Universidade Federal de Santa Catarina
Departamento de Informática e Estatística (INE)
Curso: Ciência da Computação
Disciplina: Teoria da Computação (INE5415 - 04208A)
Professor: Jerusa Marchi
Estudantes: Guilherme Adenilson de Jesus (22100620)
Vicente Cardoso dos Santos (22102203)

Relatório T2 - Análise de Complexidade

- 1a) $L = \{a^i b^j c^k \mid i, j, k \in N \text{ e } i \times j = k\}$

O funcionamento da máquina para a linguagem acima ocorre da seguinte forma:

1. Lê um ‘a’ na fita e marca com ‘X’, movendo o cabeçote para direita;

   Custo 1, pois tal operação altera apenas uma entrada na fita e, paralelamente, realiza apenas um movimento.

2. Percorre a fita até encontrar um ‘b’, marcando com ‘Y’, movendo o cabeçote para direita;

   Custo n, pois o número de movimento aumenta linearmente à medida em que se aumenta o número de itens “a” e “Y” na entrada, pois a cabeça da fita será deslocada “a” posições + “Y” posições até encontrar um valor de “b”.

3. Percorre a fita até encontrar um ‘c’, marcando com ‘Z’, movendo o cabeçote para esquerda;

   Custo n, pois o número de movimento aumenta linearmente à medida em que se aumenta o número de itens “b” e “Z” na entrada, pois a cabeça da fita será deslocada “b” posições + “Z” posições até encontrar um valor de “c”.

4. Volta na fita até encontrar um ‘Y’, movendo o cabeçote para direita;

   Custo n, pois a quantidade de movimentos aumenta linearmente à medida em que se aumenta o número de itens “Z” e “b” na entrada, pois a cabeça da fita será deslocada “Z” posições + “b” posições até encontrar um valor de “b”.

5. Repete os passos 2 a 4 até não haver mais ‘b’ para serem marcados;

   Custo n^2, pois o número de movimentos é diretamente dependente da quantidade de itens “b” na entrada da fita pois essa operação será realizada “b” vezes até ser concluída. Como os passos 2 a 4 têm um custo de n + n + n = 3n e a fita realizará tais passos “b” vezes, o passo 5 terá custo = 3*n*n = $3*n^2$

6. Quando não houver mais ‘b’ (ou seja, encontrou um ‘Z’), volta na fita, marcando os ‘Y’ que encontrar com ‘b’, até alcançar um ‘X’, movendo o cabeçote para direita;

   Custo n, pois o número de movimentos aumenta linearmente à medida em que se aumentam os elementos “Y” na fita, pois a fita se deslocará “Y” posições, marcando cada posição com “b”.

7. Repete os passos 1 a 6 até não haver mais ‘a’ para serem marcados;

   Custo ($n + 3*n^2 + n^3$) pois o número de movimentos é diretamente dependente da quantidade de itens “a” na entrada da fita pois essa operação será realizada “a” vezes até ser concluída. Como os passos 1 a 6 têm um custo de $1 + n + n + n + n^2 + n = 1 + 3*n + n^2$ e a fita realizará tais passos “a” vezes, o passo 7 terá custo = $n + 3*n^2 + n^3$

8. Quando não houver mais ‘a’ (ou seja, encontrou um ‘b’), percorre toda a fita verificando se faltou algum c para ser marcado;

   Custo n, pois o algoritmo percorre toda a fita para realizar tal operação.

9. Se chegou no final da fita, aceita, caso contrário, rejeita.

   Custo 1, pois precisa realizar apenas um movimento na fita, independente do tamanho da fita.

   Análise de custo total: Custo será calculado pela soma das partes 7, 8 e 9, pois os outros passos estão contidos dentro do loop presente no sétimo passo.

$$(n+3n^2 + n^3) + n + 1 =$$
$$\mathbf{n^3 + 3n^2 + 2n + 1}$$

Complexidade: $O(n^3)$

- 1b) L = $\{0^{2^{\wedge}n} \mid n \geq 0\}$

Para a presente linguagem, o algoritmo utilizado para sua máquina é descrito abaixo:

1. Lê um ‘0’ na fita e marca com ‘A’, movendo o cabeçote para direita;

   Custo 1, pois realiza somente um movimento de fita, marcando apenas um “0” como “A”.

2. Percorre toda a fita até achar um branco (ou seja, chegando no fim da mesma) ou um ‘X’, e move o cabeçote para a esquerda;

   Custo n, pois como percorre toda a fita, o número de operações de fita é linear ao tamanho da fita.

3. Lê um ‘0’ marcando com ‘X’, movendo o cabeçote para a esquerda;

   Custo 1, pois realiza apenas uma operação independentemente do tamanho da fita.

4. Volta na fita até encontrar um ‘A’, movendo o cabeçote para direita;

   Custo n, pois o número de movimentos necessários aumenta linearmente à medida em que se aumentam o número de itens “A” e “0” na entrada, pois a cabeça da fita será deslocada “X” posições + “0” posições até encontrar um valor de “A”.

5. Repete os passos 1 a 4 até não houver mais 0 a ser lidos (ou seja, no passo 1 será lido um ‘X’ invés de ‘0’);

   Custo $2*n + 2*n^2$, pois o número de movimentos é diretamente dependente da quantidade de itens “0” restantes na entrada da fita, pois essa operação será realizada “0” vezes até ser concluída. Como os passos 1 a 4 têm um custo de 1 + n + 1 + n = 2 + 2*n e a fita realizará tais passos “b” vezes, o passo 5 terá um custo de $n*(2 + 2*n) = 2*n + 2*n^2$.

6. Marque o ‘X’ lido no passo 5 com branco e volta no restante da fita (ou seja, até atingir um branco na esquerda) marcando com ‘0’ cada ‘A’ lido no caminho;

   Custo n, pois, como será necessário voltar até o final da fita, o número de movimentos é diretamente dependente do tamanho de entrada da fita.

7. Achando o branco, move o cabeçote para direita e repete os passos 1 a 6 até haver apenas um ‘0’ que pode ser lido (ou seja, ‘0,branco,...’), aceitando a palavra.
   A fita é reduzida pela metade a cada iteração, de forma que serão necessárias log2(n) iterações de loop. Analisando o custo de 1 a 6, tem-se: $1 + n + 1 + n + 2*n + 2*n^2 + n = log2(n)*(2*n^2 + 5*n)$.

   Análise de custo total: Custo será referente ao passo 7, pois os outros passos estão contidos dentro do loop presente no sétimo passo.
   .

$$\log2(n)*(2n^2 + 5n) =$$
$$\mathbf{2*log2(n)*n^2 + log2(n)*(5n)}$$

   Complexidade: $O(2*\log2(n)*n^2) = \mathbf{O(log(n)*n^2)}$

- 2a) $L = \{xyx^Ry^R \mid x, y \in \{0, 1\}^*\}$

A presente linguagem descrita acima foi projetada numa máquina de Turing com 2 Fitas: uma de entrada que posteriormente guardará a sequência xy e outra que armazenará x^Ry^R. O algoritmo referente pode ser visto a seguir:

1. A cada ‘0’ ou ‘1’ lido na Fita 1, marca com ‘X’ ou ‘Y’, respectivamente, movendo o cabeçote para direita;

   Custo 1, pois altera apenas um elemento e move para a direita, realizando apenas um movimento.

2. Percorre toda a Fita 1 e transfere o último elemento para Fita 2 e move o cabeçote de ambas para esquerda;

   Custo n, pois o número de movimentos aumenta à medida em que se aumenta o tamanho da Fita 1.

3. Volta na Fita 1 até achar um ‘X’ ou ‘Y’ e move o cabeçote para direita;

   Custo n, pois precisará deslocar n - X - Y posições até encontrar um valor de X ou Y.

4. Repete os passos 1 a 3 até que não haja mais ‘0’ ou ‘1’ para serem lidos na Fita 1;

Custo $2*n^2 + n$, pois o número de repetições de loop aumenta linearmente à medida em que se aumenta a quantidade de elementos "0" e "1" na entrada, de forma que cada iteração do loop terá um custo 1 + n + n. Multiplicado pelo custo do loop, tem-se $n*(1 + n + n) = 2*n^2 + n$

5. Na Fita 1, move totalmente para esquerda enquanto marca os 'X' e 'Y' no caminho com '0' e '1', respectivamente;

   Custo n/2, pois precisará se deslocar durante toda a fita, que teve seu tamanho reduzido pela metade

6. Na Fita 2, move totalmente para direita;

   Custo n/2, pois percorrerá toda a fita linearmente para realizar esta etapa.

7. Agora, fica parado na Fita 1 e move para esquerda na Fita 2 até os cabeçotes estarem apontando para o mesmo carácter, marcando-as com 'X' (para '0') ou 'Y' (para '1');

   Custo n/2, pois no pior caso, terá que se mover até o final da Fita 2 para encontrar o mesmo carácter.

8. Repete o passo 7, só que agora movendo a Fita 1 para direita, até que a Fita 2 chegue em um 'branco';

   Custo n/2, pois terá que se mover até o extremo esquerdo da Fita 2 para encontrar o branco.

9. Caso em algum momento do passo 8 ocorra dos cabeçotes apontarem para letras diferentes, volta em ambas as Fitas desmarcando os 'X' e 'Y', e continue a partir da letra à esquerda da primeira marcação na Fita 2;

   Custo (n/2 - 1)*(n/2), pois no pior caso acharia um carácter diferente até chegar na última posição que daria certo. 'n/2 - 1' referente à quantidade de vezes que faria a desmarcação dos 'X' e 'Y' e 'n/2' referente ao custo do passo 8.

10. Caso chegue em um 'branco' no passo 8, move a Fita 1 totalmente para direita;

    Custo n/2, pois a quantidade de movimentos realizados por este passo é linear à metade do tamanho da Fita 1.

11. Move a Fita 2 para direita até chegar em um ‘0’ ou ‘1’ (ou seja, o que ficou desmarcado, supostamente y^R);

    Custo n/2 - 1, visto que, no pior caso, será necessário mover por quase toda a Fita 2, mas faltando apenas um caracter que seria y^R.

12. Mova para esquerda na Fita 1 e para direita na Fita 2 enquanto estiverem apontando para mesma letra;

    Custo n/2 - 1, visto que esse passo será realizado até que encontre o momento de parada. No pior caso, $|x| = |x^R| = 1$, o que indicaria que $|y| = |y^R| = n/2 - 1$, que são a quantidade de vezes que realizaria o movimento.

13. Caso a Fita 2 atinja um ‘branco’ e a Fita 1 um ‘X’ ou ‘Y’, aceite. Se em algum momento no passo 11 apontarem para letras diferentes, rejeite.

    Custo 1, pois é apenas uma verificação final de custo constante.

    Análise de custo total: Visto que os passos 1 a 3 estão contidos no loop do passo 4, eles não farão parte da soma separadamente.

$$2n^2 + n + n/2 + n/2 + n/2 + (n/2 - 1)(n/2) + n/2 + n/2 - 1 + n/2 - 1 + 1 =$$
$$\mathbf{9n^2/4 + 7n/2 - 1}$$

    Complexidade: $O(9n^2/4) = \mathbf{O(n^2)}$

- 2b) $L = \{a^i b^j c^k \mid i, j, k \in N \text{ e } i^j = k\}$

O algoritmo em alto nível da máquina de Turing para a referente linguagem funciona da seguinte forma:

1. Inicialmente passa por todos os ‘a’ da Fita 1 até chegar ao primeiro ‘b’ e marcando-o com ‘Y’;

   Custo n, pois o tamanho da entrada será diretamente relacionado, de forma linear, com a quantidade de itens “a” na fita de entrada.

2. Volta na Fita 1 copiando todos os ‘a’ para a Fita 2;

Custo n, pois o tamanho da entrada será diretamente relacionado, de forma linear, com a quantidade de itens "a" na fita de entrada, pois terá que voltar na fita à medida em que encontra símbolos "a".

3. Percorre a Fita 1 até encontrar o próximo 'b', marcando-o com Y; O(n)

   Custo n, pois o tamanho da entrada será diretamente relacionado, de forma linear, com a quantidade de itens na fita de entrada, pois terá que avançar por todos os a's e Y's da fita até encontrar um b.

4. Volta na Fita 1 até achar um 'a';

   Custo n, pois o tamanho da entrada será diretamente relacionado, de forma linear, com a quantidade de itens na fita de entrada, pois terá que retroceder por todos os Y's da fita até encontrar um 'a'.

5. Para cada 'a' na Fita 1, copia a Fita 2 para Fita 3;

   Custo n, pois o número de operações é determinado pelo tamanho de itens "a" na fita 1, de forma que, à medida em que se aumenta o número de elementos "a" na fita 1, mais elementos serão, linearmente, copiados para a Fita 3. O valor da Fita 2 se mantém constante durante esse passo, de forma que, para cada "a" lido na Fita 1, serão realizados "k" passos, constantes, para copiar os valores para a Fita 3.

6. Concluiu de copiar, volta na Fita 2 para a posição inicial para realizar a próxima cópia;

   Custo n, pois terá que percorrer toda a fita 2, linearmente, até alcançar seu final.

7. Se chegou num 'branco' na Fita 1, move o valor da Fita 3 para Fita 2 (apaga o conteúdo da Fita 3 enquanto escreve na Fita 2);

   Custo n, pois terá que copiar o tamanho da Fita 3 aumenta linearmente à medida em que se aumenta o número de c's na Fita 1. Como move linearmente durante a Fita 3, o custo será "n".

8. Repete os passos 3 a 7 até marcar todos os 'b' na Fita 1 com Y;

   Custo $5*n^2$, visto que tal loop será repetido "b" vezes até ser concluído, de forma que o custo será dado por (n + n + n + n + n)*n = 5*n^2, pois apenas a parte externa

do loop tem custo "n", pois o número de itens b aumenta linearmente a medida em que se aumenta o tamanho da fita.

9. Marcando todos os 'b', vê se a próxima posição na Fita 1 é um c;

   Custo 1, pois tal operação altera apenas uma entrada na fita e, paralelamente, realiza apenas um movimento.

10. Para cada 'c' lido na Fita 1 e 'a' lido na Fita 2, move para direita em ambas;

    Custo n, pois o número de movimentos necessários para completar este passo é diretamente linear ao tamanho da Fita 1.

11. Se a Fita 1 e 2 chegaram num branco ao mesmo tempo, então aceite. Caso contrário, rejeite.

    Custo 1, pois tal operação altera apenas uma entrada na fita e, paralelamente, realiza apenas um movimento.

    Análise de custo total: Como os passos 3 a 7 estão contidos dentro do loop do passo 8, eles não serão considerados separadamente na soma de custos.

$$n + n + + 5n^2 + 1 + n =$$
$$\mathbf{5n^2 + 3n + 1}$$

    Complexidade: $O(5n^2) = \mathbf{O(n^2)}$

- 3a) $L = \{I^n \mid n \in N \text{ e } \textit{fibonacci(n)} >= 0\}$

A dada linguagem possui uma máquina de Turing de 2 fitas, uma para o valor de n e resultado de fibonacci(n) e outra para computação da sequência de fibonacci de 0 a n, sendo explicado seu funcionamento abaixo:

1. Computa os valores de n = 1 e n = 2 na Fita 2 enquanto apaga dois '1' na Fita 1;

   Custo = 3, pois a Fita 2 precisa escrever "1□1", necessitando de três movimentos para realizar tal passo.

2. Para cada 'l' lido a seguir na Fita 1, vai até o f(n-2) na Fita 2;

Custo f(n-2) + f(n-1) + 2. Considerando a parte relevante da Fita 2 como "f(n-2)□f(n-1)□", o cabeçote sempre está no último "□". Dessa forma, será deslocado na Fita 2 por 2 "□" e por f(n-2) + f(n-1) '1'.

3. Para cada '1' lido da Fita 2, marca com '0' e coloca '1' no fim da Fita 2;

   Custo f(n-2) + f(n-1) + 2. Considerando a parte relevante da Fita 2 como "f(n-2)□f(n-1)□". A cada '1' lido, será necessário deslocar na Fita 2 por 2 "□" e por f(n-2) + f(n-1) '1' até chegar no fim da fita para escrever.

4. Volta na Fita 2 até chegar no '0', marcando-o com '1' e movendo o cabeçote para direita;

   Custo f(n-2) + f(n-1) + 2. Considerando a parte relevante da Fita 2 como "0f(n-2)□f(n-1)□1", o cabeçote sempre está no último "1". Dessa forma, é preciso deslocar por 2 "□" e por f(n-2) + f(n-1) '1' até voltar ao '0'.

5. Repete os passos 3 e 4 até concluir f(n-2);

   Custo f(n-2)*2*(f(n-2) + f(n-1) + 2), pois é repetido para cada '1' referente ao fibonacci de n-2.

6. Concluído para f(n-2), repete os passos 3 a 5 para f(n-1);

   Custo f(n-1)*2*(f(n-2) + f(n-1) + 1), pois a única coisa que mudou foi o formato relevante da Fita 2, que agora seria "f(n-1)□f(n-2)", tendo apenas um "□" a menos. Agora ele repetirá fibonacci de n-1 vezes.

7. Repete os passos 2 a 6 até não restar mais '1' na Fita 1;

   Custo (n-2)*(f(n-2)*2*(f(n-2) + f(n-1) + 2) + f(n-1)*2*(f(n-2) + f(n-1) + 1)), pois o loop será executado n-2 vezes, de forma linear à entrada da fita, visto que os valores de n=1 e n=2 já são pré-processados.

8. Não restando mais '1' na Fita 1, copia o último valor de Fibonacci da Fita 2 para Fita 1, concluindo a computação.

   Custo f(n), pois o número de operações será equivalente ao valor do enésimo termo calculado pela entrada.

Análise de custo total: Custo será calculado pela soma das partes 1, 7 e 8, pois os outros passos estão contidos dentro do loop presente no sétimo passo.

$$3 + (n-2)*(f(n-2)*2*(f(n-2) + f(n-1) + 2) + f(n-1)*2*(f(n-2) + f(n-1) + 1)) + f(n) =$$
$$3 + (n-2)*(f(n-2)*2*(f(n) + 2) + f(n-1)*2*(f(n) + 1)) + f(n) =$$
$$3 + (n-2)*(2f(n)*(f(n-2)+f(n-1)) + 2f(n-2) + f(n-1)) + f(n) =$$
$$3 + (n-2)*(2f(n)*f(n) + f(n-2) + f(n)) + f(n) =$$
$$3 + (n-2)*(2f(n)^2 + f(n-2) + f(n)) + f(n) =$$
$$\mathbf{3 + 2*(n-2)*f(n)^2 + (n-2)*f(n-2) + (n-1)*f(n)}$$

Complexidade: $O(n*2^n)$. Ao analisar o custo total, pode-se extrair que o valor dominante é $2*(n-2)*f(n)^2$. Ao considerar que o cálculo computacional de fibonacci pode ser realizado com um custo na mesma ordem de grandeza que Φ^n, onde Φ (1.618..) é a proporção áurea, a complexidade pode ser obtida ao aproximar o custo da função para $2*(n-2)*(\Phi^n)^2 = 2*(n-2)*(\Phi^{2n})$, ao substituir f(n) por Φ^n, sem alterar sua grandeza. Pode-se também, aproximar Φ para 2 sem alterar a complexidade da função e, ademais, substituir $(n-2)(2^{2n})$ para $n*(2^{2n})$ e subsequentemente arredondar tal valor para $n*2^n$. Assim, tem-se que a complexidade pode ser expressa pelo custo de $2*n*(2^n)$, em que $O(2*n*(2^n)) = O(n*2^n)$

- 3b) $L = \{m^i n^j \mid i, j \in N \text{ e } MDC(i, j) >= 1\}$

A máquina de Turing que computa a presente linguagem possui duas fitas: uma para a entrada com sequência de ‘m’ e ‘n’ (que após o pré-processamento, armazenará o valor de m e terá o resultado final do MDC) e outra que terá o valor de n atual. Seu algoritmo está expresso a seguir:

1. Passa por todos os ‘m’ na Fita 1 até chegar num ‘n’;

   Custo n, pois o número de passos necessários aumenta linearmente à medida em que se aumentam os itens “m” na fita.

2. Transfere todos os ‘n’ encontrados na Fita 1 para Fita 2 (apaga da Fita 1 e escreve na Fita 2);

   Custo n, pois o número de passos necessários aumenta linearmente à medida em que se aumentam os itens “n” na fita.

3. Volta em ambas as Fitas até chegarem no primeiro branco antes da sequência de ‘m’ (na Fita 1) e ‘n’ (na Fita 2); O(n)

Custo n+1, pois, considerando que a Fita 1 era originalmente composta por "m" + "n" itens, e que esse passo realiza "m" movimentos na Fita 1 + "n" movimentos na Fita 2 + 1 movimento para a maior fita, então tem-se que o custo é n (tamanho da fita, diferente do símbolo "n") + 1.

4. Verifica se o quantidade de 'm' e 'n' são iguais passando por todos elementos delas (indo para direita) até chegar no fim das duas Fitas;

   Custo n, pois percorrerá min(m, n) posições até encontrar o fim das fitas, considerando que o tamanho da fita (n) aumenta à medida em que se aumentam os símbolos "m" e "n", então o custo é linear em relação ao tamanho da fita.

5. Se não for igual, uma das Fitas continuará indo pro final enquanto a outra espera; O(n)

   Custo n, pois percorrerá |"n" (símbolo) - "m"| posições até encontrar o fim das fitas, considerando que o tamanho da fita de entrada aumenta à medida em que se aumentam os símbolos "m" e "n", então o custo é linear em relação ao tamanho da fita.

6. Quando ambas chegarem no fim, a Fita com maior quantidade de elementos terá x valores apagados, sendo x a quantidade de elementos da outra Fita; O(n)

   Custo n, pois percorrerá min(m, n) posições até apagar os todos os elementos de uma fita em outra.

7. Concluindo a remoção, reseta as posições em ambas as Fitas para o primeiro elemento mais à esquerda;

   Custo n, pois percorrerá max(m, n) posições até as duas fitas chegarem ao fim, considerando que o tamanho da fita (n) aumenta à medida em que se aumentam os símbolos "m" e "n", então o custo é linear em relação ao tamanho da fita.

8. Repete os passos 5 a 7 até o passo 4 for verdadeiro (quantidade de 'm' e 'n' forem iguais);

   Custo $3n^2 - 3n$. No pior caso, em que o menor dentre os dois valores for 1, à medida em que se aumenta o outro valor, terá que ser executado tal loop um total de (n-1)

vezes. Paralelamente, considerando os custos de 5 a 7 = n + n + n, então o custo do passo é $3n*(n-1) = 3n^2 - 3n$

9. Sendo o passo 4 verdadeiro, apaga os valores da Fita 2 enquanto move ambas as Fitas para esquerda;

   Custo n/2 pois, no pior caso em que a quantidade de itens "m" é, no máximo, igual a "n" (carácter), deve-se apagar "n" valores da Fita 2. Assim, considerando que o número de itens "n" é a metade da entrada de fita, o custo será n/2 para apagar toda a Fita 2.

10. Quando ambas chegarem num 'branco', conclui-se a computação, tendo o resultado do MDC(n,m) na Fita 1.

    Custo 1, pois realiza a operação stay, verificando se ambas as fitas chegaram em um branco.

    Análise de custo total: Como os passos 5 a 7 estão dentro do loop do passo 8, eles não serão somados separadamente no custo total.

$$n + n + n + 1 + n + 3n^2 - 3n + n/2 + 1 =$$
$$\mathbf{3n^2 + 3n/2 + 1}$$

Complexidade: $O(3n^2) = \mathbf{O(n^2)}$