ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 24

Cap 7 – Complexidade de Tempo
Cap 7.1 – Medindo complexidade

Profa. Ariane Machado Lima
ariane.machado@usp.br

Decidibilidade e complexidade

- Um problema pode ser decidível, mas na **prática** ser “insolúvel” (talvez por enquanto) devido à demanda de tempo e/ou de memória
- Analisar complexidade nos ajuda a identificar
  - Quais problemas são tratáveis
  - Estimativa de tempo (ou memória) necessários
- Qual é a complexidade dos modelos que vimos até agora (AF, AP, ALL, MT)

Medindo complexidade de tempo

- Complexidade de tempo relacionada com o número de passos necessários para um algoritmo dar uma resposta
- Número de passos depende de parâmetros específicos do problema (ex: número de nós de um grafo)
- Por simplicidade, aqui representaremos como uma função do tamanho da cadeia de entrada (natural no caso de linguagens) – *n*

Medindo complexidade de tempo

- Análise de pior caso: maior tempo (número de passos) considerando todas as possíveis entradas de comprimento n
- Análise de caso médio: média dos tempos considerando todas as entradas de tamanho n
- **Definição**: Seja M uma Máquina de Turing determinística que **pára sobre todas** as entradas. O tempo de execução ou complexidade de tempo de M é a função $f: N \rightarrow N$, onde $f(n)$ é o número **máximo** de passos que M usa sobre entradas de comprimento n

Medindo complexidade de tempo

- As seguintes frases são equivalentes:
  - f(n) é o tempo de execução de M
  - M roda em tempo f(n)
  - M é uma máquina de Turing de tempo f(n)
  - M tem complexidade de tempo f(n)

Como calcular f?

- Calcular o número EXATO de passos é complicado
- Alternativa: estimar um valor aproximado
- Como? Análise assintótica (comportamento nos limites)
- Ex: $f(n) = 5n^8 + 6n^3 + 8$

o que realmente está “ditando o crescimento” de f(n)?

Como calcular f?

- Calcular o número EXATO de passos é complicado
- Alternativa: estimar um valor aproximado
- Como? Análise assintótica (comportamento nos limites)
- Ex: $f(n) = 5n^8 + 6n^3 + 8$

o que realmente está “ditando o crescimento” de $f(n)$? n^8

$O(f(n)) = n^8$

Notação assintótica (O-grande)

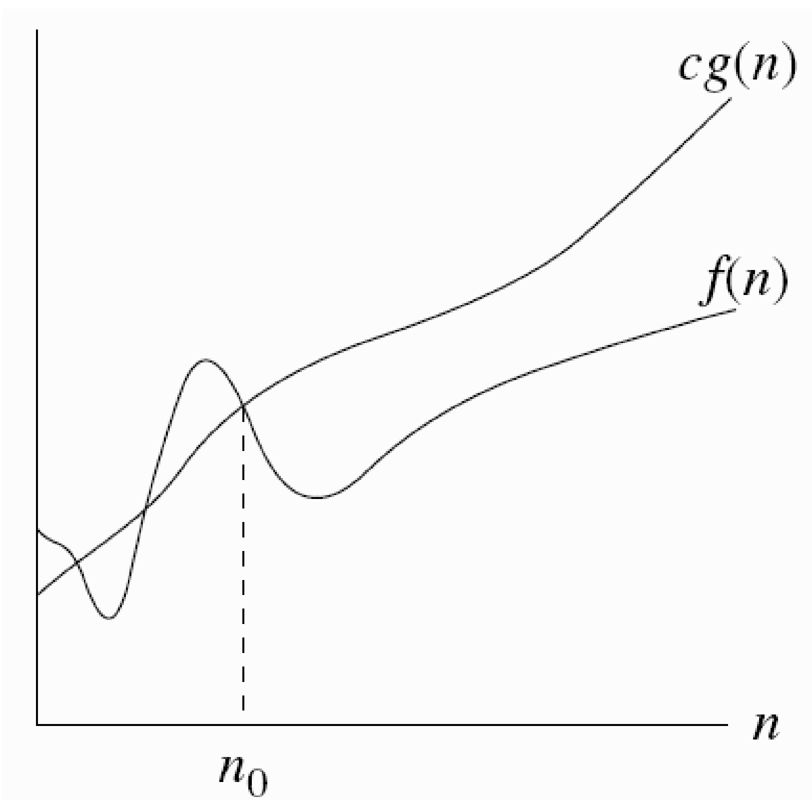
- Sejam f e g funções f, g: $N \rightarrow R^{+}$. Dizemos que $f(n) = O(g(n))$ se existem inteiros positivos c e n_0 tais que, para todo $n >= n_0$

$$f(n) <= cg(n)$$

Notação assintótica (O-grande)

- Sejam f e g funções f, g: $N \rightarrow R^+$. Dizemos que f(n) = O(g(n)) se existem inteiros positivos c e n_0 tais que, para todo n>= n_0

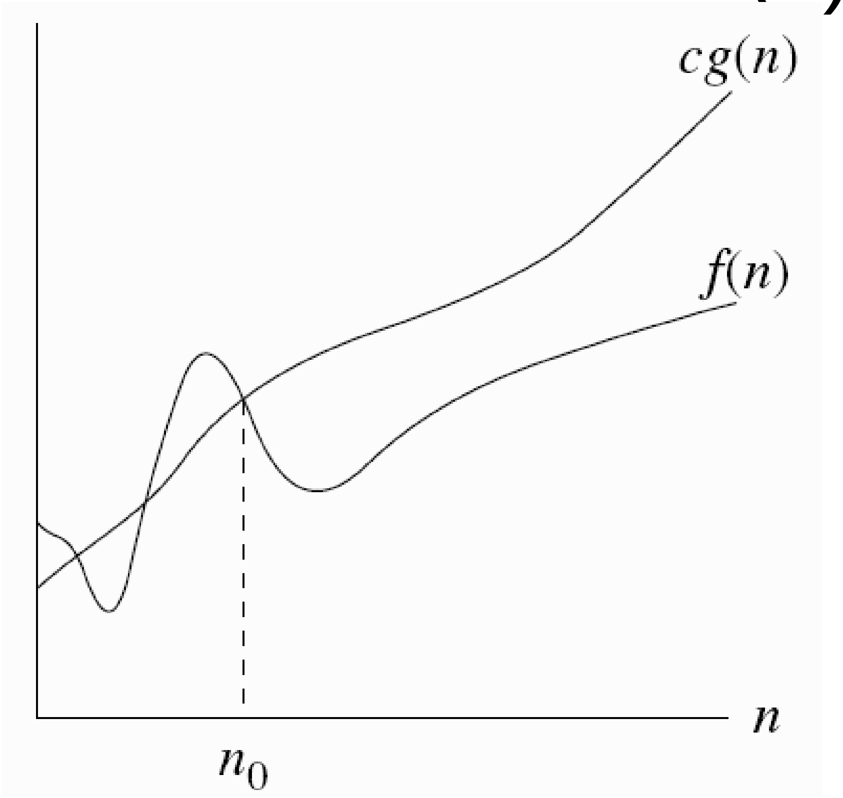
$$f(n) <= cg(n)$$



Notação assintótica (O-grande)

- Sejam f e g funções f, g: $N \rightarrow R^{+}$. Dizemos que $f(n) = O(g(n))$ se existir uma constante real positiva c e um inteiro positivo n_0 tais que, para todo $n >= n_0$

$$f(n) <= cg(n)$$

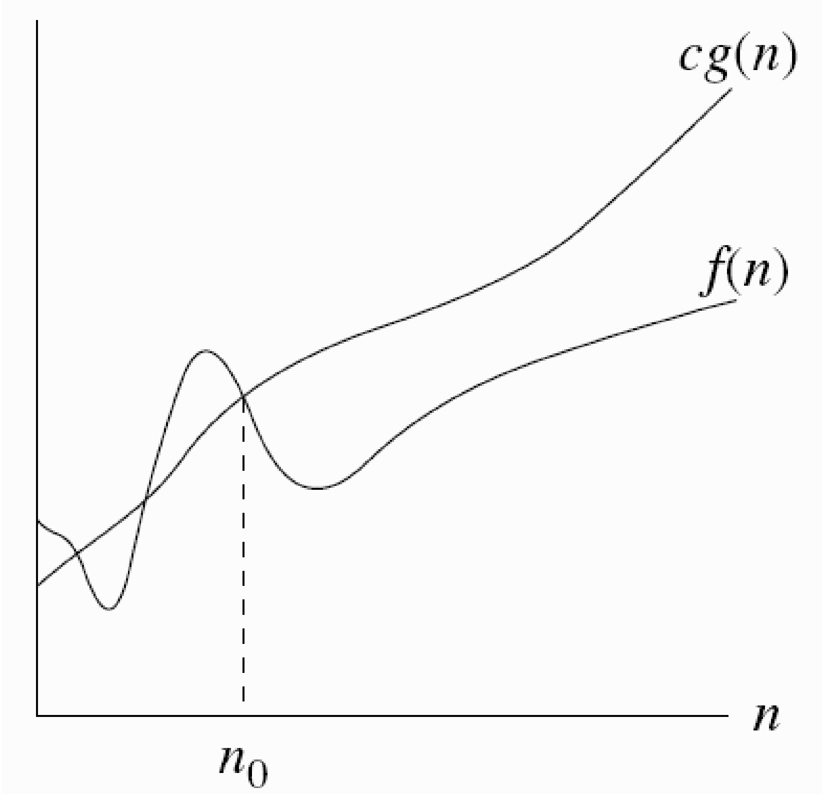


- g(n) é um limitante superior assintótico para f(n)

Notação assintótica (o-pequeno)

- Sejam f e g funções f, g: $N \rightarrow R^+$. Dizemos que f(n) = o(g(n)) se, <u>para qualquer</u> constante real positiva c, <u>existir</u> um inteiro positivo n_0 tal que, para todo $n >= n_0$

$$f(n) < cg(n)$$



Exemplos

- $\sqrt{n} = o(n)$
- $n = o(n \log \log n)$
- $n \log \log n = o(n \log n)$
- $n \log n = o(n^2)$
- $n^2 = o(n^3)$
- $n^2 = O(n^2)$ (f(n) nunca é o(f(n)))
- $O(n^2) + O(n) = O(n^2)$
- $c = O(1)$ (c é uma constante)

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*.”

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*."

Analisando algoritmos

- Uma MT para a linguagem A = $\{0^k1^k \mid k >= 0\}$:

M1 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*.”

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*.”

Analisando algoritmos

- Uma MT para a linguagem A = $\{0^k1^k \mid k >= 0\}$:

M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1 $n/2 * O(n) = O(n^2)$
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*."

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1 $n/2*O(n) = O(n^2)$
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*."

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1 $n/2*O(n) = O(n^2)$
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*."

$2n = O(n)$

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k 1^k \mid k >= 0\}$:

M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1 $n/2 * O(n) = O(n^2)$
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*." $2n = O(n)$

Total: $O(n) + O(n^2) + O(n) =$

Analisando algoritmos

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1 $2n = O(n)$
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1 $n/2 * O(n) = O(n^2)$
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*." $2n = O(n)$

Total: $O(n) + O(n^2) + O(n) = O(n^2)$

Classe de complexidade

- **Definição**: Seja uma função $t: N \rightarrow R^+$. A classe de complexidade de tempo **TIME(t(n))** é a coleção de todas as linguagens decidíveis por uma máquina de Turing de tempo O(t(n)).

- $A = \{0^k 1^k \mid k >= 0\} \in TIME(n^2)$

Classe de complexidade

- **Definição**: Seja uma função $t:N \rightarrow R^+$. A classe de complexidade de tempo **TIME(t(n))** é a coleção de todas as linguagens decidíveis por uma máquina de Turing de tempo O(t(n)).

- $A = \{0^k 1^k \mid k >= 0\}$ Є $TIME(n^2)$
- Pergunta: será que A Є TIME(t(n)) onde $t(n) = o(n^2)$

Classe de complexidade

- **Definição**: Seja uma função $t: N \rightarrow R^+$. A classe de complexidade de tempo **TIME(t(n))** é a coleção de todas as linguagens decidíveis por uma máquina de Turing de tempo O(t(n)).

- $A = \{0^k 1^k \mid k >= 0\} \in TIME(n^2)$
- Pergunta: será que A Є TIME(t(n)) onde $t(n) = o(n^2)$, ou seja, A pode ser decidida por uma MT que rode em menos tempo (mais rápida)?

- Uma MT para a linguagem A = $\{0^k1^k \mid k >= 0\}$:

M1 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*.”

Cortar dois 0s e dois 1s de cada vez resolve?

- Uma MT para a linguagem $A = \{0^k1^k \mid k >= 0\}$:

M1 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita se existem ambos, 0s e 1s, na fita:
3. Faça uma varredura na fita, cortando um único 0 e um único 1
4. Se ainda restarem 0s após todos os 1s terem sido cortados ou se ainda restarem 1s após todos os 0s terem sido cortados, *rejeite*. Caso contrário, *aceite*.”

Cortar dois 0s e dois 1s de cada vez resolve?
Corta-se o número de varreduras pela metade (2 é constante)
=> $O(n^2)$

Máquina alternativa

M2 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita enquanto alguns 0s E alguns 1s restarem na fita:
3. Faça uma varredura na fita, verificando se o número total de 0s e 1s restantes na fita é par ou ímpar. Se for ímpar, *rejeite*.
4. Faça uma varredura na fita, cortando alternadamente um 0 sim e outro não começando com o primeiro 0, e então cortando alternadamente um 1 sim e outro não começando com o primeiro 1
5. Se nenhum 0 e nenhum 1 restarem na fita, *aceite*. Caso contrário, *rejeite*.”

Analisando a complexidade

M2 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita enquanto alguns 0s E alguns 1s restarem na fita:
3. Faça uma varredura na fita, verificando se o número total de 0s e 1s restantes na fita é par ou ímpar. Se for ímpar, *rejeite*.
4. Faça uma varredura na fita, cortando alternadamente um 0 sim e outro não começando com o primeiro 0, e então cortando alternadamente um 1 sim e outro não começando com o primeiro 1
5. Se nenhum 0 e nenhum 1 restarem na fita, *aceite*. Caso contrário, *rejeite*.”

Cada estágio 1, 3, 4, 5 roda em O(n)
A questão é quantas vezes cada um deles roda?
1 e 5 rodam uma vez
3 e 4 ?

Analisando a complexidade

M2 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita e *rejeite* se for encontrado algum 0 à direita de algum 1
2. Repita enquanto alguns 0s E alguns 1s restarem na fita:
3. Faça uma varredura na fita, verificando se o número total de 0s e 1s restantes na fita é par ou ímpar. Se for ímpar, *rejeite*.
4. Faça uma varredura na fita, cortando alternadamente um 0 sim e outro não começando com o primeiro 0, e então cortando alternadamente um 1 sim e outro não começando com o primeiro 1
5. Se nenhum 0 e nenhum 1 restarem na fita, *aceite*. Caso contrário, *rejeite*.”

Cada estágio 1, 3, 4, 5 roda em O(n)

A questão é quantas vezes cada um deles roda?

1 e 5 rodam uma vez

3 e 4 rodam, no máximo, $1+\log_2 n$

Tempo total: $2*O(n) + (1+\log_2 n)*O(n) = O(n \log n)$

Classe de complexidade

- $A = \{0^k1^k \mid k >= 0\}$ Є $TIME(n^2)$
- Pergunta: será que A Є $TIME(t(n))$ onde $t(n) = o(n^2)$, ou seja, A pode ser decidida por uma MT que rode em menos tempo (mais rápida)?

Classe de complexidade

- $A = \{0^k1^k \mid k >= 0\} \in TIME(n^2)$
- Pergunta: será que $A \in TIME(t(n))$ onde $t(n) = o(n^2)$, ou seja, A pode ser decidida por uma MT que rode em menos tempo (mais rápida)?
  - Sim, $A \in TIME(n \log n)$
- Pode ser ainda mais rápido?

Classe de complexidade

- $A = \{0^k 1^k \mid k >= 0\}$ Є $TIME(n^2)$
- Pergunta: será que A Є TIME(t(n)) onde $t(n) = o(n^2)$, ou seja, A pode ser decidida por uma MT que rode em menos tempo (mais rápida)?
  - Sim, A Є TIME(n log n)
- Pode ser ainda mais rápido?
  - Não usando uma MT de fita única
  - Na verdade, só linguagens regulares podem ser reconhecidas em tempo o(n log n) em uma MT de fita única

Classe de complexidade

- $A = \{0^k1^k \mid k >= 0\}$ Є $TIME(n^2)$
- Pergunta: será que A Є TIME(t(n)) onde $t(n) = o(n^2)$, ou seja, A pode ser decidida por uma MT que rode em menos tempo (mais rápida)?
  - Sim, A Є TIME(n log n)
- Pode ser ainda mais rápido?
  - Não usando uma MT de fita única
  - Na verdade, só linguagens regulares podem ser reconhecidas em tempo o(n log n) em uma MT de fita única
- A Є TIME(n) se a máquina tiver duas fitas

Classe de complexidade

- Ideia: passar os 0s para a segunda fita e depois confrontá-los com os 1s
- M3 = “Sobre a cadeia de entrada w:

1. Faça uma varredura na fita 1 e *rejeite* se algum 0 for encontrado à direita de algum 1.
2. Faça uma varredura na fita 1 até encontrar o primeiro 1, copiando os 0s para a fita 2.
3. Faça uma varredura nos 1s sobre a fita 1 até o final da entrada. Para cada 1 lido, corte um 0 na fita 2. Se todos os 0s tiverem sido cortados antes que todos os 1s tenham sido lidos, *rejeite*.
4. Se todos os 0s tiverem sido cortados, *aceite*. Se restar algum 0, *rejeite*.

Classe de complexidade

- Dá para diminuir ainda mais a complexidade?

Classe de complexidade

- Dá para diminuir ainda mais a complexidade?
- Não, pois só para ler a cadeia demora O(n)

Classe de complexidade - Conclusões

- A classe de complexidade de uma linguagem depende do modelo de computação escolhido
- Diferentemente, em computabilidade o modelo não importa
  - Tese de Church-Turing implica que todos os modelos razoáveis de computáveis são equivalentes
- Pergunta: para classificar um problema segundo sua complexidade, que modelos escolhemos?
- Requisitos de tempo não diferem muito para os modelos determinísticos típicos, e portanto a escolha não terá muito impacto

Relacionamentos de Complexidade entre modelos

- Como a escolha do modelo computacional pode afetar a complexidade de tempo de um problema?
- Consideramos 3 modelos
  - Máquina de Turing fita-única (determinística)
  - Máquina de Turing multifita (determinística)
  - Máquina de Turing não-determinística (fita-única)

Relacionamentos de Complexidade entre modelos

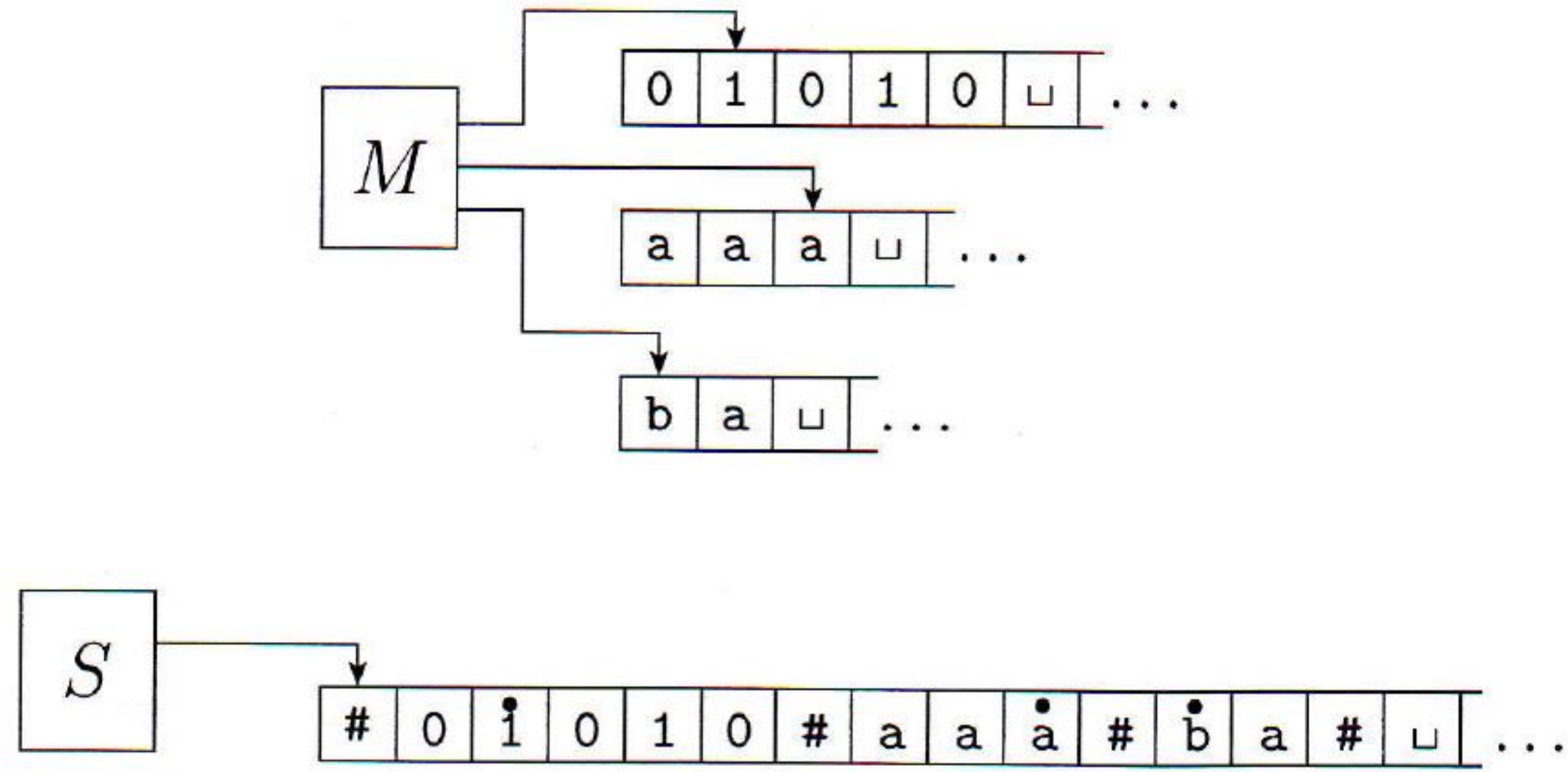
- Teorema: Seja t(n) uma função, onde t(n) >= n. Toda máquina de Turing multifita de tempo t(n) tem uma máquina de Turing fita-única equivalente de tempo $O(t^2(n))$

Relacionamentos de Complexidade entre modelos

- Teorema: Seja t(n) uma função, onde t(n) >= n. Toda máquina de Turing multifita de tempo t(n) tem uma máquina de Turing fita-única equivalente de tempo $O(t^2(n))$

- Ideia da Prova: Analisar a simulação de uma MT multifita M por uma MT fita-única S

MT multifita M e MT fita-única S PROVA



MT multifita M e MT fita-única S PROVA

- Cada passo em M, exige em S
  - Uma varredura na fita identificando as posições das cabeças de fita
  - Outra varredura na fita para atualizar conteúdo e posições de cabeça
  - Se uma cabeça se move para a direita “além do limite” (para uma posição ainda não usada na fita virtual), todo o conteúdo da fita a partir daquela posição deve ser deslocado para a direita.

MT multifita M e MT fita-única S PROVA

- Cada passo em M, exige em S
  - Uma varredura na fita identificando as posições das cabeças de fita
  - Outra varredura na fita para atualizar conteúdo e posições de cabeça
  - Se uma cabeça se move para a direita "além do limite" (para uma posição ainda não usada na fita virtual), todo o conteúdo da fita a partir daquela posição deve ser deslocado para a direita.

Para analisar a complexidade de cada tarefa, é preciso saber o comprimento da porção ativa da fita de S (um limitante superior)

MT multifita M e MT fita-única S PROVA

- Limitante superior para o comprimento da porção ativa da fita em S = soma das porções ativas das k fitas de M
  - Cada uma tem comprimento máximo t(n). (Se M roda em t(n) passos, usa no máximo t(n) células de cada fita, quando a cabeça move-se apenas para a direita)
- Limitante superior para o comprimento da porção ativa da fita em S: k*t(n) = O(t(n))
- Cada varredura na fita S: O(t(n))

MT multifita M e MT fita-única S PROVA

- Cada passo em M, exige em S
  - Uma varredura na fita identificando as posições das cabeças de fita
  - Outra varredura na fita para atualizar conteúdo e posições de cabeça
  - Se uma cabeça se move para a direita "além do limite" (para uma posição ainda não usada na fita virtual), todo o conteúdo da fita a partir daquela posição deve ser deslocado para a direita.
- Cada passo exige 2 varreduras e no máximo k deslocamentos = ?

MT multifita M e MT fita-única S PROVA

- Cada passo em M, exige em S
  - Uma varredura na fita identificando as posições das cabeças de fita
  - Outra varredura na fita para atualizar conteúdo e posições de cabeça
  - Se uma cabeça se move para a direita “além do limite” (para uma posição ainda não usada na fita virtual), todo o conteúdo da fita a partir daquela posição deve ser deslocado para a direita.
- Cada passo exige 2 varreduras e no máximo k deslocamentos = $O(t(n))$

MT multifita M e MT fita-única S PROVA

- Tempo total:
  - Início: montagem da fita de S: O(n)
  - Simulação dos passos de M:
    - Cada passo de M em O(t(n))
    - Quantos passos M realizava?
    -
  -
    -

MT multifita M e MT fita-única S PROVA

- Tempo total:
  - Início: montagem da fita de S: O(n)
  - Simulação dos passos de M:
    - Cada passo de M em O(t(n))
    - Quantos passos M realizava? t(n)
    - Logo, tempo de simulação =

MT multifita M e MT fita-única S PROVA

- Tempo total:
  - Início: montagem da fita de S: $O(n)$
  - Simulação dos passos de M:
    - Cada passo de M em $O(t(n))$
    - Quantos passos M realizava? $t(n)$
    - Logo, tempo de simulação = $O(t^2(n))$
  -
    -

MT multifita M e MT fita-única S PROVA

- Tempo total:
  - Início: montagem da fita de S: $O(n)$
  - Simulação dos passos de M:
    - Cada passo de M em $O(t(n))$
    - Quantos passos M realizava? $t(n)$
    - Logo, tempo de simulação = $O(t^2(n))$
  - Tempo total: $O(n) + O(t^2(n))$
    - Como $t(n) >= n$, tempo total:

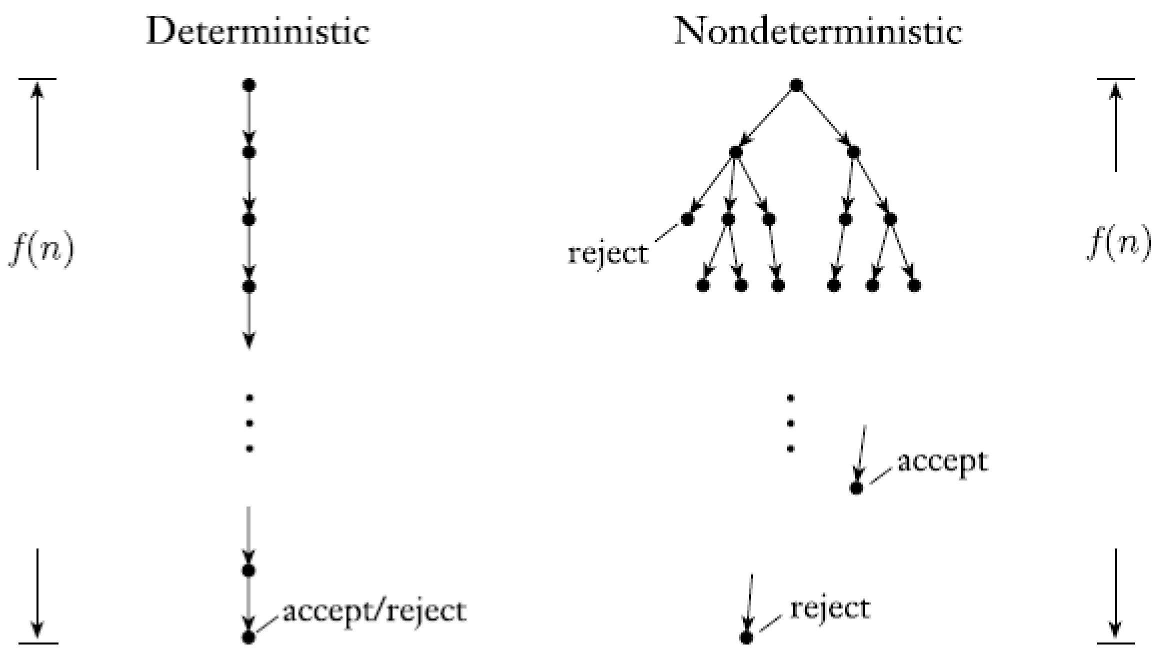
MT multifita M e MT fita-única S PROVA

- Tempo total:
  - Início: montagem da fita de S: $O(n)$
  - Simulação dos passos de M:
    - Cada passo de M em $O(t(n))$
    - Quantos passos M realizava? $t(n)$
    - Logo, tempo de simulação = $O(t^2(n))$
  - Tempo total: $O(n) + O(t^2(n))$
    - Como $t(n) >= n$, tempo total: $O(t^2(n))$

Comparando uma MT não determinística N e uma MT determinística D

- Uma MT não-determinística é decisora se TODOS os seus ramos de computação param sobre TODAS as entradas.
- **Definição**: Seja N uma MT não-determinística decisora. O tempo de execução de N é a função f:N→N, onde f(n) é o número máximo de passos que N usa sobre qualquer ramo de sua computação sobre qualquer entrada de comprimento n.

Trata-se de uma definição matemática, sem compromisso de corresponder a um dispositivo real

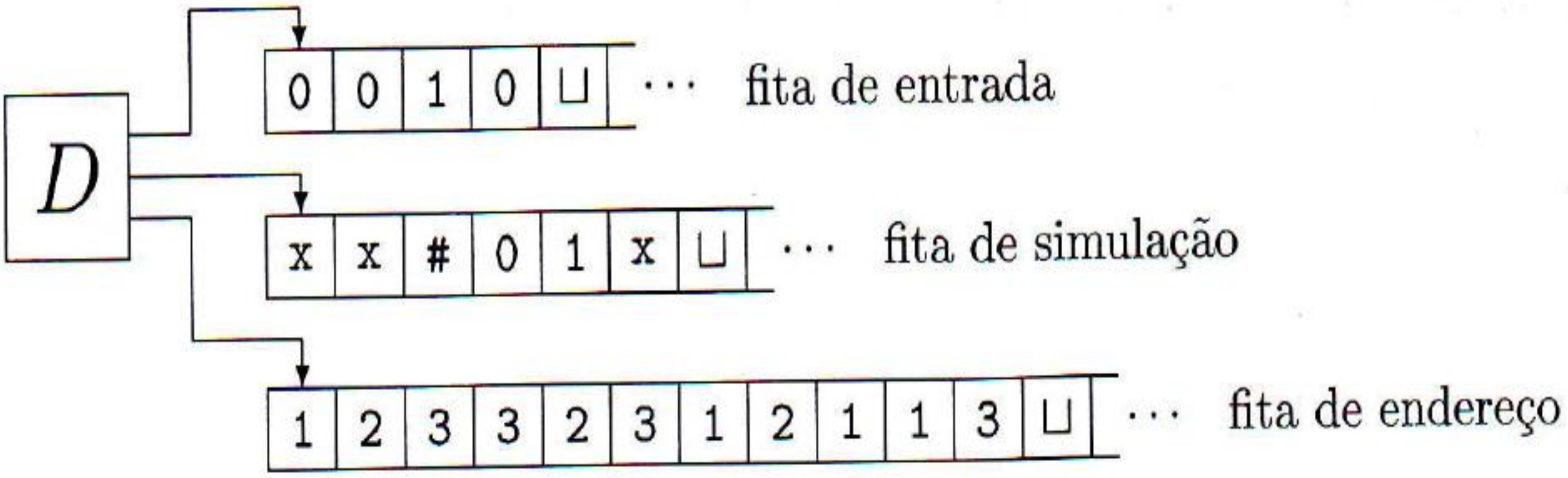


Comparando uma MT não determinística N e uma MT determinística D

- Teorema: Seja t(n) uma função, onde t(n) >= n. Toda máquina de Turing não-determinística fita-única de tempo t(n) tem uma máquina de Turing determinística fita-única equivalente de tempo $2^{O(t(n))}$

- Ideia da Prova: Analisar a simulação de uma MT não-determinística fita-única N por uma MT determinística fita-única D

Comparando uma MT não determinística N e uma MT determinística D



Comparando uma MT não determinística N e uma MT determinística D - PROVA

- Cada ramo da árvore de computação de N tem comprimento máximo t(n).
- Cada nó da árvore tem no máximo b filhos (b=número máximo de alternativas dado pela função de transição)
  - Número máximo de folhas: $b^{t(n)}$
- Simulação: busca em largura na árvore - para cada nó, desce-se da raiz até ele
  - Número máximo de nós < 2*número de folhas = $O(b^{t(n)})$
  - Tempo para descer da raiz até um nó: O(t(n))
  - Logo, o tempo total é ?

Comparando uma MT não determinística N e uma MT determinística D - PROVA

- Cada ramo da árvore de computação de N tem comprimento máximo t(n).
- Cada nó da árvore tem no máximo b filhos (b=número máximo de alternativas dado pela função de transição)
  - Número máximo de folhas: $b^{t(n)}$
- Simulação: busca em largura na árvore - para cada nó, desce-se da raiz até ele
  - Número máximo de nós < 2*número de folhas = $O(b^{t(n)})$
  - Tempo para descer da raiz até um nó: O(t(n))
  - Logo, o tempo total é $O(t(n)*b^{t(n)}) = 2^{O(t(n))}$