UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Classe de complexidade de tempo: Introdução/ P $\times$ NP

Teoria da Computação

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

06 de junho de 2023

Classe de complexidade de tempo

Introdução

- Quase todos os algoritmos que estudamos até aqui são algoritmos de tempo polinomial: para entradas de tamanho n, seu tempo de execução do pior caso é $O(n^k)$ para alguma constante k
- É natural imaginar que todos os problemas podem ser resolvidos em tempo polinomial
- A resposta é não!

Classe de complexidade de tempo

Introdução

- Por exemplo, existem problemas, como o famoso “problema da parada"de Turing, que não pode ser resolvido por qualquer computador, não importa por quanto tempo seja executado
- Também existem problemas que podem ser resolvidos, mas não no tempo $O(n^k)$ para qualquer constante k
- Em geral, pensamos que problemas que podem ser resolvidos por algoritmo de tempo polinomial são tratáveis, ou fáceis, e que problemas que exigem tempo superpolinomial são intratáveis ou difíceis

Classe de complexidade de tempo

Introdução

- Porém, o assunto dessa parte do curso é uma classe interessante de problemas, denominados problemas "NP-Completos", cujo status é desconhecido
- Ainda não foi descoberto nenhum algoritmo de tempo polinomial para um problema NP-Completo e ninguém ainda conseguiu provar que não pode existir nenhum algoritmo de tempo polinomial para nenhum deles
- Essa questão denominada P $\neq$ NP continua sendo um dos mais profundos e intrigantes problemas de pesquisa ainda em aberto na teoria da ciência da computação, desde que foi proposto pela primeira vez em 1971
- Ele é um dos sete Problemas do Prêmio Millenium selecionados pelo Clay Mathematics Institute que paga um milhão de dólares para a primeira solução correta

Classe de complexidade de tempo

Introdução

- Vários problemas NP-Completos são particularmente torturantes porque, à primeira vista, parecem ser semelhantes a problemas que sabemos resolver em tempo polinomial
- Em cada um dos pares de problemas a seguir, um deles pode ser resolvido em tempo polinomial e o outro é NP-Completo, mas a diferença entre eles parece insignificante

Classe de complexidade de tempo

Introdução

- **Caminhos simples de comprimento mínimo e caminhos simples de comprimento máximo:**Podemos determinar o primeiro em O(VE). Determinar o caminho simples de comprimento máximo entre dois vértices é difícil. Simplesmente determinar se um grafo contém um caminho simples com pelo menos um determinado número de arestas é NP-Completo.
- **Passeio de Euler e ciclo hamiltoniano:** Um passeio de Euler de um grafo conexo dirigido G=(V,E) é um ciclo que percorre cada aresta de G exatamente uma vez, embora possa visitar cada vértice mais de uma vez. Podemos determinar isso em O(E). Um ciclo hamiltoniano é um ciclo simples que contém cada vértice do grafo. Determinar se um grafo dirigido tem um ciclo hamiltoniano é NP-Completo.

Classe de complexidade de tempo

Introdução

- **Satisfazibilidade 2-CNF e satisfazibilidade 3-CNF:** Uma fórmula booleana contém variáveis cujos valores são 0 ou 1; conectivos booleanos como $\wedge$, $\vee$ e $\neg$; e parênteses. Uma fórmula é satisfazível se existe alguma atribuição dos valores 0 e 1 às suas variáveis que faça com que ela seja avaliada como 1. Determinar se uma fórmula 2-CNF é satisfazível pode ser feito em tempo polinomial. Determinar se uma fórmula 3-CNF é satisfazível é NP-completo.

Classe de complexidade de tempo

NP-Completude e as classes P e NP

- No restante deste curso, nos referiremos a três classes de problemas: P, NP e NPC, sendo a última classe a dos problemas NP-Completos
- Aqui, as descrevemos de um modo informal; mais adiante, as definiremos em linguagem mais formal

Classe de complexidade de tempo

NP-Completude e as classes P e NP

- A classe P consiste nos problemas que podem ser resolvidos em tempo polinomial
- Mais especificamente, são problemas que podem ser resolvidos no tempo $O(n^k)$ para alguma constante k, onde n é o tamanho da entrada para o problema
- A maioria dos problemas examinados até agora na disciplina pertence à classe P

NP-Completude e as classes P e NP

- A classe NP consiste nos problemas que são "verificáveis" em tempo polinomial
- O que significa ser verificável? Se tivéssemos algum tipo de "certificado" de uma solução, poderíamos verificar se o certificado é correto em tempo polinomial para o tamanho da entrada para o problema
- Por exemplo, no problema do ciclo hamiltoniano, dado um grafo dirigido G=(V,E), um certificado seria uma sequência $< v_1, v_2, v_3, ..., v_{|V|} >$ de $|V|$ vértices
- É fácil verificar em tempo polinomial que $(v_i, v_{i+1}) \in E$ para i=1,2,3,...,$|V|$-1 e também que $(v_{|V|}, v_1) \in E$

NP-Completude e as classes P e NP

- Como outro exemplo; para satisfazibilidade 3-CNF um certificado seria uma atribuição de valores às variáveis
- Poderíamos verificar em tempo polinomial se essa atribuição satisfaz a fórmula booleana

Classe de complexidade de tempo

NP-Completude e as classes P e NP

- Qualquer problema em P também está em NP visto que, se um problema está em P, podemos resolvê-lo em tempo polinomial sem nem mesmo ter um certificado
- Logo, temos que $P \subseteq NP$
- A questão em aberto é se P é ou não um subconjunto próprio de NP

NP-Completude e as classes P e NP

- Informalmente, um problema está na classe NPC - e nos referiremos a ele como uma problema NP-Completo - se ele está em NP e é tão "difícil" quanto qualquer problema em NP
- Definiremos formalmente o que significa ser "tão difícil quanto qualquer problema em NP" mais adiante
- Por enquanto, afirmaremos sem provar que, se qualquer problema NP-Completo pode ser resolvido em tempo polinomial, então todo problema em NP tem um algoritmo de tempo polinomial

Classe de complexidade de tempo

NP-Completude e as classes P e NP

- A maioria dos teóricos da ciência da computação acredita que os problemas NP-completos sejam intratáveis já que, dada a ampla faixa de problemas NP-Completos que foram estudados até hoje - sem que ninguém tenha descoberto uma solução de tempo polinomial para nenhum deles - seria verdadeiramente espantoso se todos eles pudessem ser resolvidos em tempo polinomial
- Ainda assim, dado o esforço dedicado até agora para provar que os problemas NP-Completos são intratáveis - sem um resultado conclusivo - não podemos descartar a possibilidade de que os problemas NP-Completos são de fato resolvíveis em tempo polinomial

Classe de complexidade de tempo

NP-Completude e as classes P e NP

- Para se tornar um bom projetista de algoritmos, você deve entender os rudimentos da teoria da NP-Completude
- Se puder determinar que um problema é NP-Completo, estará dando uma boa evidência de sua intratabilidade
- Então, seria melhor empregar o seu tempo no desenvolvimento de um algoritmo de aproximação ou resolvendo um caso especial tratável, em vez de procurar um algoritmo rápido que resolva o problema exatamente

Como mostrar que um problema é NP-Completo: uma visão geral

- Quando dizemos que um problema é NP-Completo, fica subentendido que trata-se de um problema difícil de resolver, e não de um problema fácil de resolver
- Não estamos tentando provar a existência de um algoritmo eficiente, mas que provavelmente não existe um algoritmo eficiente
- Contamos com três conceitos fundamentais para mostrar que um problema é NP-Completo: Problemas de decisão versus problemas de otimização, Reduções e Um primeiro problema NP-Completo

Classe de complexidade de tempo

Problemas de decisão versus problemas de otimização

- Muitos problemas de interesse são problemas de otimização, para os quais cada solução possível ("válida") tem um valor associado e para os quais desejamos encontrar uma solução viável com o melhor valor
- Por exemplo, o problema do SHORTEST-PATH, onde desejamos encontrar, em um grafo não dirigido G e vértices u e v, o caminho de u a v que utiliza o menor número de arestas
- Porém, a NP-Completude não se aplica diretamente a problemas de otimização, mas a problemas de decisão, para os quais a resposta é simplesmente "sim" ou "não"

Classe de complexidade de tempo

Problemas de decisão versus problemas de otimização

- Embora problemas NP-Completos estejam confinados ao reino dos problemas de decisão, podemos tirar proveito de uma relação conveniente entre problemas de otimização e problemas de decisão
- Normalmente, podemos expressar um determinado problema de otimização como um problema de decisão relacionado impondo um limite para o valor a ser otimizado
- Por exemplo, um problema de decisão relacionado com SHORTEST-PATH é PATH: dado um grafo dirigido G, vértices u e v, e um inteiro k, existe um caminho de u a v que consiste em no máximo k arestas?

Classe de complexidade de tempo

Problemas de decisão versus problemas de otimização

- A relação entre um problema de otimização e seu problema de decisão relacionado age a nosso favor quando tentamos mostrar que o problema de otimização é "difícil"
- Isso porque o problema de decisão é de certo modo "mais fácil" ou, ao menos, "não é mais difícil"
- Como exemplo específico, podemos resolver PATH resolvendo SHORTEST-PATH e depois comparando o número de arestas no caminho mínimo encontrado com o valor do parâmetro k do problema de decisão

Classe de complexidade de tempo

Problemas de decisão versus problemas de otimização

- Em outras palavras, se um problema de otimização é fácil, seu problema de decisão relacionado também é fácil
- Em termos mais relevantes para a NP-Completude, se pudermos apresentar evidências de que um problema de decisão é difícil, também apresentamos evidências de que seu problema de otimização relacionado é difícil
- Assim, embora restrinja a atenção a problemas de decisão, muitas vezes, a teoria da NP-Completude também tem implicações para problemas de otimização

Reduções

- Essa ideia de mostrar que um problema não é mais difícil ou não é mais fácil que outro se aplica até mesmo quando ambos são problemas de decisão
- Tiramos proveito dessa ideia em quase todas as provas da NP-Completude, como veremos mais adiante
- Vamos considerar um problema de decisão A, que gostaríamos de resolver em tempo polinomial

Reduções

- Denominamos a entrada para um determinado problema por instância desse problema; por exemplo, em PATH, uma instância seria um grafo G dado, vértices específicos u e v de G e um determinado inteiro k
- Agora, suponha que já sabemos como resolver um problema de decisão diferente, B, em tempo polinomial
- Finalmente, suponha que temos um procedimento que transforma qualquer instância α de A em alguma instância β de B com as seguintes características

Classe de complexidade de tempo

Reduções

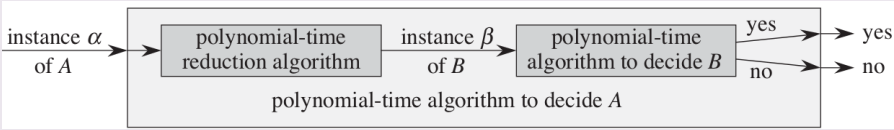

Figura: Fonte: Livro Algoritmos - Cormen

- A transformação demora tempo polinomial
- As respostas são as mesmas. Isto é, a resposta para α é "sim" se e somente se a resposta para β também é "sim"

Reduções

- Denominamos tal procedimento algoritmo de redução de tempo polinomial e, como mostra a figura anterior, ele proporciona um meio para resolver o problema A em tempo polinomial:
  1. Dada uma instância α do problema A, use um algoritmo de redução de tempo polinomial para transformá-la em uma instância β do problema B
  2. Execute o algoritmo de decisão de tempo polinomial para B para a instância β
  3. Use a resposta de β como a resposta para α

Classe de complexidade de tempo

Reduções

- Desde que cada uma dessas etapas demore tempo polinomial, as três juntas também demoram um tempo polinomial e, assim, temos um modo de decidir para α em tempo polinomial
- Em outras palavras, "reduzindo" a solução do problema A à solução do problema B, usamos a "facilidade" de B para provar a "facilidade" de A
- Lembrando que a NP-Completude consiste em mostrar o quanto um problema é difícil, em vez de mostrar o quanto ele é fácil,
- Usamos reduções de tempo polinomial ao contrário para mostrar que um problema é NP-Completo

Reduções

- Vamos avançar com essa ideia e mostrar como poderíamos usar reduções de tempo polinomial para demonstrar que não pode existir nenhum algoritmo de tempo polinomial, sem se preocupar, no momento, com a maneira de encontrar tal problema A
- Suponha ainda que tenhamos uma redução de tempo polinomial que transforma instâncias de A em instâncias de B
- Agora podemos usar uma prova simples por contradição para mostrar que não pode existir nenhum algoritmo de tempo polinomial para B
- Suponha o contrário, isto é, suponha que B tenha um algoritmo de tempo polinomial, o que contradiz nossa hipótese de inexistência de algoritmo polinomial para A

Classe de complexidade de tempo

Reduções

- Para a NP-Completude, não podemos assumir que não existe absolutamente nenhum algoritmo de tempo polinomial para o problema A
- Contudo, a metodologia da prova é semelhante no sentido de que provamos que o problema B é NP-completo considerando que o problema A também é NP-completo

Um primeiro problema NP-Completo

- Como a técnica de redução se baseia em ter um problema que já sabemos ser NP-completo para provar que um problema diferente é NP-completo, precisamos de um "primeiro" problema NP-completo
- O primeiro problema que usaremos é o da satisfazibilidade de circuitos, no qual temos um circuito computacional booleano composto por portas AND, OR e NOT, e desejamos saber se existe algum conjunto de entradas booleanas para esse circuito que faça sua saída ser 1
- Veremos que esse primeiro problema é NP-completo mais adiante

Próxima Aula

O que vem por aí?

- Redução polinomial: SAT, 3SAT e CLIQUE

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Classe de complexidade de tempo: Introdução/ P $\times$ NP

Teoria da Computação

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

06 de junho de 2023