ICC2 Aula 4

Fábio Nakano

Enfim, ICC2!!

O que queremos dos nossos programas?

O que queremos dos nossos programas?

- Serem corretos
- Executar em tempo razoável
- Usar bem os recursos da máquina

  Tempo de processamento

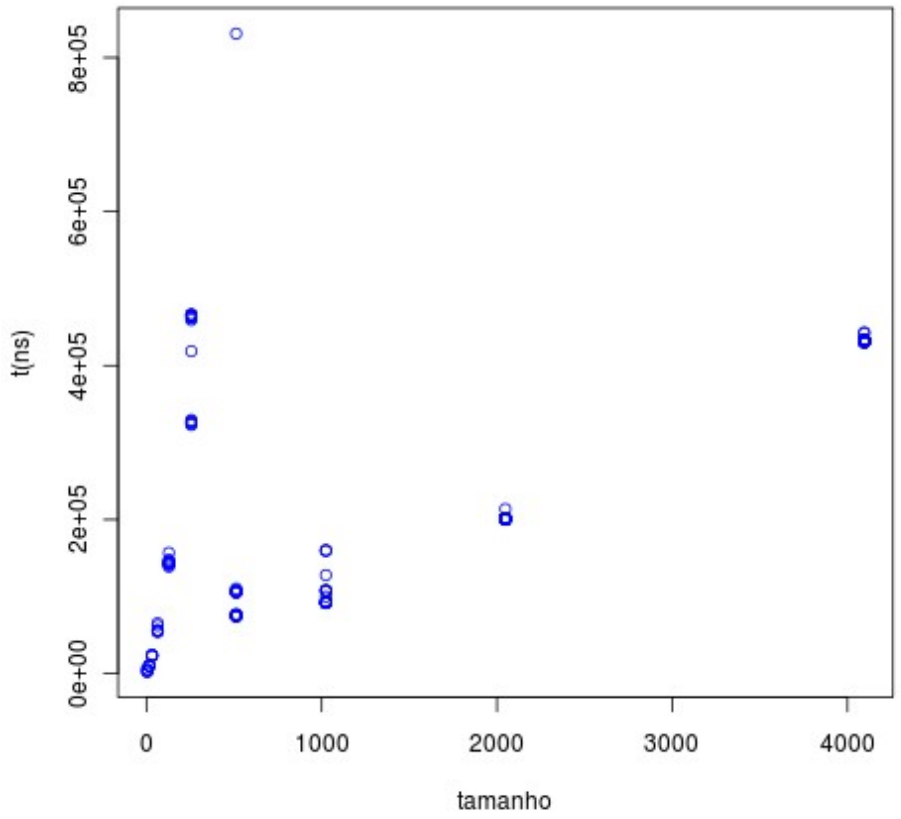
  Memória

Tempo de execução pode ser diretamente medido

```
starttime=System.nanoTime(); /* java.util.* */
```

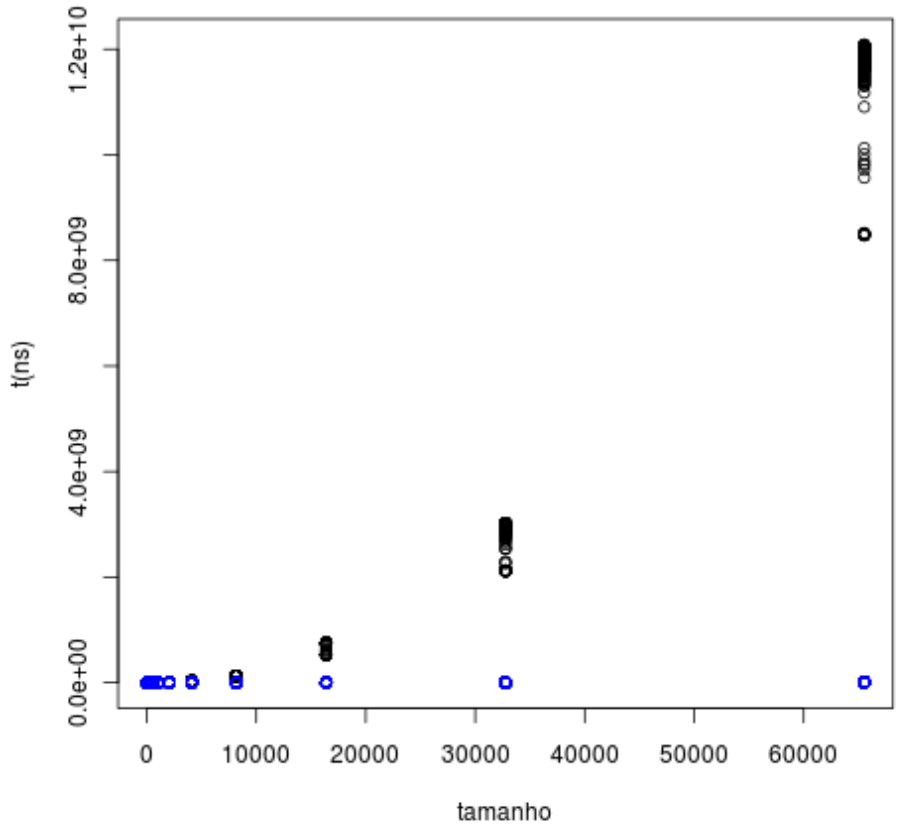
Um conhecido repaginado...

Tempo de processamento flutua



Nós analisaremos este...

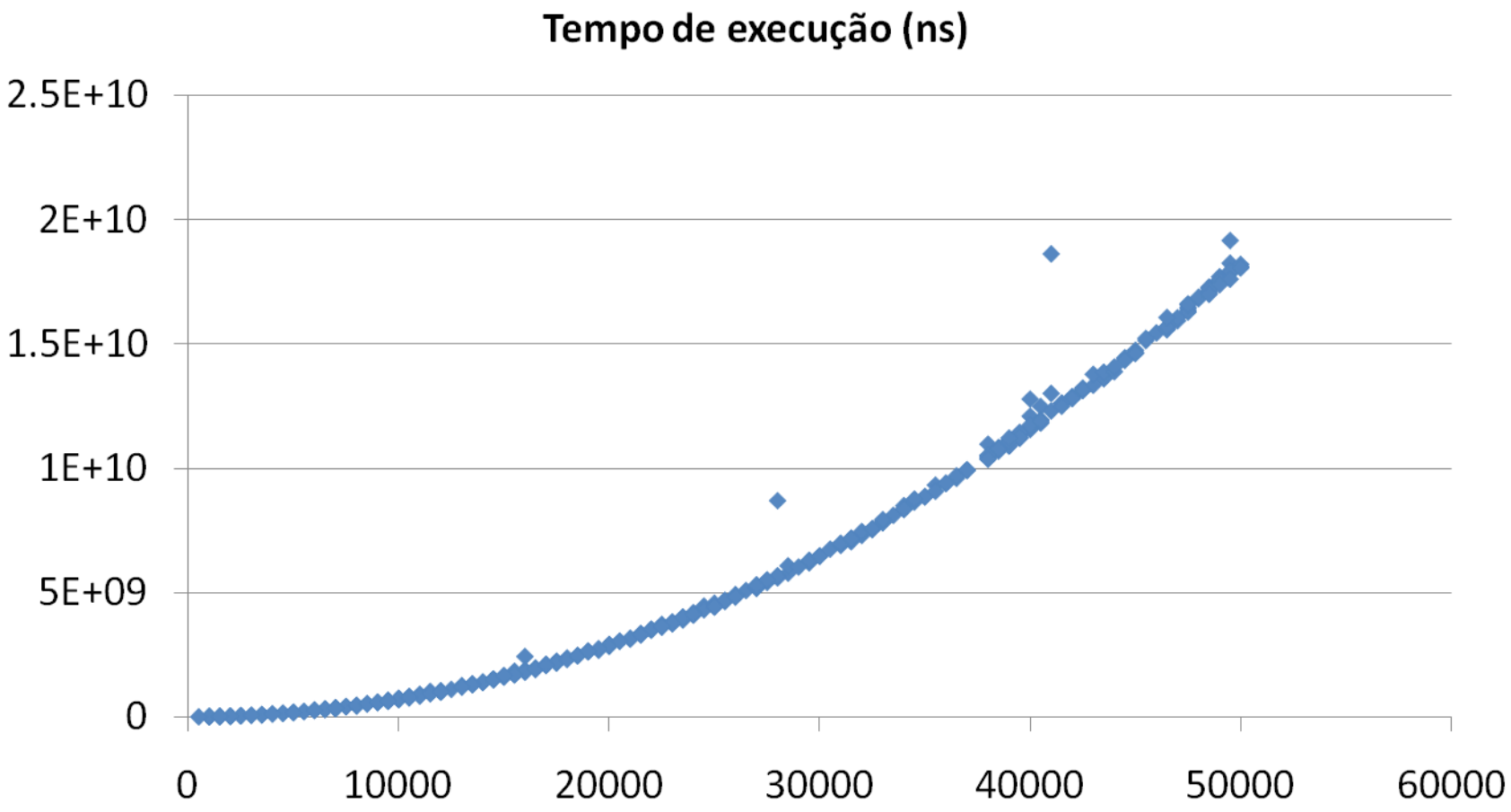
Comparativo de tempo de processamento



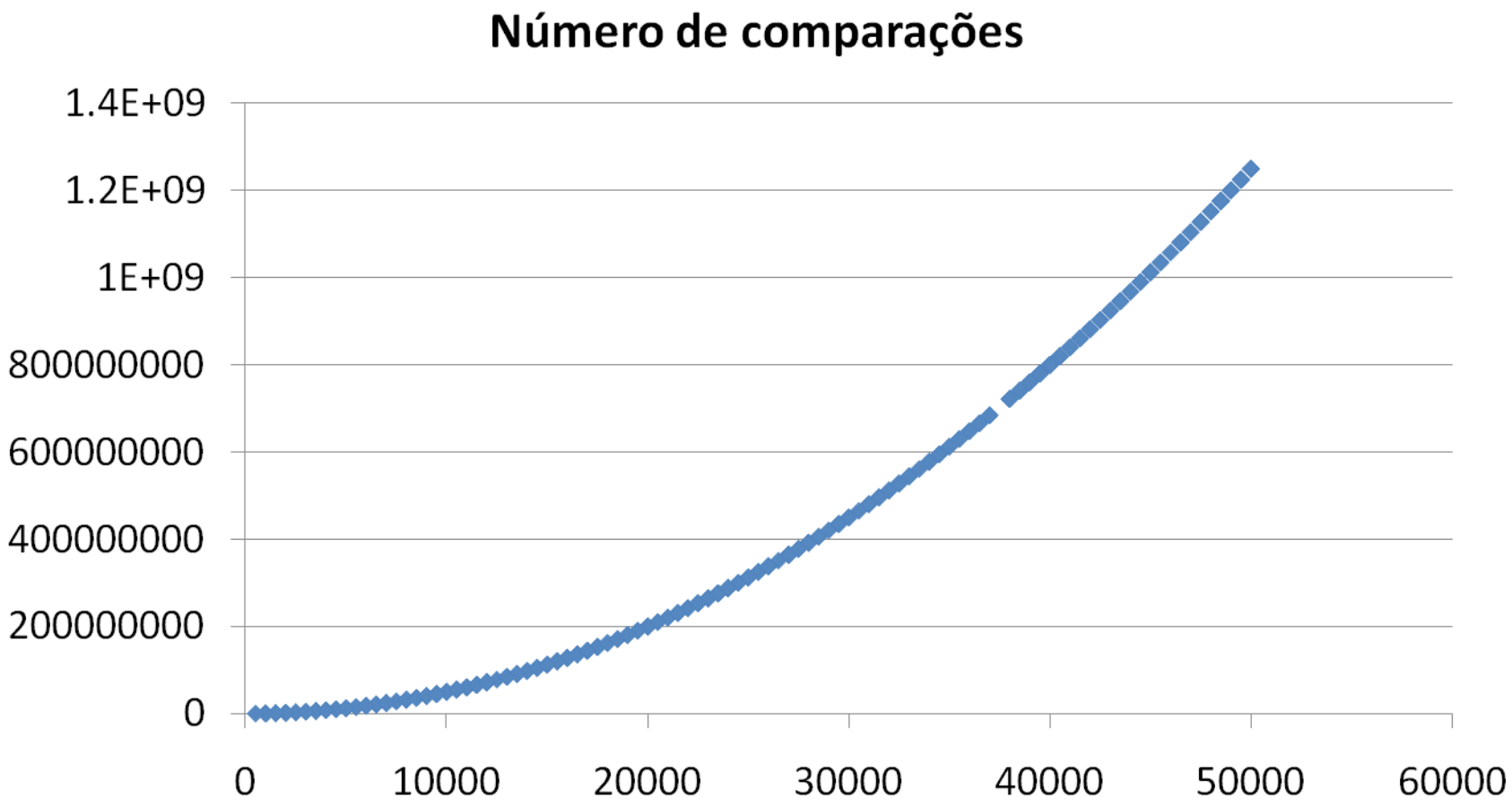
Medir tempo de execução tem seus problemas...

- Depende do tipo de máquina
- Do tempo de execução de cada instrução
- Da qualidade do programa
- Da específica instância do problema

Bubble


Tempo de execução (ns)
2.5E+10
2E+10
1.5E+10
1E+10
5E+09
0
0
10000
20000
30000
40000
50000
60000

•Bubble


Número de comparações
1.4E+09
1.2E+09
1E+09
800000000
600000000
400000000
200000000
0
0
10000
20000
30000
40000
50000
60000

O Que pode ser medido??? Tem que ser algo simples para que possamos fazer contas!!

-
- Quantidade de operações
-
- Quantidade de uma certa operação

Comentários e Conclusões até agora

- Deseja-se chegar ao algoritmo de melhor desempenho possível, onde desempenho é medido por tempo de execução e memória utilizada.
- Medida direta do tempo de execução não é uma boa idéia. Pois, entre outros fatores, depende de máquina e dos outros processos sendo executados concomitantemente.
- É possível contar número de operações executadas em uma algoritmo.
- É possível ajustar/extrair uma função que descreve o número de operações de um programa/algoritmo e esta é aderente ao tempo de execução, o que é de se esperar pois cada operação consome uma certa quantidade de tempo.
- Estas funções podem ser estudadas e comparadas.
-
- A função que descreve o número de operações ou a quantidade de memória pode ser chamada função de complexidade de tempo ou de memória, respectivamente.

Estudo comparativo de algumas funções (fazendo as hipóteses certas, essas funções “simples” são o que precisamos.)

Comparativo entre funções (n pequeno)

$$t(n) = n\,(lilás)$$

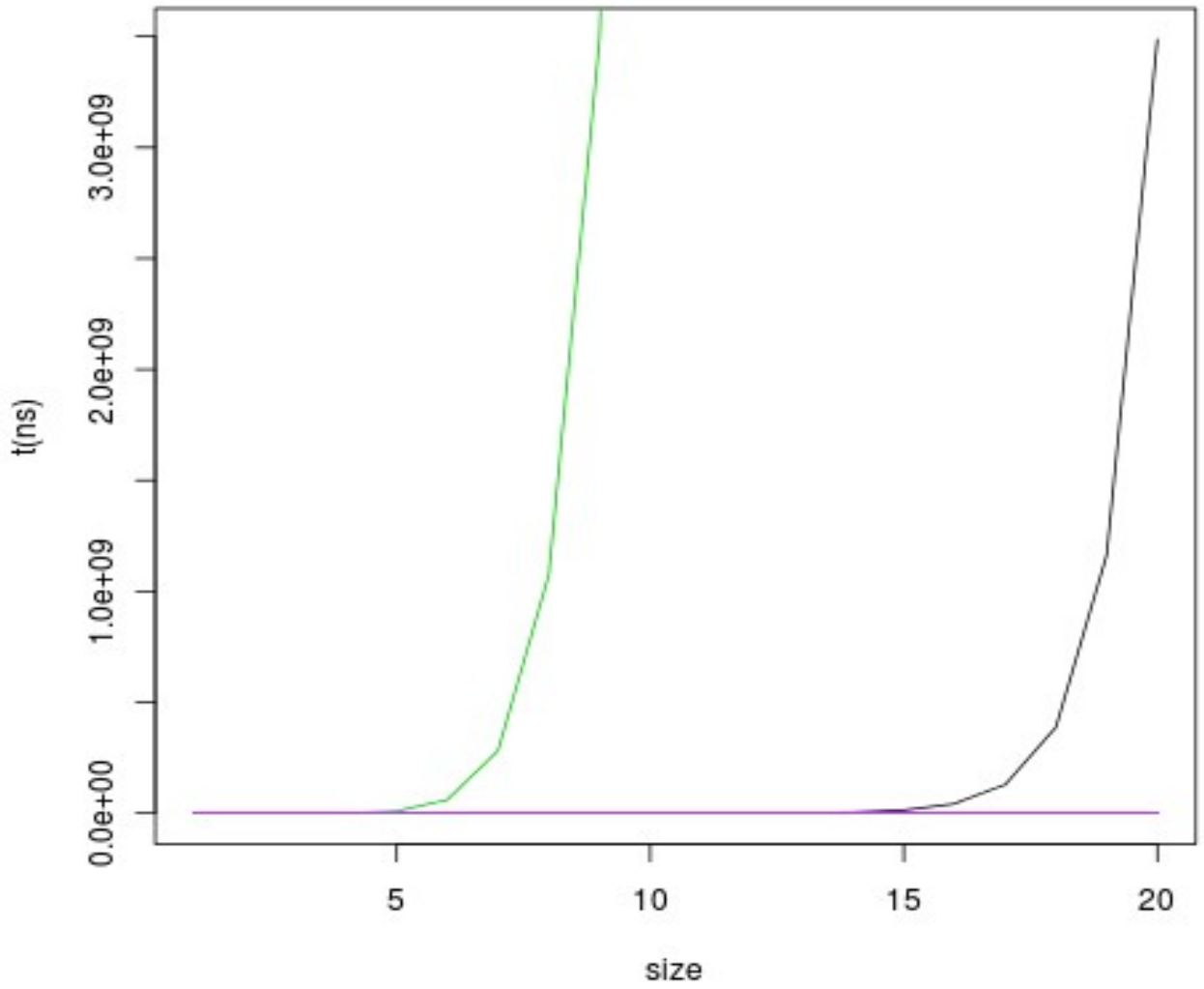
$$t(n) = n^2$$

$$t(n) = n^3$$

$$t(n) = n^{10}\,(verde)$$

$$t(n) = 2^n$$

$$t(n) = 3^n\,(preto)$$



Comparativo entre funções (n “grande”)

$$t(n) = n(lilás)$$

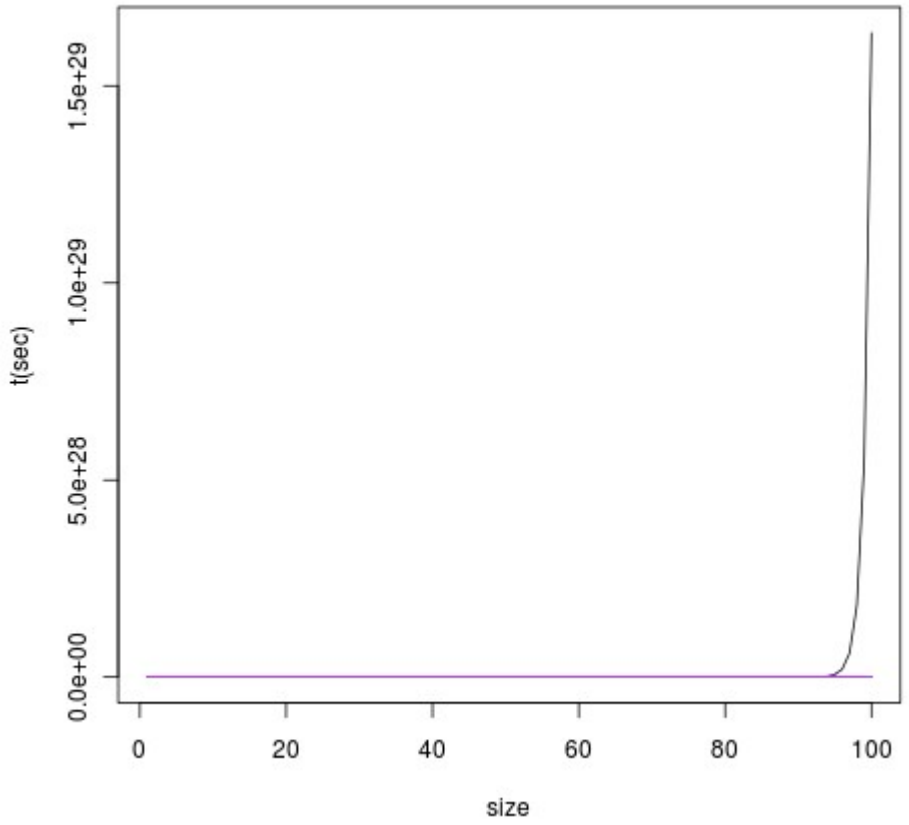
$$t(n) = n^2$$

$$t(n) = n^3$$

$$t(n) = n^{10}(verde)$$

$$t(n) = 2^n$$

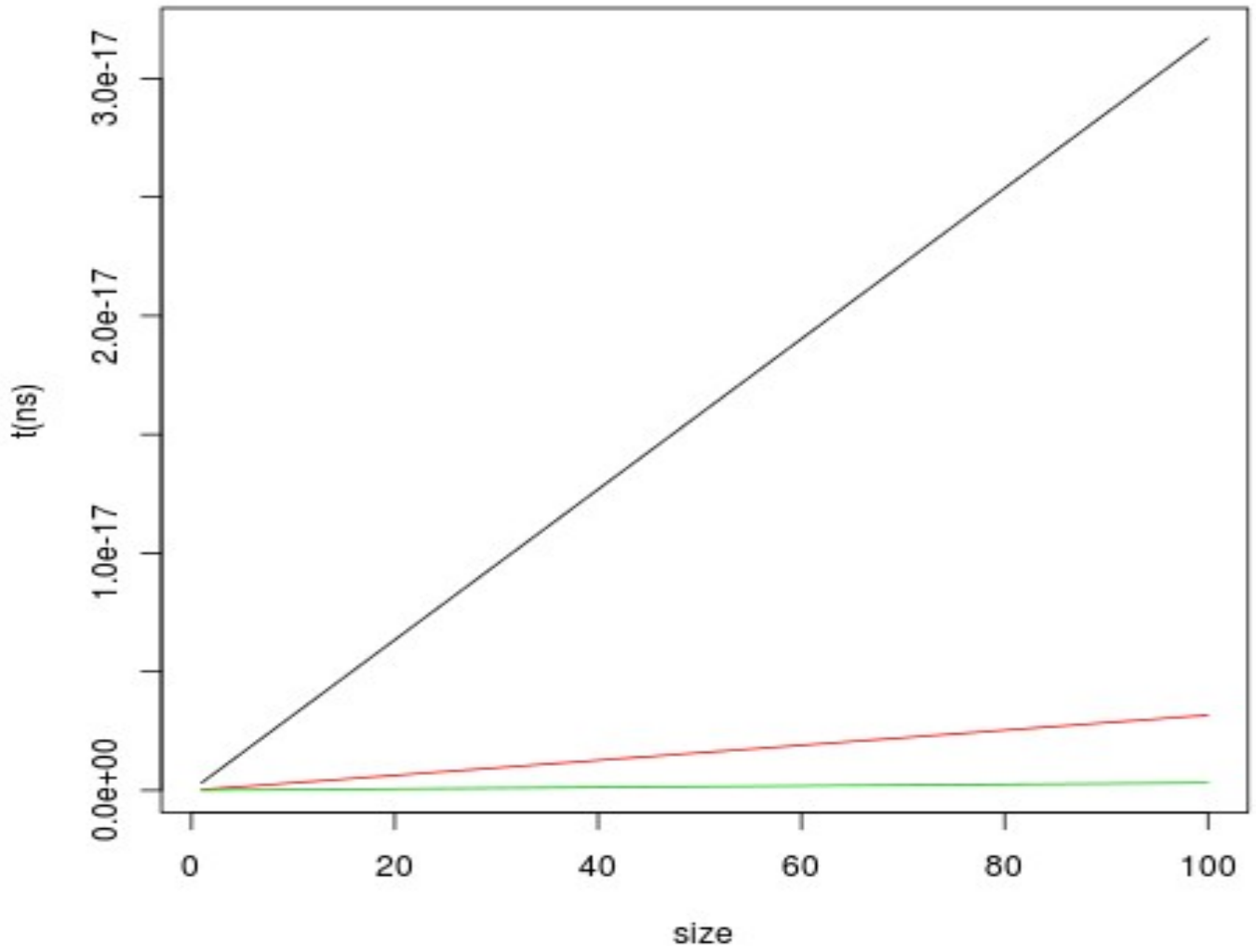
$$t(n) = 3^n(preto)$$



Aceleração (aumento de velocidade de processamento em 1x, 10x, 100x)

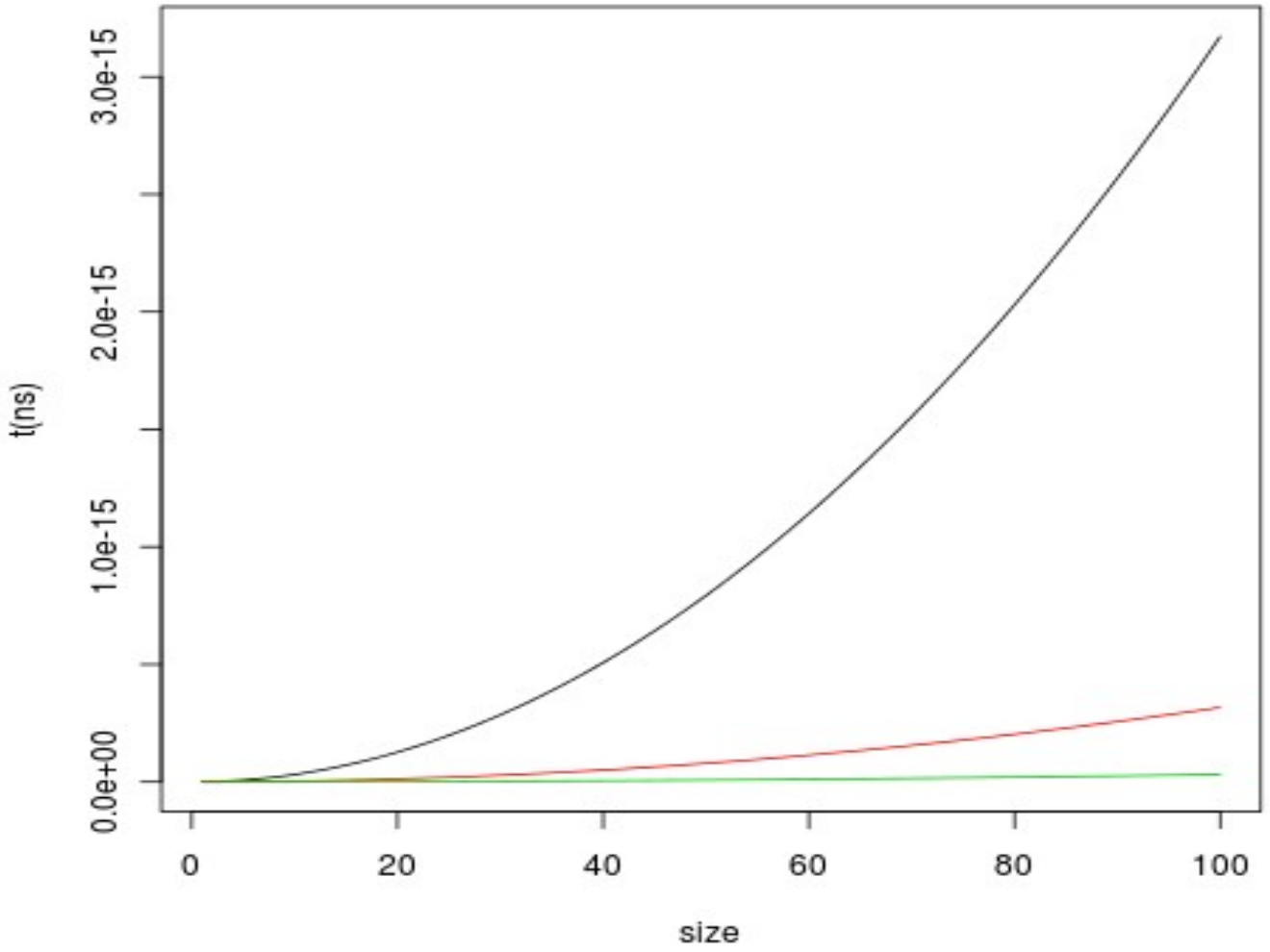
$$t(n) = n$$

Preto: 1x
Vermelho: 10x
Verde: 100x



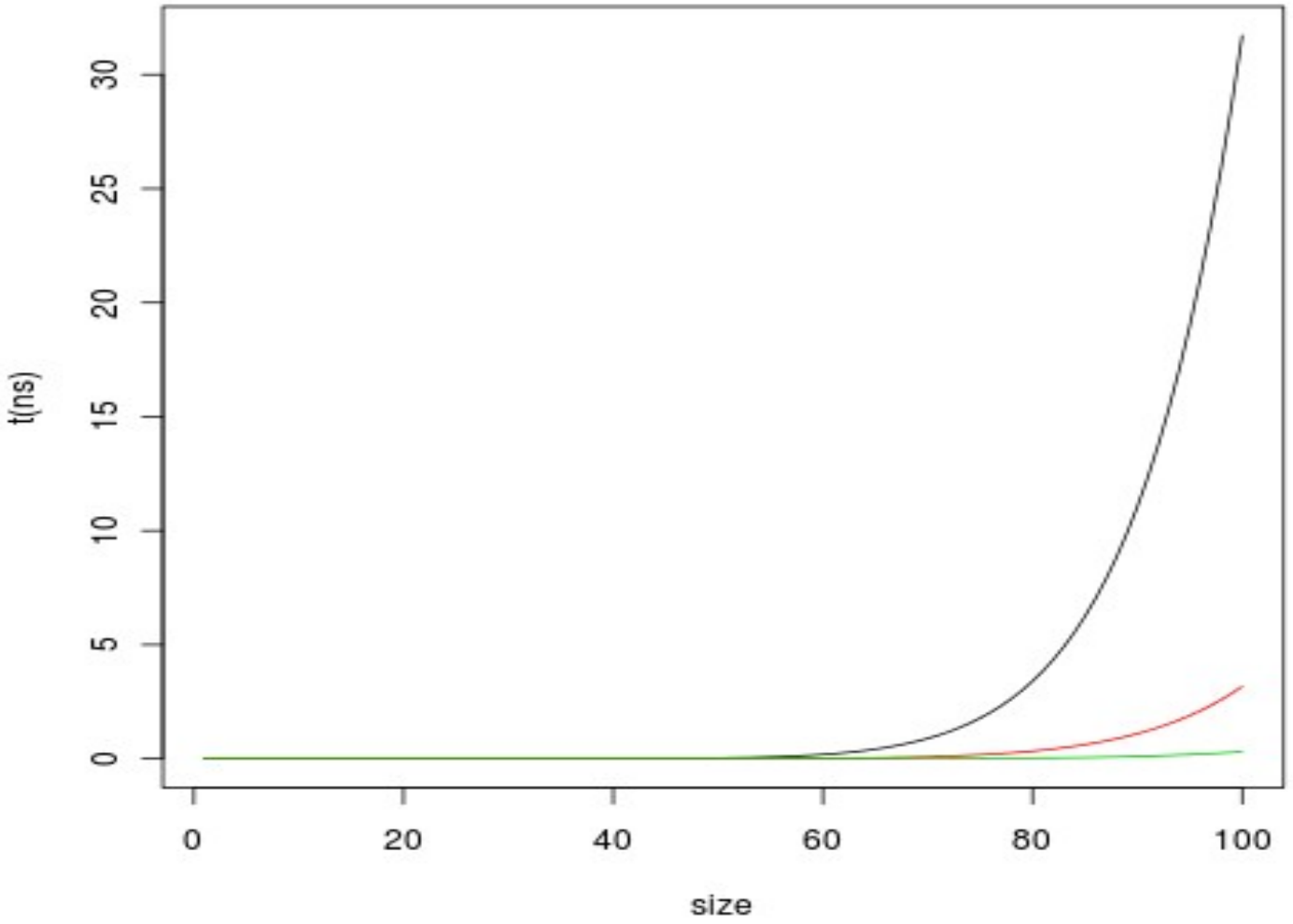
Aceleração (aumento de velocidade de processamento em 1x, 10x, 100x)

$$t(n) = n^2$$



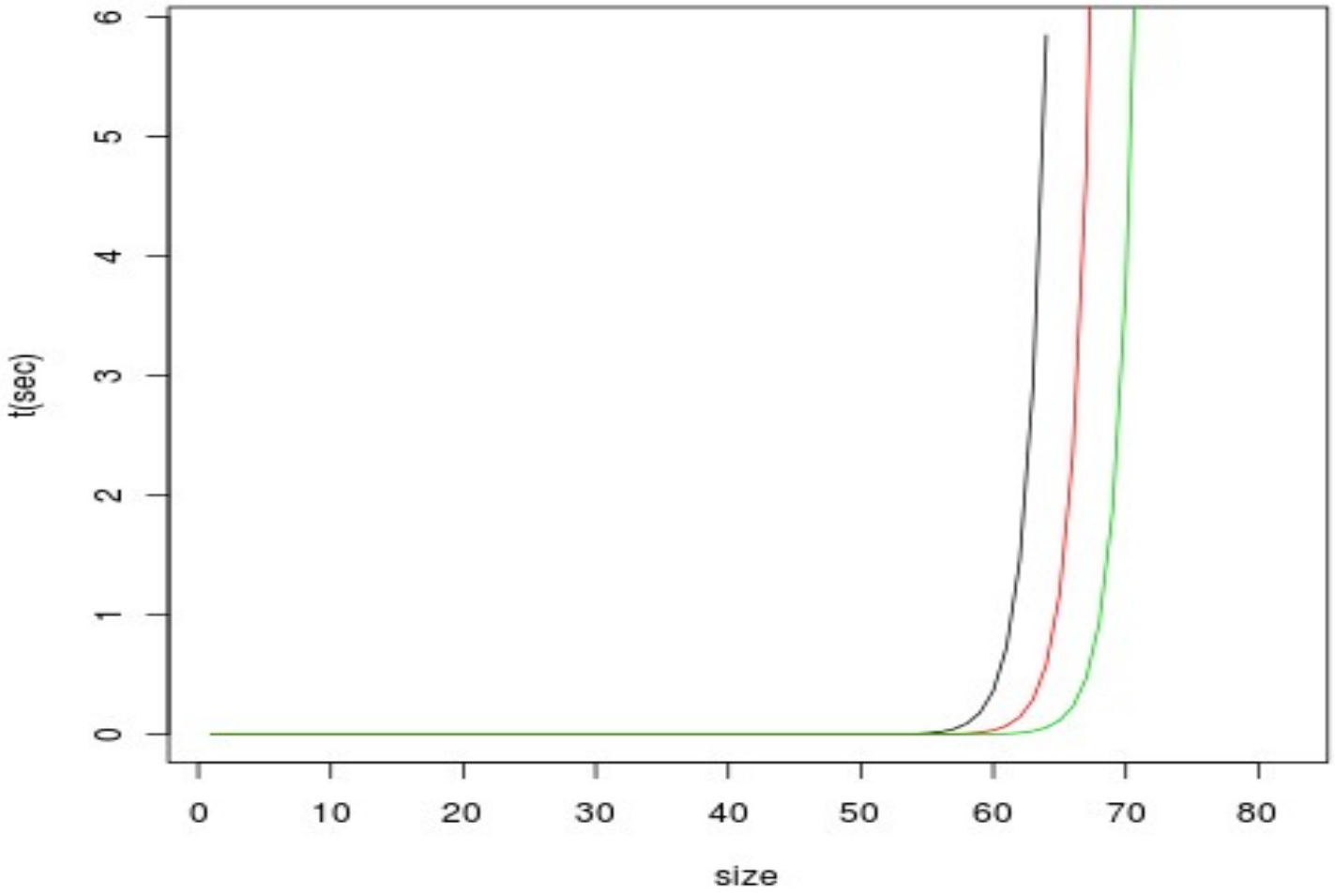
Aceleração (aumento de velocidade de processamento em 1x, 10x, 100x)

$$t(n) = n^{10}$$



Aceleração (aumento de velocidade de processamento em 1x, 10x, 100x)

$$t(n) = 2^n$$

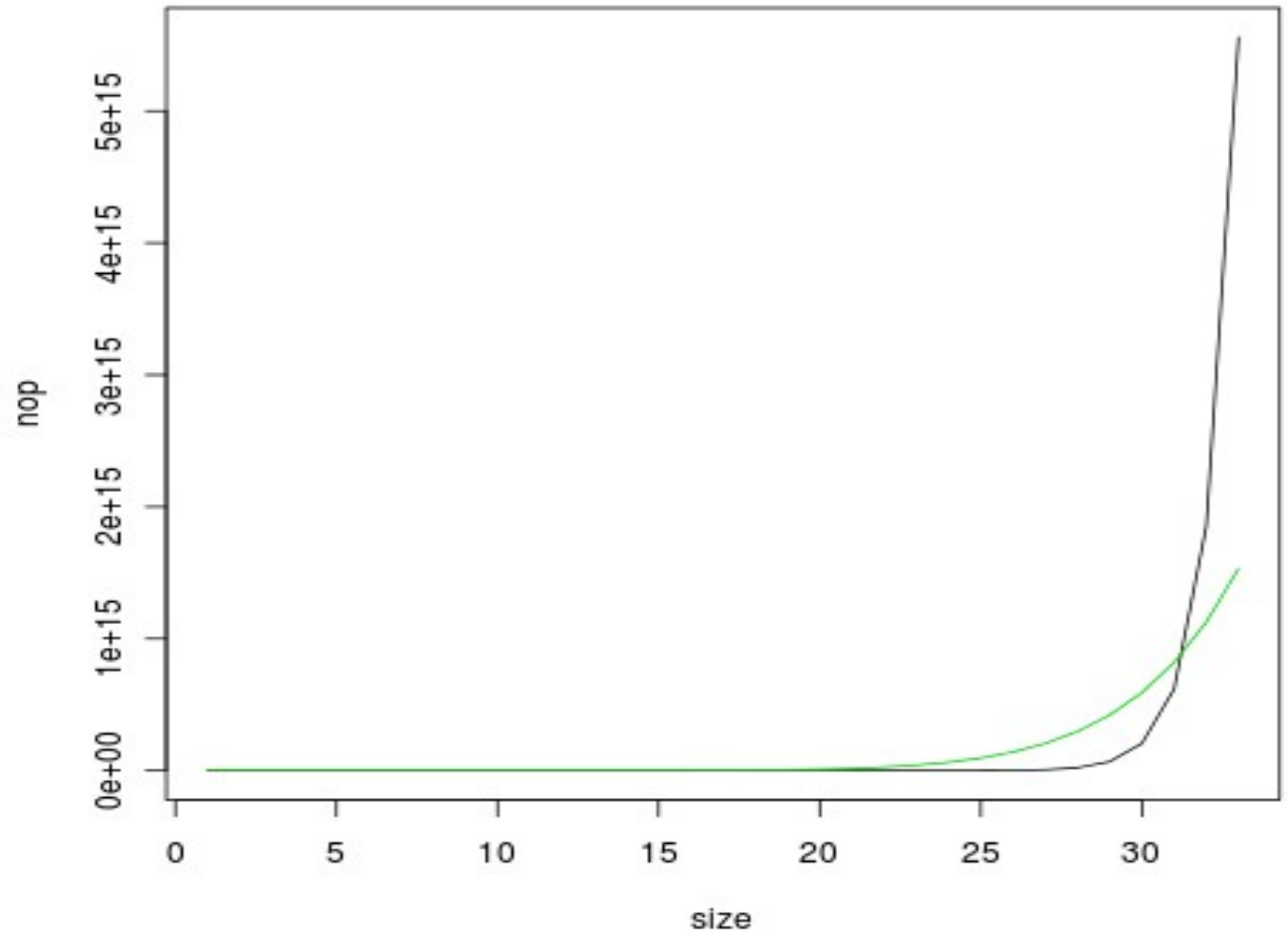


Comentários e Conclusões

- Existe diferença significativa na taxa de crescimento de funções lineares, polinomiais e exponenciais.
- O atual aumento na capacidade de processamento tem grande impacto em programas com tempo de execução linear e pequeno impacto em programas com tempo de execução exponencial
- É muito mais vantajoso usar algoritmos cujo tempo de execução segue/é ajustado por funções de taxa de crescimento menor.

Estamos interessados no comportamento da função para n “grande”. Neste caso, sempre existe n a partir do qual o termo de maior taxa de crescimento domina a função.

Comparativo entre 3^n (preto) e n^10 (verde)



Pausa para experimentação...