Ciência da Computação
Campus Arapiraca

Aula 01: Introdução

Prof. Dr. Rodolfo Carneiro
rodolfo.cavalcante@arapiraca.ufal.br

Introdução

- O que são programas?
  - Algoritmos para automatização de tarefas
- Como um programa funciona?
  - Recebe **dados** de entrada
  - Realiza um processamento nos **dados**
  - Produz uma saída

Introdução

- Dados compõem um elemento extremamente importante para o sistema computacional
- Coleções de:
  - Dados de funcionários
  - Dados de produtos
  - Dados de um experimento científico
  - Dados de uma série temporal
- Conjuntos dinâmicos
  - Algoritmos realizam operações de crescimento, encolhimento, alterações nos itens ao longo do tempo

Introdução

- É preciso decidir como estes dados serão:
  - Guardados no programa
  - Manipulados pelo programa
- Essa estrutura é chamada de estrutura de dados
  - Conjunto de dados relacionados entre si

Introdução

- Programas são compostos por
  - Algoritmos
  - Estruturas de dados
- A eficiência de um algoritmo está intrinsecamente relacionada com a estrutura de dados utilizada
  - Algoritmos exigem vários tipos diferentes de operações sobre os dados

Introdução

- A escolha de uma estrutura de dados afeta:
  - Quantidade de memória para armazenamento
  - Tempo de processamento
- Uma estrutura de dados define:
  - A forma de armazenagem
  - As operações de manipulação
    - Busca
    - Inserção
    - Remoção
    - Verificar se está vazia
    - Ordenação
    - ...

Introdução

- Existem várias estruturas de dados
  - Vetores
  - Matrizes
  - Sets
  - Tuplas
  - Dicionários
  - Listas
  - Pilhas
  - Filas
  - Árvores
  - Grafos
  - Tabelas hash
  - ....

Introdução

- Qual estrutura é a melhor?

Introdução

- Qual estrutura é a melhor?
- Antes de responder…
  - … o que significa ser melhor?

Introdução

- Qual estrutura é a melhor?
- Antes de responder…
  - … o que significa ser melhor?
- Existe uma área da computação dedicada ao estudo da eficiência de algoritmos:
  - Análise de algoritmos

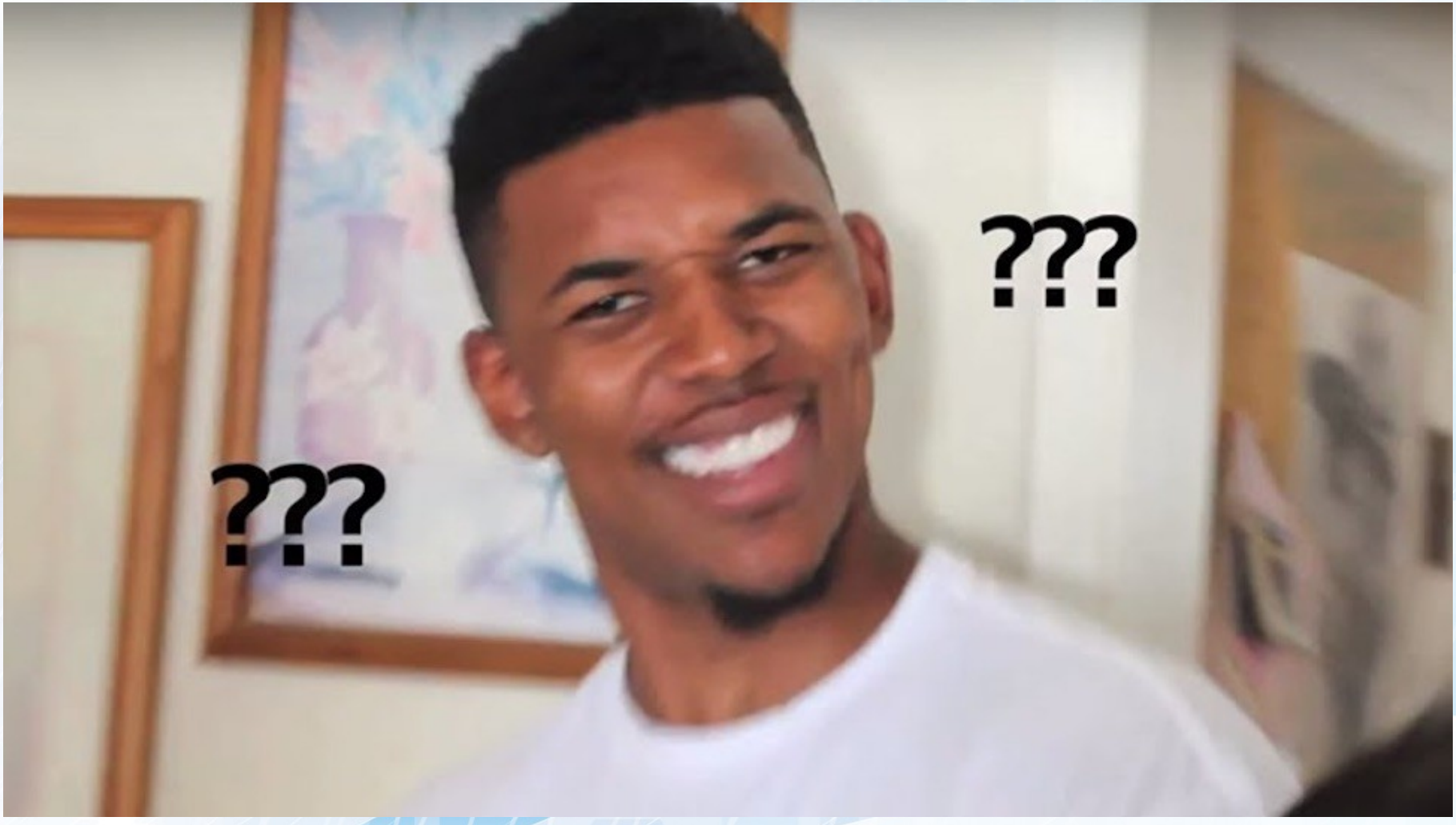
Análise de Algoritmos

- O projeto de um algoritmo deve considerar o desempenho que este terá após sua implementação
- Várias soluções podem ser propostas para um problema
  - Escolha através da análise da solução (algoritmo)
- Critérios para escolha
  - Tempo de execução
  - Espaço ocupado

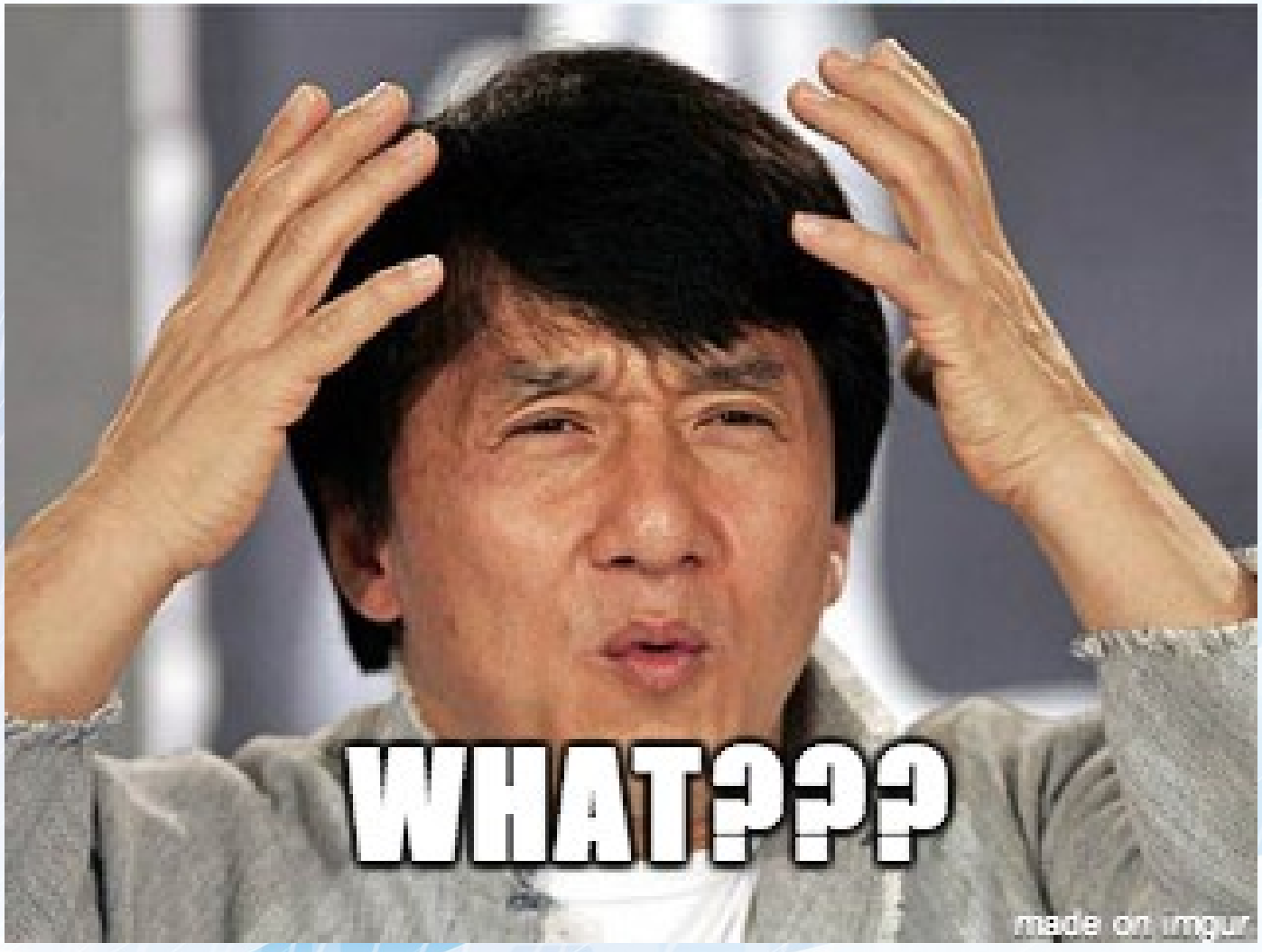
Análise de Algoritmos

- Mas entenda:
  - Tempo não é tempo e espaço não é espaço

Análise de Algoritmos



Análise de Algoritmos



Análise de Algoritmos

- Tempo não é tempo e espaço não é espaço
  - Utilizar tempo e espaço físico é difícil
    - Tempo físico – medido por relógio
    - Espaço físico – unidades de memória
  - Depende de vários fatores
    - Compilador
    - Sistema operacional
    - Tecnologia de Hardware
    - ...

Análise de Algoritmos

- Solução encontrada:
  - Medir custo por meio de modelo matemático genérico
  - Operações genéricas:
    - Aritméticas – soma, subtração, multiplicação, etc
    - movimentação de dados – armazenar, copiar excluir
    - Controle – decisões, comparações, retorno de função
  - Instruções realizadas uma após outra
    - Sem execuções concorrentes

Análise de Algoritmos

- Considera-se que cada instrução tem um custo constante
- Apenas operações mais significativas são consideradas
- Ex: ordenação
  - Considera-se o número de comparações
  - Desconsidera-se operações de substituição, movimentação de dados ou manipulações de índices

Análise de Algoritmos

- Custo de execução de um algoritmo vai ser dado por uma função de custo T
- T(n) é a medida necessária para executar um algoritmo para uma instância do problema de tamanho n
  - T(n) – função de complexidade de tempo ou de espaço do algoritmo

Análise de Algoritmos

- Processo de análise
  - Entendimento do funcionamento do algoritmo
  - Identificação da operação mais relevante
  - Montagem da função de custo
    - Função que define a complexidade do algoritmo para entrada n

Análise de Algoritmos

- Ex: Buscar pela ação com o maior lucro líquido na bovespa em 2017
  - analise

Análise de Algoritmos

- Ex: Buscar pela ação com o maior lucro líquido na bovespa em 2017

```
def getMaiorLucro(acoes_listadas):
    acao_maior_lucro = acoes_listadas[0]

    for acao in acoes_listadas[1:]:
        if acao['lucro']>acao_maior_lucro['lucro']:
            acao_maior_lucro = acao

    return acao_maior_lucro
```

Análise de Algoritmos

- Ex: Buscar pela ação com o maior lucro líquido na bovespa em 2017
  - Instrução mais relevante
    - Comparação dos lucros
  - Quantas vezes a instrução é executada
    - n-1 vezes
    - n = número de empresas
  - Função de custo de tempo
    - T(n) = n-1

Análise de Algoritmos

- Ex: Buscar pela ação com o maior lucro líquido na bovespa em 2017
  - Mas se eu estiver com sorte e a empresa com maior lucro estiver na primeira posição
  - Qual será o custo do algoritmo?

Análise de Algoritmos

- Ex: Buscar pela ação com o maior lucro líquido na bovespa em 2017
  - Mas se eu estiver com sorte e a empresa com maior lucro estiver na primeira posição
  - Qual será o custo do algoritmo?
    - Continua sendo T(n) = n-1

Análise de Algoritmos

- Ex: Buscar pela ação com lucro líquido igual a 200 milhões
  - Analise!

Análise de Algoritmos

- Ex: Buscar pela ação com lucro líquido igual a 200 milhões

```
def buscaLucro(acoes_listadas,lucro):
    i = 0
    while i<len(acoes_listadas) and acoes_listadas[i]['lucro']!=lucro:
        i+=1
    if i == len(acoes_listadas):
        return None
    else:
        return acoes_listadas[i]
```

Análise de Algoritmos

- Ex: Buscar pela ação com lucro líquido igual a 200 milhões
  - Instrução mais relevante
    - Comparação dos lucros com o lucro buscado
  - Quantas vezes a instrução é executada
    - n vezes
    - n = número de empresas
  - Função de custo de tempo
    - T(n) = n

Análise de Algoritmos

- Ex: Buscar pela ação com lucro líquido igual a 200 milhões
  - Mas se eu estiver com sorte e a empresa com o lucro buscado estiver na primeira posição
  - Quantas comparações eu irei fazer?
    - Neste caso, 1 comparação

Análise de Algoritmos

- Ex: Buscar pela ação com lucro líquido igual a 200 milhões
  - Se eu buscar um valor que nenhuma empresa apresente, qual será o custo do algoritmo?
    - n comparações

Análise de Algoritmos

- Neste segundo exemplo temos situações diferentes dependendo da instância do problema
  - Pior caso – pior situação, maior tempo de execução
  - Melhor caso – melhor situação, menor tempo
  - Caso médio – média de tempos de execução para entradas de tamanho n

Análise de Algoritmos

- Para o exemplo da busca
  - Pior caso – T(n) = n
  - Melhor caso – T(n) = 1
  - Caso médio – T(n) = (n+1)/2

Exercícios

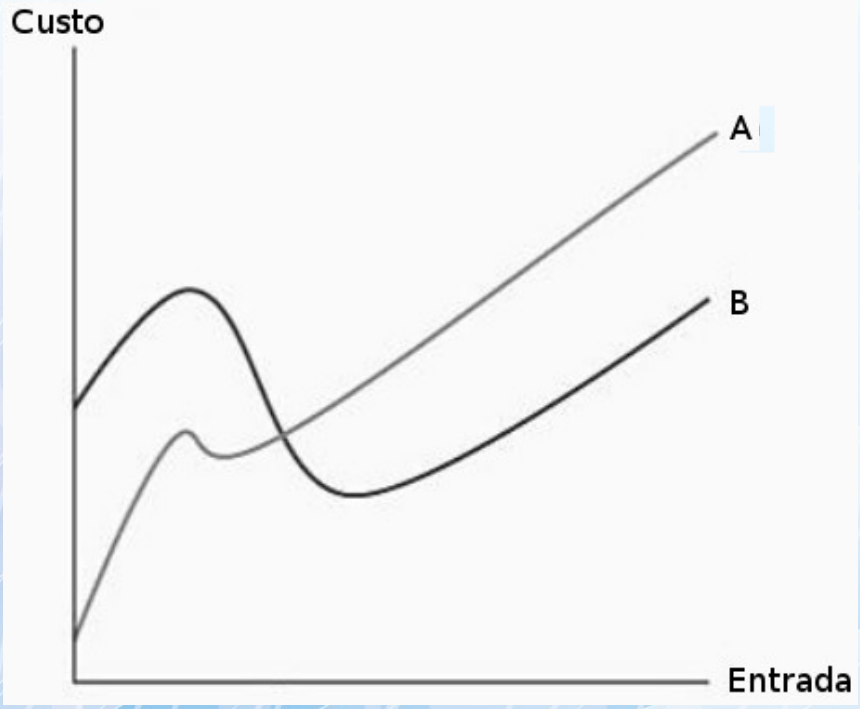
- Escreva o algoritmo e a função de custo de tempo deste para os seguintes problemas
- 1. Calcular o fatorial de um número
- 2. Contar o número de elementos negativos em um conjunto
- 3. Identificar os valores de um conjunto que estão abaixo da média do conjunto
- 4. Identificar a soma máxima entre dois elementos de um conjunto
- 5. Copiar uma lista de inteiros, retirando elementos repetidos

Análise Assintótica

- Na verdade não analisámos a eficiência de um algoritmo pelo número de execuções
  - Mas sim pela função de custo
- O que importa:
  - Crescimento da função de custo com o crescimento da entrada
- Base para a decisão de qual algoritmo utilizar:
  - Algoritmo com função de custo que cresce menos
    - Com o crescimento da entrada

Análise Assintótica

- Dados dois algoritmos A e B com as seguintes funções de custo:

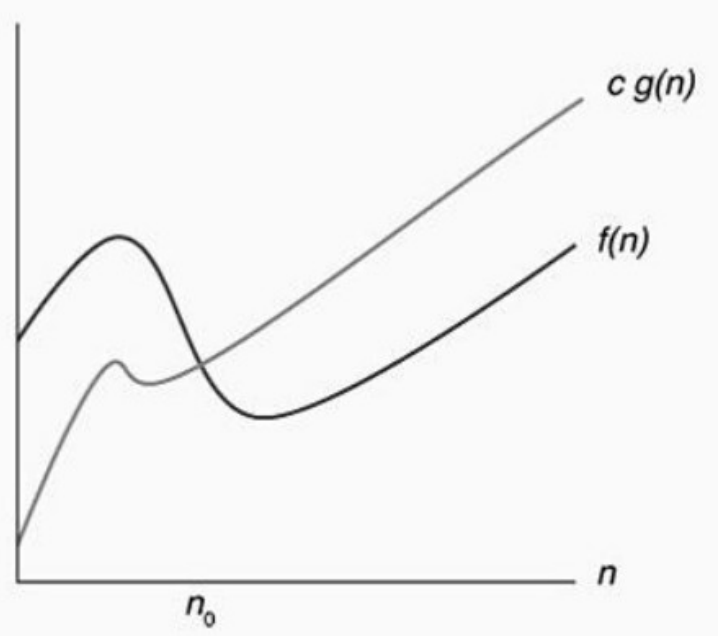


- Qual dos dois é melhor?

Análise Assintótica

- Dominação assintótica

  Uma função g(n) domina assintoticamente outra função f(n) se existem duas constantes positivas *c* e n_0 tais que, para $n \geq n_0$ temos que $|f(n)| \leq c.|g(n)|$



Análise Assintótica

- Ex: algoritmo A com custo n e algoritmo B com custo n^2.
  - Quem domina quem?
- Ex: algoritmo A com custo $(n+1)^3$ e algoritmo B com custo n^3
  - Quem domina quem?

Análise Assintótica

- Notação O (Big O ou O grande)
  - Uma função f(n) é O(g(n)) se existem duas constantes positivas *c* e n_0 tais que:
    - f(n)≤c.g(n), para todo $n \geq n_0$
  - Notação usada para dar limite assintótico superior
    - Pior caso do algoritmo
  - Ex: f(n) = 1/3n^2 – 3n é $O(n^2)$
    - c = 1/3, $n_0 = 1$
  - Ex: f(n) = $(n+1)^2$ é $O(n^2)$
    - c = 3, n0 = 2
  - Ex: f(n) = $2n^3 + 3n^2+n$ é $O(n^3)$
    - C = 4, n0 = 8
  - Ex: f(n) = n é $O(n^2)$