Técnicas e Análise de Algoritmos

Apresentação da Disciplina e Introdução

Professor: **Jeremias Moreira Gomes**
E-mail: jeremias.gomes@idp.edu.br

Introdução

Introdução - Sumário

- Introdução
- Informações Essenciais
- Metodologia
- Técnicas e Análise de Algoritmos
- Conclusão

Informações Essenciais

Informações Sobre a Disciplina

- Nome da Disciplina: Técnicas de Programação e Análise de Algoritmos
- Horários: Segundas (10 às 12h) e Quartas (08 às 10h)
- Locais:
  - Aulas Teóricas e Aulas Teórico-práticas: **sala 105**
  - Aulas Práticas: **Laboratório Dell**

Informações Sobre a Disciplina

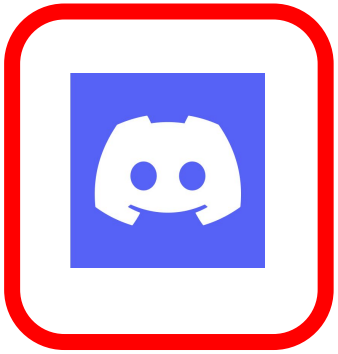
- Professor: Jeremias Moreira Gomes
- E-mail: jeremias.gomes@idp.edu.br
  - Ao enviar um email, incluir a seguinte *tag* no assunto:
    - [IDP-TAA-2025-01]

Informações Sobre a Disciplina

- Meios de Comunicação
  - Oficial: Canvas
  - Extraoficial: o que vocês têm utilizado?

discord.gg/JTwkRGUsWH

Entrada no servidor



52c83822f684

Mudança de papel

Organização das Aulas

- Aulas teóricas e práticas:
  - Aulas de segunda-feira:
    - Aulas Teórico-Práticas (primeiras semanas, mais teoria)
  - Aulas de quarta-feira:
    - Aulas Teóricas
- Exceções:
  - Provas (a ser informado durante o semestre)

Laboratório de Informática - DELL (Windows)

- Local: <vocês conhecem o IDP (provavelmente) melhor que eu>
- Proibido consumir alimentos dentro do laboratório.
- Não realizar alterações nos equipamentos sem autorização.
- Particularidades das aulas:
  - Interferência dos monitores
  - Instalação de softwares
  - Persistência de arquivos

Apoio à Aprendizagem (1/2)

- Este curso possui uma plataforma interativa de apoio à aprendizagem chamada Canvas.
  - Slides das aulas
  - Material Complementar
  - Fórum (para avisos)
  - Biblioteca Virtual

Apoio à Aprendizagem (2/2)

- Esta disciplina utilizará uma plataforma auxiliar para apoio à prática de programação chamada Codeforces:
  - A correção de exercícios e provas é feita de forma automática
  - **PLÁGIO é crime contra a propriedade intelectual**
    - Produza seus próprios códigos
    - Tirar dúvidas com os colegas é motivado, mas copiar indiscriminadamente não

Objetivos da Disciplina

- **Objetivos Gerais**
  - Estudar os métodos de análise de algoritmos e modelagem de problemas reais em termos computacionais

Objetivos da Disciplina

- **Objetivos Específicos** (1/2)

  a. Analisar soluções algorítmicas quanto a recursos de tempo e espaço em termos assintóticos

  b. Conhecer e entender os problemas clássicos em computação e as soluções envolvidas

  c. Ser capaz de resolver problemas rapidamente por meio de projeto de algoritmos

Objetivos da Disciplina

- **Objetivos Específicos** (2/2)

  d. Ser capaz de escolher soluções eficientes dentro das restrições de problemas computacionais propostos

  e. Dominar paradigmas de projetos de algoritmos envolvendo divisão e conquista, algoritmos gulosos e programação dinâmica

Ementa da Disciplina (1/3)

I. Análise de Complexidade
   A. Fundamentos
   B. Melhor caso, pior caso e caso médio
II. Busca e Ordenação
   A. Algoritmos de Busca
   B. Algoritmos de Ordenação Quadráticos
   C. Algoritmos de Ordenação em Tempo Linear

Ementa da Disciplina (2/3)

III. Árvores
    A. Florestas
    B. Árvore Geradora Mínima
    C. Filas de Prioridade
    D. Árvores Binárias de Busca com Balanceamento
IV. Hashing
    A. Tabelas de Hash
    B. Colisões

Ementa da Disciplina (3/3)

V. Grafos
   A. Buscas em Grafos
   B. Caminhos de Custo Mínimo
   C. Árvores Geradoras Mínimas

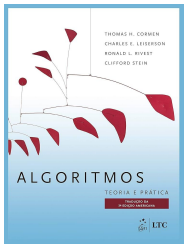
VI. Estratégias Algorítmicas
   A. Backtracking
   B. Programação Dinâmica

Referências Utilizadas na Disciplina

- CORMEN, Thomas H. et al. Algoritmos: teoria e prática. 3. ed. São Paulo: GEN LTC, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788595158092
- CORMEN, Thomas H. Desmistificando algoritmos. Rio de Janeiro: GEN LTC, 2013. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788595153929
- KNUTH, Donald Ervin et al. The art of computer programming. Reading, MA: Addison-Wesley, 1973

Referências Utilizadas na Disciplina

- **CORMEN, Thomas H. et al. Algoritmos: teoria e prática. 3. ed. São Paulo: GEN LTC, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788595158092**
- CORMEN, Thomas H. Desmistificando algoritmos. Rio de Janeiro: GEN LTC, 2013. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788595153929
- KNUTH, Donald Ervin et al. The art of computer programming. Reading, MA: Addison-Wesley, 1973



Metodologia

Metodologia

- Aula Expositiva Dialogada
  - Participação é motivada
    - Desde que focada no conteúdo da aula.
  - Exploração e Confronto de Ideias
- Listas de Exercícios (atividade em aula e lista extraclasse)
- Provas

Sistema de Avaliação

- A avaliação do aluno consistirá em:
  - Avaliações Somativas (provas ou AS)
  - Listas de Exercícios (LE)
    - Listas de Exercícios de Atividade em Aula (LEA)
    - Listas de Exercícios Extraclasse (LEE)

Sistema de Avaliação

- **Provas**
  - Serão aplicadas duas provas
  - Interpretação de texto é essencial
- Prova 01 (AS01): **Dia 23 de abril de 2025**
- Prova 02 (AS02): **Dia 23 de junho de 2025**

Sistema de Avaliação

- **Listas de Exercícios**
  - Sempre que um novo conteúdo for apresentado, duas listas (uma LEE e uma LEA) sobre o mesmo serão aplicadas
- O objetivo (idealização) é que a **cada quarta-feira alternada**, a aula será uma LEA que será iniciada e terminada em sala de aula
- Já as LEE ficarão abertas para serem realizadas por, em média, uma semana

Nota Final

- A Nota Final (NF) do aluno será calculada da seguinte forma:

$$NF = \frac{AV_1 + AV_2}{2}$$

Nota Final

- Onde AV é uma composição (não matemática) da prova (AS) daquela parte e a média aritmética das listas de exercícios, também aplicada naquela parte (NLE), dado pelo seguinte:

$$NF = \frac{AV_1 + AV_2}{2}$$

$$AV_1 = 0.4 * NLE_1 + 0.6 * AS_1$$

$$AV_2 = 0.4 * NLE_2 + 0.6 * AS_2$$

Nota Final - Pontos Extra

- Neste semestre, **é possível ganhar pontos extra**:
  - Caso o aluno participe dos contest e demonstre bom desempenho na plataforma codeforces (individual), esta será convertida em pontos

$$AV_1 = 0.4 * NLE_1 + 0.6 * AS_1 + AB_1$$

$$AV_2 = 0.4 * NLE_2 + 0.6 * AS_2 + AB_2$$

Nota Final - Pontos Extra

- **Bonificações Possíveis:**
  - 03 contests como pupil: 5% extra
  - 03 contests como specialist: 20% extra
  - 03 contests como expert: 50% extra
  - 03 contests como master candidate: 100% extra (passa sem fazer prova com total!)

Aprovação na Disciplina

- É considerado aprovado o aluno em que $NF \geq 6.0$, e
- Frequência mínima de 75%
- Há ainda a possibilidade de revisão dos resultados (também respeitando o requerimento dentro do prazo)

Prova Substitutiva

- O IDP prevê a aplicação de **uma única prova substitutiva**, caso o aluno não possa realizar alguma das duas provas aplicadas durante o semestre
    - Ou seja, **o aluno pode perder somente uma prova**, pelo motivo que for
- O conteúdo dessa prova é todo o material aplicado no semestre
- Data da prova substitutiva: **30 de junho**

Frequência Discente e Atestados

- 15 min de tolerância para atraso
- Chamada automática por reconhecimento facial
  - Necessário 75% do tempo na sala de aula
  - No início terá lista de chamada também (não confio muito em tecnologia)
- Atestados:
  - 5 dias para a entrega dos atestados médicos
  - Devem ser entregues na Central de Relacionamento

Técnicas e Análise de Algoritmos

Sobre a Disciplina

“A disciplina de Técnica e Análise de Algoritmos (TAA) tem como objetivo fornecer uma compreensão profunda das técnicas fundamentais utilizadas na concepção, desenvolvimento e análise de algoritmos eficientes.”

- Compreender as principais estratégias de projeto de algoritmos
- Analisar a eficiência de algoritmos
- Adquirir habilidades para resolver problemas complexos através da seleção e adaptação de algoritmos apropriados

Algoritmo

“Informalmente, um Algoritmo é um procedimento para resolver um problema.”

- Um problema pode ser descrito como uma conjunto de instâncias de entrada e a respectiva saída de determinada Instância

Introdução

- **Considere um problema de localizar números em uma lista**
  - Entrada: Sequência de n elementos $\{a_0, a_1, \ldots, a_n\}$ e um inteiro v, a ser procurado na lista
  - Saída: Verdadeiro, se o elemento se encontrar na lista, ou falso caso contrário

Qual o papel dos algoritmos no mundo?

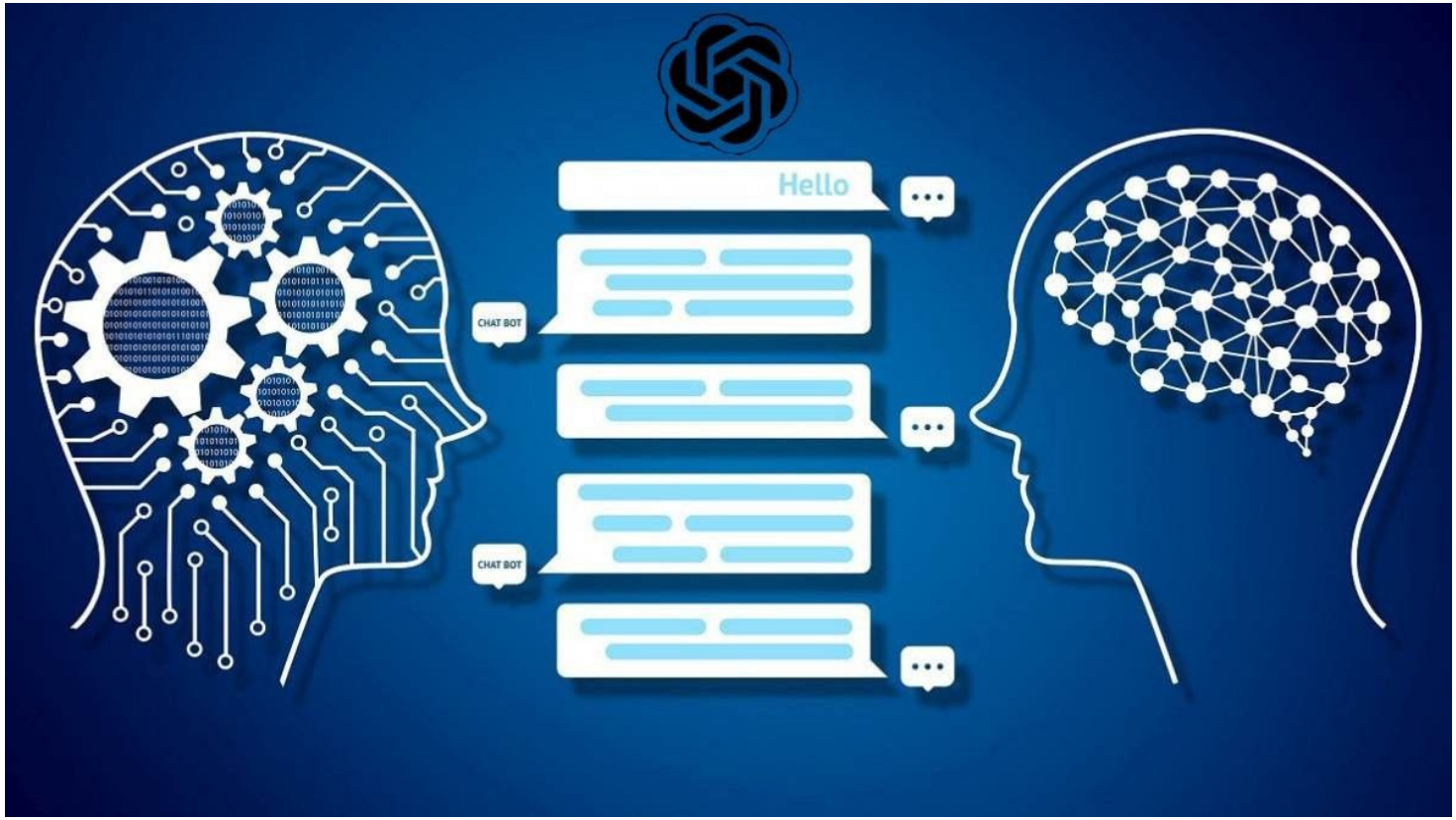
Redes de Computadores

Sistemas Operacionais

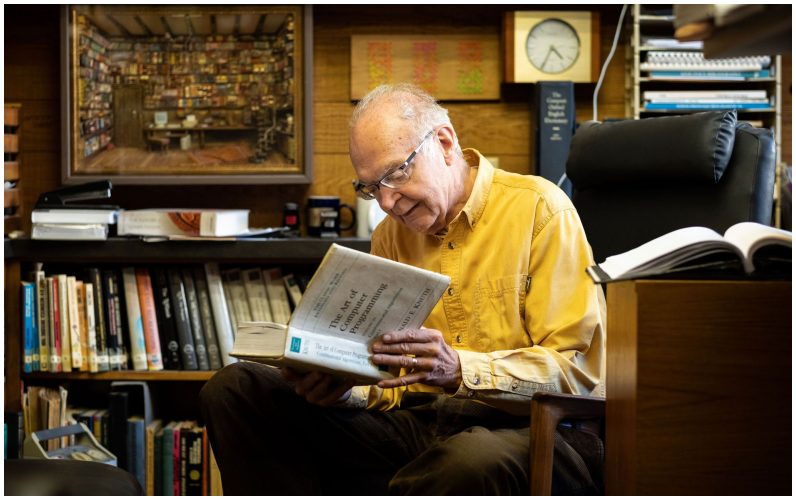
Criptografia

Processamento de Sinais e Imagens

Large Language Models (LLMs)



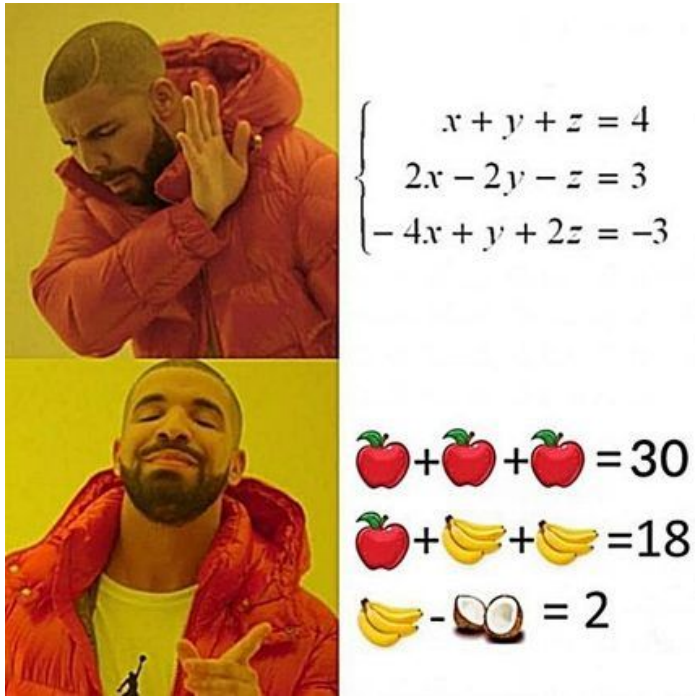
Algoritmos são realmente importantes?



“Computer Science is algorithms. That’s it. Not just theory. All. Everything. If there is not an algorithm lurking around, then it is not Computer Science.”

Motivação

- **Criação de algoritmos está ligado a resolução de problemas**



Motivação

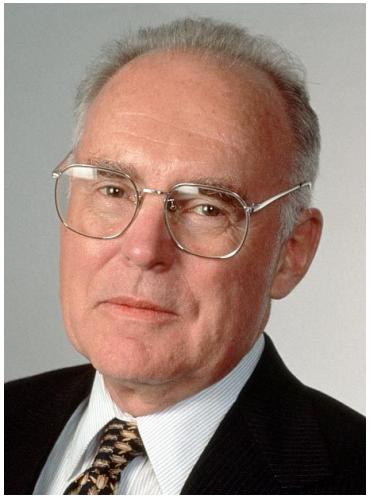
- **Por que precisamos conhecer problemas?**
  - Diversos problemas entram na classe de **problemas bem conhecidos**
  - Soluções para esses problemas tem ficado cada vez mais complexas
  - Como executar uma tarefa mais rápido?

Motivação

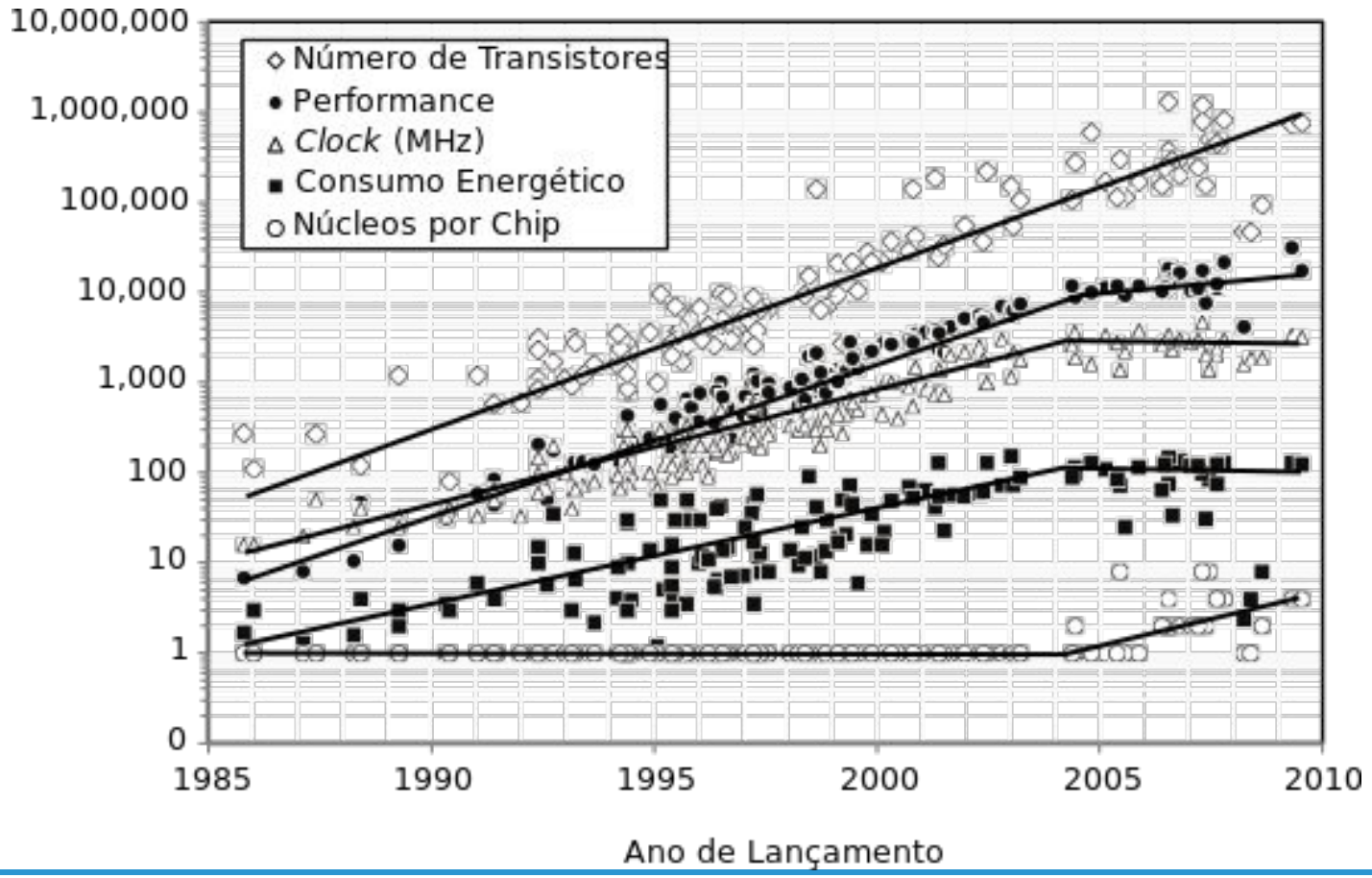
- **Como executar uma tarefa mais rápido?**
  - Existem, normalmente, três formas de se executar uma tarefa mais rápido:
    i. Trabalhar mais rápido
    ii. Solicitar apoio
    iii. Trabalhar de maneira mais inteligente

i - Trabalhar mais rápido

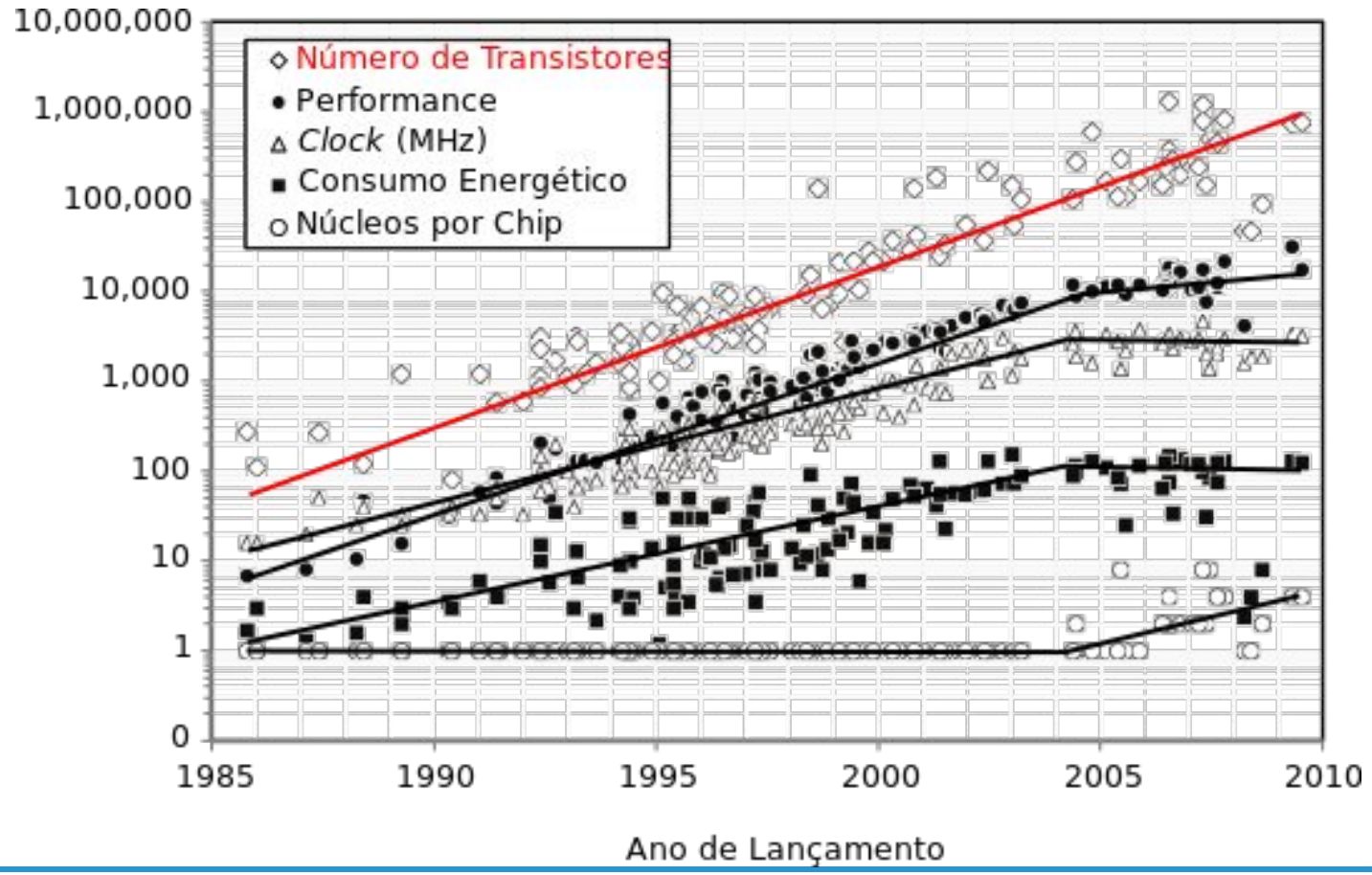
- “Lei de Moore”:
  - “A complexidade para componentes com custos mínimos tem aumentado em uma taxa de aproximada um fator de dois por ano.”



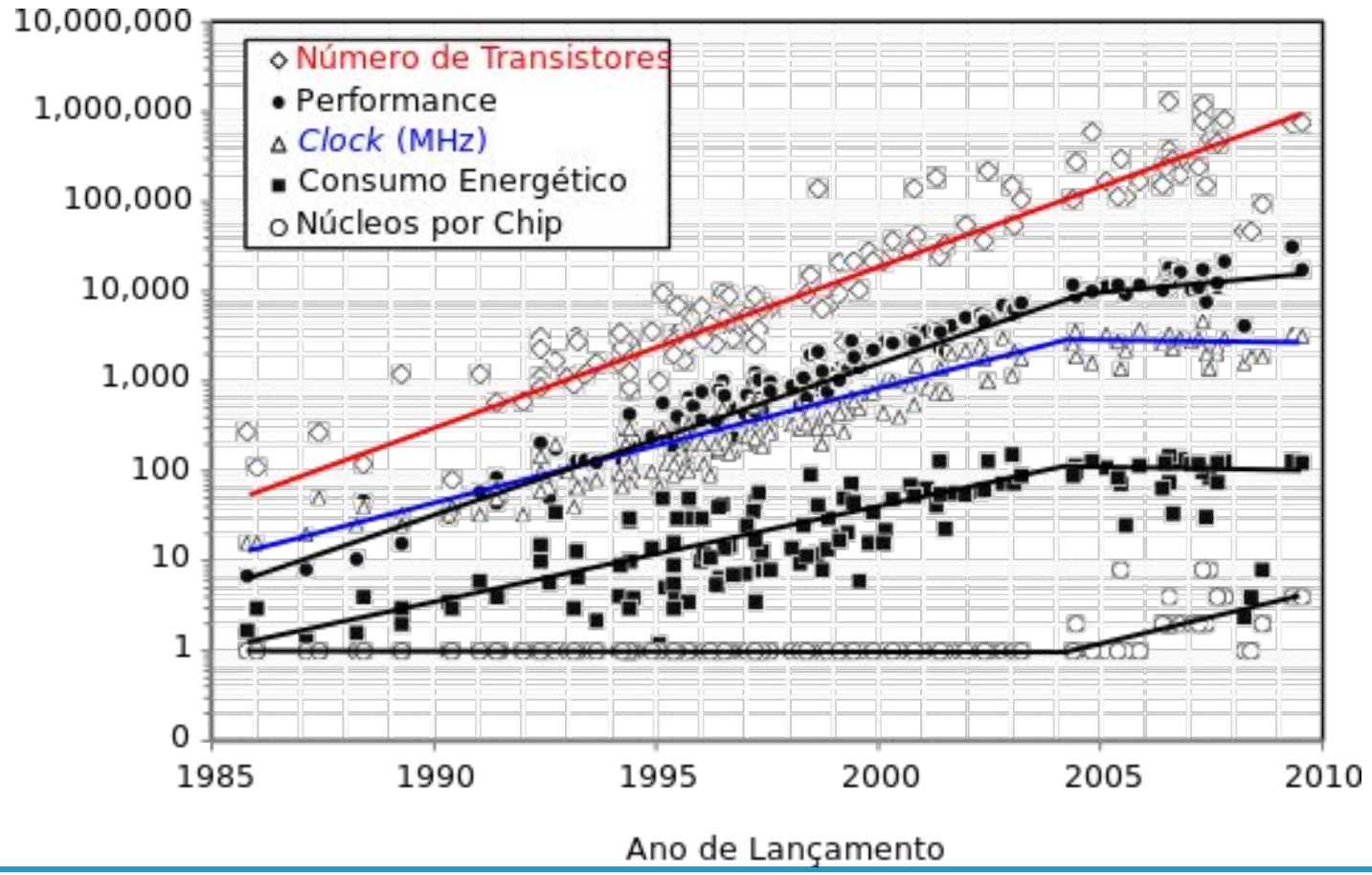
Performance Computacional



Performance Computacional



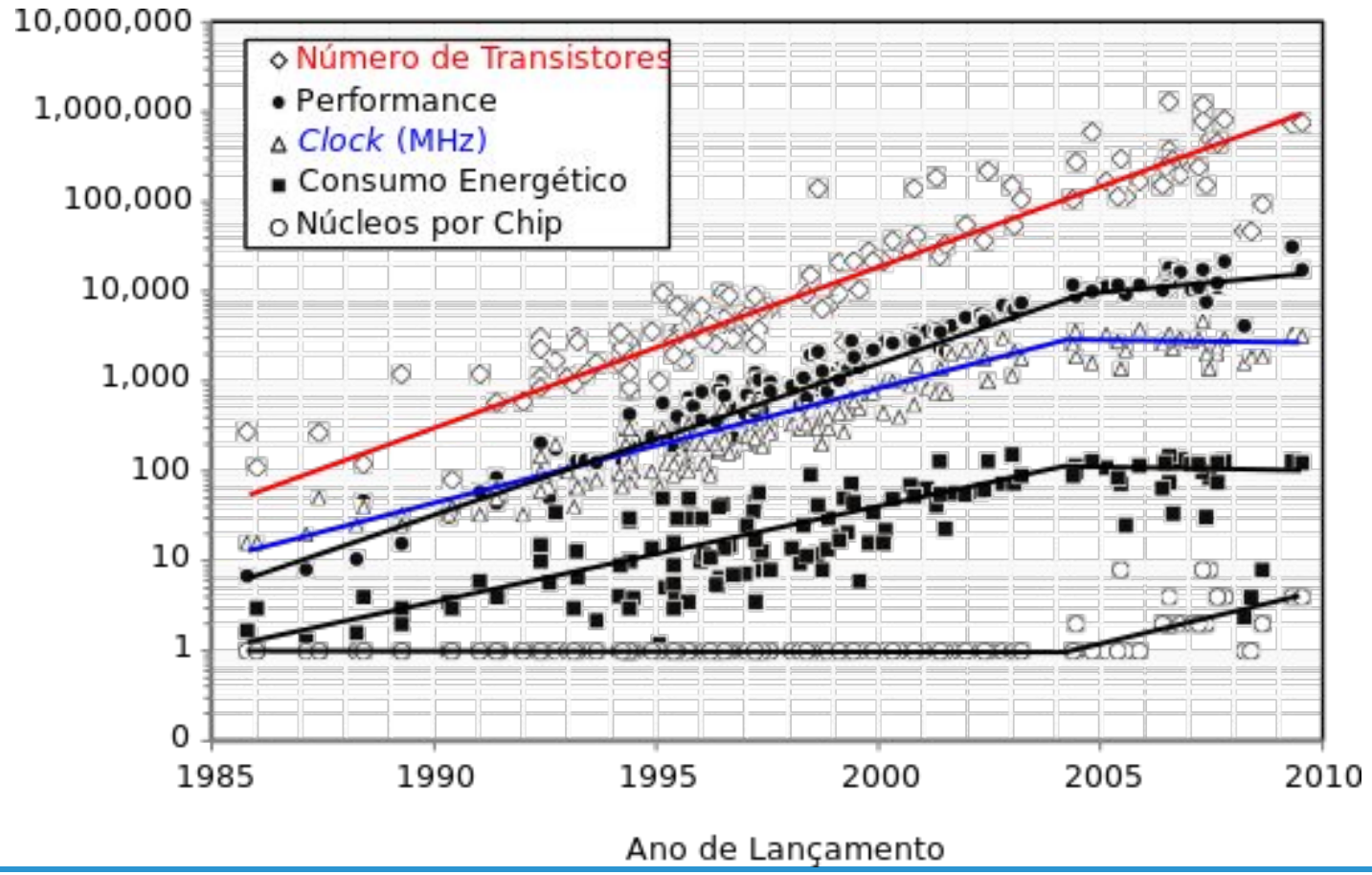
Performance Computacional



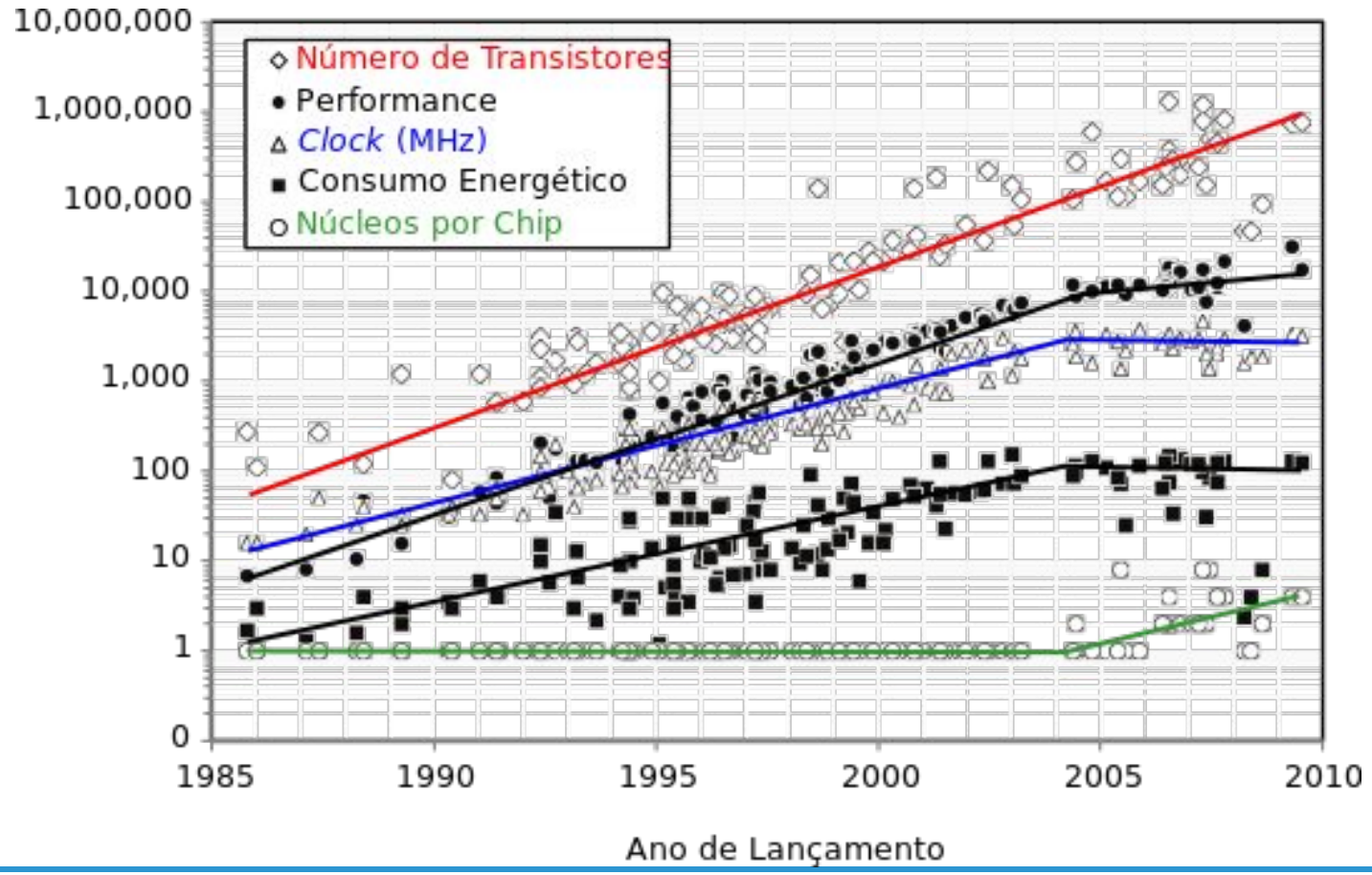
ii - Solicitar apoio

- Infelizmente, o aumento do número de transistores por área possui um limite físico
  - Problemas de geração e dissipação de calor
- A saída da indústria para superar foi disponibilizar mais núcleos nos processadores
- Além disso, conectar um ou mais dispositivos para trabalhar de maneira conjunta
  - Computação distribuída

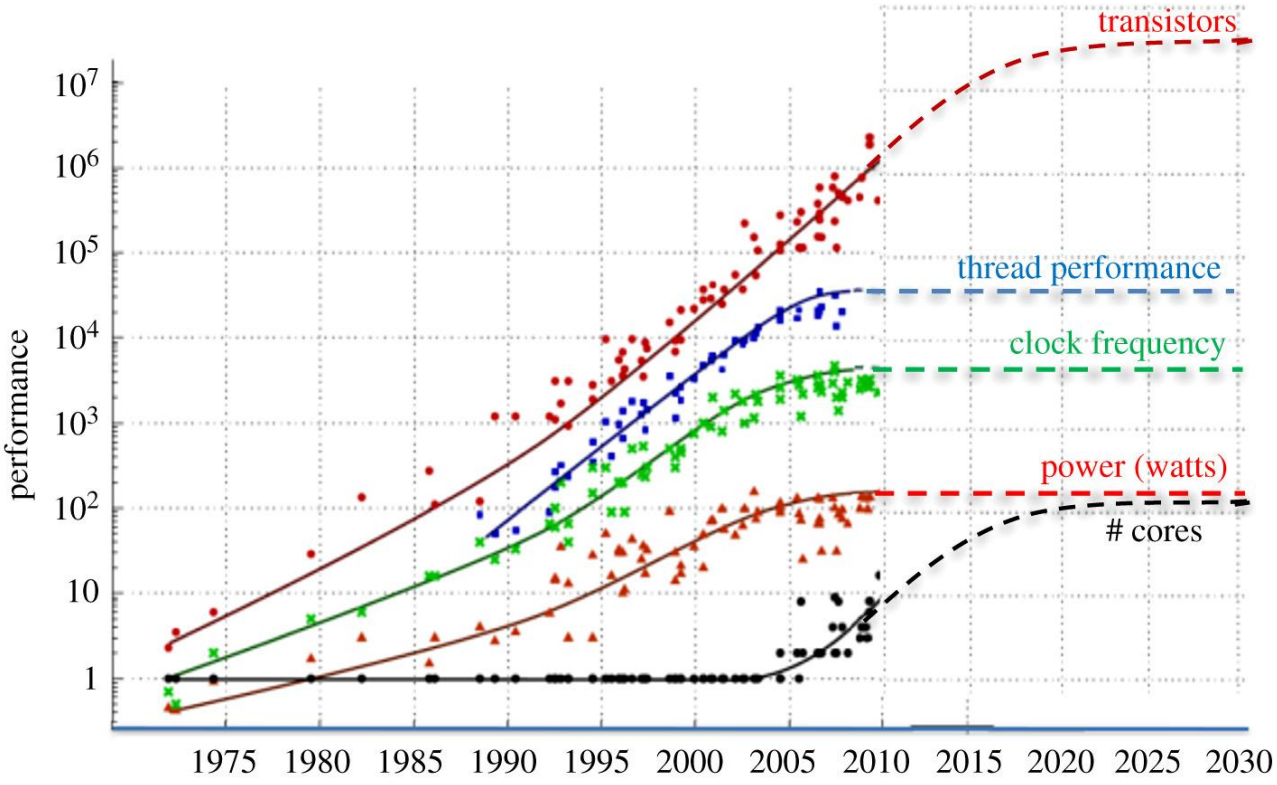
Performance Computacional



Performance Computacional



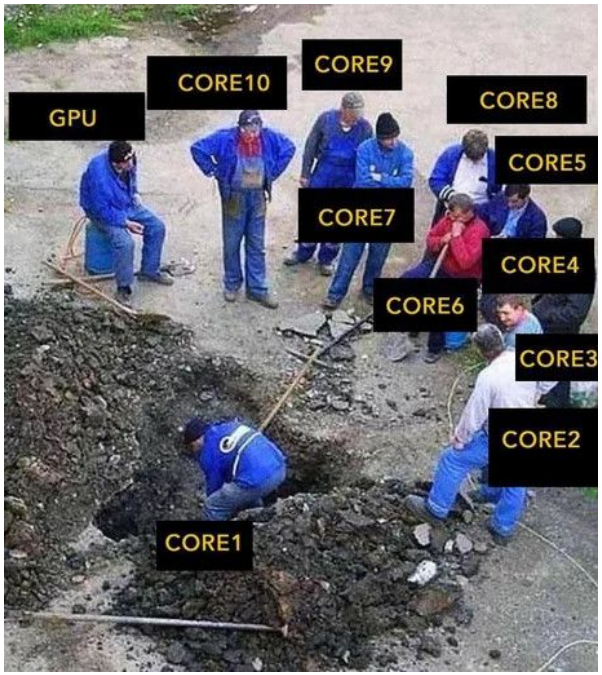
Performance Computacional



Fonte: The future of computing beyond Moore's Law

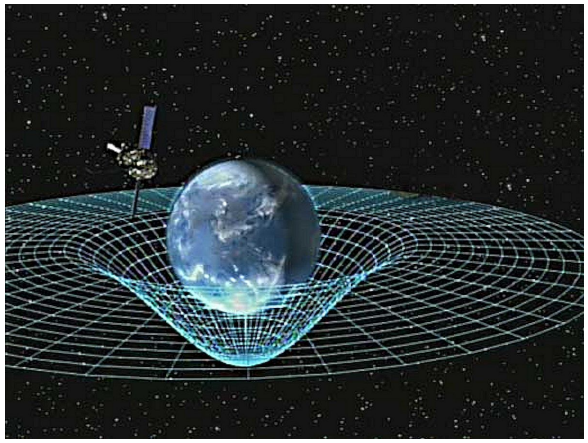
ii - Solicitar apoio

- Uma consequência (aumento de núcleos) é a necessidade de programação específica para arquiteturas paralelas



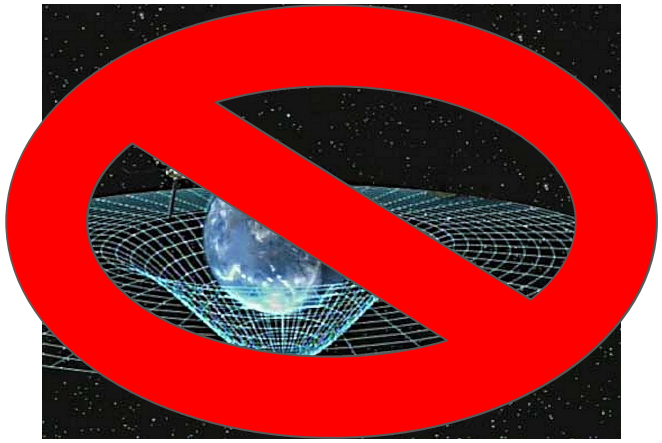
iii - Trabalhar de Maneira mais Inteligente

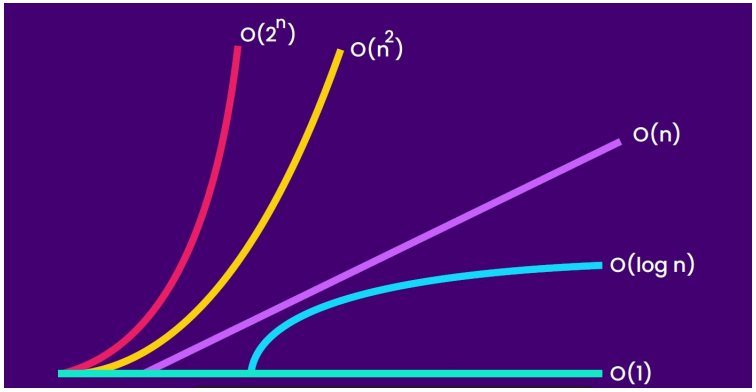
- Busca pelos algoritmos que melhor resolva o problema
  - Área conhecida como **Análise de Algoritmos**
  - Nesse contexto, o que quer dizer “melhor”?
    i. Conhecimento e controle de tempo e espaço



iii - Trabalhar de Maneira mais Inteligente

- Busca pelos algoritmos que melhor resolva o problema
  - Área conhecida como **Análise de Algoritmos**
  - Nesse contexto, o que quer dizer “melhor”?
    i. Conhecimento e controle de tempo e espaço





Motivação

- **Como executar uma tarefa mais rápido?**

  a. **Trabalhando mais rápido**

     - Aumento do *clock*.

  b. **Solicitando apoio**

     - Executar, de alguma forma, em paralelo

  c. **Trabalhando de maneira mais inteligente**

Disciplina Arquitetura de Computadores

Disciplinas de Programação Concorrente, Paralela e Distribuída

Essa disciplina!!!

Análise de Algoritmos

- Analisar algoritmos é essencial para que se possa prever recursos, tais como tempo e espaço, a serem utilizados
- Para realizar uma análise, deve-se definir um modelo computacional*:
  - Máquinas de Turing
  - Cálculo-**λ**
  - Código de Máquina
  - Código em C
  - etc

* Diferentes modelos computacionais possuem diferentes complexidades nas operações

Modelo RAM

- O Modelo RAM corresponde bem à maioria das Arquiteturas encontradas no mundo real
  - Instruções aritméticas (`add, div,. . .`)
  - Instruções de dados (`load, str, cpy`)
  - Instruções de controle(`branches, loops, . . .`)
- Cada operação básica leva um passo e, neste modelo, o tempo é mensurado pelo número de passos
- Já o espaço, é mensurado pelo número de posições de memória utilizados

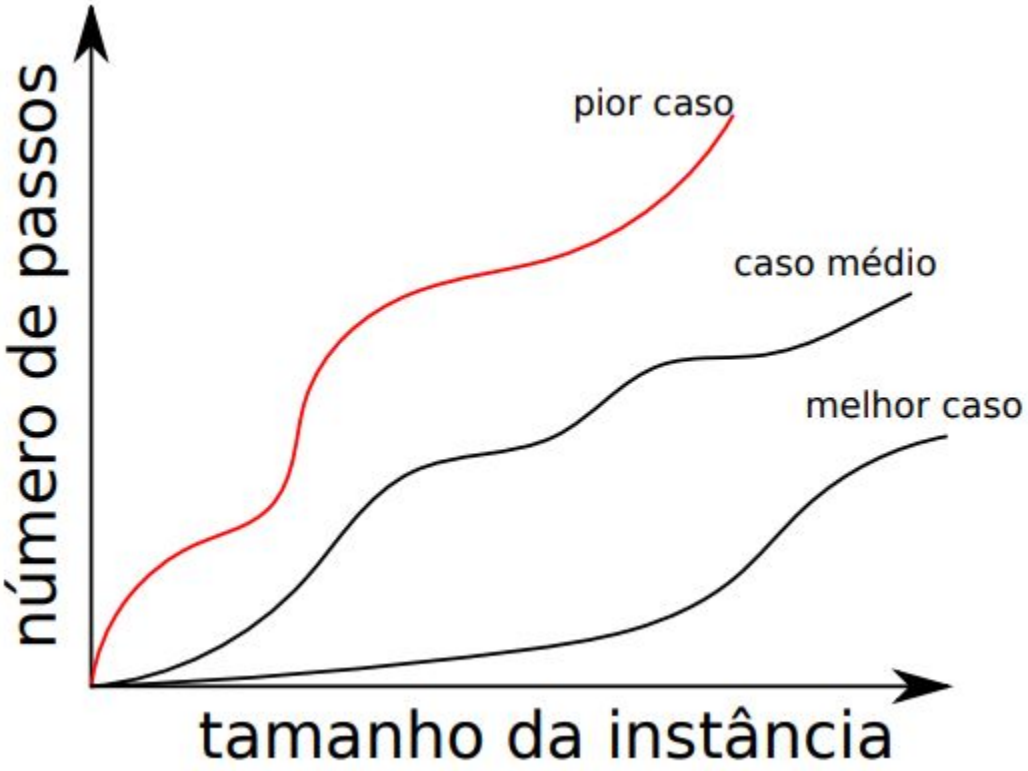
Modelo de Cenários

- Uma outra forma de avaliar cenários de desempenho e eficiência, é verificar operações em relação ao tamanho de entrada
- Este modelo examina o comportamento em diferentes cenários:
  - Pior Caso
  - Caso Médio
  - Melhor Caso

Pior Caso, Caso Médio e Melhor Caso

- O **pior caso** de complexidade de um algoritmo é a função definida pelo maior número de passos com tamanho de instância n
- O **melhor caso** de complexidade é dada pela função definida pelo menor número de passos com tamanho de instância n
- O **caso médio** de complexidade do algoritmo é dado por uma função definida pela média do número de passos sobre todas as instâncias de tamanho n

Pior Caso, Caso Médio e Melhor Caso



Bubble Sort

- É um dos algoritmos mais simples de ordenação
- Utiliza a ideia de flutuação para deslizar os maiores valores para o fim da lista
    - Em um recipiente com diversas bolhas, estas tendem a se ajustar por nível, deslizando umas pelas outras

Bubble Sort

- O método inicia em ler todo um vetor, comparando elementos vizinhos entre si
  - Sempre que dois elementos estão fora de ordem, eles trocam a posição entre si
  - O algoritmo vai checando todos os pares do início ao fim do vetor

Bubble Sort

- Exemplo de ordenação em ordem crescente.

(5 1 4 2 8)

Bubble Sort

- Exemplo de ordenação em ordem crescente.

(5 1 4 2 8)

Bubble Sort

- Exemplo de ordenação em ordem crescente.

```
( 5 1 4 2 8 ) -> ( 1 5 4 2 8 ), Compara o primeiro par de elementos (5 e 1)
                                e os troca por o 5 ser maior que 1.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente.

```
( 5 1 4 2 8 ) -> ( 1 5 4 2 8 ), Compara o primeiro par de elementos (5 e 1)
                                e os troca por o 5 ser maior que 1.
( 1 5 4 2 8 ) -> ( 1 4 5 2 8 ), O próximo par é (5 e 4) que também estão
                                desordenados e são trocados.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente.

```
( 5 1 4 2 8 ) -> ( 1 5 4 2 8 ), Compara o primeiro par de elementos (5 e 1)
                                e os troca por o 5 ser maior que 1.
( 1 5 4 2 8 ) -> ( 1 4 5 2 8 ), O próximo par é (5 e 4) que também estão
                                desordenados e são trocados.
( 1 4 5 2 8 ) -> ( 1 4 2 5 8 ), (5 e 2) também são trocados, por estarem
                                desordenados.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente.

```
( 5 1 4 2 8 ) -> ( 1 5 4 2 8 ), Compara o primeiro par de elementos (5 e 1)
                                e os troca por o 5 ser maior que 1.
( 1 5 4 2 8 ) -> ( 1 4 5 2 8 ), O próximo par é (5 e 4) que também estão
                                desordenados e são trocados.
( 1 4 5 2 8 ) -> ( 1 4 2 5 8 ), (5 e 2) também são trocados, por estarem
                                desordenados.
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Por úlitmo, são comparados os valores do par
                                (5, 8), porém, como estes já estão ordenados,
                                nada acontece (feijoada).
```

Bubble Sort

- Exemplo de ordenação em ordem crescente.

(5 1 4 2 8) -> (1 5 4 2 8), Compara o primeiro par de elementos (5 e 1) e os troca por o 5 ser maior que 1.

(1 5 4 2 8) -> (1 4 5 2 8), O próximo par é (5 e 4) que também estão desordenados e são trocados.

(1 4 5 2 8) -> (1 4 2 5 8), (5 e 2) também são trocados, por estarem desordenados.

(1 4 2 5 8) -> (1 4 2 5 8), Por úlitmo, são comparados os valores do par (5, 8), porém, como estes já estão ordenados, nada acontece (feijoada).

Bubble Sort

- Após a execução de todos esses passos, o vetor está mais próximo de estar ordenado, mas ainda não está.
  - Repare que conseguimos colocar o 8 na última posição.

(1 4 2 5 8)

  - E se aplicarmos o passo de deslizar novamente?
    - Spoiler: mais valores vão ficar ordenados.

Bubble Sort

- Exemplo de ordenação em ordem crescente. Segunda passada.

(1 4 2 5 8)

Bubble Sort

- Exemplo de ordenação em ordem crescente. Segunda passada.

```
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Nenhuma troca acontece.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente. Segunda passada.

```
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Nenhuma troca acontece.

( 1 4 2 5 8 ) -> ( 1 2 4 5 8 ), Nesse passo, o par (4 e 2) estão
                                desordenados e são trocados.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente. Segunda passada.

```
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Nenhuma troca acontece.

( 1 4 2 5 8 ) -> ( 1 2 4 5 8 ), Nesse passo, o par (4 e 2) estão desordenados
                                e são trocados.
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente. Segunda passada.

```
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Nenhuma troca acontece.

( 1 4 2 5 8 ) -> ( 1 2 4 5 8 ), Nesse passo, o par (4 e 2) estão desordenados
                                e são trocados.
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.

( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
```

Bubble Sort

- Exemplo de ordenação em ordem crescente. Segunda passada.

```
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Nenhuma troca acontece.

( 1 4 2 5 8 ) -> ( 1 2 4 5 8 ), Nesse passo, o par (4 e 2) estão desordenados
                                e são trocados.
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.

( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
```

Bubble Sort

- Após a execução da segunda passada, o vetor coincidentemente encontra-se ordenado. Mas quando parar?

(1 2 4 5 8)

  - Se executar essas passagens **N** vezes, você irá deslocar todos os **N** até o seu devido lugar

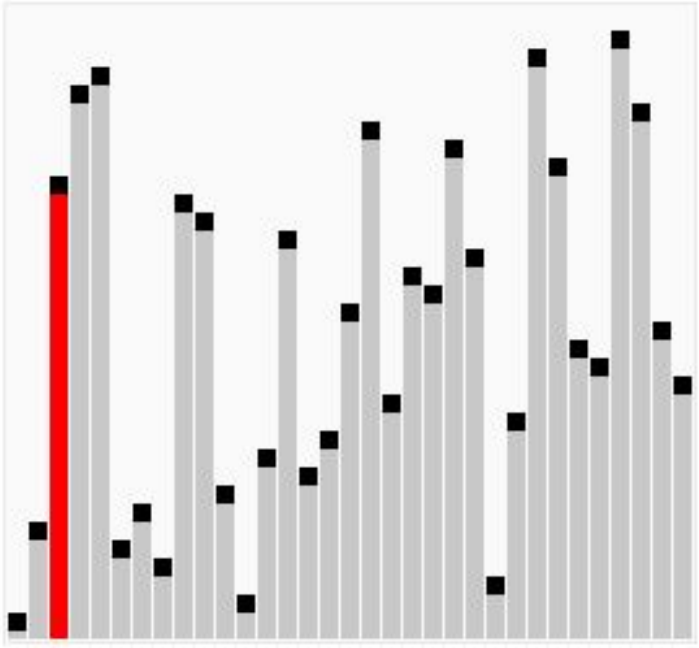
Bubble Sort

- Exemplo de ordenação em ordem crescente. Terceira passada.

```
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 ), Nenhuma troca acontece.
```

Análise de Complexidade do Bubble Sort

- Qual o melhor caso para o Bubble Sort?
- Qual o pior caso para o Bubble Sort?



Análise de Complexidade

- Contar precisamente o número de passos executados numa máquina pelo modelo RAM é muito trabalhoso e depende muitas vezes de detalhes específicos de implementação
- Na prática, o que se faz é colocar tudo em função de uma cota superior e uma cota inferior de complexidade de tempo e espaço
- Essa função e análise é chamada de **Notação Assintótica**

Notação Assintótica

- É uma ferramenta que simplifica o processo de análise de complexidade e permite compararmos a eficiência relativa de algoritmos
  - O
  - Ω
  - Θ
  - o
  - ω

Conclusão