# Estrutura de Dados e Algoritmos

## Análise de Complexidade

# Análise de Complexidade

- **Algoritmo**: conjunto claramente especificado de instruções a seguir para resolver um problema;

# Análise de Complexidade

- **Análise de algoritmos**:
  - Provar que um algoritmo está correto;
  - Determinar recursos exigidos por um algoritmo (tempo, espaço, etc.);
  - Comparar os recursos exigidos por diferentes algoritmos que resolvem o mesmo problema (um algoritmo mais eficiente exige menos recursos para resolver o mesmo problema)
  - Prever o crescimento dos recursos exigidos por um algoritmo à medida que o tamanho dos dados de entrada cresce;

# Complexidade espacial e temporal

- Complexidade espacial de um programa ou algoritmo: espaço de memória que necessita para executar até ao fim
  - S(n) - espaço de memória exigido em função do tamanho (n) da entrada;
- Complexidade temporal de um programa ou algoritmo: tempo que demora a executar (tempo de execução)
  - T(n) - tempo de execução em função do tamanho (n) da entrada;

# Complexidade espacial e temporal

- Complexidade↑ versus Eficiência↓;
- Por vezes estima-se a complexidade para o "melhor caso" (pouco útil), o "pior caso" (mais útil) e o "caso médio" (igualmente útil);

# O Conceito de Complexidade

- A Complexidade Computacional é um ramo da Matemática Computacional que estuda a eficiência dos algoritmos.
- Para medir a eficiência de um algoritmo freqüentemente usamos um tempo teórico que o programa leva para encontrar uma resposta em função dos dados de entrada.

- PROBLEMA DO CAIXEIRO VIAJANTE:
  - “Suponha que um caixeiro viajante tenha de visitar n cidades diferentes, iniciando e encerrando sua viagem na primeira cidade. Suponha, também, que não importa a ordem com que as cidades são visitadas e que de cada uma delas pode-se ir diretamente a qualquer outra. O problema do caixeiro viajante consiste em descobrir a rota que torna mínima a viagem total”.

# Complexidade em Algoritmos Computacionais

- O problema do caixeiro viajante é um problema de otimização combinatória.
  - Como transforma-lo num problema de enumeração?
  - Como determinar todas as rotas do caixeiro?
  - Como saber qual delas é a menor?
- SOLUÇÃO: São (n-1)! Rotas
  - É um trabalho fácil para a máquina?

# Complexidade em Algoritmos Computacionais

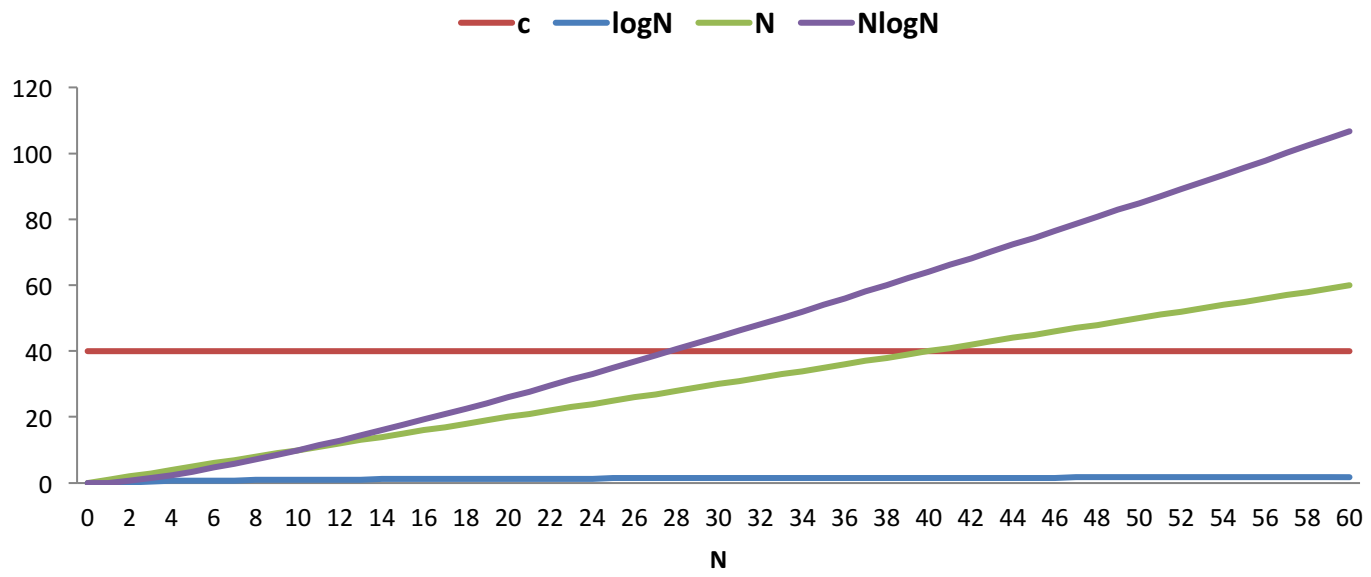
| n | Rotas por segundo | (n - 1)! | Cálculo total |
|---|---|---|---|
| 5 | 250 milhões | 24 | insignificante |
| 10 | 110 milhões | 362 880 | 0.003 seg |
| 15 | 71 milhões | 87 bilhões | 20 min |
| 20 | 53 milhões | $1.2 \times 10^{17}$ | 73 anos |
| 25 | 42 milhões | $6.2 \times 10^{23}$ | 470 milhões de anos |

# Complexidade em Algoritmos Computacionais

- Se descobrirmos como resolver o problema do caixeiro viajante em tempo polinomial, seremos capazes de resolver, também em tempo polinomial, uma grande quantidade de outros problemas matemáticos importantes.
- Se um dia alguém provar que é impossível resolver o problema do caixeiro em tempo polinomial no número de cidades, também se terá estabelecido que uma grande quantidade de problemas importantes não tem solução prática.

# Análise Assintótica

- Motivação: Definir ordem entre funções.
- Avaliação pontual → Não tem sentido:
  - f(N) < g(N) ?

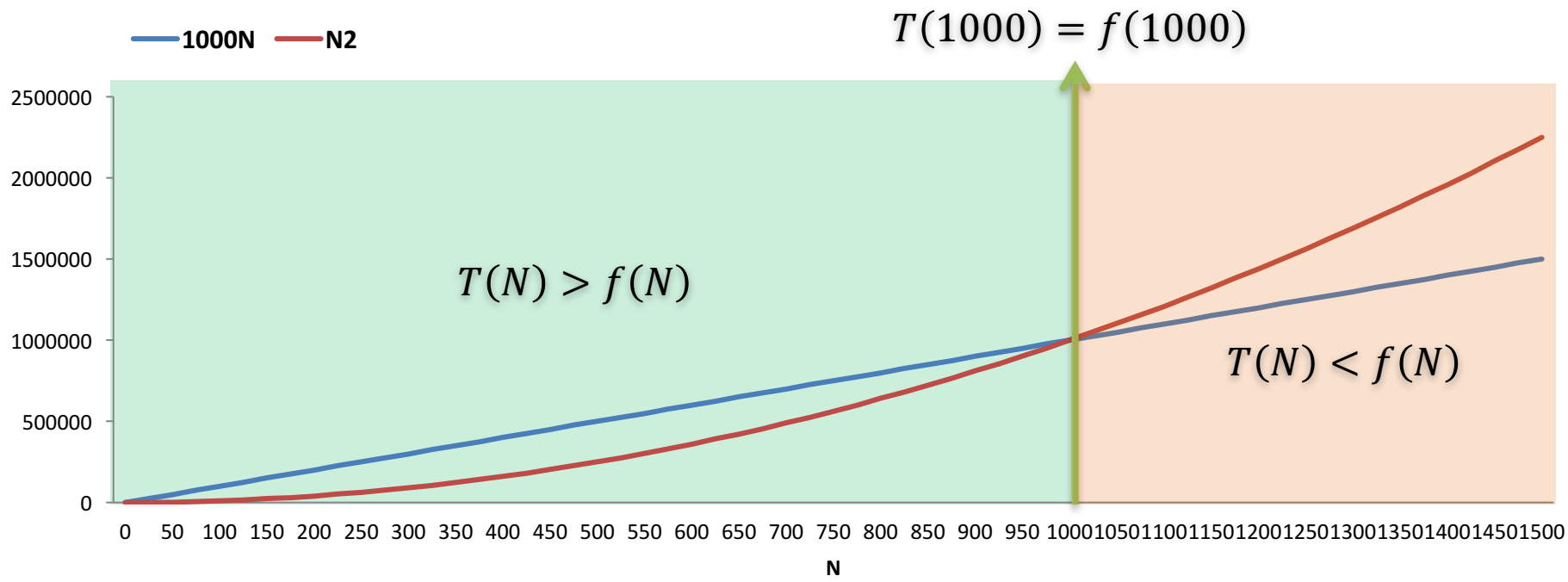


# Análise Assintótica

- Forma de Análise: Taxa de crescimento.

$T(N) = 1000N$

$f(N) = N^2$

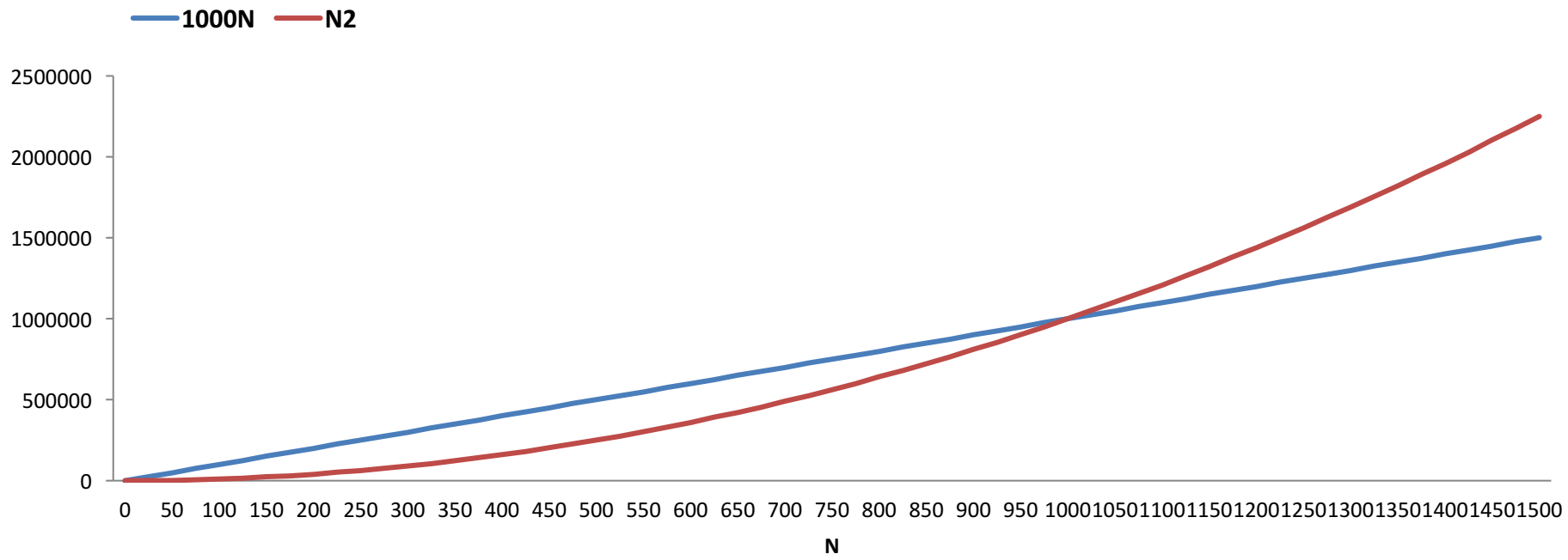


# Análise Assintótica

- Apesar de 1000N ser maior que $N^2$ para N pequenos, a taxa de crescimento de $N^2$ é maior e ultrapassa 1000N para N>1000. Logo, $N^2$ é maior que 1000N.
- Há um valor de N ($n_0$) a partir do qual c.f(N) será sempre, no mínimo, tão grande quanto T(N). Neste caso: $n_0$=1000 e c=1. Outra possibilidade seria $n_0$=100 e c=10.

# Análise Assintótica

- Concluindo, podemos dizer que a taxa de crescimento de T(N)=1000N é menor ou igual à taxa de crescimento de f(N)=$N^2$.



# Análise Assintótica

- Taxas de Crescimento Típicas:
  - $c$ – *Constante;*
  - $logN$ – *Logarítmica;*
  - $log^2 N$ – *Log quadrática;*
  - $N$ – *Linear;*
  - $N.logN$
  - $N^2$ – *Quadrática;*
  - $N^3$ – *Cúbica;*
  - $2^N$ – *Exponencial;*

# Análise Assintótica

- Taxas de Crescimento Típicas: