GSI011 – Estrutura de Dados 2

Profª: Christiane Regina Soares Brasil

GSI011 – Estrutura de Dados 2 – Aula 1

- Análise de Algoritmos
  - Correção: o algoritmo resolve o problema?
  - Desempenho: como o algoritmo resolve? Em quanto tempo?

GSI011 – Estrutura de Dados 2 – Aula 1

- Todo problema computacional tem uma coleção de instâncias (amostras).

- Exemplo: inserção em uma lista linear simplesmente encadeada para instância (14, 25, 0, -7, 31). O tamanho desta instância é 5. A instância (25, -7, 14, 31, 0) é outra entrada possível.

GSI011 – Estrutura de Dados 2 – Aula 1

- O que influencia no desempenho?
  - Tamanho da instância (quantidade de elementos)
  - Qualidade da instância (ordenada ou desordenada, por exemplo)
  - Recursos computacionais (memória, processador)

- Complexidade Computacional
  - Memória + tempo

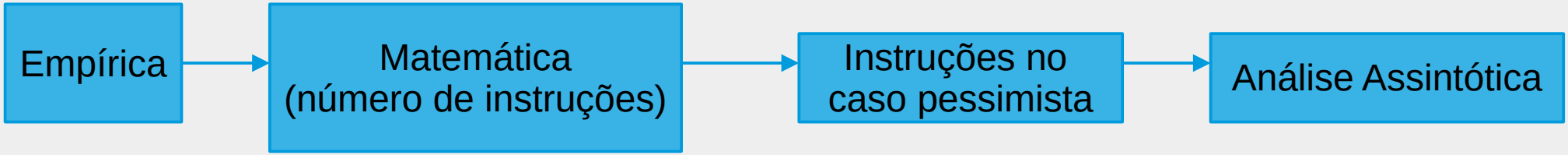
GSI011 – Estrutura de Dados 2 – Aula 1

- Análise Empírica:
  - Realização de testes
  - Tempo de execução do programa

- Vantagens:
  - Comparar configurações do computador
  - Comparar sistema operacional
  - Comparar linguagens de programação

- Desvantagens:
  - Necessidade de implementação (habilidade do programador)
  - Resultado “mascarado” (linguagem, hardware e software)
  - Depende das instâncias (tamanho, qualidade)

- Hierarquia de abstração para análise de complexidade:



- Análise Matemática:
  - “Contar” o número de instruções com tempo O(1), ou seja, tempo constante.
    - Operações básicas
      - Atribuição;
      - Incremento/decremento;
      - Teste de condição;
      - Entrada/Saída.

- Exemplo 1:

```
int a;
scanf(“%d”, &a);
a++;
printf(“%d”, a);
```

Caso geral:
F(n): 3
Logo, o custo é constante.

F(n) é a função de custo, onde n é a entrada dos dados.

- Exemplo 2:

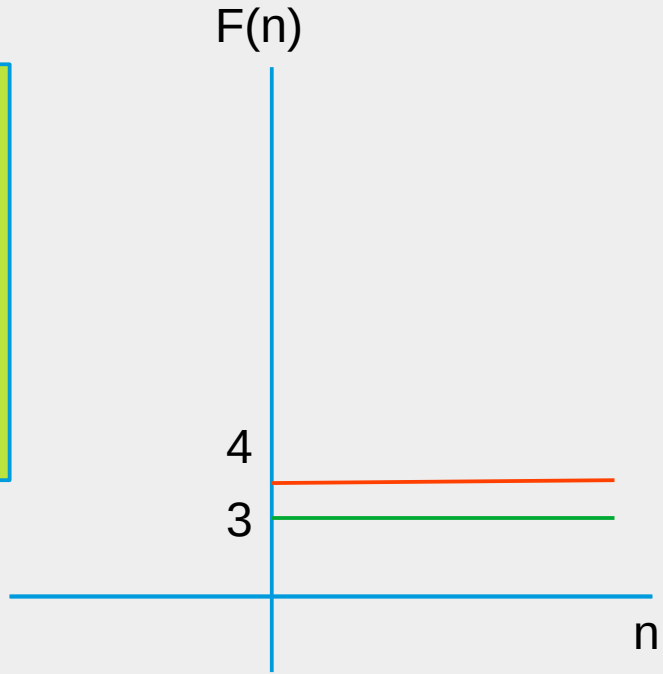
```
int a;
scanf(“%d”, &a);
a++;
if (a>0)
        printf(“%d”, a);
```

Melhor caso (não positivo):
F(n): 3

Pior caso (positivo):
F(n): 4

Melhor caso:
F(n): 3

Pior caso:
F(n): 4



n	Melhor caso	Pior caso
-1	3	-
-10	3	-
1	-	4
10000	-	4

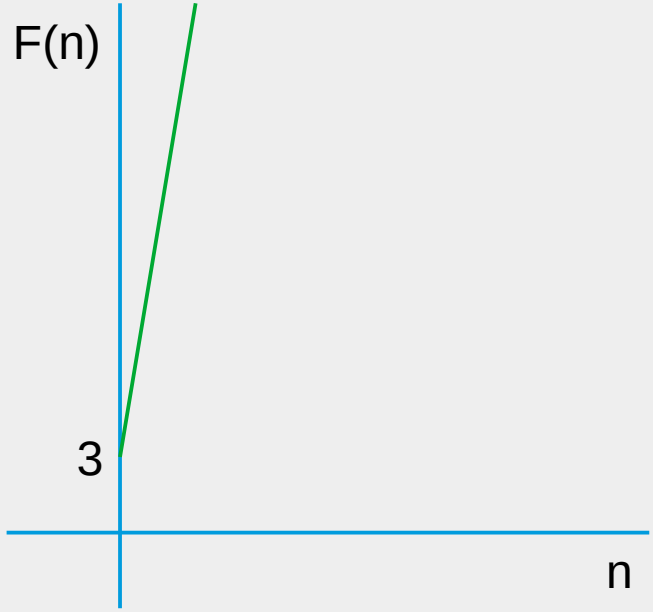
- Exemplo 3:

```
int a, x, i;
scanf(“%d %d”, &a, &x);
for(i=0; i<x; i++)
    a++;
```

Caso geral:
F(n): 1 + 1 + 1 + n + n + n
Portanto, temos:
F(n): 3n + 3

Logo, o custo é linear em função de n.

- Gráfico do exemplo 3:



Caso geral:
F(n): 1 + 1 + 1 + n + n + n
Portanto, temos:
F(n): 3n + 3

Logo, o custo é **linear** em função de n.

Quanto maior o **coeficiente angular**, mais custoso é nosso programa.

- Exemplo 4:

```
int i, x, a;
scanf(“%d”, &x);
for(i=0; i<x; i++){
      scanf(“%d”, &a);
       if(a>0) printf(“positivo”);
       }
```

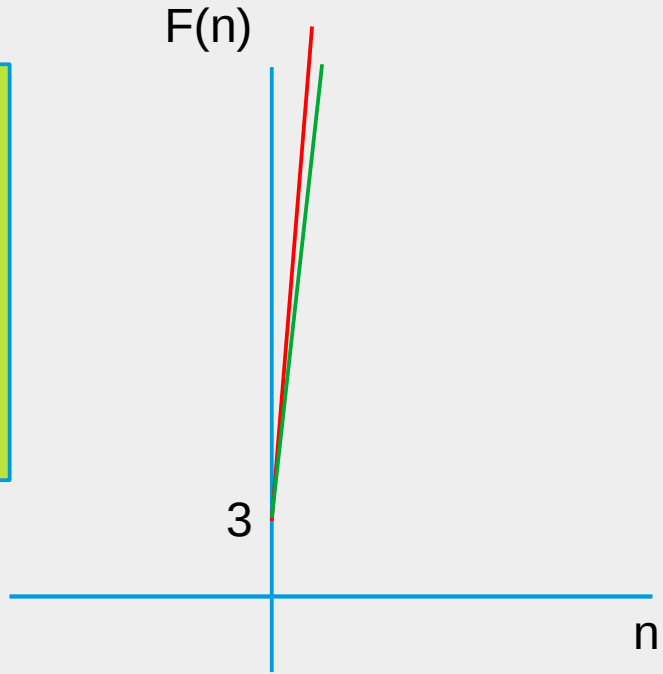
Melhor caso (nenhum positivo):
F(n): 1 + 1 + 1 + n + n + n + n
Portanto, temos:
F(n): 4n + 3

Pior caso (todos são positivos):
F(n): 1 + 1 + 1 + n + n + n + n + n
Portanto, temos:
F(n): 5n + 3

Ambas tem custo linear em n, no entanto, a segunda é **mais custosa** que a primeira.

Melhor caso:
F(n): 4n + 3

Pior caso:
F(n): 5n+ 3



n	Melhor caso	Pior caso
1	7	8
10	43	53
1000	4 003	5 003
10000	40 003	50 003