Algoritmos e Fundamentos da Teoria de Computação

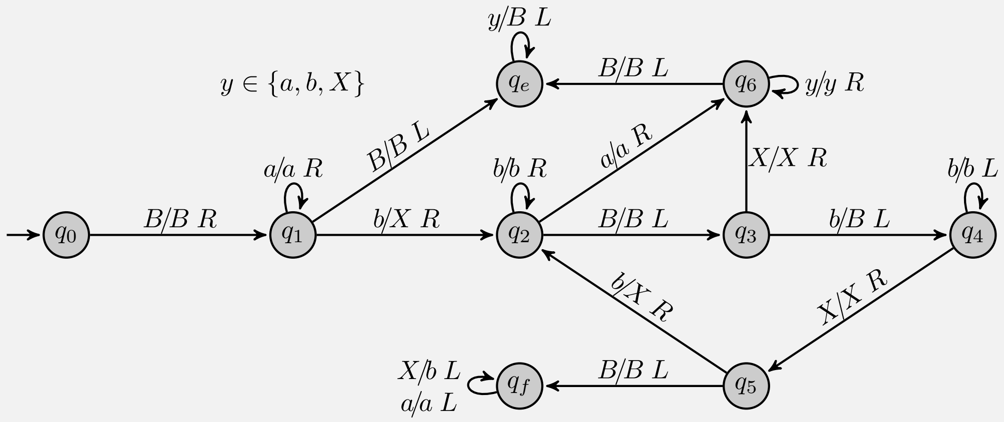
Lista de Exercícios 06

1 Construa uma DTM R que reduz a linguagem $\mathsf{L} = \{a^i(bb)^i \mid i > 0\}$ para a linguagem $\mathsf{Q} = \{a^ib^i \mid i > 0\}$ em tempo polinomial. Usando a notação de Big Oh, apresente a complexidade de tempo de R.

A ideia geral da redução computada por R é eliminar um segundo b do final da *string* sempre que um b for lido no início. Vale lembrar que a redução precisa mapear corretamente as instâncias $w \in \mathsf{L}$ em $r(w) \in \mathsf{Q}$, e também as instâncias $w \notin \mathsf{L}$ em $r(w) \notin \mathsf{Q}$. A tabela abaixo mostra a redução para alguns exemplos de entrada.

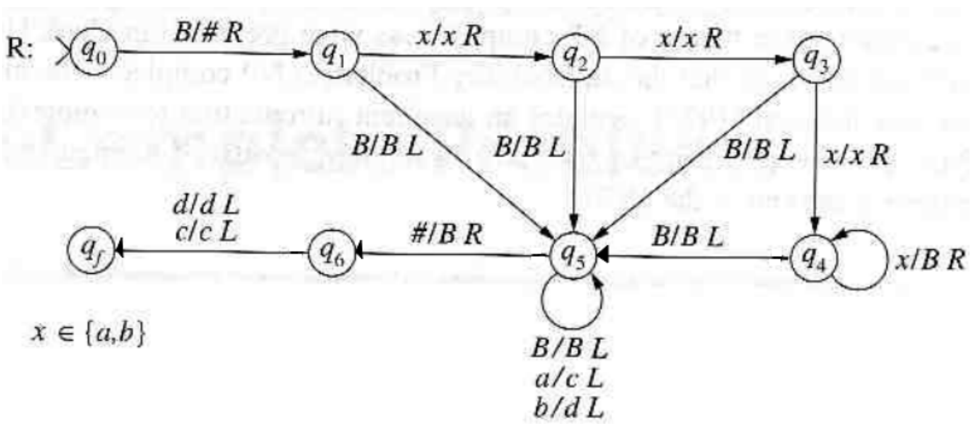
w	$w \in \mathsf{L}$?	$r(w)$	$r(w) \in \mathsf{Q}$?
$aabbbb$	sim	$aabb$	sim
$abbbb$	não	abb	não
$aabbb$	não	λ	não
aa	não	λ	não
$bbbb$	não	bb	não
bbb	não	λ	não
$abab$	não	λ	não

O diagrama da DTM R é mostrado abaixo. Eventuais a's do prefixo da entrada são lidos no estado q_1. Sempre que a entrada possui um número par de b's consecutivos, o *loop* entre os estados q_2, q_3, q_4 e q_5 elimina metade dos b's no final da *string*. A cada passo do *loop* o b mais à esquerda da sequência é marcado com um X e a máquina vai até o final para apagar um b. Se o *loop* termina corretamente, isto é, se há uma sequência de tamanho par de b's, todos os b marcados por X são desmarcados e a máquina termina lendo a posição 0 da fita no estado q_f. Quando a entrada possui uma sequência de tamanho ímpar de b's, ou com a's e b's alternados, a máquina avança até o final da *string* no estado q_6 e retorna apagando toda a fita, parando na posição 0.



O pior caso da computação de R ocorre quando a entrada é da forma b^{2i}, para $i > 0$. Nesse caso, a máquina faz um zig-zag sobre a entrada, marcando um b no início e apagando um b no final. Fazendo uma analogia com a DTM padrão que aceita palíndromos (Aula 05, slide 25), que possui o mesmo comportamento, vemos que a complexidade assintótica de R está em $O(n^2)$.

2 A máquina R abaixo computa uma função de $\{a, b\}^*$ para $\{c, d\}^*$.



a. Faça o *trace* da computação de R para a entrada $abba$.

b. Descreva a *string* de tamanho n para a qual a computação de R requer o número máximo de transições.

c. Apresente a função tc_{R}.

d. A máquina R reduz a linguagem $\mathsf{L} = \boldsymbol{abb(a \cup b)^*}$ para a linguagem $\mathsf{Q} = \boldsymbol{(c \cup d)cdd^*}$? Se sim, prove que a função computada por R é uma redução. Se não, apresente uma *string* que demonstra que o mapeamento da função não é uma redução entre essas duas linguagens.

a. O *trace* pedido é apresentado abaixo.

$q_0 BabbaB$	$\vdash \#abq_5bBB$
$\vdash \#q_1abbaB$	$\vdash \#aq_5bdBB$
$\vdash \#aq_2bbaB$	$\vdash \#q_5addBB$
$\vdash \#abq_3baB$	$\vdash q_5\#cddBB$
$\vdash \#abbq_4aB$	$\vdash Bq_6cddBB$
$\vdash \#abbBq_4B$	$\vdash q_fBcddBB$
$\vdash \#abbq_5BB$	

b. Qualquer *string* de entrada causa o máximo número de transições. A computação lê a entrada toda da esquerda para a direita, e a seguir novamente da direita para a esquerda, terminando com duas transições para apagar o marcador da posição 0.

c. A função de complexidade de tempo é $tc_{\mathsf{R}}(n) = 2n + 4$.

d. A função computada por R não é uma redução de L para Q. A *string* ab, que não pertence a L, é mapeada para cd, que pertence a Q. Isso viola a condição fundamental de uma redução: manter as mesmas respostas através das instâncias mapeadas.

3 Para resolver ambos os itens abaixo, assuma que $\mathcal{P} = \mathcal{NP}$.

a. Seja L uma linguagem em $\mathcal{NP}$ com $\mathsf{L} \neq \emptyset$ e $\overline{\mathsf{L}} \neq \emptyset$. Isto é, tanto L quando o seu complemento $\overline{\mathsf{L}}$ são linguagens não-vazias e o problema de decidir se uma *string* pertence a L está na classe $\mathcal{NP}$. Prove que esse problema de decisão para L é NP-completo.

b. Por que $\mathcal{NPC}$ (a classe de complexidade que contém os problemas NP-completos) é um subconjunto próprio de $\mathcal{NP}$?

A suposição de que $\mathcal{P} = \mathcal{NP}$ é essencial para os argumentos que justificam os itens abaixo. Claro que não se sabe realmente se tal suposição é verdadeira.

a. Seja L uma linguagem em $\mathcal{NP}$ onde tanto L quando o seu complemento $\bar{\mathsf{L}}$ não são vazias. Isto quer dizer que existem *strings* u e v tal que $u \in \mathsf{L}$ e $v \in \bar{\mathsf{L}}$.

   Para provar que L é NP-complete, devemos mostrar que qualquer linguagem Q em $\mathcal{NP}$ é redutível a L em tempo polinomial. Uma vez que $\mathcal{P} = \mathcal{NP}$, sabemos que existe uma Máquina de Turing M que aceita Q em tempo polinomial. Uma redução em tempo polinomial de Q para L pode ser obtida como a seguir. Para uma *string* w:

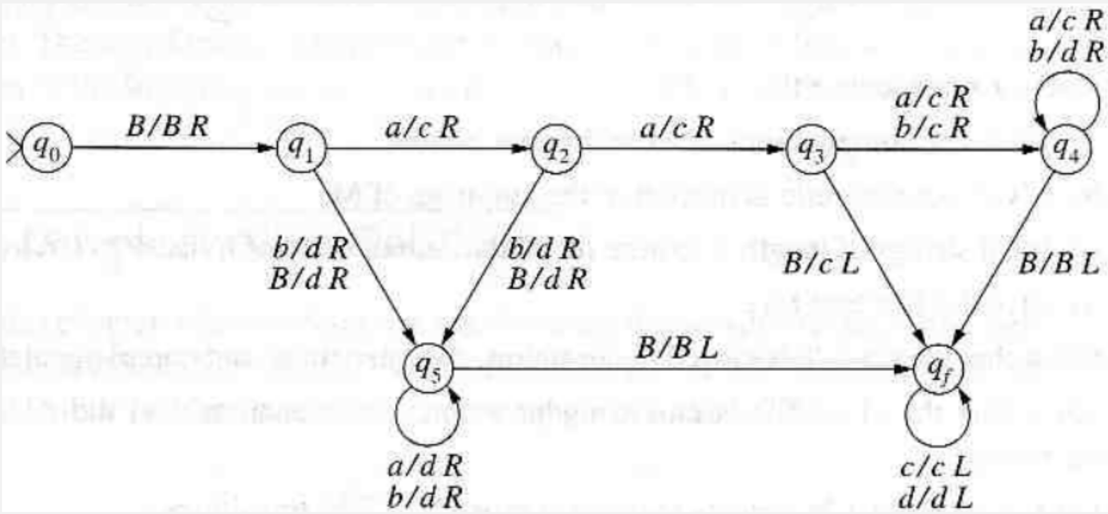
   1. Execute M para determinar se $w \in Q$.
   2. Se $w \in Q$, então apague a fita e escreva u na posição de entrada. Senão, apague a fita e escreva v na posição de entrada.

   Pode-se ver de imediato que o algoritmo acima é uma redução de Q para L. Além disso, tal redução pode ser realizada em tempo polinomial uma vez que a computação de Q é polinomial.

b. No item a) foi provado que qualquer linguagem em $\mathcal{NP}$ exceto i) $\emptyset$, e ii) o seu complemento são NP-complete. As linguagens i) e ii) claramente estão em $\mathcal{P}$ ($= \mathcal{NP}$), mas não são NP-complete. Tal fato se deve à impossibilidade de se reduzir uma linguagem não-vazia Q para $\emptyset$, uma vez que em tal redução seria necessário mapear *strings* em Q para uma *string* em $\emptyset$. Como não há *strings* em $\emptyset$ tal mapeamento é impossível. Uma vez que i) e ii) estão em $\mathcal{NP}$ mas não estão em $\mathcal{NPC}$, temos que $\mathcal{NPC}$ é um subconjunto próprio de $\mathcal{NP}$.

4 Construa uma DTM R que reduz a linguagem $\mathsf{L} = \boldsymbol{aa(a \cup b)^*}$ para a linguagem $\mathsf{Q} = \boldsymbol{ccc(c \cup d)^*}$ em tempo polinomial. Usando a notação de Big Oh, apresente a complexidade de tempo assintótica para R.

A máquina abaixo realiza a redução pedida. É simples de perceber que a máquina percorre a *string* na fita duas vezes. Assim, temos que a complexidade assintótica é **linear** em relação ao tamanho da entrada, isto é, pertence a $O(n)$.



5 Três alunos (A, B e C) tentaram encontrar a classe de complexidade de um problema de decisão X. O aluno A construiu uma DTM M_A que decide X com complexidade assintótica de tempo $O(n!)$ e concluiu que $X \notin \mathcal{P}$. O aluno B construiu uma outra DTM M_B (diferente de M_A) com complexidade assintótica de tempo $O(2^n)$ e concluiu que $X \in \mathcal{NP}$. Por fim, o aluno C construiu uma redução polinomial (determinística) de SAT para X e concluiu que $X \in \mathcal{NPC}$.

Assumindo que as máquinas e a redução construídas estão corretas, bem como a determinação da complexidade assintótica, explique por que as conclusões dos três alunos estão *erradas*, **justificando adequadamente todas as respostas**. A seguir, explique o quê pode ser concluído sobre a complexidade de X a partir das informações acima.

A conclusão do aluno A está errada porque não é possível afirmar que $X \notin \mathcal{P}$ a partir da máquina M_A. A única coisa que se pode afirmar nesse caso é que M_A é uma solução ineficiente para X, mas isso não quer dizer que não é possível construir uma outra máquina que decida X de forma eficiente (tempo polinomial determinístico).

A conclusão do aluno B está errada porque a classe $\mathcal{NP}$ é formada pelos problemas que possuem solução por **NTM** em tempo polinomial. A complexidade de M_B não implica nessa condição da classe $\mathcal{NP}$.

A conclusão do aluno C está errada porque ele não provou que $X \in \mathcal{NP}$, logo, ele não poderia afirmar que $X \in \mathcal{NPC}$. A redução que ele fez somente permite afirmar que X é **NP-hard**. Essa é a única conclusão que podemos tirar a partir das informações fornecidas.