ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 19

Cap 3.3 – Definição de Algoritmo

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

Aulas passadas

- Máquinas de Turing: falamos que era um modelo de computação, mas só mostramos um dispositivo para reconhecer (ou decidir) linguagens
  - Descrição formal
  - Descrição de implementação
  - Descrição de alto-nível
- Variantes de máquinas de Máquinas de Turing
  - Determinística multi-fita
  - Não-determinística
  - Enumeradores
  - Fita ilimitada de ambos os lados (exercício)

Aula de hoje

- Máquinas de Turing como um modelo de computação de fato, como um sinônimo de algoritmo

O que é um algoritmo?

O que é um algoritmo?

Muito “usado” há tempos, mas formalmente definido apenas no século XX

Charles Babbage (inglês)

- Notou:
  - Muitos erros em tabelas de cálculos
  - Que muito do que se fazia em matemática poderia ser automatizado
- Iniciou projeto da máquina de diferenças (*Difference Engine*).
- Capaz de resolver equações polinômicas

1812

Máquina de Diferenças

- 1822: terminou um protótipo de máquina e obteve financiamento do governo inglês para construí-la.

1822 - 1823

Máquina de Diferenças

- 1823: iniciou construção (usaria motor a vapor, seria totalmente automática, imprimiria o resultado e teria um programa fixo).
  - Mudar o cálculo implicaria em mudar as engrenagens

1822 - 1823

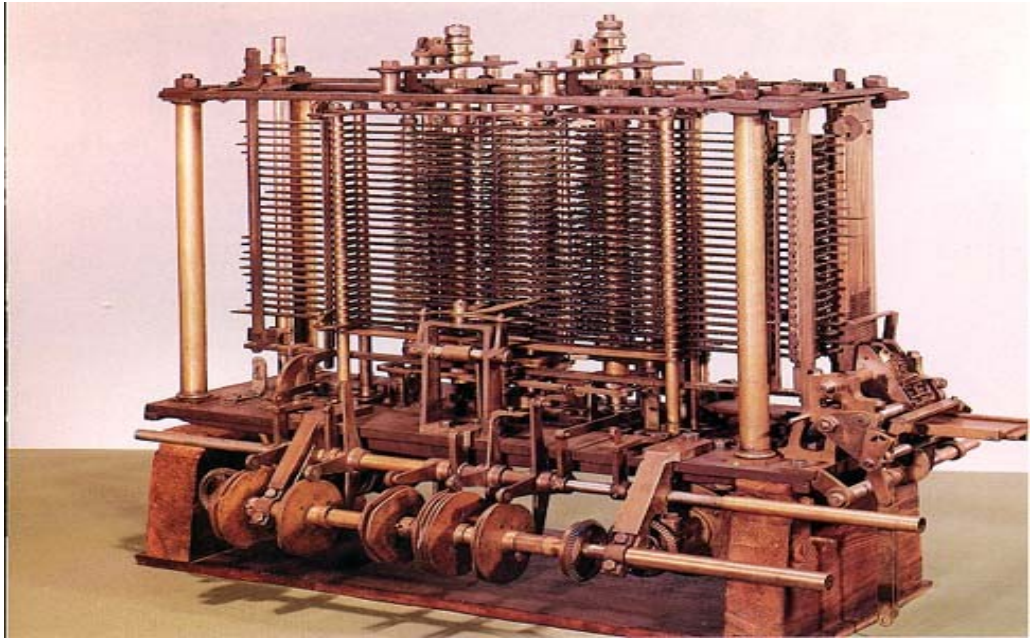
Mas...

- 1833: Teve uma idéia melhor e abandonou tudo.
- Nova idéia: máquina *programável*, de propósito geral: **máquina analítica** (*Analytical Engine*).
- Ponto de partida para os computadores eletrônicos!

1833

Máquina Analítica - programável

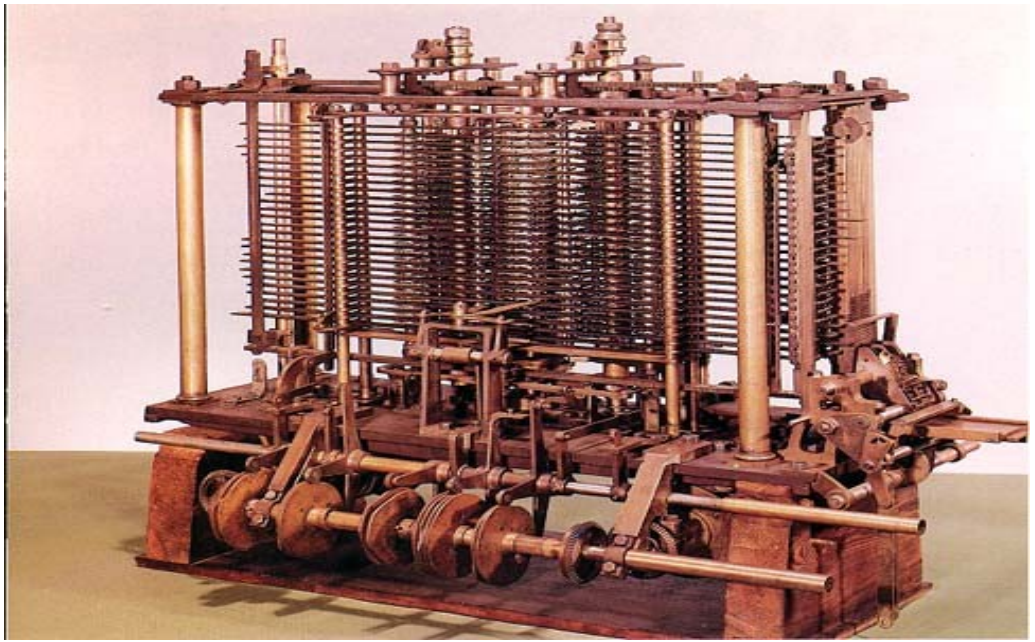
- Manipularia números de 50 dígitos
- Memória de 1000 dígitos
- Estações de leitura: cartões perfurados (programas!)
- Pai da computação



1833

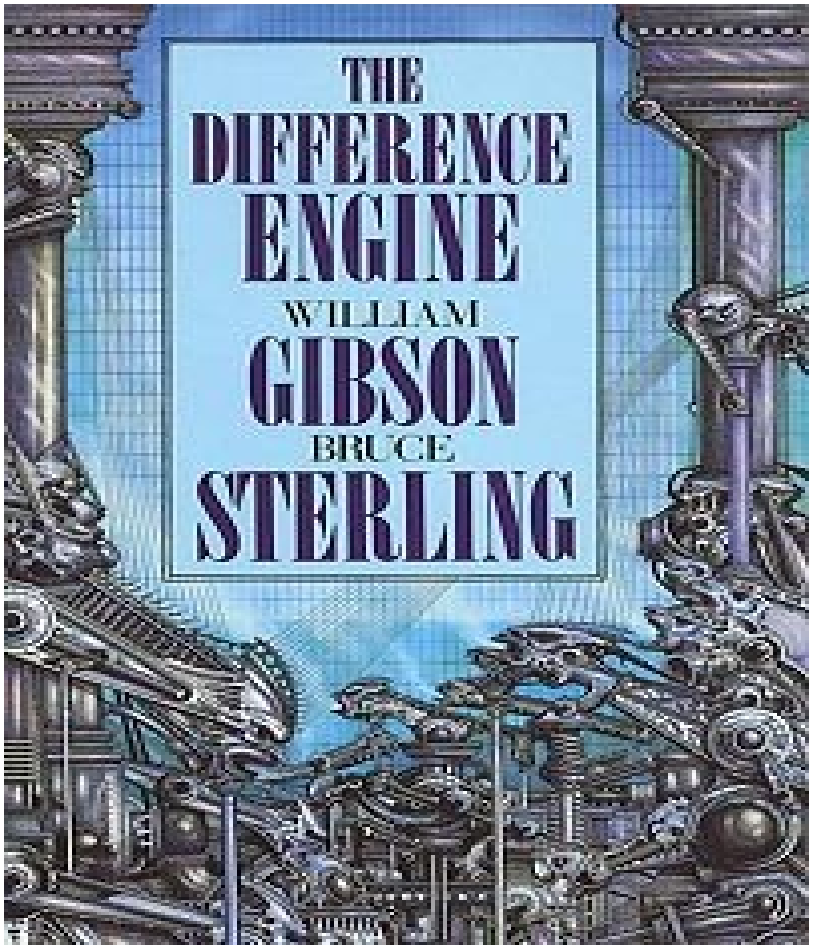
Máquina Analítica - programável

- Continuou a trabalhar no projeto até sua morte
- Não conseguiu construir
  - Tecnologia mecânica da época era insuficiente (???)
  - Pouca gente via a necessidade para tal máquina.



1833-1871

Livro de ficção



Programas para a máquina analítica

- Ada Lovelace (*mãe da programação*)
- Criou sub-rotinas, loops, saltos condicionais para a futura máquina
- Cunhou a palavra **algoritmo** em homenagem ao matemático Al-Khawarizmi (820 D.C.) (pronúncia: algo como "Al-rruá-riz-mi")

1833-1871

Programas para a máquina analítica

- **Algoritmo:** sequência de operações ou comandos que, aplicada a um conjunto de dados, permite solucionar classes de problemas semelhantes.
- Ex:
  - algoritmo da divisão,
  - da raiz cúbica,
  - resolução de equações de segundo grau,
  - etc.

1833-1871

Programas para a máquina analítica

- **Algoritmo:** sequência de operações ou comandos que, aplicada a um conjunto de dados, permite solucionar classes de problemas semelhantes.
- Ex:
  - algoritmo da divisão,
  - da raiz cúbica,
  - resolução de equações de segundo grau,
  - etc.

1833-1871

Vídeo interessante e lúdico sobre ela: https://youtu.be/jPq9wVsCv9I

Um pouco depois...

- 1900 – publicação do matemático David Hilbert de 23 desafios matemáticos para o próximo século
  - Décimo problema: “um processo pelo qual possa ser determinado, com um número finito de operações, se um polinômio tem raízes inteiras”.

Um pouco depois...

- 1900 – publicação do matemático David Hilbert de 23 desafios matemáticos para o próximo século
  - Décimo problema: “um processo pelo qual possa ser determinado, com um número finito de operações, se um polinômio tem raízes inteiras”.

Um pouco depois...

- 1900 – publicação do matemático David Hilbert de 23 desafios matemáticos para o próximo século
  - Décimo problema: “um processo pelo qual possa ser determinado, com um número **FINITO** de operações, se um polinômio tem raízes inteiras”.

ESTAVAM ATRÁS DE UM ALGORITMO!

Um pouco depois...

- 1936 – artigos de Alonzo Church e Alan Turing definindo formalmente um algoritmo
  - Church com lambda-cálculo
  - Turing com Máquinas de Turing
  - As duas formulações são equivalentes

Vale honrarmos sua memória

- Realizou inúmeras contribuições
  - Máquinas de Turing
  - Criptografia
  - Enigma (estimativas que a 2ª Guerra Mundial acabou 2 anos antes devido à sua contribuição, salvando milhares de vidas)
  - Teste de Turing (Inteligência Artificial)
- Era homossexual:
  - Sofreu preconceito
  - Perdeu seu trabalho
  - Obrigado a sujeitar-se a uma castração química
  - Tudo isso levou-o a suicidar-se em 1954 (com 42 anos)
- Somente em 2013 o primeiro ministro inglês e a rainha Elizabeth II reconhecem o erro em um pedido póstumo de desculpas
  - A "lei Alan Turing" é agora um termo informal para uma lei britânica de 2017 que retroativamente “perdoou” homens advertidos ou condenados sob a legislação histórica que proibia atos homossexuais.

Em 1999, a revista Time nomeou Turing como uma das 100 pessoas mais importantes do século XX e declarou: "O fato é que todo mundo que toca em um teclado, abrindo uma planilha ou um programa de processamento de texto, está trabalhando em uma encarnação de uma máquina de Turing."

Tese de Church-Turing

Noção intuitiva de algoritmos	é igual a	*algoritmos de máquina de Turing*

Turing machines explained visually

https://www.youtube.com/watch?v=-ZS_zFg4w5k

Algoritmo para o problema de Hilbert

- Problema de Hilbert:

"um processo pelo qual possa ser determinado, com um número finito de operações", se um polinômio tem raízes inteiras

- 1970 – Yuri Matijasevic mostrou que não existe tal "processo" (ou algoritmo)
  - Não é Turing-decidível…
  - Mas como ver esse problema como uma linguagem?

Algoritmo para o problema de Hilbert

- Problema de Hilbert:

  D = { p | p é um polinômio com uma raiz inteira}

  D é decidível?

Algoritmo para o problema de Hilbert

- Problema de Hilbert:

  D = { p | p é um polinômio com uma raiz inteira}

  D é decidível?

- 1970 – Yuri Matijasevic mostrou que não
- Próximas aulas: como fazer esse tipo de prova.

Problema simplificado

$$D_1 = \{p | \ p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}.$$

Problema simplificado

$$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}.$$

Aqui está uma MT M_1 que reconhece D_1:

M_1 = "A entrada é um polinômio p sobre a variável x.

Note que agora especificamos quem é a cadeia de entrada

Problema simplificado

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}$.

Aqui está uma MT M_1 que reconhece D_1:

Note que agora especificamos quem é a cadeia de entrada

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Está MT é reconhecedora?

Problema simplificado

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}$.

Aqui está uma MT M_1 que reconhece D_1:

Note que agora especificamos quem é a cadeia de entrada

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Está MT é reconhecedora? SIM!
É decidível?

Problema simplificado

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}$.

Aqui está uma MT M_1 que reconhece D_1:

Note que agora especificamos quem é a cadeia de entrada

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Está MT é reconhecedora? SIM!
É decidível? Essa Máquina não, mas é possível escrever uma MT decisora SIM!!!
As raízes de um polinômio de uma só variável devem residir entre os dois valores:

$$\pm k \frac{c_{\text{máx}}}{c_1},$$

onde k é o número de termos no polinômio, $c_{\text{máx}}$ é o coeficiente com o maior valor absoluto, e c_1 é o coeficiente do termo de mais alta ordem. Se uma raiz não for encontrada dentro desses limitantes, a máquina *rejeita*.

Exercício: transforme essa MT em decisora

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}.$

Aqui está uma MT M_1 que reconhece D_1:

Note que agora especificamos quem é a cadeia de entrada

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Está MT é reconhecedora? SIM!
É decidível? Essa Máquina não, mas é possível escrever uma MT decisora SIM!!!
As raízes de um polinômio de uma só variável devem residir entre os dois valores:

$$\pm k \frac{c_{\text{máx}}}{c_1},$$

onde k é o número de termos no polinômio, $c_{\text{máx}}$ é o coeficiente com o maior valor absoluto, e c_1 é o coeficiente do termo de mais alta ordem. Se uma raiz não for encontrada dentro desses limitantes, a máquina *rejeita*.

Máquina M_{D1} que DECIDE D_1

$$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}.$$

M_{D1} = “Sobre a entrada *p* que representa um polinômio sobre x,

1. Calcule $max = k * (c_{max} / c_1)$, sendo *k* o número de termos de *p*, c_{max} o coeficiente de maior valor absoluto e c_1 o coeficiente do termo de mais alta ordem.
2. $i = 0$
3. Enquanto $i <= max$, calcule o valor de *p* com o valor de *x* substituído por *i* e *-i*. Se para um deles o resultado for igual a 0, *aceite; senão* $i = i + 1$
4. *Rejeite.*

MT versus programas

Que paralelo você faria?

MT versus programas

Que paralelo você faria?

MT são como funções booleanas (que retornam true ou false):

- aceite = return true
- rejeite = return false

A questão é como descrever qualquer problema em termos de linguagem
(ou seja, de forma a ser resolvível por uma função booleana)

O problema original

Matijasevic mostra que, para polinômios com várias variáveis, não é possível calcular tais limitantes

Logo, D = { p | p é um polinômio (geral) com uma raiz inteira} é

O problema original

Matijasevic mostra que, para polinômios com várias variáveis, não é possível calcular tais limitantes

Logo, D = { p | p é um polinômio (geral) com uma raiz inteira} é Turing-reconhecível mas não Turing-decidível

Terminologia para descrever Máquinas de Turing

- Mudança de foco no curso: algoritmos
  - Máquina de Turing como modelo de computação
  - Precisamos estar convencidos de que podemos descrever qualquer algoritmo com uma máquina de Turing

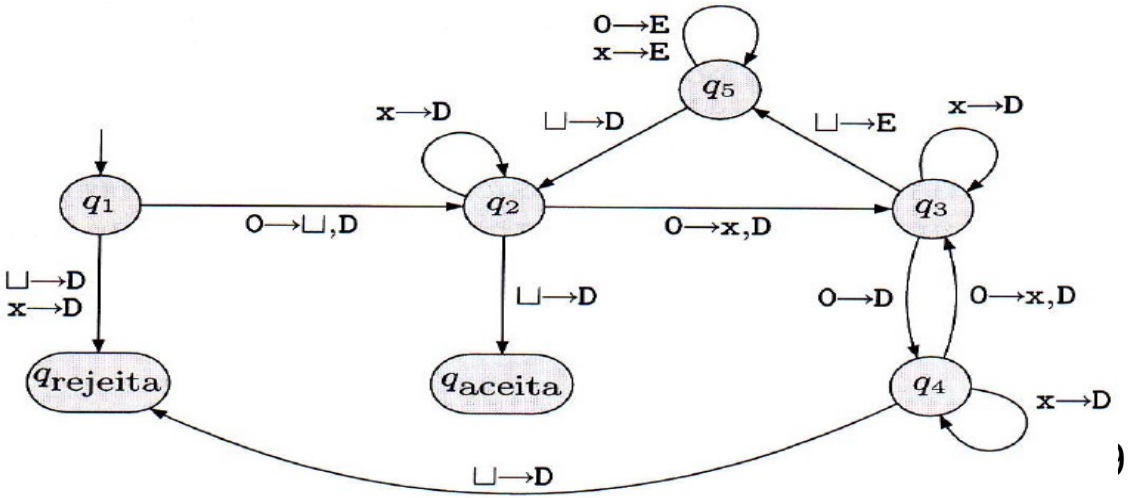
Terminologia para descrever Máquinas de Turing

- 3 níveis de descrição de algoritmos:
  - **Descrição formal**: detalhes da máquina: estados, função de transição, etc.
  - **Descrição de implementação**: escrito em língua natural para descrever como a máquina move a cabeça da fita, lê e escreve dados, etc (sem descrever estados ou função de transição)
  - **Descrição de alto nível**: escrito em língua natural para descrever um algoritmo, omitindo detalhes de implementação

Exemplo – descrição formal (se o nr de zeros de uma cadeia é uma potência de 2)

Agora, damos a descrição formal de $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{aceita}}, q_{\text{rejeita}})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{aceita}}, q_{\text{rejeita}}\}$,
- $\Sigma = \{0\}$ e
- $\Gamma = \{0, x, \sqcup\}$.
- Descrevemos δ com um diagrama de estados (veja a Figura 3.8).
- Os estados inicial, de aceitação e de rejeição são q_1, q_{aceita} e q_{rejeita}.



Exemplo – descrição de implementação (se o nr de zeros de uma cadeia é uma potência de 2)

EXEMPLO 3.7

Aqui descrevemos uma máquina de Turing (MT) M_2 que decide $A = \{0^{2^n} \mid n \geq 0\}$, a linguagem consistindo em todas as cadeias de 0s cujo comprimento é uma potência de 2.

M_2 = "Sobre a cadeia de entrada w:

1. Faça uma varredura da esquerda para a direita na fita, marcando um 0 não, e outro, sim.
2. Se no estágio 1, a fita continha um único 0, *aceite*.
3. Se no estágio 1, a fita continha mais que um único 0 e o número de 0s era ímpar, *rejeite*.
4. Retorne a cabeça para a extremidade esquerda da fita.
5. Vá para o estágio 1."

Exemplo – descrição de alto nível (se um polinômio sobre x tem raiz inteira)

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Terminologia para descrever Máquinas de Turing

- Até agora usamos as descrições formais e de implementação
- Passaremos a usar mais a descrição de alto nível
  - Objetos (O) convertidos em cadeias (<O>)
    - Ex de O: polinômio p sobre a variável x
  - Vários objetos em uma única cadeia ($<O_1, O_2, ..., O_k>$)
  - Assumimos que as MTs são capazes de decodificar essas cadeias

Descrição de alto nível de Máquinas de Turing

- M = “ …

“

- Primeira linha: entrada da máquina
  - w é cadeia
  - <w> é objeto codificado em cadeia – implicitamente MT testa se a codificação está ok, se não estiver rejeita

M_1 = “A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores 0, 1, −1, 2, −2, 3, −3, . . . Se em algum ponto o valor do polinômio resulta em 0, *aceite*.”

EXEMPLO **3.23**

Vamos considerar o problema de verificar se um dado grafo G não direcionado é conexo

Como seria a função booleana? (Quando retornaria true e false?)

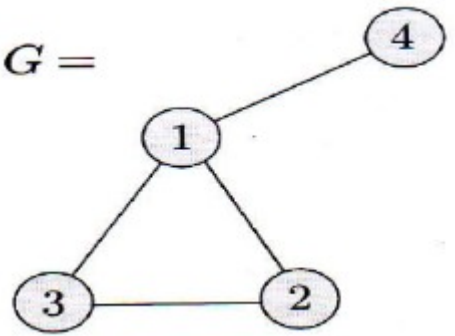
Então como seria a linguagem?

EXEMPLO 3.23

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Lembre-se de que um grafo é ***conexo*** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos

$$A = \{\langle G \rangle \mid G \text{ é um grafo não-direcionado conexo}\}.$$

$G =$



EXEMPLO 3.23

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Lembre-se de que um grafo é ***conexo*** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos

$$A = \{\langle G \rangle \mid G \text{ é um grafo não-direcionado conexo}\}.$$

O que se segue é uma descrição de alto nível de uma MT M que decide A.

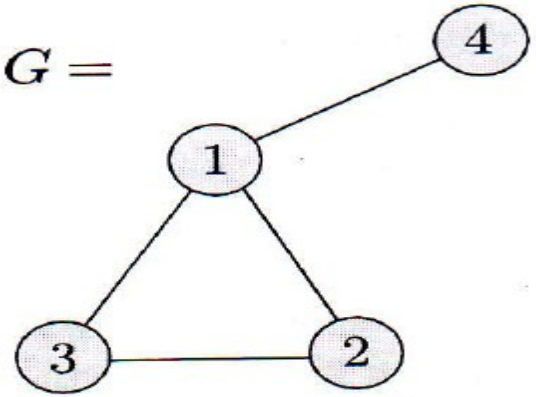
M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
3. Para cada nó em G, marque-o, se ele estiver ligado por uma aresta a um nó que já esteja marcado.
4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*; caso contrário, *rejeite*."

Detalhes de implementação (só desta vez...)

- Codificação:
  - G = (V,A) no qual V é o conjunto de vértices e A é o conjunto de arestas
  - <G> = lista de vértices (números decimais) e lista de arestas (pares desses números)

$G =$



$\langle G \rangle =$

(1,2,3,4)((1,2),(2,3),(3,1),(1,4))

Detalhes de implementação (só desta vez...)

Dá para imaginar os detalhes de implementação (até mesmo o diagrama de estados):

- Teste da codificação: a cadeia w deve
  - Conter duas listas, uma de decimais e outra de pares de decimais
  - A lista de nós não deve ter repetições
  - A lista de arestas só pode ter nós da lista de nós
- Obs.: verificação da distinção de elementos – exemplo 3.12 do livro do Sipser

EXEMPLO 3.23

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Lembre-se de que um grafo é ***conexo*** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos

$$A = \{\langle G \rangle \mid G \text{ é um grafo não-direcionado conexo}\}.$$

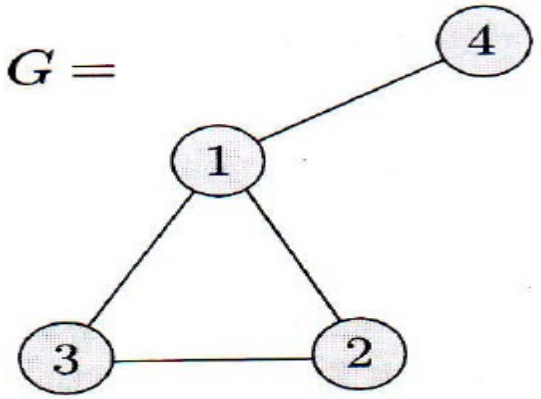
O que se segue é uma descrição de alto nível de uma MT M que decide A.

M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
3. Para cada nó em G, marque-o, se ele estiver ligado por uma aresta a um nó que já esteja marcado.
4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*; caso contrário, *rejeite*."

Detalhes de implementação (só desta vez...)

- Estágio 1: M marca o primeiro nó com um ponto no dígito mais à esquerda



$\langle G \rangle =$

(1*,2,3,4)((1,2),(2,3),(3,1),(1,4))

EXEMPLO 3.23

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Lembre-se de que um grafo é ***conexo*** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos

$$A = \{\langle G \rangle \mid G \text{ é um grafo não-direcionado conexo}\}.$$

O que se segue é uma descrição de alto nível de uma MT M que decide A.

M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
3. Para cada nó em G, marque-o, se ele estiver ligado por uma aresta a um nó que já esteja marcado.
4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*; caso contrário, *rejeite*."

Detalhes de implementação (só desta vez...)

- Estágios 2 e 3:
  - a) Varra a lista de nós procurando um nó não marcado com ponto (n_1)
    - Marque n_1 com sublinhado
  - b) Varra a lista de nós novamente procurando um nó marcado com ponto (n_2)
    - Marque n_2 com sublinhado
  - c) Varra a lista de arestas procurando uma aresta entre n_1 e n_2
    - Se achar, tire o sublinhado de n_1 e n_2 e marque n_1 com ponto e volte para o início do estágio 2
    - Senão, mova o sublinhado de n_2 para outro nó marcado (chame esse de n_2) e repita o passo c)
  - d) Se já analisou todos os nós marcados (n_1 não está conectado a nenhum nó marcado até o momento)
    - Se ainda houver nós não marcados, mova o sublinhado de n_1 para o próximo nó não marcado e repita os passos b) e c).
    - Senão vá para o estágio 4 (não conseguiu marcar nenhum nó novo)

EXEMPLO 3.23

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Lembre-se de que um grafo é ***conexo*** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos

$$A = \{\langle G \rangle \mid G \text{ é um grafo não-direcionado conexo}\}.$$

O que se segue é uma descrição de alto nível de uma MT M que decide A.

M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
3. Para cada nó em G, marque-o, se ele estiver ligado por uma aresta a um nó que já esteja marcado.
4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*; caso contrário, *rejeite*."

Detalhes de implementação (só desta vez...)

- Estágio 4: varre a lista de nós verificando se todos estão com ponto
  - Se sim, entra em um estado de aceitação
  - senão, entra em um estado de rejeição

Problema ↔ Linguagem

Algoritmo ↔ Máquina de Turing

EXEMPLO 3.23

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Lembre-se de que um grafo é ***conexo*** se todo nó pode ser atingido a partir de cada um dos outros nós passando pelas arestas do grafo. Escrevemos

$$A = \{\langle G \rangle \mid G \text{ é um grafo não-direcionado conexo}\}.$$

O que se segue é uma descrição de alto nível de uma MT M que decide A.

M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
3. Para cada nó em G, marque-o, se ele estiver ligado por uma aresta a um nó que já esteja marcado.
4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*; caso contrário, *rejeite*."