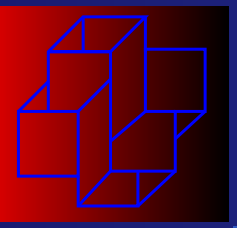
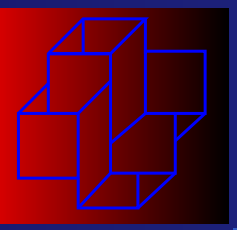


Introdução à Teoria da Complexidade Computacional Clássica

Verão – 2010

Fábio Borges & Emmanuel Felix

LNCC

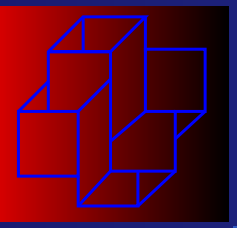


Computação Paralela

- Se temos k processadores, em geral, não mudamos de classe.
- Se temos complexidade $O(2^n)$, então

$$\lim_{n \to \infty} \frac{2^n}{k} \Rightarrow O(2^n)$$

- Comunicação entre os processos e overhead
- Processamento massivamente paralelo
- Não resolve NP

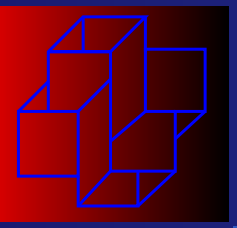


Lei de Amdahl

Se P é a proporção de um programa que pode ser paralelizado e $(1 - P)$ é a proporção que não pode, então o speedup máximo com N processadores é

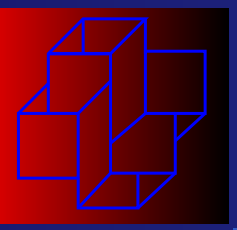
$$\frac{1}{(1 - P) + \frac{P}{N}}$$

Com $N \to \infty$ o speedup máximo tende para $1/(1 - P)$.
O ganho cai rapidamente como N



Exemplo da Lei de Amdahl

- Se P é de 90%, então $(1 - P)$ é de 10%, e o problema pode acelerar até o máximo de 10 vezes, não importa quão grande seja o valor de N.
- O grande trabalho consiste em tentar reduzir o componente $(1 - P)$ para o menor valor possível.



Lei de Gustafson

Seja n o tamanho de um problema mensurável.
A execução de um programa é decomposto em

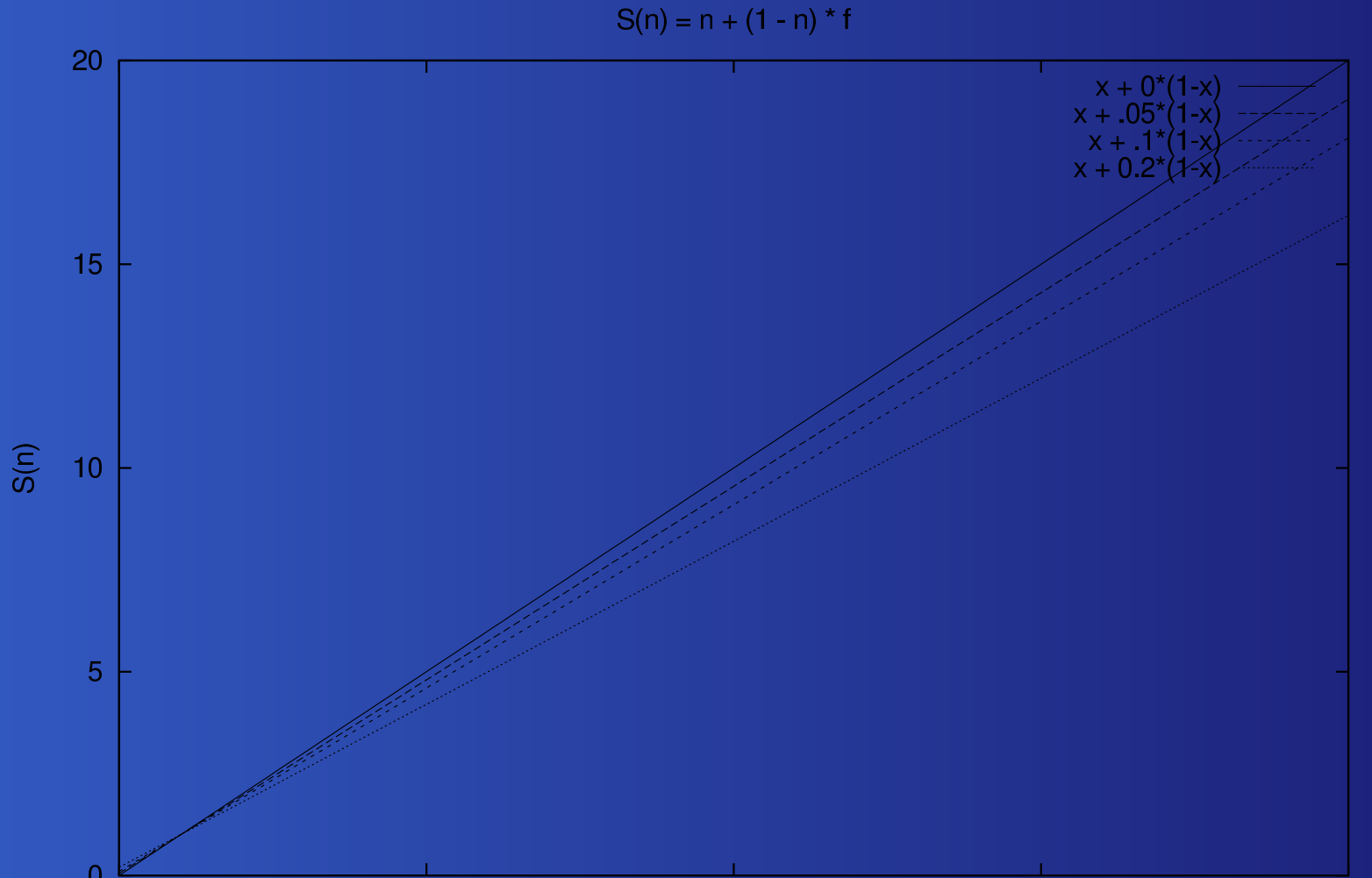
$$a(n) + b(n) = 1$$

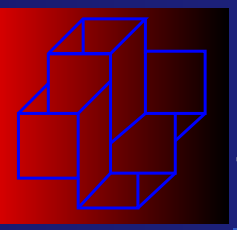
Onde a é a parte sequencial e b a parte paralela.
O tempo de execução seria $a(n) + p \cdot b(n)$, onde p é o número de processadores.
Logo o speedup S é dado por

$$S = (a(n) + p \cdot b(n))$$

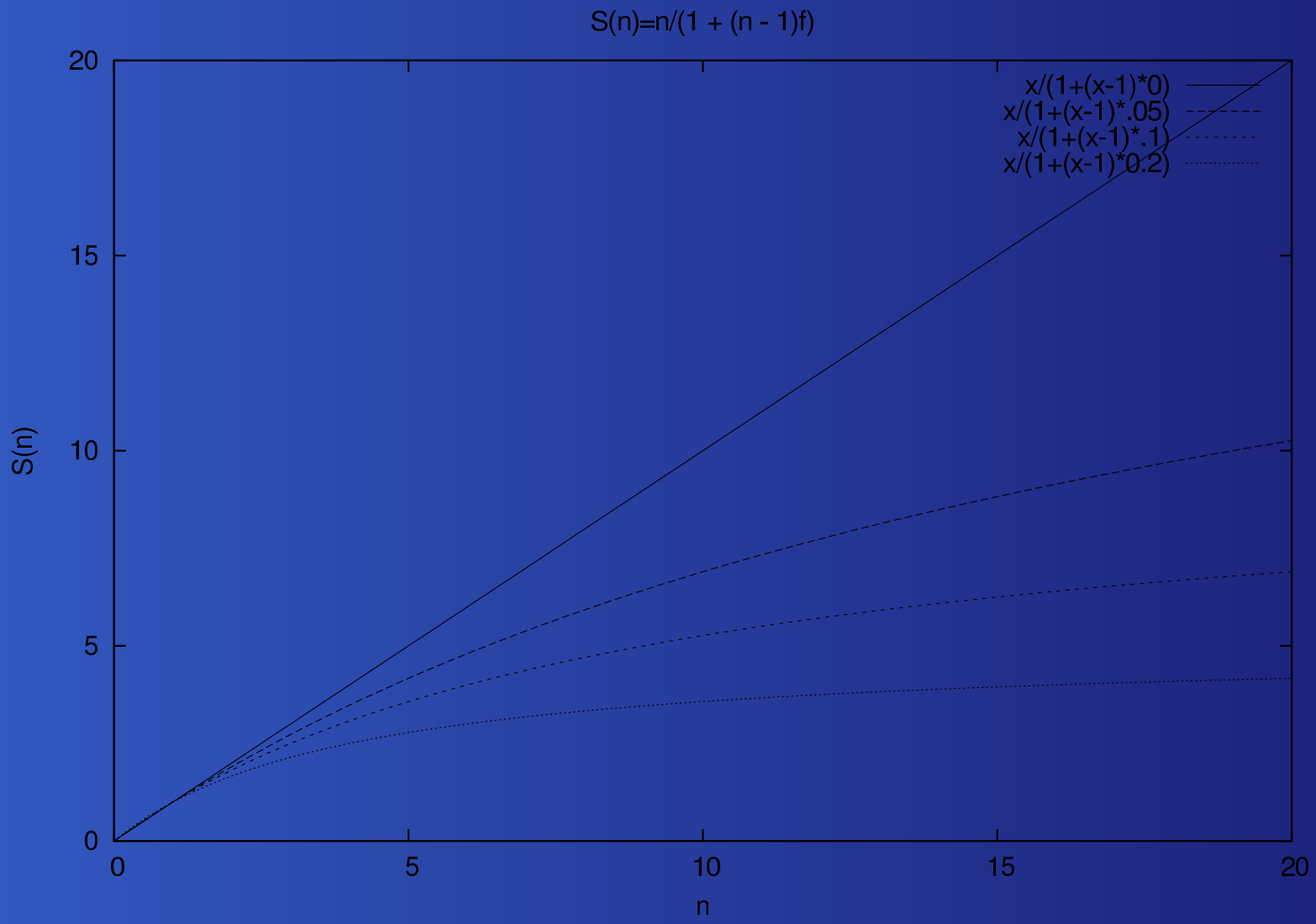
$$\therefore \qquad S = a(n) + p \cdot (1 - a(n))$$

Gustafson

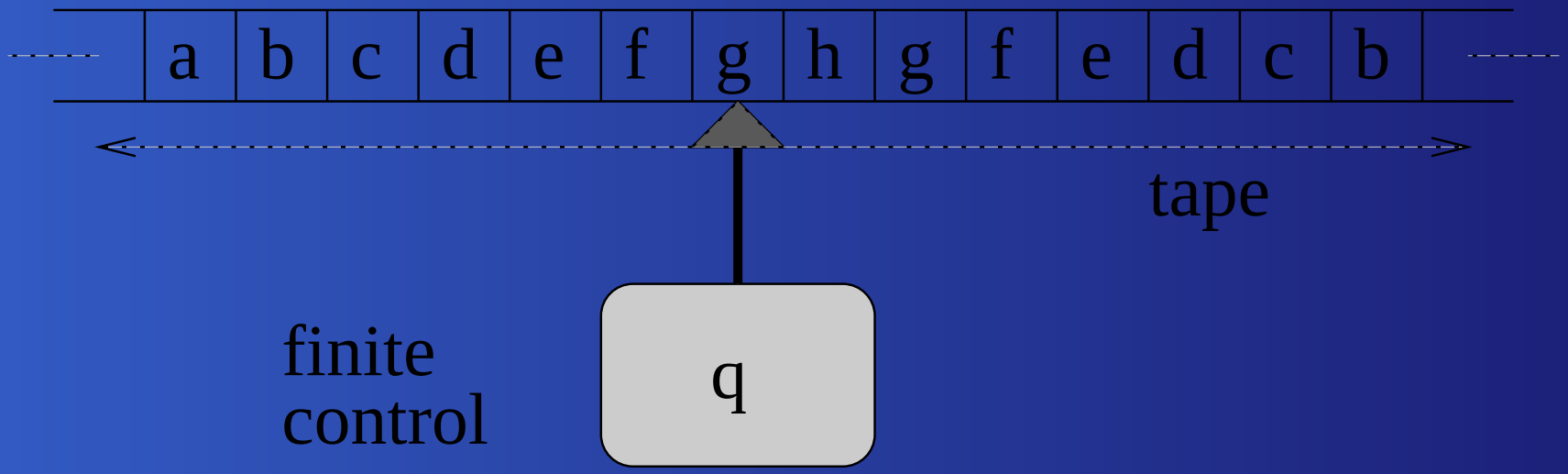




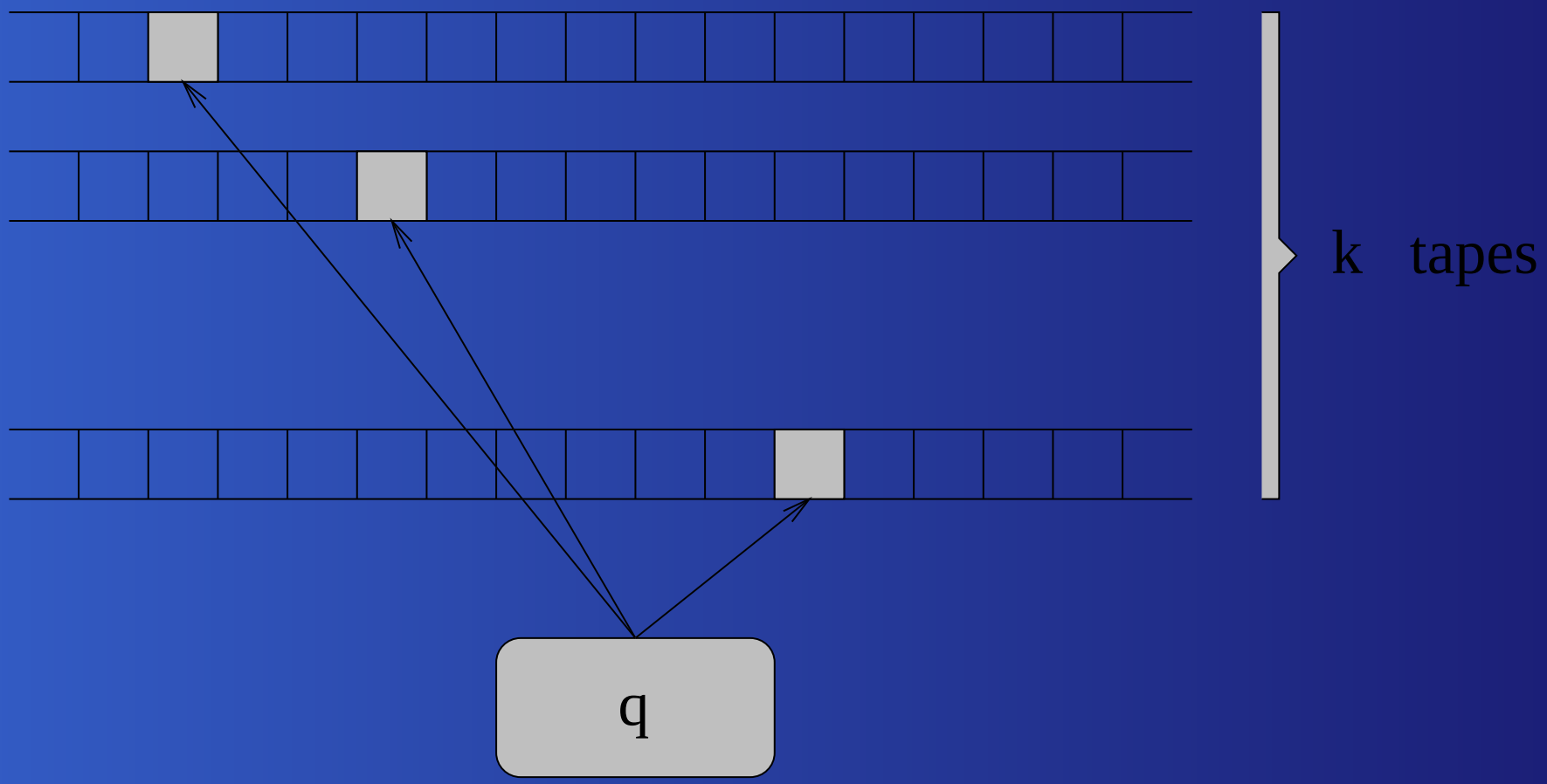
Amdahl

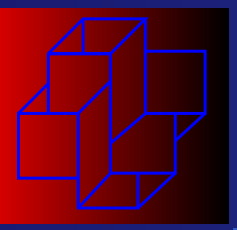


TM



Multitape Turing Machine

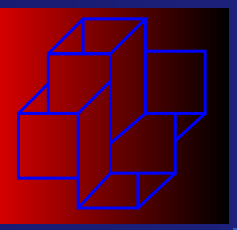




Definição

$$\delta : \Gamma \times \Sigma \times \Sigma \mapsto \Gamma \times \Sigma \times \Sigma \times \{\leftarrow, \Box, \rightarrow\} \times \{\leftarrow, \Box, \rightarrow\}$$

$$\delta : \Gamma \times \Sigma^k \mapsto \Gamma \times \Sigma^k \times \{\leftarrow, \Box, \rightarrow\}^k$$



Multiplicação na MTM

$$(\gamma_0, \&, *, \gamma_0, \&, *, \rightarrow, \square)$$

$$(\gamma_0, 1, *, \gamma_0, 1, 1, \rightarrow, \rightarrow)$$

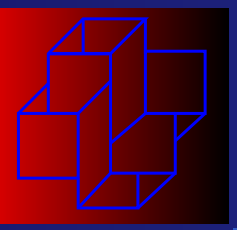
$$(\gamma_0, \times, *, \gamma_1, \times, *, \rightarrow, \square)$$

$$(\gamma_1, 1, *, \gamma_2, *, *, \rightarrow, \square)$$

$$(\gamma_1, *, *, \gamma_1, *, *, \rightarrow, \square)$$

$$(\gamma_2, 1, *, \gamma_3, 1, *, \leftarrow, \square)$$

$$(\gamma_2, =, *, \gamma_p, =, *, \square, \square)$$



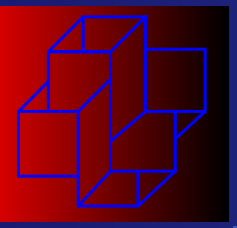
Multiplicação na MTM

$$(\gamma_3, 1, *, \gamma_3, 1, *, \leftarrow, \square)$$

$$(\gamma_3, *, *, \gamma_3, *, *, \leftarrow, \square)$$

$$(\gamma_3, \times, *, \gamma_3, \times, *, \leftarrow, \square)$$

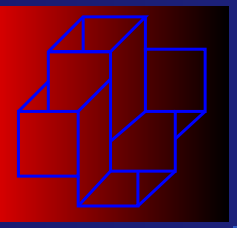
$$(\gamma_3, \&, *, \gamma_0, \&, *, \rightarrow, \square)$$



Equivalência da MTM

Teorema: Toda MTM tem uma máquina de Turing de uma fita equivalente.

[Sipser, pág. 137]

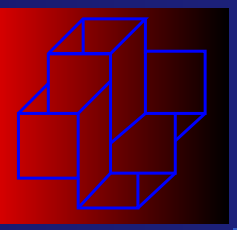


Multiplicação Matricial

- Considere as matrizes $C = A \cdot B$ de ordem $n \times n$

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}, \qquad i, j = 1, \ldots, n.$$

- Temos $O(n^3)$ operações, mas se tivermos n^3 processadores o tempo fica linear $O(n)$ com $n - 1$ adições

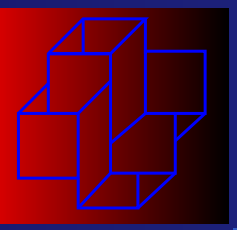


Otimizando

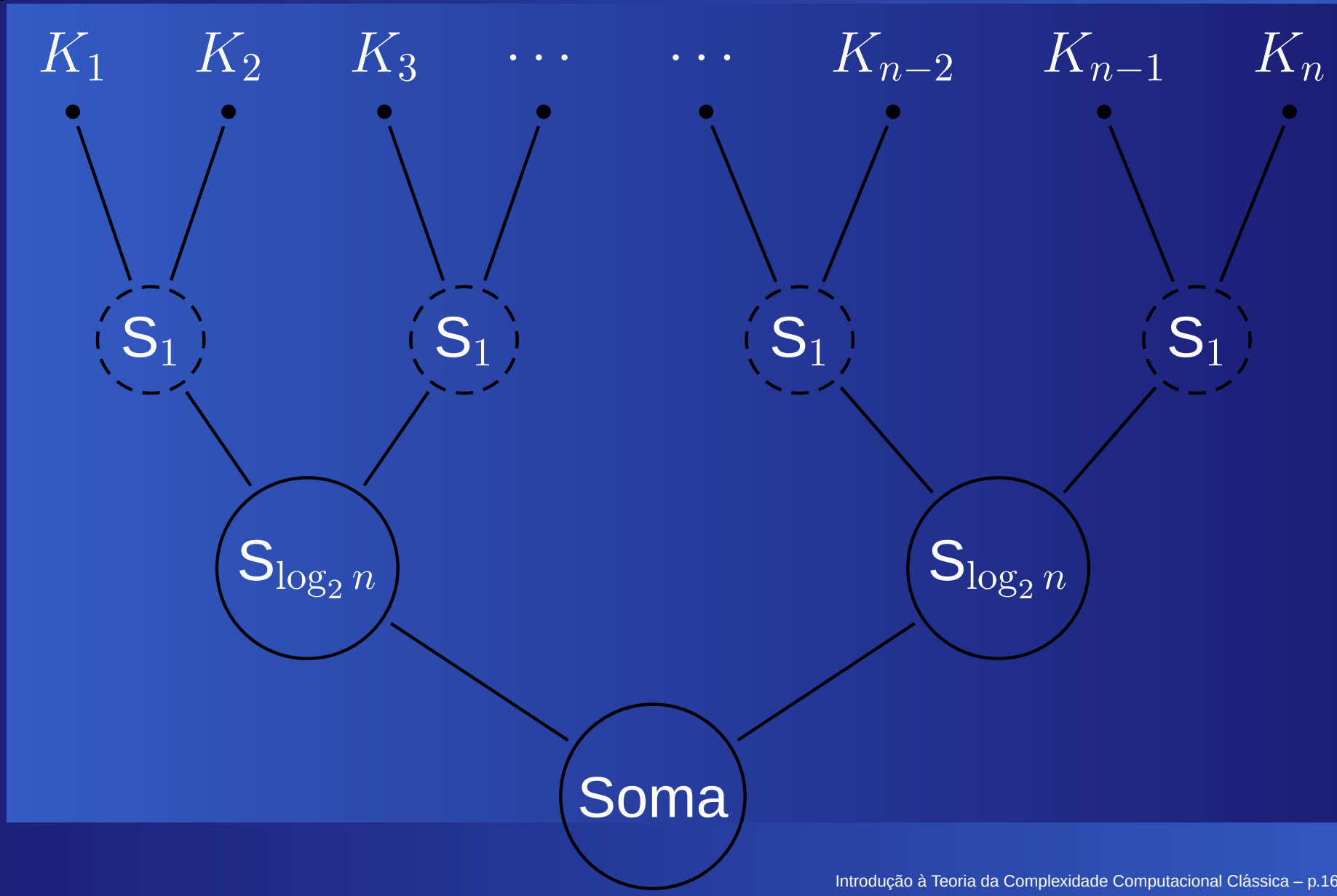
Podemos fazer a soma em $O(\log_2 n)$. No primeiro passo fazemos

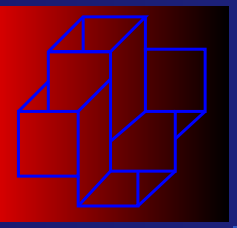
$$(i, k, j) + (i, k + 1, j)$$

Depois somamos os resultados de dois em dois, e assim por diante. Desta forma, temos C_{ij} com $\log_2 n + 1$ passos paralelos em n^3 processadores.



Somas Parciais



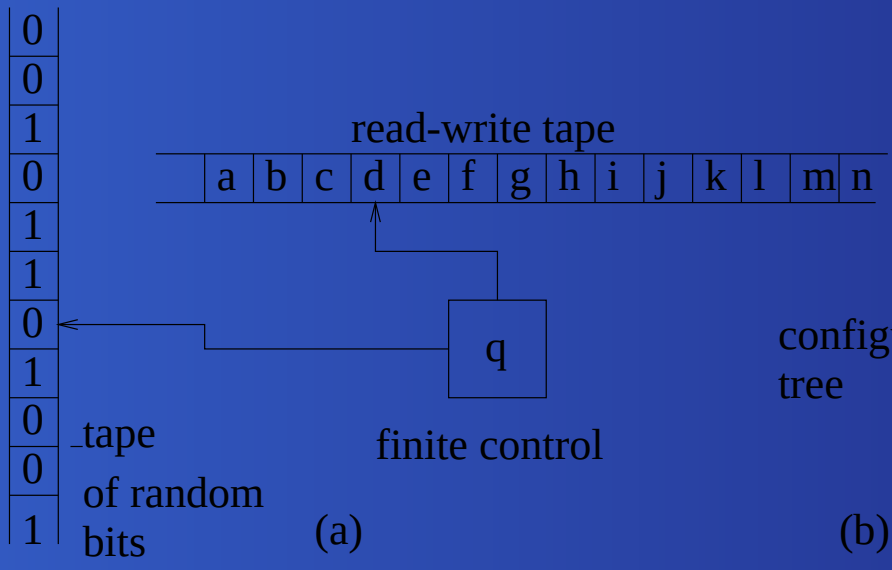


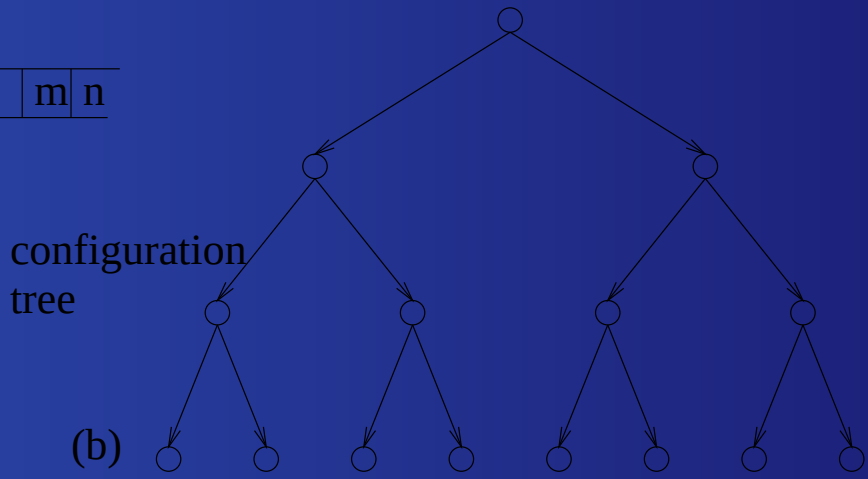
Nick's Class

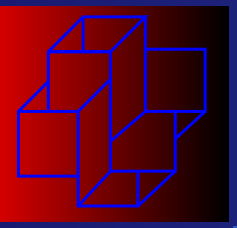
Resolvemos o problema da multiplicação matricial com tamanho de $O(n^3)$ e uma profundidade de $O(\log_2 n)$.
A classe NC é o conjunto de todos os problemas resolvidos com size-depth $(O(n^k), O(\log^k n))$ para uma constante $k \geqslant 1$, dizemos que $\text{NC}^k \subseteq \text{NC}$.

[Sipser, pág. 369]

Nondeterministic Turing Machine



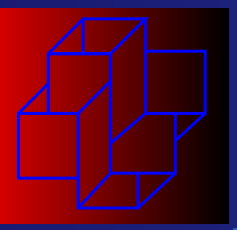




Equivalência da NTM

Teorema: Toda NTM tem uma máquina de Turing determinística equivalente.

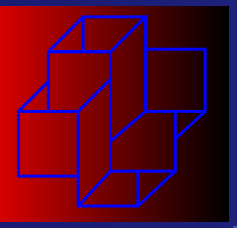
[Sipser, pág. 138]



Probabilistic Turing Machine

Existem várias definições interessantes:

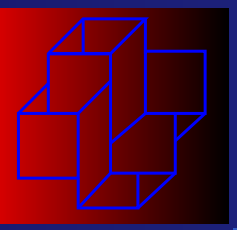
- Uma NTM que escolhe aleatoriamente entre transições disponíveis em cada ponto de acordo com alguma distribuição de probabilidade
- Um tipo de NTM onde cada passo é chamado de *coin-flip step* e tem dois próximos passos legais
- Uma MT em que algumas transições são escolhas aleatórias entre as alternativas finitas



PTM $\subset$ NTM

"A probabilistic Turing machine M is a type of nondeterministic Turing machine where each nondeterministic step is called coin-flip step and has two legal next move."

[Sipser, pág. 336]



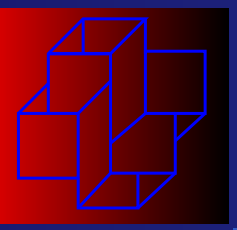
Probabilidades

Assumimos a probabilidade de cada vértice b de M computações com uma entrada w. Definimos a probabilidade do vértice b como

$$Pr[b] = 2^{-k}$$

onde k é o número de coin-flip step que ocorre no vértice b. Definimos a probabilidade de M aceitar w como

$$Pr[M \text{ aceitar } w] = \sum_{b \text{ á um vértice aceito}} Pr[b]$$



Reconhecimento em PTM

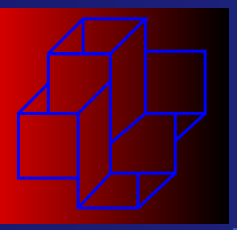
M reconhece a linguagem A com probabilidade de erro ϵ se:

- $w \in A$ implica $Pr[M \text{ aceitar } w] \geqslant 1 - \epsilon$
- $w \notin A$ implica $Pr[M \text{ rejeitar } w] \geqslant 1 - \epsilon$

Aceita todas as sequências na linguagem e rejeita todas as sequências fora da linguagem:

$$Pr[M \text{ rejeitar } w] = 1 - Pr[M \text{ aceitar } w]$$

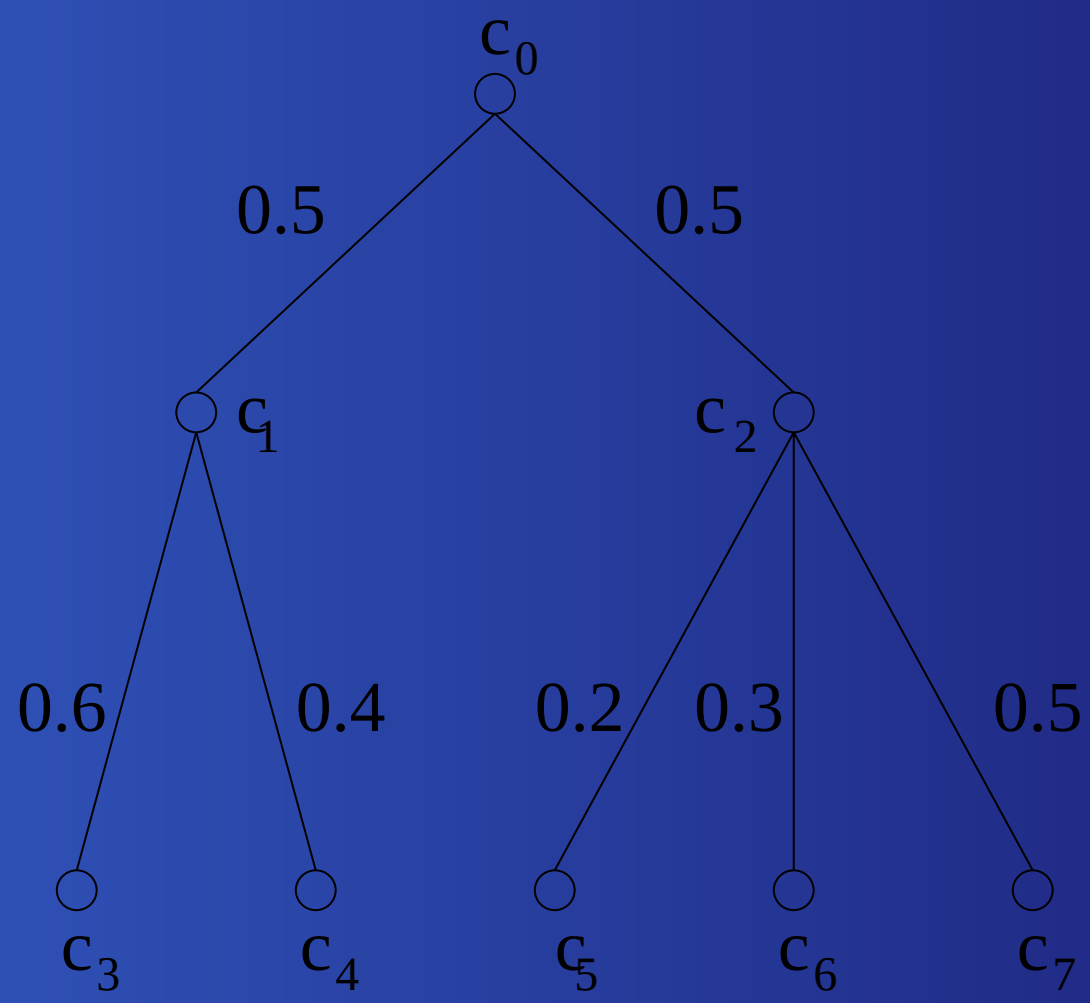
No entanto, um PTM terá uma probabilidade de erro $\epsilon = 2^{-n}$, onde n é a entrada.

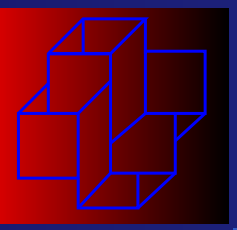


Fatos

- Cada “vértice” na computação de PTM tem uma probabilidade
- Tem resultados estocásticos

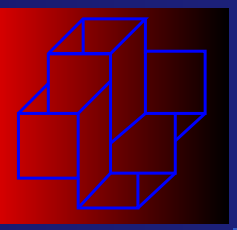
PTM





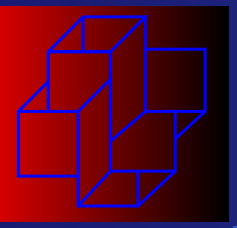
Fatos

- Assim, uma dada entrada:
  - pode ter tempo de execução diferente
  - pode não parar
- Portanto, pode aceitar a entrada em uma determinada execução, mas rejeitar outra entrada
- A complexidade de tempo e espaço pode ser medida através do pior caso



Lema da Amplificação

Seja uma constante fixada estritamente entre 0 e $1/3$. Então para qualquer tempo polinomial $\mathrm{poly}(n)$ em uma PTM M_1 que opera com probabilidade de erro ϵ existe uma PTM M_2 que opera com probabilidade de erro $2^{-\mathrm{poly}(n)}$ em tempo polinomial.

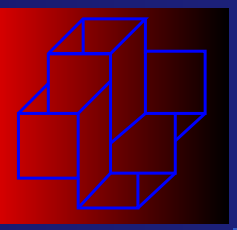


Teste de Primalidade

- Se p é primo então $a^{p-1} \equiv 1 \mod p \; \forall \; a \in \mathbb{Z}_p^*$

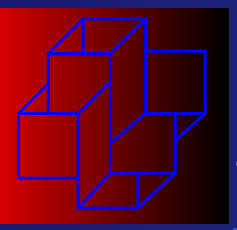
$$2^{7-1} \equiv 64 \equiv 1 \mod 7$$

$$2^{6-1} \equiv 32 \equiv 2 \mod 6$$



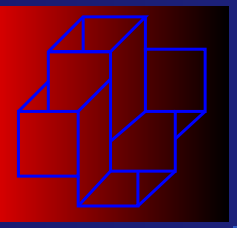
Erros do Teste de Primalidade

- $2^{340} \equiv 1 \mod 341$ é pseudoprimo na base 2
- $3^{340} \equiv 56 \mod 341$
- Existem 245 pseudoprimos na base 2 menores que um milhão
- A maioria não é pseudoprimo em outra base
- Existe apenas 2163 n. de Carmichael menores que 2.5×10^{10}



Algoritmo Probabilístico

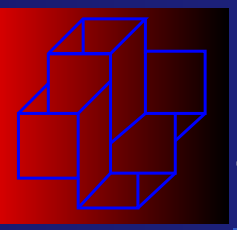
- Um algoritmo concebido para utilizar o resultado de um processo aleatório
- Muitas vezes, o algoritmo tem acesso a um gerador de números pseudo-aleatórios
- O algoritmo utiliza bits aleatórios para ajudar a fazer escolhas (na esperança de obter um melhor desempenho)



Por que usar algoritmos aleatórios

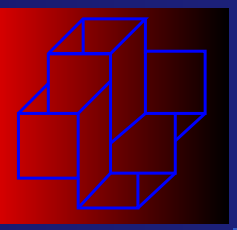
Algoritmos probabilísticos são úteis porque:

- É demorado calcular a “melhor resposta”
- Estimativa poderia introduzir um viés indesejado que invalida os resultados



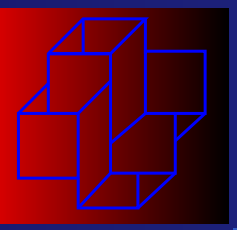
Amostra Aleatória

- A amostragem aleatória é usada para obter informações sobre os indivíduos em uma grande população
- Consultar a todos levaria muito tempo
- Consultar um subconjunto não aleatório selecionado pode influenciar os resultados



Bounded error Probability in Polynomi

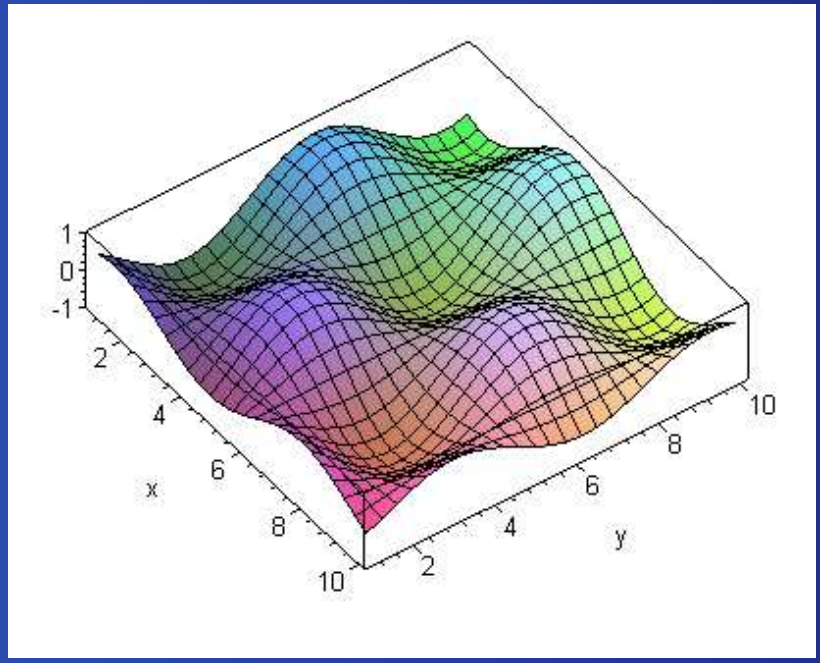
- A classe de linguagens que são reconhecidas por TM em tempo polinomial probabilístico com uma probabilidade de erro de $1/3$ (ou menor)
- A classe de linguagens que uma PTM pára em tempo polinomial com uma resposta de aceitar ou rejeitar de pelo menos $2/3$ do tempo

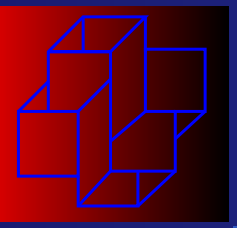


Monte Carlo

- Como vimos na classe BPP, executando as repetições adicionais no algoritmo irá diminuir a chance do algoritmo dar a resposta errada
- Muitas vezes referimos a este tipo de algoritmo como algoritmo de Monte Carlo

Integral





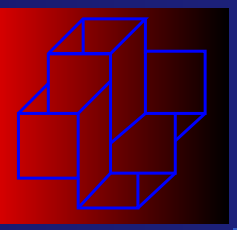
Integração por Monte Carlo

Escolhendo aleatoriamente n pontos $x_1, x_2, \ldots, x_n$ em um volume V para determinar a integral de uma função f fazemos

$$\int f dV \approx V \left(\langle f \rangle \pm \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{n}} \right)$$

onde

$$\langle f \rangle = \frac{1}{n} \sum_{i=1}^{n} f(x_i) \qquad \langle f^2 \rangle = \frac{1}{n} \sum_{i=1}^{n} f^2(x_i)$$



Erro da Integração

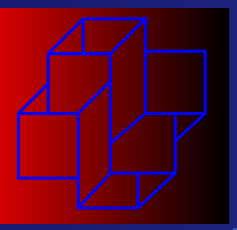
O desvio padrão é dado por

$$\sigma = \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{n}}$$

Logo o erro decresce conforme n cresce, isto é

$$\frac{1}{\sqrt{n}}$$

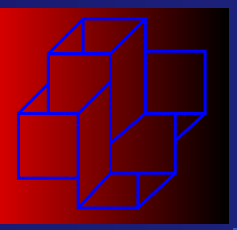
Este algoritmo pertence à BPP.



Randomized Polynomial time

A classe de problemas que serão executados em tempo polinomial em uma PTM com as seguintes propriedades:

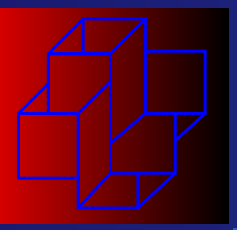
- Se a resposta correta é
  - não, sempre retorna não
  - sim, retornar sim com probabilidade de pelo menos $1/2$
- Caso contrário, não retorna



Simplificando

A classe de linguagens para as quais o resultado pode ser determinado em tempo polinomial por uma PTM sem falsas aceitações e menos da metade das rejeições são falsas.

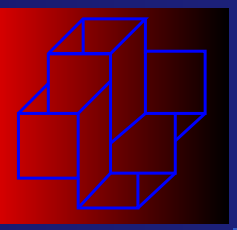
- Se o algoritmo retorna uma resposta sim, então sim é a resposta correta. Não existe falso positivo.
- Se o algoritmo retorna uma resposta não, então ela pode ou não ser correto



Co-RP

A classe de problemas que serão executados em tempo polinomial em uma PTM com as seguintes propriedades:

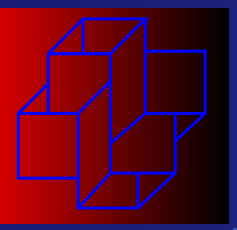
- Se a resposta correta é
  - Sim, sempre retornam sim
  - Não, retornam não com probabilidade de pelo menos $1/2$
- Caso contrário, retorna um sim



Simplificando

A classe de linguagens para as quais o resultado pode ser determinado em tempo polinomial por uma PTM sem falsas rejeições e menos da metade das aceitações são falsas.

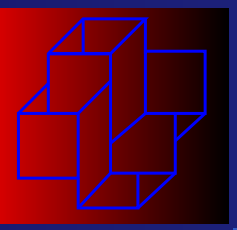
- Se o algoritmo retorna uma resposta não, então não é a resposta correta. Não exite falso negativo.
- Se o algoritmo retorna uma resposta sim, então ela pode ou não ser correto



Zero-error Probabilistic Polinomial

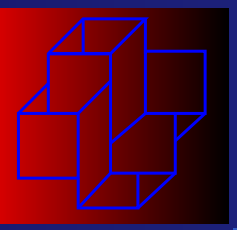
A classe de linguagens que uma PTM pára em tempo polinomial, sem falsas aceitações ou rejeições, mas às vezes dá um “não sei” como resposta. Em outras palavras:

- sempre retorna uma resposta sim ou não garantida
- pode retornar um “não sei” como resposta



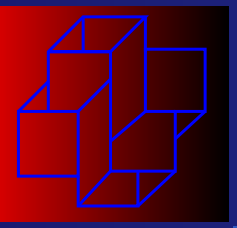
ZPP

- A execução é ilimitada
  - Mas tem tempo polinomial em média (para qualquer entrada)
  - Espera-se parar em tempo polinomial



ZPP

- Similar a definição de P, exceto:
  - ZPP permite que TM tenha “aleatoriedade”
  - O tempo esperado de execução é o médido (em vez do pior caso)
- Muitas vezes referida como um algoritmo de Las Vegas



Relações

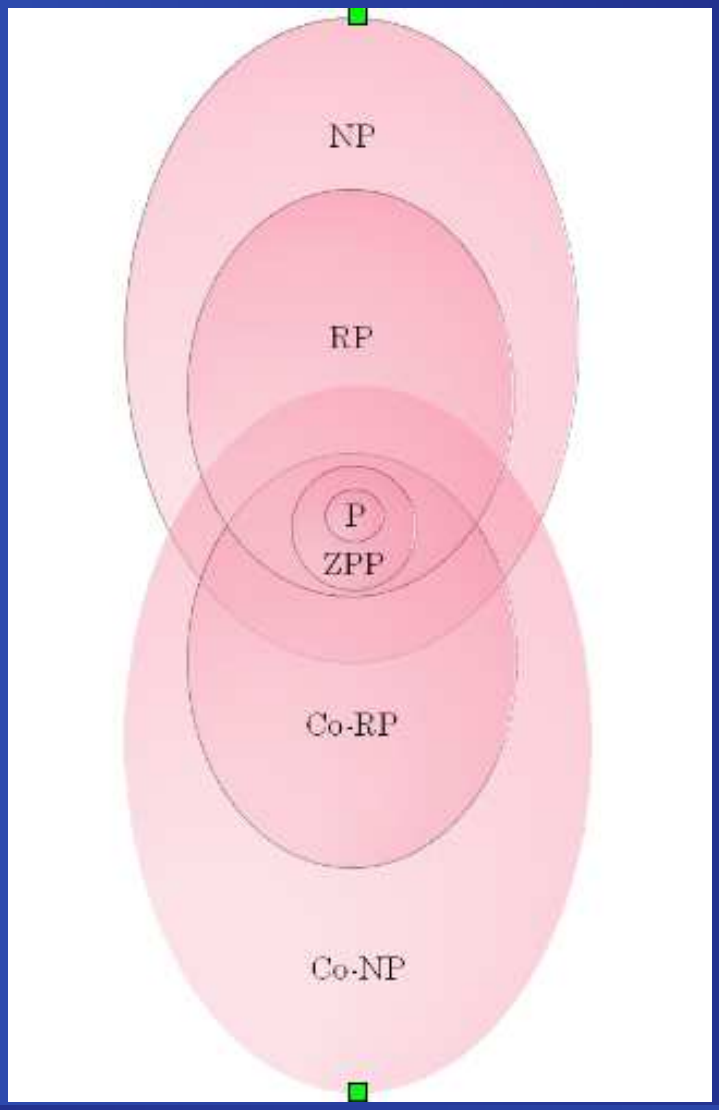
$$\text{LOGSPACE} \subseteq \text{NC} \subseteq \text{P} \subseteq \text{RP} \subseteq \text{NP} \subseteq \text{PSPACE}$$

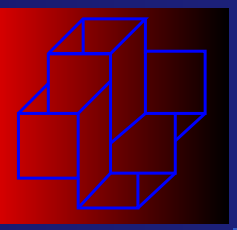
[claymath.org, Palestra P$\times$NP]

$$\text{P} \subseteq \text{ZPP} = \text{RP} \cap \text{co-RP} \subseteq \text{RP} \subseteq \text{RP} \cup \text{co-RP} \subseteq \text{BPP}$$

[Talbot, pág. 83]

Relações

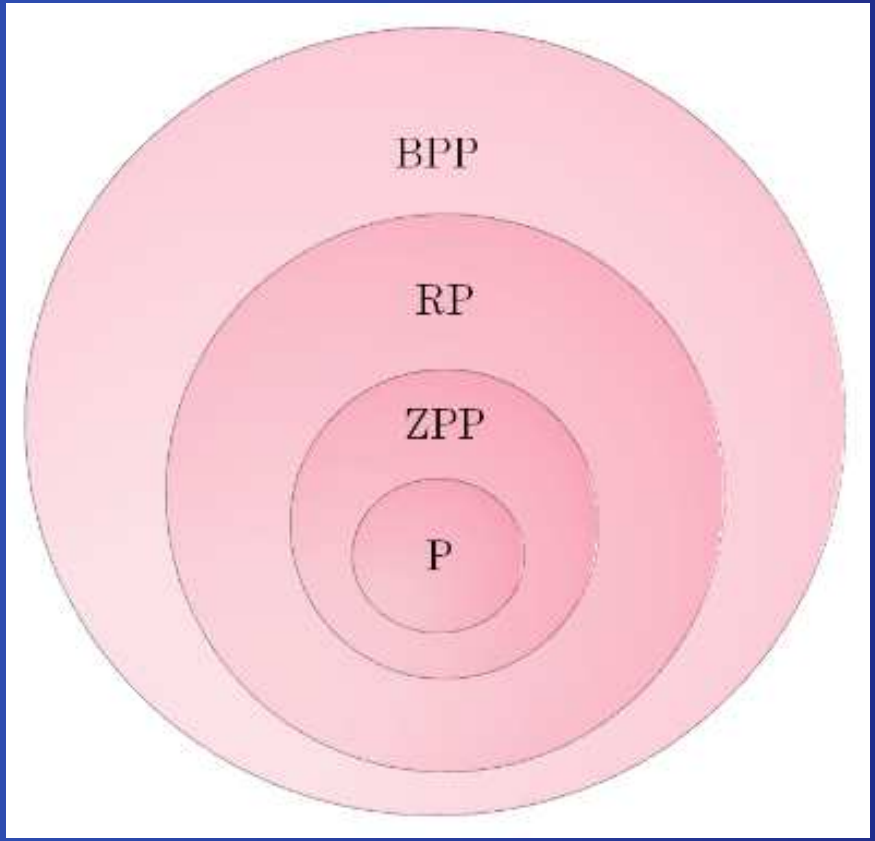


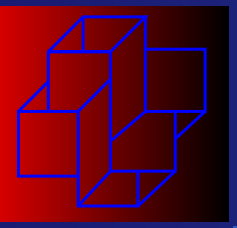


BPP$\subseteq$?

- Ainda é uma questão em aberto se NP é um subconjunto de BPP ou BPP é um subconjunto de NP
- No entanto, acredita-se que RP seja um subconjunto de BPP

Relações BPP ?

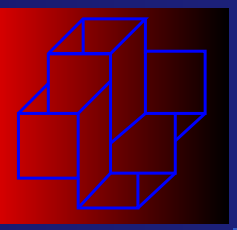




Problemas em aberto

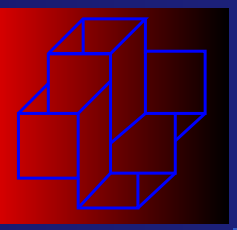
1. $\mathrm{P} = \mathrm{ZPP}$?
2. $\mathrm{RP} = \text{co-RP}$?
3. $\mathrm{RP} = \mathrm{NP} \cap \text{co-NP}$?
4. $\mathrm{BPP} \subseteq \mathrm{NP}$?
5. $\mathrm{NP} \subseteq \mathrm{BPP}$?
6. $\mathrm{RP} \cup \text{co-RP} = \mathrm{BPP}$?

[Talbot, pág. 83]



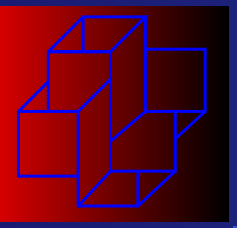
Provas

- Para provar que P $=$ NP basta encontrar um algoritmo polinomial para um problema NP-Completo
- Existem muitas “provas” erradas de P $=$ NP



Consequências

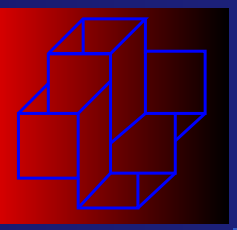
- Se NP $=$ P, muitos problemas importantes podem ser resolvidos deterministicamente e de forma muito rápida
- Se NP $=$ P, muitos códigos criptográficos serão quebrados
- Se NP $\neq$ P, muitos problemas importantes são intratáveis
- Se NP $\neq$ P, existem infinitas classes entre P e NP-Completo (Ladner 1975)



Complexity Zoo

Várias classes podem ser encontradas em:
http://qwiki.stanford.edu/wiki/Complexity_Zoo

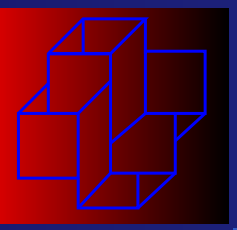
Existem 489 classes catalogadas até agora!



Desconhecidos

Problemas em NP que não sabemos se estão em P ou NP-Completo.

1. Isomorfismo de Grafos
2. Problema do Logaritmo Discreto
3. Problema da Fatoração de Inteiros
   - NP∩Co-NP se PFI∈NP-Completo então NP=Co-NP
   - General Number Field Sieve
   - $O\left(e^{(c+o(1))(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}}\right) = L_n\left[1/3, c\right]$



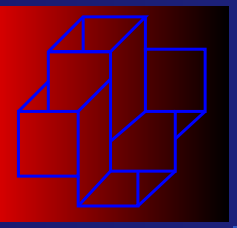
Notação L

A notação L é definida como

$$L_n[\alpha, c] = O\left(e^{(c+o(1))(\ln n)^{\alpha}(\ln\ln n)^{1-\alpha}}\right)$$

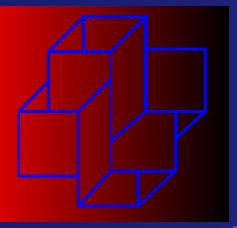
com $0 \leqslant \alpha \leqslant 1$.
A existência do AKS mostra que o teste de primalidade pode ser feito em P com complexidade

$$L_n[0, c] = O((\ln n)^{c+o(1)})$$



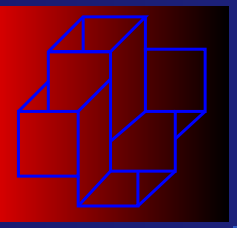
Criptografia

- One-time pad (Vigenère-Vernam)



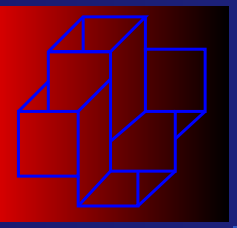
Criptografia

- One-time pad (Vigenère-Vernam)
- Simétricas (Força-bruta)



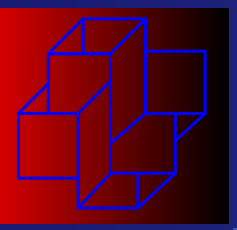
Criptografia

- One-time pad (Vigenère-Vernam)
- Simétricas (Força-bruta)
- Assimétricas ($H(f_D(C)|C) = 0$)



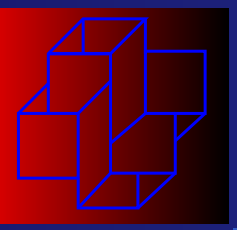
Criptografia

- One-time pad (Vigenère-Vernam)
- Simétricas (Força-bruta)
- Assimétricas ($H(f_D(C)|C) = 0$)
- Carência de prova da segurança



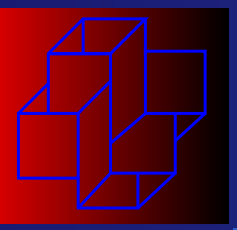
Criptografia

- One-time pad (Vigenère-Vernam)
- Simétricas (Força-bruta)
- Assimétricas ($H(f_D(C)|C) = 0$)
- Carência de prova da segurança
- Se provarmos que encontrar k não está em P então P$\neq$NP



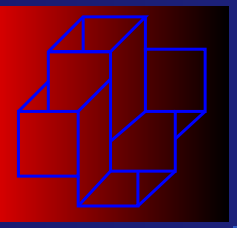
Criptografia

- One-time pad (Vigenère-Vernam)
- Simétricas (Força-bruta)
- Assimétricas ($H(f_D(C)|C) = 0$)
- Carência de prova da segurança
- Se provarmos que encontrar k não está em P então P$\neq$NP
- Se quebramos um código baseado em um problema NP-completo temos que P=NP



Criptografia

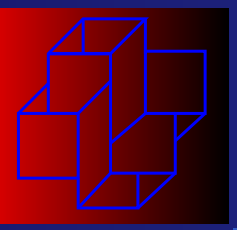
- One-time pad (Vigenère-Vernam)
- Simétricas (Força-bruta)
- Assimétricas ($H(f_D(C)|C) = 0$)
- Carência de prova da segurança
- Se provarmos que encontrar k não está em P então P$\neq$NP
- Se quebramos um código baseado em um problema NP-completo temos que P=NP
- Quebrar códigos significa mudar a classe de um problema ou definir a relação entre classes



One-way Functions

Uma função $f : \Sigma^* \longrightarrow \Sigma^*$ preserva o comprimento se o comprimento de w e $f(w)$ são iguais.

Uma função que preserva o comprimento é uma permutação se $f(x) \neq f(y)$ sempre que $x \neq y$.

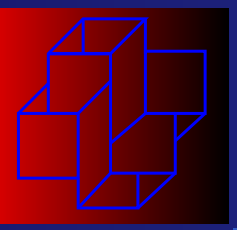


One-way permutation

Uma permutação One-way é uma permutação f que preserva o comprimento com as duas propriedades:

1. É computável em tempo polinomial
2. Para toda PTM M, todo k e n suficientemente grande, se pegarmos um w aleatório de comprimento n e executarmos M com a entrada w, então

$$Pr_{M,w}[M(f(w)) = w] \leqslant n^{-k}$$

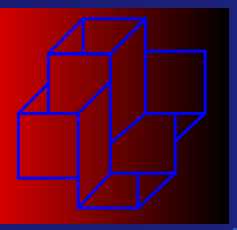


One-way function

Uma função One-way é uma função f que preserva o comprimento com as duas propriedades:

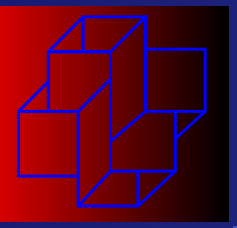
1. É computável em tempo polinomial
2. Para toda PTM M, todo k e n suficientemente grande, se pegarmos um w aleatório de comprimento n e executarmos M com a entrada w, então

$$Pr_{M,w}[M(f(w)) = y \text{ onde } f(y) = f(w)] \leqslant n^{-k}$$



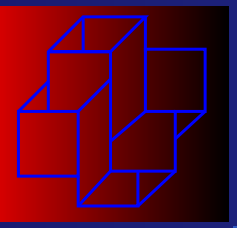
Pseudoaleatório

- $$x^s \equiv y \mod z$$
- Dado x, s e z temos y é pseudo-randômico
- Dado x, y e z temos s secreto



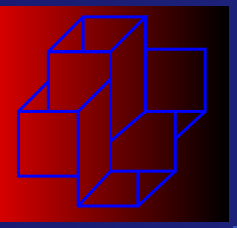
Problema do Logaritmo Discreto

- Com k, pq, k^r e k^s
- Poderia calcular s ou r e depois b_A
- Equivalente ao Problema da Fatoração de Inteiros



Intruso e o Logaritmo Discreto

- Com $k = 256$, $pq = 8383$, $k^r = 5835$ e $k^s = 1438$
- o intruso calcula $256^{109} = 1438$
- $s = 109$
- $b_A = (k^r)^s = 5835^{109} = 3439$

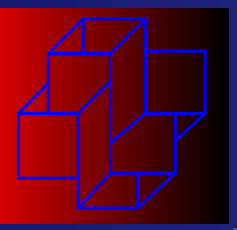


Função de indexação

Uma família de funções $\{f_i\}$ com $i \in \Sigma^*$ pode ser representada por

$$f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

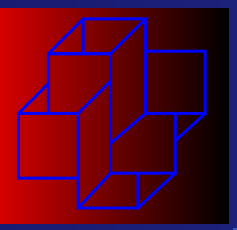
onde $F(i, w) = f_i(w)$.



Trapdoor function

Uma função Trapdoor é uma função de indexação $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ que preserva o comprimento com uma PTM G auxiliar e uma função $h : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ auxiliar.

Sendo que $f,\ G$ e h satisfazem três condições.



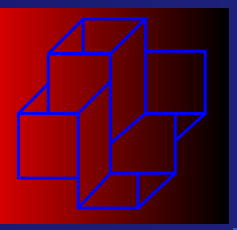
Condições da Trapdoor

1. f e h são computáveis em tempo polinomial
2. Para cada PTM E e todo k, dado um n suficientemente grande e um resultado aleatório $\langle i, t\rangle$ de G de 1^n e um $w \in \Sigma^n$ aleatório, então

$$Pr[E(i, f_i(w)) = y, \text{ onde } f_i(y) = f_i(w)] \leqslant n^{-k}$$

3. Para $\forall n$, $\forall w$ de comprimento n, e toda saída $\langle i, t\rangle$ de G que ocorre com probabilidade não-nula para uma entrada de G

$$h(t, f_i(w)) = y, \text{ onde } f_i(y) = f_i(w).$$



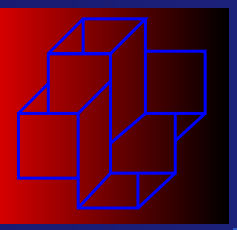
RSA

$$\varphi = \varphi(pq) = (p-1)(q-1)$$

$$(a, \varphi) = 1$$

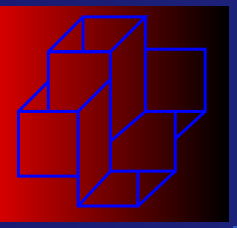
$$ab \equiv 1 \mod \varphi$$

$$x^{ab} \equiv x \mod pq \quad \forall x \in \mathbb{Z}$$



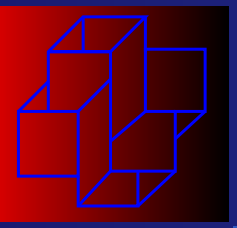
Cifra Monoalfabética

- A $\leftrightarrow$ Q



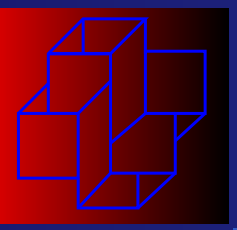
Cifra Monoalfabética

- A $\leftrightarrow$ Q
- B $\leftrightarrow$ V



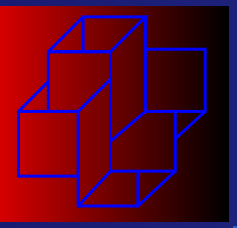
Cifra Monoalfabética

- A $\leftrightarrow$ Q
- B $\leftrightarrow$ V
- C $\leftrightarrow$ D



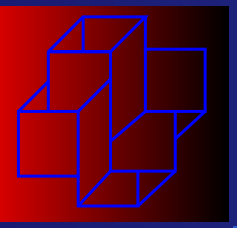
Cifra Monoalfabética

- A $\leftrightarrow$ Q
- B $\leftrightarrow$ V
- C $\leftrightarrow$ D
- $\vdots$



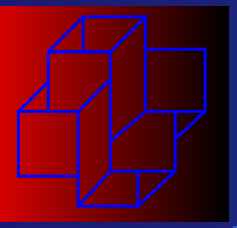
Cifra Monoalfabética

- A $\leftrightarrow$ Q
- B $\leftrightarrow$ V
- C $\leftrightarrow$ D
- $\vdots$
- Z $\leftrightarrow$ E



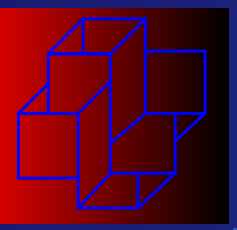
Cifra Monoalfabética

- A $\leftrightarrow$ Q
- B $\leftrightarrow$ V
- C $\leftrightarrow$ D
- $\vdots$
- Z $\leftrightarrow$ E
- "ABCDEFGHIJKLMNOPQRSTUVWXYZ"



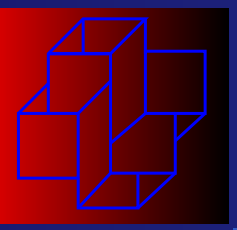
Cifra Monoalfabética

- $A \leftrightarrow Q$
- $B \leftrightarrow V$
- $C \leftrightarrow D$
- $\vdots$
- $Z \leftrightarrow E$
- "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
- "QVDIJTPOCYHNGXAZWUSMFKRLBE"



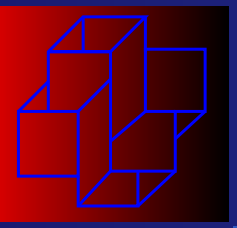
Cifra Monoalfabética

- $A \leftrightarrow Q$
- $B \leftrightarrow V$
- $C \leftrightarrow D$
- $\vdots$
- $Z \leftrightarrow E$
- "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
- "QVDIJTPOCYHNGXAZWUSMFKRLBE"
- $26! - 1 = 403291461126605635583999999$



Cifra Monoalfabética

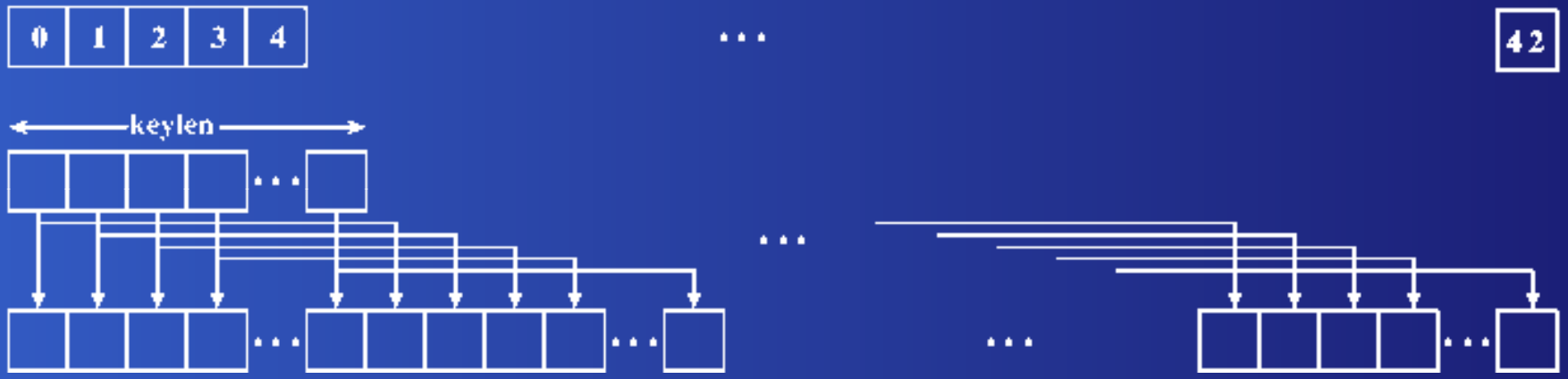
- A $\leftrightarrow$ Q
- B $\leftrightarrow$ V
- C $\leftrightarrow$ D
- $\vdots$
- Z $\leftrightarrow$ E
- "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
- "QVDIJTPOCYHNGXAZWUSMFKRLBE"
- $26! - 1 = 403291461126605635583999999$
- $26! \approx 4.03\ 10^{26}$

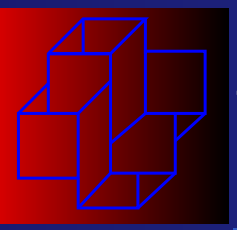


Nomenclaturas

Constante	1
Logarítmo	$\log n$
Linear	n
Log-linear	$n \log n$
Quadratica	n^2
Cubica	n^3
Polinomial	n^p
Sub-exponencial	$b^{\log n}$
Exponential	b^n
Factorial	∞

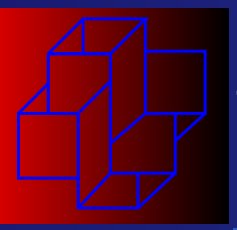
Esquema Vigenère





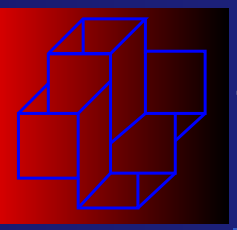
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)



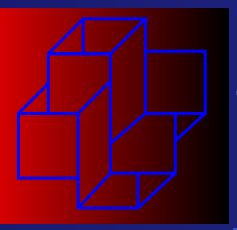
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A⊘MENINA⊘BRINCA



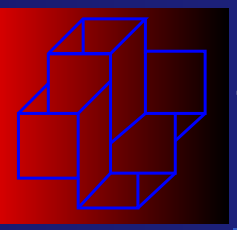
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A∅MENINA∅BRINCA
  - 01,00,13,05,14,09,14,01,00,02,18,09,14,03,01



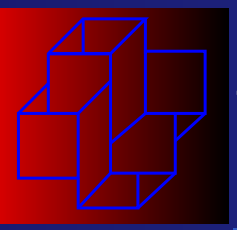
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A∅MENINA∅BRINCA
  - 01,00,13,05,14,09,14,01,00,02,18,09,14,03,01
  - SENHASENHASENHA



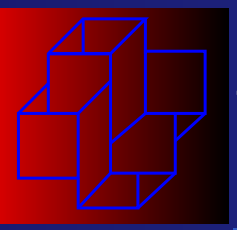
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A∅MENINA∅BRINCA
  - 01,00,13,05,14,09,14,01,00,02,18,09,14,03,01
  - SENHASENHASENHA
  - 19,05,14,08,01,19,05,14,08,01,19,05,14,08,01



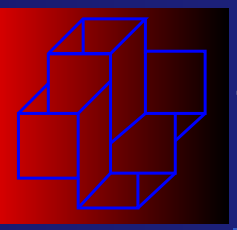
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A␣MENINA␣BRINCA
  - 01,00,13,05,14,09,14,01,00,02,18,09,14,03,01
  - SENHASENHASENHA
  - 19,05,14,08,01,19,05,14,08,01,19,05,14,08,01
  - 20,05,00,13,15,01,19,15,08,03,10,14,01,11,02



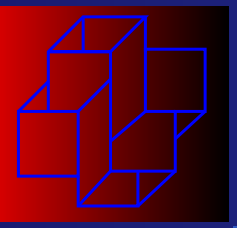
Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A␣MENINA␣BRINCA
  - 01,00,13,05,14,09,14,01,00,02,18,09,14,03,01
  - SENHASENHASENHA
  - 19,05,14,08,01,19,05,14,08,01,19,05,14,08,01
  - 20,05,00,13,15,01,19,15,08,03,10,14,01,11,02
  - TE␣MOASOHCJNAKB



Vigenère

- Gerando uma senha do tamanho do texto, a partir de uma senha menor (Keystream)
  - A␣MENINA␣BRINCA
  - 01,00,13,05,14,09,14,01,00,02,18,09,14,03,01
  - SENHASENHASENHA
  - 19,05,14,08,01,19,05,14,08,01,19,05,14,08,01
  - 20,05,00,13,15,01,19,15,08,03,10,14,01,11,02
  - TE␣MOASOHCJNAKB
- $27^5 = 14348907 \approx 1.24\ 10^7$



Último Slide

- Obrigado.
- Quaisquer sugestões serão muito bem-vindas.

www.lncc.br/∼borges