Algoritmos e Fundamentos da Teoria de Computação

Lista de Exercícios 04

1 Considere o problema de decisão cujas instâncias são definidas pela pergunta: **"O número natural n é ímpar?"** Defina uma *representação* para as instâncias deste problema e a seguir apresente uma máquina de Turing que decide as instâncias codificadas.

É fácil resolver essa questão utilizando uma representação *binária* para os números naturais. Assim, temos que se o número é par, o seu bit menos significativo é 0, e se é ímpar, o bit é 1. Logo, basta criar uma DTM Padrão que percorre a entrada até o final e inspeciona o bit menos significativo. Se for 0, rejeita; se for 1, aceita. Como a DTM sempre para, ela decide esse problema.

Não é muito mais difícil utilizar uma representação unária para as instâncias, mas nesse caso a TM precisa ficar em *loop* contando o número de 1s já vistos. O objetivo desse exercício é ilustrar como a escolha da representação das instâncias influencia depois no projeto da máquina.

2 Seja L a linguagem contendo apenas uma única *string* s, aonde

$$s = \begin{cases} 0 & \text{se nunca será encontrada vida em Marte.} \\ 1 & \text{se algum dia será encontrada vida em Marte.} \end{cases}$$

A linguagem L é decidível? Justifique adequadamente sua resposta.
Obs.: Para os propósitos deste problema, assuma que a questão de existência de vida em Marte possui uma resposta definitiva 'sim' ou 'não'.

Essa questão é trivial porque qualquer linguagem representada por um conjunto finito é decidível. Como *card* $(\mathsf{L}) = 1$, temos que L é decidível. O resto do enunciado é somente ruído.

3 Usando a representação unária de números naturais, projete uma TM que decide se um número natural é *primo*. *Obs. 1:* Assuma a existência de uma TM DIV que recebe como entrada $BmBnB$ aonde $m, n > 0$ estão escritos em notação unária na fita. A máquina DIV retorna como saída o resultado m/n se este for um número natural e 0, caso contrário. *Obs. 2:* Apresente a sua solução como um algoritmo, isto é, não é necessário apresentar o diagrama completo da máquina, só descrever como ela funciona.

Vamos usar uma DTM de 3 fitas para resolver esse problema. A fita 1 contém a entrada, o número natural m que deve ser testado para primalidade. A fita 2 guarda o valor do divisor n, que vai variar de 2 até $m - 1$. A fita 3 é utilizada para executar a TM DIV. A sequência de passos da máquina é a seguinte.

1. Escreva 2 na fita 2.
2. Zere a fita 3 e a seguir copie o valor de m da fita 1 seguido do valor de n da fita 2, gerando a *string* $BmBnB$ na fita 3. Execute DIV na fita 3.
3. Se o resultado de DIV é zero e o valor de n na fita 2 é $m - 1$, a máquina para e aceita. Se o resultado de DIV é zero e valor de n na fita 2 é menor que $m - 1$, incremente n na fita 2 e volte para o passo 2.
4. Se o resultado de DIV não é zero, a máquina para e rejeita.

4 Considere o problema de decisão abaixo.

DAG (***Directed Acyclic Graph***)

Input: um grafo direcionado G.

Output: sim; se G é acíclico.
não; caso contrário.

Esse problema **DAG** é *decidível*, *semi-decidível* ou *indecidível*?

1. Se a sua resposta for *decidível*, você deve apresentar uma máquina de Turing que *sempre termina* e responde sim ou não para as instâncias do problema.
2. Se a sua resposta for *semi-decidível*, você deve apresentar uma máquina de Turing que sempre *termina para as instâncias cuja resposta é sim*. Para as instâncias cuja resposta é não, a máquina não precisa terminar.
3. Se a sua resposta for *indecidível*, você deve *provar* porque é impossível projetar uma máquina de Turing para decidir esse problema.

Obs.: Nos casos 1 ou 2, você pode descrever a máquina de Turing como um algoritmo, isto é, não é necessário apresentar o diagrama completo da máquina, só descrever como ela funciona.

O problema **DAG** é *decidível* e vamos projetar uma máquina de Turing não-determinística M com 4 fitas para resolver esse problema. Um nó v_i de um grafo G é dito *acíclico* se não é possível construir um caminho em G que parte de v_i e retorna a v_i passando por pelo menos uma aresta. Um grafo G é *acíclico* se todos os seus nós são acíclicos. Partindo dessa definição, fica fácil construir M: a máquina fica em um *loop* testando se cada nó de G é acíclico. Se algum nó for cíclico a máquina termina e responde não. Se todos os nós passarem no teste, a máquina termina e responde sim. Vamos assumir que o grafo G é conexo (isto é, não existe nenhum nó sem aresta incidente) e que os nós de G são numerados de v_1 a v_n. Seja $R(G)$ a representação de G que é entregue como entrada para M na fita 1. Na fita 2 vamos guardar v_n, na fita 3, o nó v_i atualmente sendo testado, e na fita 4, o caminho que parte de v_i. Os passos de M são descritos abaixo.

1. Teste se a entrada está correta, isto é, se tem a forma $R(G)$. Se não estiver, M termina e rejeita a entrada.
2. Percorra a lista de arestas em $R(G)$ para determinar o maior nó v_n. Escreva v_n na fita 2.
3. Escreva v_1 na fita 3.
4. Seja v_i $(0 < i \leq n)$ na fita 3. Escreva $en(v_i)0$ na fita 4.
5. Seja v_s o nó mais à direita na fita 4. Escolha um arco $[v_s, v_t]$ de forma não-determinística. Se $v_t = v_i$, M termina e rejeita, pois v_i é cíclico. Se $v_t \neq v_i$, M escreve $en(v_t)0$ na fita 4 e esse passo é repetido. Se não existe um arco para ser escolhido nesse passo, vá para o passo 6.
6. Comparar os nós da fita 2 e 3. Se $v_i = v_n$, M termina e aceita, pois todos os nós foram testados e nenhum é cíclico. Senão, escreva v_{i+1} na fita 3 e volte para o passo 4.

5 Dada uma TM M arbitrária com *string* de entrada w, é possível determinar se a computação de M para a entrada w *termina* em *menos* de 100 transições. Descreva uma máquina de Turing que resolve esse problema de decisão.

É simples de se modificar a máquina do Exemplo 11.5.2 (Aula 04, slide 28), que resolve o problema de decidir se uma TM para com exatamente n transições. A solução fica como abaixo.

Modifique a máquina universal U para U', adicionando uma quarta fita para registrar o número de transições na simulação de M. Represente uma instância do problema como $R(\mathsf{M})w$. A computação de U' segue os seguintes passos.

1. Se a *string* na fita 1 não tem a forma $R(\mathsf{M})w$, U' para e rejeita.
2. Escreve 1^{99} na fita 4. Posiciona a cabeça no início da fita 4.

3. Copia w na fita 3. Escreve $en(q_0)$ na fita 2.
4. Simula uma transição de M, usado a fita 4 como contador de transições (mover para a direita sempre que um 1 é lido).
   - Se M termina e U' não lê branco (lê 1), U' para e aceita.
   - Se M deve tomar uma nova transição mas U' está lendo branco, U' para e rejeita pois M fez no mínimo 100 transições.

6 Mostre que o problema de decisão abaixo é *decidível*.

Input: máquina de Turing M
Output: sim; se existe uma *string* $w \in \Sigma^*$ para a qual a computação de M leva mais de 10 transições.
não; caso contrário.

Modifique a máquina de Turing universal apresentada no slide 25 da Aula 04 para realizar a verificação do problema acima. Apresente a máquina como um algoritmo, como feito no slide.

A princípio pode parecer que o problema requer que a máquina M seja testada com todas as possíveis *strings* de entrada $w \in \Sigma^*$. No entanto, não é este o caso, pois as primeiras 10 transições de qualquer computação podem ler no máximo os 10 primeiros símbolos da *string* de entrada. Assim, só é necessário analisar a computação de M para todas as *strings* de tamanho 10 ou menos. Se uma computação sobre esse conjunto finito de *strings* de tamanho 10 ou menos precisa de mais de 10 transições para terminar, então a resposta para o problema é 'sim'. Se as computações de M terminam com 10 transições ou menos para todas essas *strings*, a resposta é 'não'. (Note que qualquer outra *string* $w \in \Sigma^*$ que tem comprimento maior que 10 possui uma das *strings* testadas como prefixo. Se todos esses prefixos fazem a máquina parar, qualquer parte da *string* além do prefixo é ignorada.)

Uma modificação da máquina universal pode ser feita para esse problema. Duas novas fitas são adicionadas à máquina universal: uma (fita 4) é usada para a geração das *strings* de Σ^* com comprimento 10 ou menos; e a outra (fita 5) é usada para contar o número de transições na simulação da computação de M. A entrada da máquina universal é a representação de M.

A computação consiste do seguinte ciclo:

1. A *string* nula é escrita na fita 4.
2. A *string* 1^{10} é escrita na fita 5.
3. A *string* na fita 4 é copiada para a fita 3 e a codificação do estado q_0 é escrita na fita 2.
4. A computação de M com a *string* de entrada da fita 4 é simulada nas fitas 2 e 3. A cada transição simulada, um 1 é apagado da fita 5.
5. Se a simulação de M especifica uma transição e a fita 5 está vazia, a computação para e aceita.
6. Se a simulação de M para e um 1 está sendo lido na fita 5, então:
   - As fitas 2, 3 e 5 são apagadas;
   - A *string* 1^{10} é escrita na fita 5; e
   - A próxima *string* é gerada na fita 4.
7. Se a *string* na fita 4 tem comprimento maior do que 10, a computação para e rejeita a entrada. Caso contrário, a computação continua com o passo 3.