unipac.br
Barbacena

Discentes:

- **Bernardo Resende Andrés**
- **Rafael De Souza Damasceno**

Análise e Projeto de Algoritmo

Classe de Problemas

Exercício 1

Definir as classes de problemas P, NP, NP Completo e NP Difícil

Explicar:

O que é:

O que representa

Para que serve:

Incluir exemplos de problemas.

- **P**

  A classe de problema P é uma classe de problemas computacionais caracterizada pelo fato de que existe um algoritmo polinomial para resolvê-los. Em termos simples, isso significa que o tempo de cálculo necessário para resolver esses problemas cresce de forma polinomial em relação ao tamanho da entrada do problema. Problemas em P são considerados "fáceis" ou "tratáveis" do ponto de vista algorítmico, uma vez que o tempo de execução do algoritmo de resolução não cresce exponencialmente com o aumento do tamanho do problema.

  A classe P é fundamental na teoria da complexidade computacional, pois abrange uma ampla gama de problemas práticos que podem ser resolvidos eficientemente. Esses problemas têm importância significativa em diversas áreas, como otimização, programação linear, grafos e muitos outros campos da ciência da computação e matemática aplicada. A classe P é crucial para avaliar a eficiência dos algoritmos e para classificar problemas de acordo com sua complexidade computacional. A identificação de um problema como pertencente a P fornece uma garantia de que existe uma solução eficiente para ele.

  Exemplos de problemas:

  - Ordenação de Lista.
  - Busca Binária:

- Encontrar o Maior Elemento
- Problema do Caixeiro-Viajante

- **NP**

A classe de problemas de tempo polinomial não determinístico (NP) é uma classe importante na teoria da complexidade computacional. Consiste no problema de ser capaz, dada uma solução candidata, de determinar se a solução está correta em tempo polinomial. Isso significa que, embora possa ser difícil encontrar uma solução, testar uma solução proposta é relativamente fácil e rápido. Os problemas NP incluem uma variedade de problemas do mundo real, como o problema do caixeiro viajante e o problema do máximo de cliques em um gráfico. O termo “não determinístico” refere-se ao fato de que a solução correta pode ser adivinhada imediatamente a partir de um modelo computacional teórico.

A classe NP faz parte de muitos problemas do mundo real que enfrentamos na computação cotidiana. A teoria da complexidade também tem uma questão chave: P versus P. NP pergunta se todos os problemas em NP também são P (ou seja, se problemas cujas soluções podem ser rapidamente identificadas também podem ser encontrados rapidamente). Esta questão continua sendo um dos problemas não resolvidos mais importantes da ciência da computação.

Exemplo de Problemas:

- Problema da Mochila (Knapsack)
- Problema do Subconjunto com Soma

- **NP Completo**

A classe NP-completa (tempo polinomial não determinístico completo) é uma classe especial de problemas NP com propriedades interessantes. Se for NP, o problema é NP-completo e, em certo sentido, tão difícil quanto

o problema mais difícil em NP. Ou seja, se você conseguir encontrar um algoritmo polinomial para resolver um problema NP-completo, você terá automaticamente um algoritmo polinomial para resolver todos os problemas NP. Inclui também problemas para os quais ainda não conhecemos soluções eficazes. O conceito central que liga problemas NP-completos é o de redução polinomial. Se a entrada de um problema NP-completo puder ser rapidamente transformada em um problema concreto em tempo polinomial, qualquer algoritmo que resolva o segundo problema poderá ser usado para resolver o primeiro problema. O primeiro problema NP-completo conhecido foi o Problema de Correspondência Booleana (SAT), proposto por Stephen Cook em 1971. Desde então, muitos outros problemas demonstraram ser NP-completos usando redução polinomial SAT e outros problemas NP-completos. A importância prática do problema NP-completo reside na hipótese de que não existe um algoritmo polinomial eficiente para resolver o problema. Isto significa que uma vez encontrado um algoritmo eficiente (polinomial) para um problema NP completo, ele pode ser usado para resolver qualquer problema NP de forma eficiente, desafiando a visão atual de que P (problema polinomial) difere de NP (problema polinomial) em polinômios. significa. tempo disponível para confirmação)

Exemplo de Problemas:

- Problema do Circuito Hamiltoniano
- Problema da Partição

- **NP Difícil**

Um problema na classe NP-hard é quando o mesmo se ele for tão difícil quanto os problemas mais difíceis em NP, mas não é necessário pertencer à classe NP. Em outras palavras, não há a exigência de que um problema NP-hard tenha uma solução verificável em tempo polinomial, como é o caso dos problemas em NP. A principal diferença entre NP-hard e NP-

completo é que todos os problemas NP-completos estão em NP, enquanto os problemas NP-hard podem estar ou não em NP. Um problema NP-completo é um problema NP-hard específico que também pertence à classe NP. A ideia principal é que se você puder resolver um problema NP-difícil em tempo polinomial, poderá resolver qualquer problema em NP de forma eficiente, mas não necessariamente em tempo polinomial verificável. A Teoria dos Problemas NP-Difíceis é importante para a compreensão da complexidade computacional porque ajuda a definir os limites da complexidade do problema. Os problemas do mundo real são muitas vezes difíceis de provar que os NPs são complexos e não se podem esperar soluções eficientes.

Exemplo de Problemas:

- Problema do Ciclo Hamiltoniano em Grafos
- Problema do Emparelhamento Máximo em Grafos

2)

Sobre redução de problemas, explicar:

Como funciona

Para que serve

Onde se aplica

Incluir exemplos

- **Como funciona:**

A redução de problemas geralmente segue um padrão lógico, onde um problema complexo (chamado de problema A) é transformado em um problema mais simples (problema B) que já conhecemos a solução ou que é mais fácil de resolver. Se conseguirmos resolver o problema B, podemos utilizar essa solução para resolver o problema A original.

- **Para que serve:**

Simplificação de Complexidade: Permite lidar com problemas complexos, que podem ser difíceis de resolver diretamente, transformando-os em problemas mais gerenciáveis.

Demonstração de Indecidibilidade: É frequentemente usado para provar que um problema é indecidível ao mostrar que, se pudéssemos resolver um problema conhecido (que já sabemos ser indecidível), poderíamos resolver o problema em questão.

- **Onde se aplica:**

Ciência da Computação: Em teoria da computação, a redução é frequentemente utilizada para mostrar a equivalência entre problemas, como na redução de problemas NP-completos.

Matemática: Pode ser usado para transformar problemas matemáticos em formas mais simples, facilitando a solução.

- **Exemplos:**

Redução de Problemas NP-completos:

Problema A: O problema do caixeiro-viajante.

Problema B: O problema do circuito hamiltoniano.

Se conseguirmos resolver eficientemente o problema B, podemos usar essa solução para resolver o problema A.

Redução em Matemática:

Problema A: Provar que um número é primo.

Problema B: Fatorar o número.

Se pudermos eficientemente fatorar o número (problema B), podemos concluir que o número não é primo (problema A).

3) Um dos problemas computacionais mais conhecidos é o do Caixeiro Viajante (TSP).

Pesquise esse problema computacional e informe:

• A sua classe de problema (P, NP, NP-Completo ou NP-Difícil).

• Por qual motivo ele está categorizado nessa classe?

O Problema do Caixeiro Viajante (TSP) pertence à classe de problemas NP-Completo.

O TSP é classificado como NP-Completo devido à sua natureza combinatória e à falta de um algoritmo polinomial conhecido para resolvê-lo de forma eficiente.

- **Motivo da Categorização:**
  Natureza de Decisão: O TSP pode ser formulado como um problema de decisão, onde a pergunta é se existe um ciclo hamiltoniano que visite cada cidade exatamente uma vez e que tenha um comprimento total menor ou igual a um valor dado.
  Dificuldade de Solução Eficiente: Apesar de existirem algoritmos que conseguem resolver instâncias pequenas do TSP de maneira eficiente, não há um algoritmo conhecido que possa resolver todas as instâncias em tempo polinomial. A complexidade aumenta exponencialmente com o número de cidades, tornando a resolução prática impraticável para instâncias maiores.

Referências

http://www.faccamp.br/osvaldo/PAAII/ClassesComplexidadeProblemas.pdf

https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/NPcompleto.html

https://www2.fct.unesp.br/docentes/dmec/olivete/tc/arquivos/Aula11.pdf

https://mat.unb.br/~ayala/LECTURES/AA/notasaulaComplexidade.pdf

https://brasil.elpais.com/brasil/2017/05/19/tecnologia/1495202801_698394.html#:~:text=Os%20problemas%20que%20s%C3%A3o%20resolvidos,polin%C3%B4mio%20no%20tamanho%20dos%20dados.

https://pt.wikipedia.org/wiki/P_(complexidade)

https://pt.wikipedia.org/wiki/NP_(complexidade)

https://pt.wikipedia.org/wiki/NP-completo

https://pt.wikipedia.org/wiki/NP-dif%C3%ADcil#:~:text=NP%2Ddif%C3%ADcil%20(ou%20NP%2D,problemas%20mais%20dif%C3%ADceis%20em%20NP%22.

https://www.yumpu.com/pt/document/read/19642531/classes-de-problemas-faccamp

https://docs.fct.unesp.br/docentes/dmec/olivete/tc/arquivos/Aula9.pdf

https://www.studocu.com/pt-br/document/universidade-regional-do-noroeste-do-estado-do-rio-grande-do-sul/complexidade-computacional/metodos-para-reducao-de-problemas/11222900

https://danielsaad.com/teoria-da-computacao/assets/aulas/redutibilidade.pdf

http://www.mat.ufrgs.br/~portosil/caixeiro.html

https://www2.fct.unesp.br/docentes/dmec/olivete/tc/arquivos/Aula11.pdf

https://www.inf.ufrgs.br/~prestes/Courses/Complexity/aula26.pdf

https://abepro.org.br/biblioteca/TN_STO_263_509_35790.pdf

https://www.youtube.com/watch?v=y5UdMdcJ1ow