Complexidade Assintótica

Complexidade Assintótica

- Linguagem Recursiva
  - Problema decidível
  - Possui solução algorítmica

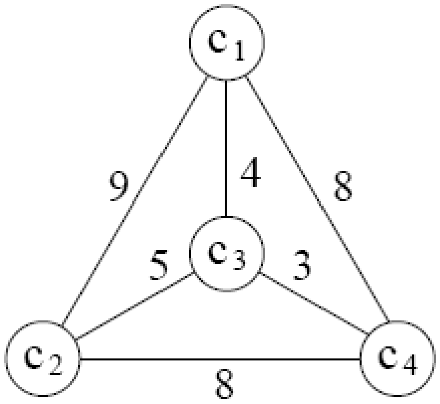
  - Solução é tratável computacionalmente?

Complexidade Assintótica

- O fato de um algoritmo resolver um dado problema não significa que seja aceitável na prática
- Exemplo:
  - Caixeiro Viajante
    - Um **caixeiro viajante** deseja visitar n cidades de tal forma que sua viagem inicie e termine em uma mesma cidade, e cada cidade deve ser visitada uma única vez
    - Supondo que sempre há uma estrada entre duas cidades quaisquer, o problema é encontrar a menor rota para a viagem.

Complexidade Assintótica

- A figura ilustra o exemplo para quatro cidades *c1*, *c2*, *c3*, *c4*, em que os números nos arcos indicam a distância entre duas cidades.
- O percurso < *c1*, *c3*, *c4*, *c2*, *c1*> é uma solução para o problema, cujo percurso total tem distância 24.



Complexidade Assintótica

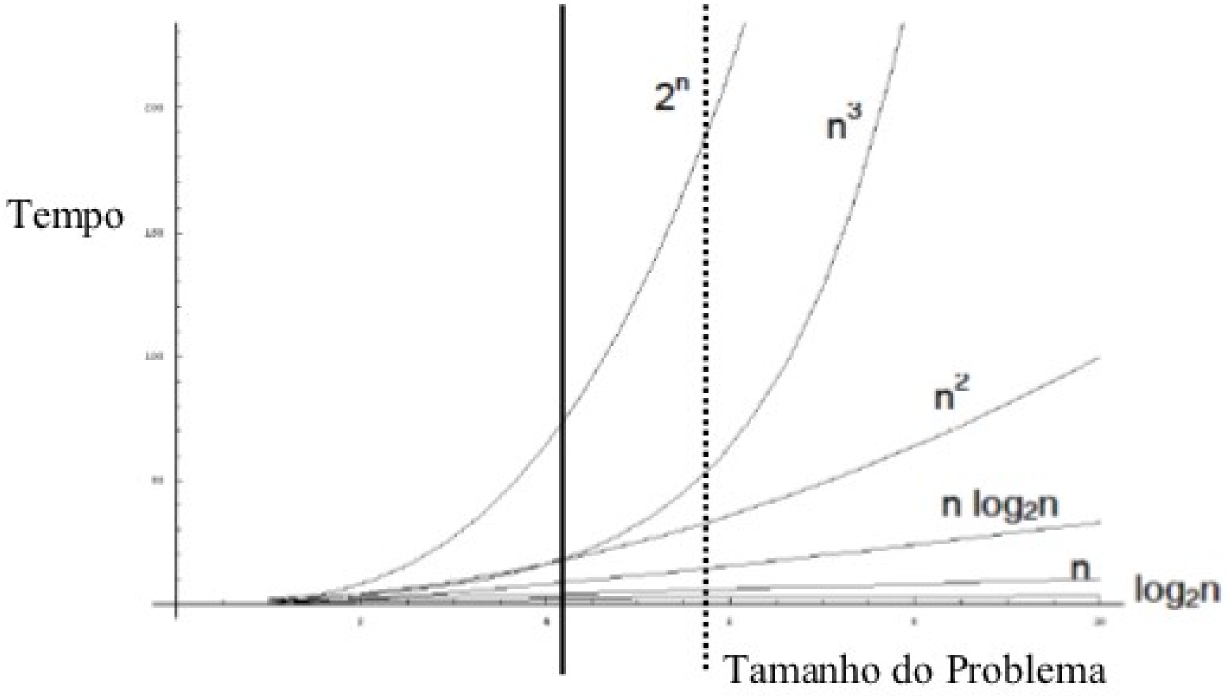
- Um algoritmo simples seria verificar todas as rotas e escolher a menor delas.

- Há (*n* - 1)! rotas possíveis e a distância total percorrida em cada rota envolve *n* adições, logo o número total de adições é *n*!

- Suponha agora 50 cidades: o número de adições seria $50! \approx 10^{64}$
- Em um computador que executa 10^9 adições por segundo, o tempo total para resolver o problema com 50 cidades seria maior do que 10^{45} séculos só para executar as adições

Complexidade Assintótica

- Classes de complexidade



Complexidade Assintótica

- Classes de Complexidade

	10	100	10^3	10^4	10^5	10^6
$\log_2 n$	3	6	9	13	16	19
n	10	100	1000	10^4	10^5	10^6
$n \log_2 n$	30	664	9965	10^5	10^6	10^7
n^2	100	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{300}	10^{300}	10^{3000}	10^{300000}

1 ano = $365 \times 24 \times 60 \times 60 \approx 3 \times 10^7$ segundos

1 século $\approx 3 \times 10^9$ segundos

1 milénio $\approx 3 \times 10^{10}$ segundos

Complexidade Assintótica

- Custo Assintótico
  - O custo assintótico de uma função f(n) representa o limite do comportamento de custo quando n cresce
  - Notação O
  - Notação Ω
  - Notação Θ

Complexidade Assintótica

- Notação O
  - Trazida da matemática por Knuth (1968):
  - $g(n) = O(f(n))$
  - Lê-se:
    - $g(n)$ é de ordem no máximo $f(n)$
    - $f(n)$ domina assintoticamente $g(n)$
    - ($f(n)$ é um limite assintótico superior para $g(n)$)

Formalmente:

$$g(n) = O(f(n)),\ c > 0 \text{ e } n0 \mid 0 <= g(n) <= c.f(n),\ \forall\ n >= n0$$

Complexidade Assintótica

– Se dizemos que g(n) = O(n2), significa que existem constantes c e m | g(n) <= c. n2 para n >= m

– Exemplo:

g(0) = 1

g(1) = 4

g(n) = (n+1)2

(n+1)2 <=cn2

n2 + 2n + 1 <= cn2

2n + 1 <= (c-1)n2

Complexidade Assintótica

- Notação Ω
  - Ω define um limite inferior para a função, por um fator constante.
    - Escreve-se g(n) = Ω(f(n)), se existirem constantes positivas c e n0 | para n >= n0, o valor de g(n) é maior ou igual a c.f(n)
  - Neste caso, diz-se que f(n) é um limite assintótico inferior para g(n).

Formalmente:

$$g(n) = \Omega(f(n)),\ c > 0 \text{ e } n_0 \mid 0 \le c.f(n) \le g(n),\ \forall\, n \ge n_0$$

Complexidade Assintótica

- Notação Θ
  - A notação Θ limita a função por fatores constantes
    - $g(n) = \Theta(f(n))$ se existirem constantes positivas c1 e c2 e n0 tais que para n >= n0, o valor de $g(n)$ está sempre entre $c1.f(n)$ e $c2.f(n)$ inclusive.

Formalmente:

$$g(n) = \Theta(f(n)),\ c1 > 0 \text{ e } c2 > 0 \text{ e } n0 \mid$$

$$0 <= c1.f(n) <= g(n) <= c2.f(n),\ \forall\ n >= n0$$