a opção correta.

- Todo problema em NP-Completo é NP, mas nem todo NP-Completo é NP-Difícil.
- A classe P inclui todos os problemas que podem ser resolvidos em tempo polinomial por um algoritmo não determinístico.
- Todo problema NP-Difícil é, por definição, também um problema NP-Completo.
- Se P = NP, então não existe algoritmo de tempo polinomial para qualquer problema NP-Completo.

  Correto. Se P = NP, então problemas NP-Completo não podem ser resolvidos em tempo polinomial por um algoritmo determinístico.

Pergunta 8

1 / 1 pts

No vasto e infinito universo da Informática, constelações de problemas brilham com intensidades variáveis, cada uma refletindo sua singular complexidade. Assim como um astrônomo mapeia as estrelas, a Teoria da Complexidade tem o seu papel de cartógrafo no cosmos da computação. Sua incumbência, grandiosa e significativa, se destina a classificar esses problemas cósmicos com base em suas dificuldades inatas e a tecer constelações de conexões entre eles. Esse exercício misterioso e belo não apenas adiciona ordem ao caos, mas também permite aos navegantes dessa imensidão estelar — os profissionais da informática — avaliar a eficácia de suas ferramentas computacionais, ou seja, os algoritmos, aprimorando-os em sua jornada de solução de problemas.

A Teoria da Complexidade é, no entanto, permeada por enigmas e paradoxos, muito semelhante ao próprio universo que busca mapear. O mais famoso destes, a questão P = NP?, paira como uma supernova ainda não completamente entendida. Esta intrigante indagação questiona se todos os problemas cuja solução pode ser verificada rapidamente por um computador (problemas NP) também podem ser resolvidos com igual agilidade (problemas P). É uma pergunta que ecoa através dos corredores da ciência da computação, às vezes parecendo estar à beira de uma resposta, às vezes recuando para as sombras do desconhecido.

Nesse cenário, torna-se imprescindível interpretar e selecionar a afirmação correta. Os enunciados abaixo tentam lançar luz sobre diferentes aspectos da teoria, e apenas um deles é correto. Cabe a nós analisar e selecionar a opção que se alinha à verdade no misterioso cosmos da Teoria da Complexidade da Computação.

- Se um problema é NP-Completo, então é impossível encontrar uma solução em tempo polinomial.
- P = NP foi provado e é um dos grandes resultados da Teoria da Complexidade.
- Todos os problemas NP são problemas P e todos os problemas NP-Completo são NP-Difícil.
- A classe P contém apenas problemas que podem ser resolvidos em tempo polinomial de forma determinística.

  Correto. Um problema pertence à classe P quando existe um algoritmo determinístico polinomial para resolvê-lo.

Pontuação do teste: de 8

‹ Anterior

Próximo ›