DCC207 – Algoritmos 2

Aula 08 – Introdução a Teoria da Complexidade (Parte 02)

Professor Renato Vimieiro

DCC/ICEx/UFMG

Introdução

- Na aula passada, vimos três classes de complexidade de tempo em que problemas podiam ser classificados
- Vimos que a classe dos problemas NP são aqueles que possuíam soluções não-determinísticas polinomiais
- Essas podiam ser transformadas em determinísticas com um custo exponencial, através da simulação da computação de uma MT não-determinística por uma determinística
- Uma pergunta que surge é: todos os problemas de uma mesma classe possuem a mesma dificuldade?
- Nessa aula, veremos como certos problemas resumem a dificuldade da classe:
  - Alguns problemas dessa classe são pelo menos tão difíceis quanto qualquer outro problema dela
- Esse conceito está apoiado na definição de redução de problemas
  - Focaremos no caso particular de redução polinomial entre problemas NP

Redução de problemas

- A ideia de se reduzir um problema (A) a outro (B) consiste na computação de uma função f: A -> B que:
  - Dada uma entrada do problema A, a converte para uma do problema B
  - Existe solução para o problema A, se existe solução para o problema f(A)
- Do ponto de vista de paradigmas de desenho de algoritmos, essa técnica pode ser usada também na construção de algoritmos
  - Reduz-se um problema para o qual não se conhece uma solução a outro que possua solução conhecida
- Em outras palavras, essa técnica possui utilidade tanto prática quanto teórica

Redução polinomial de problemas

- Uma linguagem (problema) A é redutível a outra linguagem B em tempo polinomial, $A \leq_p B$, se existe uma função f: A -> B, computável deterministicamente em tempo polinomial, tal que, para toda palavra w:
  - $w \in A \longleftrightarrow f(w) \in B$
- Teorema: Se $A \leq_p B$ e $B \in P$, então $A \in P$.
  - Prova: Segue direto da definição de redução.
- Exemplo: $3SAT \leq_p CLIQUE$

Redução polinomial de problemas

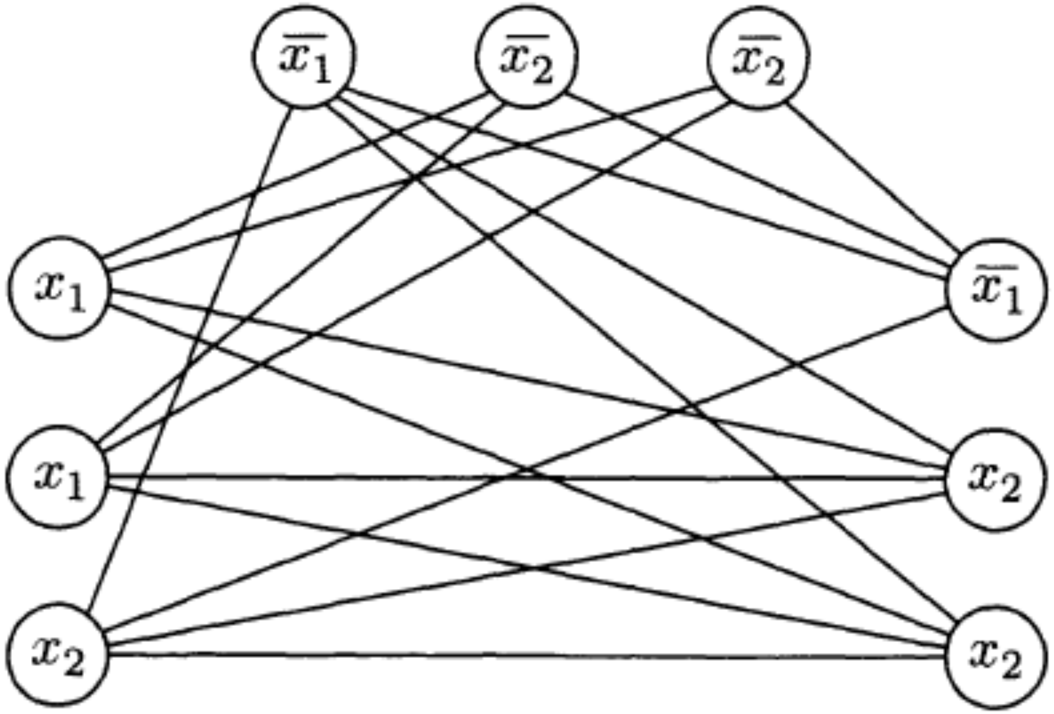
- Uma fórmula/função booleana é composta por literais (variáveis booleana ou sua negação) conectadas por operadores booleanos (conjunção/disjunção)
  - $f(x,y,z) = (x \land y) \lor (y \land \neg z)$
- Uma função booleana está na forma normal conjuntiva (CNF) se ela é composta pela conjunção de max-termos
  - Um max-termo é a disjunção (soma) de literais
- O problema do 3SAT consiste em determinar se uma função booleana na CNF em que cada max-termo tem tamanho 3 é satisfazível
  - Isto é, admite-se uma atribuição de valores às variáveis de forma que a função avalie para 1
  - $f(x,y,z) = (x \lor y \lor z) \land (\neg x \lor y \lor \neg z)$
  - A função avalia para 1 com $f(x=1,y=1,z=0)$

Redução polinomial de problemas

- Teorema: O problema 3SAT é redutível em tempo polinomial ao problema de CLIQUE
- Prova: Para demonstrar o teorema, precisamos descrever o algoritmo polinomial que mapeie um problema ao outro.
- Considere uma função booleana f na CNF, com k max-termos de tamanho 3.
  - $f = \bigwedge (a_i \vee b_i \vee c_i)$
- Dessa função, geramos o seguinte grafo:
  - V = {x | x é um literal de algum max-termo}
  - E = {{a,b} | a e b pertencem a max-termos distintos, e a não é o complemento de b}

Redução polinomial de problemas

- $\phi = (x_1 \vee x_1 \vee x_2) \ \wedge \ (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \ \wedge \ (\overline{x_1} \vee x_2 \vee x_2)$



Redução polinomial de problemas

- Note que uma solução para o 3SAT contém uma clique nesse grafo
  - Um vértice de cada max-termo é escolhido
  - Como um literal e seu complemento não podem ambos ser verdadeiros ao mesmo tempo, somente um dos dois consta na solução. Portanto, todos os literais escolhidos estão conectados entre si.
- Da mesma forma, uma k-clique nesse grafo é solução para o 3SAT. Basta atribuir verdadeiro a cada um dos literais escolhidos
- Como a construção do grafo pode ser feita em tempo $O(n^2)$, 3SAT $\leq_p$ CLIQUE
- Assim, podemos resolver 3SAT em tempo polinomial de forma não determinística

Problemas NP-Completos

- Como discutimos, existem problemas na classe NP que são pelo menos tão difíceis quanto qualquer outro da classe.
- Esses problemas são conhecidos como problemas NP-completos
- Formalmente, um problema B é NP-completo se:
  - $B \in NP$
  - Todo problema $A \in NP$ é redutível em tempo polinomial a B, $A \leq_p B$. (**NP-Difícil**)
- Teorema: Se B é NP-completo e $B \in P$, então P=NP.
- Prova: Segue da definição de redução.

Problemas NP-completos

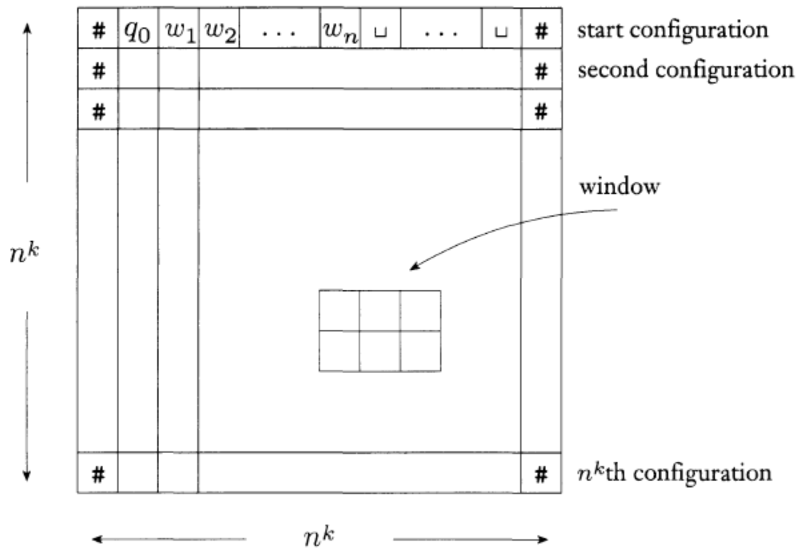
- Um corolário do teorema anterior é que, se A é NP-completo, B $\in$ NP, e A $\leq_p$ B, então B também é NP-completo.
- Esse corolário permite demonstrar que um problema é NP-Completo a partir de um outro conhecido.
- No entanto, como determinar o 'primeiro' problema NP-Completo, a partir do qual os outros podem ser determinados?
- Esse é o teorema postulado, de forma independente, pelos cientistas da computação Stephen Cook (americano-canadense) e Leonid Levin (ucraniano-soviético) na década de 1970

Teorema de Cook-Levin

- Teorema (Cook-Levin): SAT é NP-Completo.
- Prova: Discutiremos somente a intuição.
- A demonstração de que SAT $\in$ NP é relativamente fácil de ser obtida. Podemos construir uma MT que escolha de forma não-determinística um subconjunto de literais a serem marcados como verdadeiro. Depois, a máquina verifica se a função avalia a 1, retornando verdadeiro/falso.
- A segunda parte, de que todo problema pode ser reduzido em tempo polinomial a SAT, é consideravelmente mais complexa.
- A ideia da prova é mostrar como as computações da MT não-determinística podem ser modeladas como funções booleana.

Teorema de Cook-Levin

- Seja A ∈ NP e a MT M sua solução não-determinística polinomial (ela existe pela definição de NP)
- Suponha que M execute suas computações em tempo n^k
- Podemos representar as computações do ramo que aceita ou rejeita uma palavra em forma tabular como:



Teorema de Cook-Levin

- M termina a computação em estado de aceitação se em alguma linha da matriz aparece algum estado final
- Analogamente, a primeira linha deve conter a configuração inicial da máquina
- Assim criamos uma função que, quando satisfeita, garanta que cada linha da matriz contenha uma configuração instantânea válida da máquina; ela inicie na configuração inicial; execute transições (mude de uma configuração para outra conforme sua função de transição); e termine num estado de aceitação
  - $f(A) = f_{matriz} \wedge f_{inicial} \wedge f_{transição} \wedge f_{final}$

Teorema de Cook-Levin

- f_{matriz} = $\displaystyle\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{\substack{s,t\in C\\ s\ne t}}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$
- $f_{inicial}$ = $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge$
  $x_{1,3,w_1}\wedge x_{1,4,w_2}\wedge\ldots\wedge x_{1,n+2,w_n}\wedge$
  $x_{1,n+3,\sqcup}\wedge\ldots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
- $f_{transição}$ = $\displaystyle\bigwedge_{1\le i<n^k,\ 1<j<n^k}$ (the (i,j)-window is legal)
- f_{final} = $\displaystyle\bigvee_{1\le i,j\le n^k} x_{i,j,q_{\text{accept}}}$

Teorema de Cook-Levin

- As transições entre configurações da máquina usam somente informações locais
- Pode-se usar uma janela 2x3 para verificar se a mudança de uma configuração (linha da matriz) para outra é válida
- Considere: δ(q1,a) = {(q1,b,D)}; e δ(q1,b) = {(q2,c,E),(q2,a,D)}

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

Transições legais

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

Transições ilegais

Teorema de Cook-Levin

- Note que todas as funções são montadas sobre células da tabela
- Assim, o número total de literais será da ordem $O(n^{2k})$; ou seja, ele é polinomial. Portanto, a redução executada dessa forma será realizada em tempo polinomial.
- Como descrevemos um mecanismo genérico para reduzir a computação de uma MT não-determinística a uma função booleana, qualquer problema NP poderá ser reduzido a SAT.

3SAT é NP-completo

- Uma forma de demonstrar que 3SAT é NP-completo é reduzir polinomialmente SAT a ele
- Essa tarefa não é exatamente difícil já que toda função booleana possui uma equivalente na CNF
- A partir daí poderíamos restringir que cada max-termo tivesse exatamente 3 literais
- Alternativamente, podemos impor tal restrição já na construção da função booleana obtida na prova do teorema de Cook-Levin
- Uma expressão (a1 ∨ a2 ∨ a3 ∨ a4) pode ser reescrita como:
  - (a1 ∨ a2 ∨ z) ∧ (¬z ∨ a3 ∨ a4)
  - No caso geral, (a1 ∨ a2 ∨ ... ∨ ak) -> (a1 ∨ a2 ∨ z1) ∧ (¬z1 ∨ a3 ∨ z2) ∧ (¬z2 ∨ a4 ∨ z3) ∧ ... (¬zk-3 ∨ ak-1 ∨ ak)
- Essa transformação introduz uma quantidade linear de novos literais, e, portanto, mantém a complexidade da redução polinomial

CLIQUE é NP-completo

- A redução que fizemos mais cedo de 3SAT a CLIQUE prova que o último também é NP-completo
- Como 3SAT é NP-completo, por transitividade da redução polinomial (a composição de duas reduções polinomiais continua sendo polinomial), CLIQUE também é NP-completo

Leitura

- Seções 7.4 (Introduction to the Theory of Computation, 2nd ed., Michael Sipser)

DCC207 – Algoritmos 2

Aula 08 – Introdução a Teoria da Complexidade (Parte 02)

Professor Renato Vimieiro

DCC/ICEx/UFMG