ITC: Introdução à Teoria da Computação

Marcos Castilho

DInf/UFPR

4 de agosto de 2021

Complexidade computacional

- Existem problemas que são solúveis apenas teoricamente, mas na prática, demoram demais e não tem efetivamente uma solução em tempo razoável.
- Os problemas que tem solução em tempo razoável são denominados de tratáveis, os outros de intratáveis. Tratabilidade é um tema importante.
- Podemos classificar os problemas por sua similaridade e definir classes de complexidade.

Complexidade computacional

- Existem vários problemas para os quais até hoje ninguém apresentou um algoritmo (ou uma MT) que possa ser resolvido em tempo polinomial.
- Exemplos: SAT, circuito Hamiltoniano, etc.
- Alguns destes problemas possuem soluções polinomiais se usarmos MT's não determinísticas.
- Vamos estudar o relacionamento entre soluções que usam MT's determinísticas e não determinísticas.
- Não sabemos se um problema que pode ser resolvido polinomialmente por uma MT não determinística também pode ser por outra determinística.
- Esta problemática é a principal questão em aberto em teoria da computação.

Problemas de decisão tratáveis e intratáveis

- Uma linguagem L é decidível em tempo polinomial se existe um algoritmo que determina pertinência em L para a qual o tempo requerido por uma computação cresce polinomialmente com o tamanho da entrada.
- Como existem muitos algoritmos que resolvem o mesmo problema, consideramos que a complexidade de tempo para L é a taxa de crescimento mais eficiente conhecida que resolve o problema.
- Por exemplo, para a linguagem dos palíndromes que estudamos, conseguimos uma MT padrão que aceita a linguagem em tempo $O(n^2)$.

Definição

Uma linguagem L é decidível em tempo polinomial se existe uma MT padrão M que aceita L com $tc_M(n) = O(n^r)$, onde r é um número independente de n.

- A família de linguagens decidíveis em tempo polinomial é denotada $\mathcal{P}$.

Observações

- Teoria de computabilidade é o estudo de quando problemas de decisão são solúveis teoreticamente.
- Teoria de complexidade estuda se é possível distinguir problemas que são solúveis na prática daqueles que são solúveis apenas em teoria.
- Problemas podem não ser solúveis na prática por limitações de recursos, tais como memória ou tempo de processamento.
- Problemas para os quais não há algoritmos eficientes são ditos serem intratáveis.
- A questão é decidir se um problema pode ser resolvido em tempo razoável, mesmo para casos onde a entrada é grande. A classificação entre tempo polinomial e tempo não polinomial é cruscial aqui.

A classe $\mathcal{P}$

- Esta classe foi definida em termos de complexidade de tempo de uma implementação em uma MT padrão, embora outros modelos de MT's poderiam ter sido utilizados.
- Isto por causa dos teoremas vistos anteriormente, isto é, o fator polinomial não se perde quando se usam diferentes tipos de MT's.

Complicação

- A forma de se representar a entrada é determinante neste estudo, pois o tamanho da entrada é relacionado com a complexidade de tempo, conforme definimos.
- Por exemplo, se a entrada for um natural n vai requerer que se definam n transições.
- Usando-se representação unária, o número de transições é linear com respeito ao tamanho da entrada.
- Mas se a representação de n for binária, a entrada tem tamanho $\log n$. Consequentemente, a função que relaciona o tamanho da entrada ao número de transições é exponencial.
- Soluções pseudo-polinomiais são esses que usam representação binária em tempo polinomial, mas que para a representação unária é polinomial.

A classe $\mathcal{NP}$

- A computação em uma MT não determinística resolve um problema de decisão examinando uma das possíveis soluções.
- A habilidade de não deterministicamente selecionar uma solução particular, ao invés de analisar todas as possíveis soluções, evidentemente reduz a complexidade da computação na MT não determinística.

Definição

Uma linguagem L é dita ser aceita em tempo polinomial não determinístico se existe uma MT M não determinística que aceita L com $tc_M = O(n^r)$, onde r é um natural independente de n.

- A família de linguagens aceitas em tempo polinomial não determinístico é denotada $\mathcal{NP}$.

A classe $\mathcal{NP}$

- A família $\mathcal{NP}$ é um subconjunto das linguagens recursivas.
- De fato, como existe um limite polinomial no número de transições, a computação eventualmente termina.
- Evidentemente, como uma MT determinística é também não determinística, $\mathcal{P} \subseteq \mathcal{NP}$.
- A questão de 1 milhão de dólares é saber se $\mathcal{NP} \subseteq \mathcal{P}$.

Exemplo: circuito Hamiltoniano

- Seja G um grafo direcionado com n vértices numerados de 1 até n.
- Uma aresta do vértice i ao vértice j é representado por um par ordenado $[i, j]$. Um circuito Hamiltoniano é um caminho $i_0, i_1, \ldots, i_n$, em G que satisfaz:
  - $i_0 = i_n$;
  - $i_i \neq i_j$, quando $i \neq j$ e $0 \leq i, j < n$.
- Isto é, um circuito Hamiltoniano é um caminho que visita cada vértice exatamente uma vez e que termina no ponto de partida, isto é, se existe um passeio do primeiro vértice até ele mesmo passando por todos os outros uma única vez.
- O problema do circuito Hamiltoniano é saber se existe um passeio em um grafo G.

MT determinística

- Seje G um grafo direcionado com n vértices. Construímos uma MT com 4 fitas que resolve o problema assim:
- Um vértice do grafo é denotado em sua representação binária.
- O alfabeto da representação é $\{0, 1, \#\}$
- Um grafo com n vértices e m arestas é representado pela entrada:

$$\bar{x_1}\#\bar{y_1}\#\# \ldots \#\#\bar{x_m}\#\bar{y_m}\#\#\#\bar{n}$$

onde $[x_i, y_i]$ são as arestas e $\bar{x}$ denota a representação binária de x.

Exemplo, continuação

- A fita 1 vai manter a representação dos arcos.
- A computação vai gerar e examinar sequências de $n+1$ vértices $1, i_1, \ldots, i_{n-1}$ para determinar se formam um ciclo.
- As sequências são geradas em ordem numérica na fita 2.
- A representação da sequência $1, n, \ldots, n, 1$ é escrita na fita 4 e usada para disparar a condição de parada (usamos uma MT que gera isso para provar que uma MT não determinística é equivalente à uma determinística).

Como é a computação?

- É um laço que:
  - Gera uma sequência $B\bar{1}B\bar{i_1}B\bar{i_2}B \ldots B\bar{i_{n-1}}B\bar{1}B$ na fita 2.
  - Para se as fitas 2 e 4 forem idênticas.
  - Examina a sequência $1, i_1, \ldots, i_{n-1}, 1$ e para se ela for um passeio no grafo.

Exemplo, continuação

- Se a computação para no passo 2, todas as sequências foram examinadas e o grafo não contém um circuto Hamiltoniano.

Exemplo, continuação

A análise no passo 3 começa com a máquina com a configuração seguinte:

Fita 1 $B\bar{1}(B\bar{n})^{n-1}B\bar{1}B$

Fita 2 $B\bar{1}B$

Fita 3 $B\bar{1}B\bar{i_1}B\ldots B\bar{i_{n-1}}B\bar{1}B$

Fita 4 $B\bar{x_1}\#\bar{y_1}\#\#\ldots\#\#\bar{x_m}\#\bar{y_m}\#\#\#\bar{n}B$

Exemplo, continuação

- Os vértices $i_1, \ldots, i_{n-1}$ são sequencialmente analisados.
- Um vértice i_j é adicionado à sequência na fita 3 se:
  - $i_j \neq 1$
  - $i_j \neq i_k$, para $1 \leq k \leq j-1$
  - existe um arco $[i_{j-1}, i_j]$ representado na fita 1.
- Isto é, se $1, i_1, \ldots, i_j$ é um caminho acíclico no grafo.
- Se todo vértice i_j, $j = i, \ldots, n-1$, na sequência na fita 2 é adiconado na fita 3 e existe uma aresta de i_{n-1} para 1 o caminho na fita 2 é um passeio e a computação aceita a entrada.

Finalmente (quase)

- O algoritmo acima é exponencial.
- Uma computação examina e rejeita cada sequência quando a entrada no grafo não contém um passeio.
- Para um grafo com n vértices, existem n^{n-1} sequências.

Finalmente (agora sim)

- O número de sequências cresce exponencialmente com o número de vértices no grafo.
- Como usamos a representação binária, aumentar o número de vértices para $2n$, sem acrescentar arestas, aumenta o tamanho da entrada em um único símbolo.
- Assim, aumentar o tamanho da entrada provoca que o número de possibilidades que tem que ser examinadas aumenta exponencialmente.

Observações

- Na MT utilizada, provamos que o problema é solúvel em tempo exponencial,
- mas isso não quer dizer que ele não possa ser revolvido em tempo polinomial...
- Ou a solução polinomial não existe, ou não encontramos um algoritmo polinomial...

Versão não determinística

- Podemos obter uma solução em tempo polinomial com uma MT não determinística com 3 fitas.
- A quarta fita, que é usada para terminar a computação quando o grafo não contém um passeio, não é necessária.

Continuação

A computação

- Para e rejeita a entrada quando existem menos de $n+1$ arestas no grafo.
- Não deterministicamente gera uma sequência $1, i_1, \ldots, i_{n-1}, 1$ na fita 2
- Usa as fitas 1 e 3 para determinar quando a sequência na fita 2 define um passeio.

Continuação

- Esta MT não determinística é polinomial.
- De fato, construímos um limite superior no número de transições da computação.
- O máximo número de transições ocorre quando a palavra define um passeio.
- Senão, a computação termina sem examinar cada nodo representado na fita 2.
- Como os nodos são representados em binário, a máxima quantidade de fita necessária para codificar qualquer nodo é $logn + 1$.

Licença

- Slides feitos em LaTeX usando beamer e tikz, editados com vim.
- Licença