UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Classe P, classe NP e NP-Completude

Teoria da Computação

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

13 de junho de 2022

A classe P

Tempo Polinomial

- Algoritmos de tempo exponencial surgem tipicamente quando resolvemos problemas por meio de busca exaustiva em um espaço de soluções, a denominada **busca pela força bruta**
- Exemplo: fatoração em primos
- Às vezes, a busca por força bruta pode ser evitada através de um entendimento mais profundo de um problema, que pode revelar um algoritmo de tempo polinomial de utilidade maior

A classe P

Exemplos de problemas em P

- CAM = $\{< G, s, t > |$G é um grafo direcionado que tem um caminho direcionado de s para t$\}$
- CAM está em P?

A classe P

Exemplos de problemas em P

- Prova: Um algoritmo de tempo polinomial M para CAM opera da seguinte forma
- M="Sobre uma entrada $< G, s, t >$ onde G é um grafo direcionado com nós s e t:
  1. Ponha uma marca sobre o nó s
  2. Repita o seguinte até que nenhum nó adicional seja marcado:
  3. Faça uma varredura em todas as arestas de G. Se uma aresta (a,b) for encontrada indo de um nó marcado a para um nó não marcado b, marque o nó b
  4. Se t estiver marcado, aceite. Caso contrário, rejeite"

A classe P

Exemplos de problemas em P

- Agora analisamos esse algoritmo para mostrar que ele roda em tempo polinomial
- Obviamente, os estágios 1 e 4 são executados apenas uma vez
- O estágio 3 roda no máximo m vezes, pois em cada vez, exceto a última, ele marca um nó adicional em G
- Por conseguinte, o número total de estágios usados é no máximo 1+1+m, dando um tempo polinomial no tamanho de G
- Logo, M é um algoritmo de tempo polinomial para CAM

A classe P

Exemplos de problemas em P

- Vamos ver outro exemplo
- Dizemos que dois números são primos entre si se 1 é o maior número inteiro que divide ambos
- Por exemplo, 10 e 21 são primos entre si, muito embora nenhum deles seja um número primo por si só
- 10 e 22 não são primos entre si porque ambos são divisíveis por 2
- Seja PRIM-ES o problema de se testar se dois números são primos entre si
- Portanto, PRIM-ES=$\{< x, y > \mid$ x e y são primos entre si$\}$

A classe P

Exemplos de problemas em P

- Prova: O algoritmo euclideano E é como segue
- E="Sobre a entrada <x,y>, onde x e y são números naturais em binário:
  1. Repita até que y=0:
  2. Atribua x $\leftarrow$ x mod y
  3. Intercambie x e y
  4. Dê como saída x"
- O algoritmo R resolve PRIM-ES, usando E como uma sub-rotina
- R="Sobre a entrada $< x, y >$, onde x e y são números naturais em binário:
  1. Rode E sobre $< x, y >$
  2. Se o resultado for 1, aceite. Caso contrário, rejeite"

Exemplos de problemas em P

- Claramente, se E roda em tempo polinomial, assim também o faz R; portanto, precisamos apenas analisar E com relação a tempo e correção
- A correção desse algoritmo é bem conhecida
- E quanto a complexidade?

A classe NP

Introdução

- Para alguns problemas, não se sabe se existem algoritmos de tempo polinomial que os resolvem
- Por que não tivemos tido sucesso em encontrar algoritmos de tempo polinomial para certos problemas?
- Não sabemos a resposta
- Talvez esses problemas possuam algoritmos de tempo polinomial ainda não descobertos e que se baseiem em princípios desconhecidos
- Ou possivelmente alguns desses problemas simplesmente não podem ser resolvidos em tempo polinomial
- Eles podem ser intrinsecamente difíceis

A classe NP

Introdução

- Uma descoberta notável concernente a essa questão mostra que as complexidades de muitos problemas estão interligadas
- Um algoritmo de tempo polinomial para um desses problemas pode ser usado para resolver uma classe inteira de problemas
- Para entender esse fenômeno, vamos começar com um exemplo

A classe NP

Caminho Hamiltoniano

- Um caminho hamiltoniano em um grafo direcionado G é um caminho direcionado que passa por cada nó exatamente uma vez
- Vamos considerar o problema de se testar se um grafo direcionado contém um caminho hamiltoniano conectando dois nós especificados, como podemos ver na figura no quadro
- Seja CAMHAM $= \{< G, s, t > \mid$ G é um grafo direcionado com um caminho hamiltoniano de s para t$\}$
- Podemos facilmente obter um algoritmo de tempo exponencial para o problema CAMHAM modificando o algoritmo de força bruta para CAM
- Precisamos apenas adicionar um teste para verificar que o caminho potencial é hamiltoniano
- Ninguém sabe se CAMHAM é solucionável em tempo polinomial

A classe NP

Caminho hamiltoniano

- O problema CAMHAM de fato tem uma característica chamada verificabilidade polinomial que é importante para entender sua complexidade
- Muito embora não conheçamos uma forma rápida (isto é, de tempo polinomial) de determinar se um grafo contém um caminho hamiltoniano, se este fosse descoberto de alguma forma (talvez usando um algoritmo de tempo exponencial) poderíamos facilmente convencer uma outra pessoa de sua existência, simplesmente apresentando-o
- Em outras palavras, **verificar** a existência de um caminho hamiltoniano pode ser muito mais fácil que **determinar** sua existência

A classe NP

Compostos

- Outro problema polinomialmente verificável é o problema de um número ser composto
- Lembre-se de que um número natural é composto se ele é o produto de dois números inteiros maiores que 1 (isto é, um número composto é aquele que não é um número primo)
- Seja COMPOSTOS=$\{x|$ x = pq, para inteiros p,q $> 1\}$
- Podemos facilmente verificar que um número é composto - tudo o que é necessário é um divisor desse número
- Recentemente, um algoritmo de tempo polinomial para testar se um número é primo ou composto foi descoberto, mas ele é consideravelmente mais complicado que o método precedente para verificar o problema

A classe NP

Verificador - Definição

Um verificador para uma linguagem A é um algoritmo V onde

$$A = \{w | \text{ V aceita } < w, c > \text{ para alguma cadeia c}\}$$

Medimos o tempo de um verificador em termos apenas do comprimento de w, portanto um verificador de tempo polinomial roda em tempo polinomial no comprimento de w. Uma linguagem A é polinomialmente verificável se ela tem um verificador de tempo polinomial.

A classe NP

Verificação em tempo polinomial

- Um verificador usa informação adicional, representada pelo símbolo c, para verificar que uma cadeia w é um membro de A
- Essa informação é chamada de certificado ou prova da pertinência a A
- Para o problema CAMHAM, um certificado para uma cadeia $< G, s, t > \in$ CAMHAM é simplesmente o caminho hamiltoniano de s a t
- Para o problema COMPOSTOS, um certificado para o número composto x é simplesmente um de seus divisores
- Em ambos os casos, o verificador pode checar em tempo polinomial que a entrada está na linguagem quando ela recebe o certificado

A classe NP

Definição - Classe NP

NP é a classe das linguagens que têm verificadores de tempo polinomial

A classe NP

Importância da classe NP

- A classe NP é importante porque contém muitos problemas de interesse prático
- O tempo NP vem de tempo polinomial não-determinístico
- Problemas em NP são às vezes chamados problemas NP

Verificador de tempo polinomial para CAMHAM

- A seguir está uma máquina de Turing não-determinística (MTN) que decide o problema CAMHAM em tempo polinomial não-determinístico
- N_1="Sobre a entrada $< G, s, t >$, onde G é um grafo direcionado com nós s e t:
  1. Escreva uma lista de m números, p_1,...,p_m, onde m é o número de nós em G. Cada número na lista é selecionado não-deterministicamente entre os números 1 a m
  2. Verifique se há repetições na lista. Se alguma for encontrada, rejeite
  3. Teste se s=p_1 e t=p_m. Se um dos testes falhar, rejeite
  4. Para cada i entre 1 e m-1, verifique se (p_i, p_{i+1}) é uma aresta de G. Se alguma não for, rejeite. Caso contrário, todos os testes foram positivos, portanto, aceite"
- Tempo polinomial?

A classe NP

Exemplos de problemas em NP

- Uma clique em um grafo não-direcionado é um subgrafo no qual todo par de nós está conectado por uma aresta
- Uma k-clique é uma clique que contém k nós
- No quadro vemos o exemplo de uma 5-clique
- O problema da clique é determinar se um grafo contém uma clique de um tamanho especificado
- Seja CLIQUE = $\{< G, k > \mid$ G é um grafo não-direcionado com uma k-clique$\}$

A classe NP

Exemplos de problemas em NP

- Vamos mostrar que CLIQUE está em NP
- A clique é o certificado
- Aqui está um verificador V para CLIQUE
- V="Sobre a entrada $<< G, k >, c >$:
  1. Teste se c é um conjunto de k nós em G
  2. Teste se G contém todas as arestas conectando nós em c
  3. Se ambos os testes retornam positivo, aceite; caso contrário, rejeite

Exemplos de problemas em NP

- Consideramos agora o problema SOMA-SUBC que vimos em Programação Dinâmica
- Nesse problema, temos uma coleção de números $x_1, ..., x_k$ e um número-alvo t
- Desejamos determinar se a coleção contém uma subcoleção que soma t
- SOMA-SUBC = $\{< S, t > |\text{S}=\{x_1, ..., x_k\}$ e para algum $\{y_1, ..., y_l\} \subseteq \{x_1, ..., x_k\}$, temos $\sum y_i = \text{t}\}$

Exemplos de problemas em NP

- Vamos mostrar que SUMA-SUBC está em NP
- O subconjunto é o certificado
- Prova: O que segue é um verificador V para SOMA-SUBC
- V="Sobre a entrada $<< S, t >, c >$:
  1. Teste se c é uma coleção de números que somam t
  2. Teste se S contém todos os números em c
  3. Se ambos os testes retornem, aceite; caso contrário, rejeite"

A classe P e a classe NP

Resumo

- P = a classe das linguagens para as quais pertinência pode ser decidida rapidamente
- NP = a classe das linguagens para as quais pertinência pode ser verificada rapidamente

Introdução

- Um avanço importante na questão P versus NP veio no início dos anos 1970 com o trabalho de Stephen Cook e Leonid Levin
- Eles descobriram certos problemas em NP cuja complexidade individual está relacionada àquela da classe inteira
- Se existe um algoritmo de tempo polinomial para quaisquer desses problemas, todos os problemas em NP seriam solúveis em tempo polinomial
- Esses problemas são chamados NP-completos

SAT

- O primeiro problema NP-completo que apresentamos é chamado problema da satisfazibilidade (SAT, para os íntimos)
- Lembre-se de que variáveis que podem tomar os valores VERDADEIRO ou FALSO são chamadas variáveis booleanas
- Geralmente representamos VERDADEIRO por 1 e FALSO por 0
- As operações booleanas E, OU e NÃO, representadas pelos símbolos $\wedge$, $\vee$ e $\neg$, respectivamente
- Usamos a barra superior como uma abreviação para o símbolo $\neg$, portanto $\overline{x}$ significa $\neg x$
- Uma fórmula boolena é uma expressão envolvendo variáveis booleanas e operações, por exemplo

$$\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$$

SAT

- Uma fórmula booleana é satisfazível se alguma atribuição de 0s e 1s às variáveis faz a fórmula ter valor 1
- A fórmula precedente é satisfazível porque a atribuição x=0, y=1 e z=0 faz ϕ ter valor 1
- Dizemos que a atribuição satisfaz ϕ
- O problema da satisfazibilidade é testar se uma fórmula booleana é satisfazível. Seja

$$SAT = \{< \phi > | \phi \text{ é uma fórmula booleana satisfazível } \}$$

Teorema de Cook-Levin

SAT é NP-Completo

- Ideia da prova: Mostrar que SAT está em NP é fácil
- A parte difícil da prova é mostrar que qualquer linguagem em NP é redutível em tempo polinomial a SAT
- Para fazer isso, construímos uma redução de tempo polinomial para cada linguagem A em NP para SAT
- A redução para A toma uma cadeia w e produz uma fórmula boolena ϕ que simula a máquina NP para A sobre a entrada w
- Se a máquina aceita, ϕ tem uma atribuição que a satisfaz que corresponde à computação de aceitação
- Se a máquina não aceita, nenhuma atribuição satisfaz ϕ
- Consequentemente, w está em A se e somente se ϕ é satisfazível

- Construir realmente a redução para funcionar dessa maneira é uma tarefa conceitualmente simples, embora devamos lidar com muitos detalhes
- Uma fórmula booleana pode conter as operações booleanas E, OU, NÃO e essas operações formam a base para os circuitos usados em computadores eletrônicos
- Logo, o fato de que podemos projetar uma fórmula booleana para simular uma máquina de Turing não é surpreendente

Redutibilidade em tempo polinomial

- Uma redução é uma maneira de converter um problema em outro de forma que uma solução para o segundo problema possa ser usada para resolver o primeiro
- Essas redutibilidades aparecem frequentemente no dia-a-dia, mesmo que em geral não nos referimos a elas dessa forma
- Por exemplo, suponha que você deseje se orientar em uma nova cidade
- Você sabe que seria fácil fazer isso se tivesse um mapa
- Consequentemente, você pode reduzir o problema de se orientar na cidade ao problema de se obter um mapa da cidade

Redutibilidade em tempo polinomial

- A redutibilidade sempre envolve dois problemas, que denominamos A e B
- Se A se reduz a B, podemos usar uma solução para B para resolver A
- Assim, em nosso exemplo, A é o problema de se orientar na cidade e B é o problema de se obter um mapa
- Note que redutibilidade não diz nada sobre resolver A ou B sozinhos, mas somente sobre a solubilidade de A na presença de uma solução para B

Redutibilidade em tempo polinomial

- Outro exemplo: o problema de se viajar de Boston a Paris se reduz ao problema de se comprar uma passagem aérea entre as duas cidades
- Esse problema, por sua vez, se reduz ao problema de se ganhar dinheiro para a passagem
- E esse último problema se reduz ao problema de se encontrar um emprego

Redutibilidade em tempo polinomial

- A redutibilidade também ocorre em problemas matemáticos
- Por exemplo, o problema de se medir a área de um retângulo se reduz ao problema de se medir seu comprimento e largura
- O problema de se resolver um sistema de equações lineares se reduz ao problema de se inverter uma matriz
- Quando A é redutível a B, resolver A não pode ser mais difícil que resolver B, porque uma solução para B dá uma solução para A

Teorema

Se $A \leq_P B$ e $B \in P$, então $A \in P$

Prova

- Seja M o algoritmo de tempo polinomial que decide B e f a redução de tempo polinomial de A para B
- Descrevemos um algoritmo de tempo polinomial N que decide A da seguinte forma
- N="Sobre a entrada w:
  1. Compute f(w)
  2. Rode M sobre a entrada f(w) e dê como saída o que M der como saída"

Redutibilidade em tempo polinomial

- Temos $w \in A$ sempre que $f(w) \in B$ porque f é uma redução de A para B
- Por conseguinte, M aceita f(w) sempre que w $\in$ A
- Além do mais, N roda em tempo polinomial
- Note que o estágio 2 roda em tempo polinomial porque a composição de polinômios é um polinômio

Próxima Aula

O que vem por aí?

- 3SAT
- CLIQUE

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Classe P, classe NP e NP-Completude

Teoria da Computação

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

13 de junho de 2022