Algoritmos

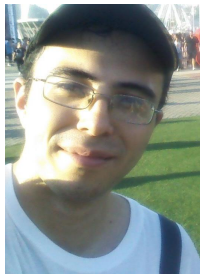
Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

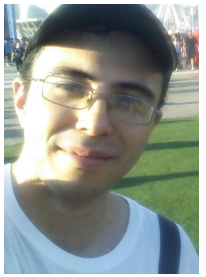
Qualquer erro é de minha responsabilidade.

Sobre o docente



- 2002 - 2004: Técnico em Programação e Desenvolvimento de Sistemas (CEFET-SP)
- 2006 - 2009: Bacharelado em Ciência da Computação (UNICAMP)
  - 2007-2008: IC - Algoritmos e Heurísticas para Empacotamento Tridimensional
  - 2008-2009: IC - Algoritmos e Heurísticas para o Problema de Roteamento Tridimensional
- 2009 - 2011: Mestrado em Ciência da Computação (UNICAMP) - O Problema do Caixeiro Viajante com Restrições de Empacotamento Tridimensional
- 2011 - 2016: Doutorado em Ciência da Computação (UNICAMP) - Algoritmos para Problemas com Restrições de Empacotamento

Sobre o docente



- 2016 - 2018: Pós-doutorado (UFSCar)
- 2018 - atual: Professor Adjunto no Instituto de Matemática e Computação da Universidade Federal de Itajubá
  - Programa de Pós-Graduação em Ciência e Tecnologia da Computação.
  - Orientador de IC, TFG e Pós-Graduação: Algoritmos, Otimização, Teoria dos Jogos, Aprendizado de Máquina, etc..
  - Coordenador do Projeto de Extensão DevU - Desenvolvimento de Jogos
  - Coordenador de Mobilidade Acadêmica dos Cursos de Sistemas de Informação e Ciência da Computação.

O que é um algoritmo?

- Um conjunto de passos bem definidos para resolver um problema computacional.

Exemplos:

- Você tem vários números e precisa deles ordenados.
- Você tem um mapa e precisa encontrar o menor caminho entre uma origem e um destino.
- Você tem várias tarefas distintas, cada uma com uma data de entrega e cada uma demora um certo tempo para ser realizada. E você quer completar todas sem atraso.

Etimologia

- Uma provável origem da palavra algoritmo é a latinização do nome do persa *Muhammad ibn Musa al-Khwarizmi*
- Matemático, astronomo, geógrafo, e acadêmico na *House of Wisdom* em Bagdá.
- Por volta de 825 ele escreveu um tratado sobre o sistema numérico Arábico-Hindu que foi traduzido para o Latin no século 12 com o título *Algoritmi de numero Indorum*. Com o sentido de *Algoritmi (al-Khwarizmi) sobre os números dos Hindus.*

Por que estudar algoritmos?

Entender as bases de algoritmos e estruturas de dados é essencial para fazer um trabalho sério em qualquer ramo da ciência da computação.
Exemplos:

- Roteamento de redes, utiliza princípios de algoritmos de caminhos mínimos
- Criptografia de chave pública se baseia em algoritmos de teoria dos números
- Computação gráfica precisa de algoritmos geométricos.
- Bancos de Dados se baseiam em árvores balanceadas.
- Biologia computacional usa programação dinâmica para medir a similaridade entre genomas.
- etc, etc, etc...

Por que estudar algoritmos?

São a chave fundamental para inovação tecnológica moderna.
Um exemplo importante:

- Os mecanismos de busca se baseiam nos fundamentos de algorítimos para computar de forma eficiente a relevância de várias páginas web para uma dada consulta.
- Desses o algoritmo mais famoso é o algoritmo *page rank* usado pelo Google.
- Talvez você já tenha pensado que a evolução dos hardwares é a única responsável pelo progresso da tecnologia. Porém não é bem assim.
- O próximo slide apresenta um trecho do relatório do conselho de ciência e tecnologia para a Casa Branca de dezembro de 2010: **Progress in Algorithms Beats Moore's Law**

Everyone knows Moore's Law – a prediction made in 1965 by Intel co-founder Gordon Moore that the density of transistors in integrated circuits would continue to double every 1 to 2 years. Fewer people appreciate the extraordinary innovation that is needed to translate increased transistor density into improved system performance. This effort requires new approaches to integrated circuit design, and new supporting design tools, that allow the design of integrated circuits with hundreds of millions or even billions of transistors, compared to the tens of thousands that were the norm 30 years ago. It requires new processor architectures that take advantage of these transistors, and new system architectures that take advantage of these processors. It requires new approaches for the system software, programming languages, and applications that run on top of this hardware. All of this is the work of computer scientists and computer engineers.

Even more remarkable – and even less widely understood – is that in many areas, performance gains due to improvements in algorithms have vastly exceeded even the dramatic performance gains due to increased processor speed.

The algorithms that we use today for speech recognition, for natural language translation, for chess playing, for logistics planning, have evolved remarkably in the past decade. It's difficult to quantify the improvement, though, because it is as much in the realm of quality as of execution time.

In the field of numerical algorithms, however, the improvement can be quantified. Here is just one example, provided by Professor Martin Grötschel of Konrad-Zuse-Zentrum für Informationstechnik Berlin. Grötschel, an expert in optimization, observes that a benchmark production planning model solved using linear programming would have taken 82 years to solve in 1988, using the computers and the linear programming algorithms of the day. Fifteen years later – in 2003 – this same model could be solved in roughly 1 minute, an improvement by a factor of roughly 43 million. Of this, a factor of roughly 1,000 was due to increased processor speed, whereas a factor of roughly 43,000 was due to improvements in algorithms! Grötschel also cites an algorithmic improvement of roughly 30,000 for mixed integer programming between 1991 and 2008.

The design and analysis of algorithms, and the study of the inherent computational complexity of problems, are fundamental subfields of computer science.

Por que estudar algoritmos?

- É desafiador!
  - Existem uma miríade de Algoritmos e Técnicas de Projeto de algoritmos que mal imaginamos. Veremos algumas das mais importantes nessa disciplina.
- É divertido!

Multiplicação de Inteiros

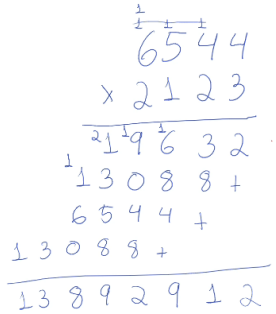
Vamos definir o problema da multiplicação de inteiros:

Multiplicação de Inteiros

Dado dois inteiros x e y de n dígitos cada. Encontrar o produto $x \cdot y$.

- Exemplo: 6544 e 2123
- Você provavelmente aprendeu no ensino fundamental a fazer essa conta. De fato o que você aprendeu foi um algoritmo que é capaz de resolver esse problema.

Multiplicação de 6544 e 2123 (!)

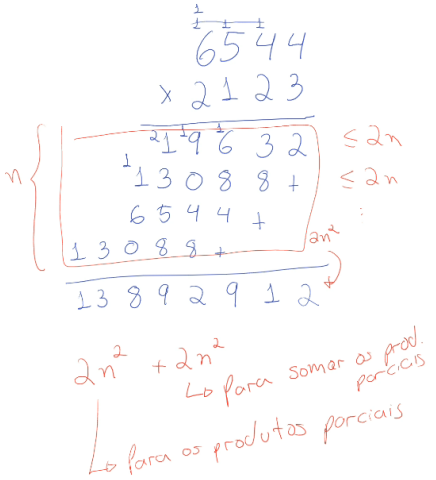


Resposta: 13892912

Multiplicação de Inteiros

- Intuitivamente você sabe que esse algoritmo está CORRETO, ou seja, dados quaisquer números x e y seguindo adequadamente os passos do algoritmo você terminaria com o produto de x e y.
- Mas quantas "contas"elementares tivemos que fazer para realizar essa multiplicação?
- Vamos chamar de operações elementares a soma ou multiplicação de números com um único digito. Quantas operações básicas são necessárias nesse algoritmo?

Número de operações na multiplicação de 6544 e 2123 (!)



Resposta: $4n^2$

Multiplicação de Inteiros

- **Conclusão:** Precisamos aproximadamente de uma constante (aprox. 4) vezes n^2 operações para realizar essa multiplicação. Então o que acontece se você dobrar o número de dígitos? E se você quadruplicar o número de dígitos?
- **Será que podemos fazer melhor?** De fato essa é a pergunta que faremos constantemente.

  Perhaps the most important principle for the good algorithm designer is to refuse to be content. (Aho, Hopcroft e Ullman. The Design and Analysis of Computer Algorithms, 1974)
- Veremos a seguir um algoritmo diferente (melhor?) para multiplicar inteiros.

Karatsuba

- Anatolii Alexeievitch Karatsuba, matemático nascido na União Soviética (1937-2008).
- Passou a maior parte da vida acadêmica na Faculdade de Mecânica e Matemática da Universidade Estadual de Moscow
- Descobriu em 1960 e publicou em 1962 um novo método que usa o paradigma de divisão e conquista para multiplicar números grandes.

Algoritmo de Karatsuba

Multiplicação de Inteiros

Dado dois inteiros x e y de n dígitos cada. Encontrar o produto $x \cdot y$.

- Dividir x em duas partes, digamos a e b de $n/2$ dígitos.

$$x = a \cdot 10^{n/2} + b, \text{ no nosso exemplo } 6544 = 65 \cdot 10^2 + 44$$

- Dividir y em duas partes, digamos c e d de $n/2$ dígitos.

$$y = c \cdot 10^{n/2} + d$$

$$x = a \cdot 10^{n/2} + b$$

$$y = c \cdot 10^{n/2} + d$$

$$\begin{aligned} x \cdot y &= (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d) \\ &= ac \cdot 10^{n/2} \cdot 10^{n/2} + ad \cdot 10^{n/2} + bc \cdot 10^{n/2} + bd \\ &= ac \cdot 10^n + 10^{n/2}(ad + bc) + bd \end{aligned}$$

- Então se calcularmos ac, ad, bc, e bd que são todas multiplicações de números com $n/2$ dígitos (portanto menor que os x e y com n dígitos) podemos fazer algumas operações simples e obter o resultado para o problema original.
- Obviamente podemos calcular ac, ad, bc, e bd com 4 chamadas recursivas para esse algoritmo. O caso base é quando temos números de 1 dígito que sabemos calcular.

$$x \cdot y = ac \cdot 10^n + 10^{n/2}(ad + bc) + bd$$

$$6544 \cdot 2123$$

$$a = 65, b = 44, c = 21, d = 23$$

$$65 \cdot 21 \cdot 10^n + 10^{n/2}(65 \cdot 23 + 44 \cdot 21) + 44 \cdot 23$$

$$65 \cdot 21 \cdot 10^4 + 10^2(65 \cdot 23 + 44 \cdot 21) + 44 \cdot 23$$

$$1365 \cdot 10^4 + 10^2(1495 + 924) + 1012$$

$$1365 \cdot 10^4 + 10^2(2419) + 1012$$

$$\begin{array}{r} 13650000 \\ 241900 \\ 1012 \\ \hline 13892912 \end{array}$$

- Será que esse algoritmo é mais rápido (em termos de número de operações elementares) do que o algoritmo do primário?
- Veremos nessa disciplina como analisar esse tipo de algoritmo
- Por enquanto acredite que o número de operações desses algoritmo é a mesma coisa que o algoritmo do primário. :(
- Mas perceba o seguinte, na formula

$$x \cdot y = ac \cdot 10^n + 10^{n/2}(ad + bc) + bd$$

não estamos de fato interessados no valor de ad e bc, mas apenas na sua soma.

- Então será que ao invés de obter 4 valores através de chamadas recursivas, podemos obter apenas os 3 que nos interessa?

Carl Friedrich Gauss

- Johann Carl Friedrich Gauss (1777 - 1855) foi um matemático, astrônomo e físico alemão.
- Dentre diversas contribuições para a ciência, Gauss notou que no produto de dois números complexos

$$(a+bi)(c+di) = ac-bd+(ad+bc)i$$

parece envolver a multiplicação de 4 números.

- Mas de fato pode ser feito com 3 multiplicações, uma vez que

$$ad + bc = (a+b)(c+d) - ac - bd$$

Algoritmo de Karatsuba

$$x \cdot y = ac \cdot 10^n + 10^{n/2}(ad + bc) + bd$$

- Observe que:

$$ad + bc = (a+b)(c+d) - ac - bd$$

- Dessa forma podemos fazer o seguinte:
  1. calcular ac
  2. calcular bd
  3. calcular $(a+b)(c+d)$
  4. calcular $e = (a+b)(c+d) - ac - bd$
  5. somar $a.c.10^n + b.d + e.10^{n/2}$

1. calcular ac
2. calcular bd
3. calcular $(a+b)(c+d)$
4. calcular $e = (a+b)(c+d) - ac - bd$
5. somar $a.c.10^n + e.10^{n/2} + b.d$

$$6544 \cdot 2123$$

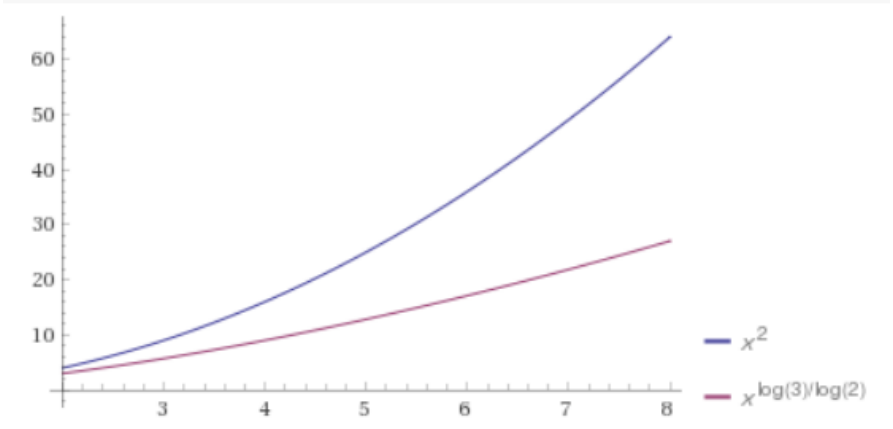
$$a = 65, b = 44, c = 21, d = 23$$

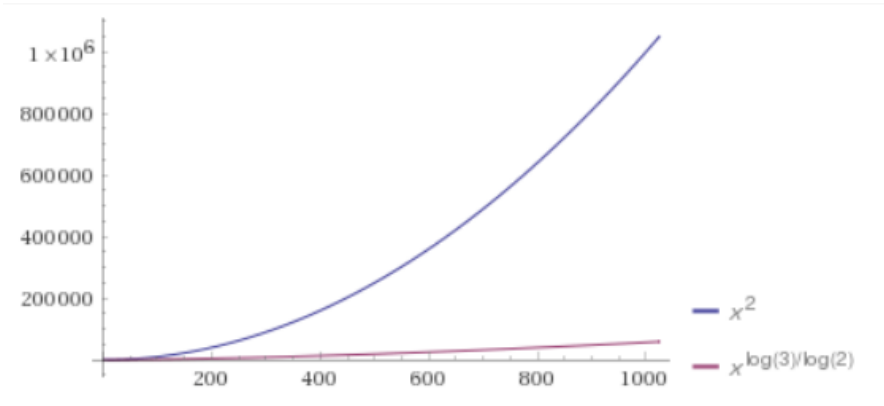
1. $ac = 1365$
2. $bd = 1012$
3. $(a+b)(c+d) = (109 \cdot 44) = 4796$
4. $e = 4796 - 1365 - 1012 = 2419$
5. $13650000 + 241900 + 1012$

$$13892912$$

Algoritmo de Karatsuba

- Pelas provas que fizemos você deve estar convencido de que o algoritmo de fato funciona!
- Perceba que como um problema tão comum quanto a multiplicação de inteiros pode ter algoritmos diferentes para resolve-los.
- Mas será que esse algoritmo é mais rápido que o do primário, ou do que a versão que faz 4 chamadas recursivas?
- Veremos com detalhes essa análise posteriormente, mas por enquanto acredite, ao invés de cn^2 esse algoritmo faz $c'n^{\log_2 3} = c'n^{1.586}$





Objetivos da Disciplina

- Familiarizar o aluno com o vocabulário da Ciência da Computação
  - Notação O, *Big O*, Ó grande, Ózão. E seus colegas.
  - Encontrar uma forma de comparar dois algoritmos de forma simples porém precisa.
- Técnicas de Projeto de algoritmos:
  - Divisão e Conquista
  - Algoritmos Gulosos
  - Aleatorização de algoritmos
  - Programação dinâmica
- Algoritmos em Grafos
- Estruturas de dados

Habilidades que você vai adquirir

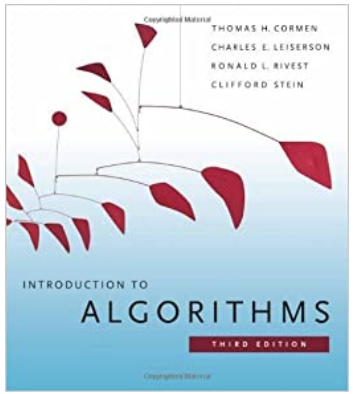
- Tornar-se um programador melhor
- Afiar a sua habilidade matemática e analítica
- Pensar de forma algorítmica
- Familiaridade com a literatura de ciência da computação, algoritmos clássicos, teorema fundamentais, personalidades importantes.
- Vai ajudar a passar em entrevistas de emprego!

Pré-requisitos

- Saber programação, ser capaz de entender uma ideia e transforma-la em código
- Estruturas de dados simples: Vetor, Listas, Pilha, Fila, Heaps, Árvores.
- Matemática discreta: Conjuntos, Números, Provas Matemáticas.

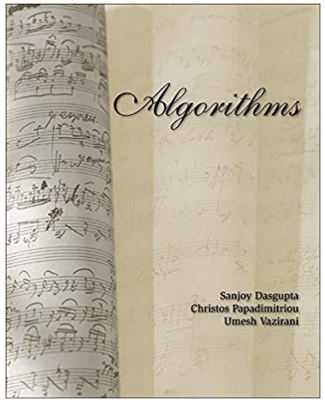
Referências

- Introduction to Algorithms (3rd Edition)
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein
- Tem edição em português.



Referências

- Algorithms
- Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Tem uma versão livre na internet.
- Tem edição em português.



Referências

- Algorithms Illuminated
- Tim Roughgarden
- Tem uma série de vídeo aulas que cobrem o material do livro.

Referências

- Análise de Algoritmos e Estruturas de Dados
- Carla Negri Lintzmayer e Guilherme Oliveira Mota
- Em português e tem versão online.
- `http://professor.ufabc.edu.br/~carla.negri/cursos/materiais/Livro-Analise.de.Algoritmos.pdf`

Avaliação

- A chamada será feita durante a aula, e através de questionários em aula.
- Nota T_t dos trabalhos práticos do bimestre t de 0 a 10.
- Nota P_t da prova do bimestre t de 0 a 10.
- Nota do bimestre t, $N_t = T_t \times P_t/10$.
- Compromisso: De em todas as aulas houver pelo menos 50% dos alunos, $P_t = 10$.
- Média Parcial $M = \frac{N_1+N_2}{2}$.
- Se frequência menor que 75% o aluno reprovou-se.
- Senão se, $M \geq 6$ e presença maior que 75% o aluno aprovou-se.
- Senão se, $M < 6$ e presença maior que 75% o aluno pode fazer uma substitutiva que substitui a menor entre N_1 e N_2.
- Em caso de plágio, fraude, tentativa de burlar os sistemas, nota zero será aplicado na disciplina a todos os envolvidos e estarão automaticamente reprovado.

Comunicação

- Site da disciplina https://hokama.com.br
- Email hokama@unifei.edu.br, melhor enviar pela sua conta @unifei.edu.br e usar a tag [CIC110] no início do assunto.
- Nas aulas é obrigatório o uso de uma conta @unifei.edu.br