Teoria da Computação
Complexidade de Tempo

Leonardo Takuno
{leonardo.takuno@gmail.com}

Centro Universitário Senac

Sumário

Sumário

Teoria da complexidade

Até agora investigamos se um problema é em princípio solucionável algoritmicamente, isto é, questionamos se uma linguagem particular é,

- Decidível: se a máquina pára sobre todas as entradas
- Turing-Reconhecíveis: se a máquina entra em loop sobre alguma entrada (funções parcialmente computáveis)

Teoria da complexidade

Entretanto, nós não investigamos o custo da computação - a quantidade de recursos que a computação utiliza (tempo, espaço, etc.).
A seguir, discutimos a complexidade de tempo e de espaço. Além disso, assumimos o tratamento de funções computáveis, isto é, a linguagem decidível.

- Se nós podemos resolver um certo problema P, quão fácil ou quão difícil é fazê-lo?
- Teoria da complexidade tenta resolver esta questão.

Sumário

Contagem de Recursos

Teoria da Complexidade é a 'arte de contar recursos'

- **Teoria da Complexidade:** Estudo de problemas computacionalmente viáveis (tratáveis) com recursos limitados.
- **Complexidade de Tempo:** Quanto tempo é necessário para resolver a instância de um problema?
- **Complexidade de Espaço:** Quantos bits de memória são necessários para a computação ?

Contagem de Recursos

Ex: $A = \{0^n1^n | n \geq 0\}$

- Claramente, a linguagem é decidível.
- Quanto tempo uma máquina de Turing precisa para decidí-la?

Contagem de Recursos

Exemplo: Dada a linguagem decidível $A = \{0^k 1^k | k \geq 0\}$ e uma MT M1 que a decide, onde
M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e rejeite se um 0 é encontrado à direita de algum 1.
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1.
4. Se nem 0s nem 1s restarem sobre a fita *aceite*, caso contrário *rejeite*"

Questão

- Quanto tempo MT M1 precisa para executar dada uma entrada?
- O número de passos podem depender de uma série de parâmetros.

Questão

- Exemplo: Se a entrada fosse um grafo, poderia depender do número de
  - Nós
  - Arestas
  - Grau máximo
  - Todos, ou alguns fatores combinados.

Sumário

Complexidade de Tempo

Definição: Seja M uma MT determinística que pára sobre todas as entradas. O **tempo de execução** ou **complexidade de tempo** de M é a função

$$f : N \rightarrow N,$$

onde $f(n)$ é o número de passos que M usa sobre qualquer entrada de tamanho n.
Se $f(n)$ é o tempo de execução de M, então dizemos que M executa em tempo $f(n)$ e que M é uma MT de tempo $f(n)$.
Como de costume usaremos n para representar o tamanho da entrada.

Complexidade de Tempo

Nossa complexidade de tempo é chamado de **análise de pior caso** pois consideraremos somente o **número máximo de passos** que uma máquina usa para uma entrada n.

Sumário

Notação O-grande

Definição: Sejam f e g funções $f, g : N \rightarrow R^{+}$. Dizemos que $f(n) = O(g(n))$ se existirem inteiros positivos c e n_0 tal que para todo inteiro $n \geq n_0$,

$$f(n) \leq cg(n).$$

Quando $f(n) = O(g(n))$ dizemos que $g(n)$ é um **limitante superior (assintótico)** para $f(n)$.

Notação O-grande (Exemplo)

Seja $f(n) = 5n^3 + 2n^2 + 22n + 6$, então só considerando o termo de mais alta ordem e desconsiderando todas as constantes e coeficientes temos:

$$f(n) = O(n^3).$$

Podemos mostrar que isto satisfaz nossa definição formal de análise assintótica com $c = 6$ e $n_0 = 10$. Então para qualquer $n > 10$ temos $f(n) \leq 6n^3$.

Notação O-grande

- Seja $f(n) = 3n\log_2 n + 5n + 3$, então $f(n) = O(n \log n)$. Note que eliminamos a base pois $\log_b n = \frac{1}{\log_2 b} \log_2 n$ para qualquer base b, ou seja, diferentes logaritmos estão relacionados por um fator constante.
- $f(n) = O(n^2) + O(n) \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{O(n)} \Rightarrow f(n) \leq 2^{cn}$ para algum c e algum valor n_0 tal que $n > n_0$

Notação O-grande

- Limitantes da forma $O(n^k)$ onde $k > 0$ são chamados **limitantes polinomiais**.
- Limitantes da forma $2^{O(n^k)}$ onde $k > 0$ são chamados **limitantes exponenciais**.

Analisando algoritmos

Dado a linguagem $A = \{0^k 1^k | k \geq 0\}$ e a MT M1 que a decide, onde

M1 $=$ "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e rejeite se um 0 é encontrado à direita de algum 1.
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1.
4. Se nem 0s nem 1s restarem sobre a fita *aceite*, caso contrário *rejeite*"

Analisando algoritmos

Para analizar a complexidade de tempo dessa máquina analisamos separadamente cada estágio.

- **estágio 1**: A máquina varre a fita para verificar que a entrada é da forma 0^*1^*. Executando esta varredura usa n passos onde n é o tamanho da entrada. Reposicionando a cabeça para o início da fita que toma outros n passos. Portanto, este estágio toma $2n$ ou $O(n)$ passos

Analisando algoritmos

- **estágio 2,3**: Aqui a máquina varre a entrada repetidamente. Cada varredura toma $O(n)$. Como cada varredura corta dois símbolos por vez, ocorrem pelo menos $n/2$ varreduras. Ou seja $(n/2)O(n) = O(n^2)$

Analisando algoritmos

- **estágio 4**: A máquina faz uma simples varredura para decidir se aceita ou rejeita - $O(n)$ passos.

Analisando algoritmos

A complexidade tempo total de M1 sobre a entrada n é $2O(n) + O(n^2) = O(n^2)$. Podemos encontrar um algoritmo ou modelo computacional mais rápido?

Analisando algoritmos

Considere a máquina de 2-fitas M2
M2 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita 1 e rejeite se um 0 é encontrado à direita de algum 1.
2. Faça uma varredura nos 0s sobre a fita 1 até o primeiro 1. Ao mesmo tempo, copie os 0s para a fita 2.
3. Faça uma varredura nos 1s sobre a fita 1 até o final da entrada. Para cada 1 lido sobre a fita 1, corte um 0 sobre a fita 2. Se todos os 0s estiverem cortados antes que todos os 1s sejam lidos, rejeite.
4. Se todos os 0s tiverem sido cortados, aceite. Se restar algum 0, rejeite."

Analisando algoritmos

É fácil ver que cada estágio da máquina toma $O(n)$ passos - complexidade de tempo é $O(n)$ ou linear!

Analisando algoritmos

- É interessante notar que mesmo que a computabilidade não dependa do modelo preciso de computação escolhido - a complexidade de tempo depende!
- Vimos que ambos M1 e M2, decidem a linguagem A, mas M1 decide com complexidade de tempo $O(n^2)$ e M2 decide a linguagem em complexidade $O(n)$.

Notação o-pequeno

Definição: Sejam f e g funções $f, g : N \rightarrow R^+$. Dizemos que $f(n) = o(g(n))$ se

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Em outras palavras, $f(n) = o(g(n))$ significa que, para qualquer número real $c > 0$, existe um número n_0, onde $f(n) < cg(n)$ para $n \geq n_0$.

Na notação O-grande a relação é "menor ou igual" enquanto a notação o-pequeno a relação é "estritamente menor".

Notação o-pequeno

- Limite superior não justo.
- $2n = o(n^2)$, mas $2n^2 \neq o(n^2)$
- É fácil observar que:
    - se $f(n) = o(g(n))$ então $f(n) = O(g(n))$

Notação o-pequeno (Exemplo)

Mostre que $2n$ é $o(n^2)$.
De acordo com a nossa definição da notação o, nós temos que encontrar uma constante real c e uma constante inteira n_0 tais que $c > 0$ e $n_0 \geq 1$, e

$$2n < c \cdot n^2$$

para todo $n \geq n_0$.
Intuitivamente, na notação o, a função $f(n)$ torna-se insignificante em relação a $g(n)$ quando n vai para o infinito.

$$\lim_{n \to \infty} \frac{2n}{n^2} = \lim_{n \to \infty} \frac{2}{n} = 0$$

Portanto, $n = o(n^2)$

Sumário

Relacionamento entre modelos de MT (única fita vs multifita)

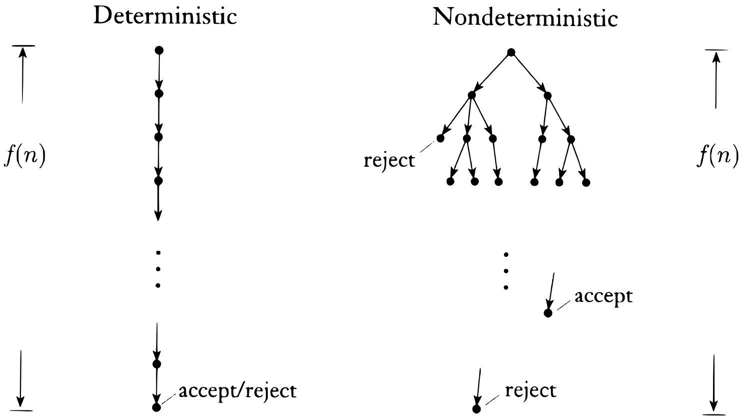
Teorema: Seja $t(n)$ uma função, onde $t(n) \geq n$. Então toda máquina de Turing multifita de tempo $t(n)$ tem uma máquina de Turing de uma única fita equivalente de tempo $O(t^2(n))$.

Idéia da prova: É possível mostrar que a simulação de cada passo de computação da máquina de multifita em uma máquina de uma máquina de fita única usa, no máximo, $O(t(n))$ passos. Portanto, para simular a computação completa de uma máquina multifita em uma máquina de fita única levaria $O(t^2(n))$ passos.

Complexidade de tempo não-determinístico

Definição: Seja N uma máquina de Turing não-determinística decisora. O **tempo de execução** de N é a função $f : N \to N$, onde $f(n)$ é o número máximo de passos que N usa sobre qualquer ramo de sua computação sobre qualquer entrada de comprimento n.

Complexidade de tempo não-determinístico



Relacionamento entre modelos de MT

Teorema: Seja $t(n)$ uma função, onde $t(n) \geq n$, então para toda máquina de Turing não-determinística de fita única de tempo $t(n)$ existe uma máquina de Turing de fita única determinística equivalente de tempo $2^{O(t(n))}$.

Idéia da prova: Lembrar que a simulação de uma MT não-determinística em uma MT determinística pode ser visto como uma busca em uma árvore de computação não-determinística por estados de aceitação. Como a MTnão-determinística é uma máquina de tempo $O(t(n))$, então o caminho da raiz até as folhas é limitado por $O(t(n))$ passos. A busca por um estado de aceitação é uma operação exponencial limitado por $2^{O(t(n))}$ passos.