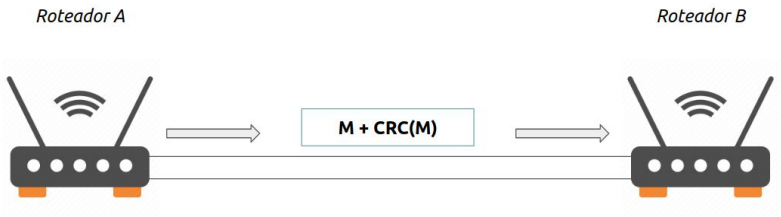
MC102W - Algoritmos e Programação de Computadores

Lab05: Checagem de Redundância

Prazo: 26 de Abril de 2020. **Peso da Atividade:** 2

Cyclic Redundancy Check (CRC) é um código criado na década de 60 com o intuito de detectar erros em uma transmissão de dados. Esse código é amplamente utilizado atualmente, como, por exemplo, na Internet, no GSM, e no Rádio. O CRC é gerado a partir do conteúdo da mensagem enviada. O código é enviado de um roteador A para um roteador B, por exemplo, juntamente com o conteúdo da mensagem. Quando a mensagem chega no Roteador B, o CRC é gerado novamente e comparado com o enviado. Se o código gerado em B não for igual ao calculado em A, sabemos que a mensagem não está correta.



O código CRC é o resto da divisão polinomial do conteúdo da mensagem. Para calcular o CRC é preciso fazer sucessivas operações de deslocamento e operações de XOR (ambas explicadas abaixo) entre a mensagem e o polinômio. **Ao utilizar um polinômio divisor de grau N obtemos um CRC de N-1 bits**.

Um polinômio de grau N com coeficientes binários é representado por uma sequência de N bits, indicando os seus coeficientes do maior grau para o menor grau. Exemplos de polinômio com coeficientes binários e sua representação em bits:

Grau	Polinômio	Representação Binária	Observação
0	1	**1**	-
1	$x + 1$	**11**	-
2	$x^2 + 1$	**101**	*Nesse caso o coeficiente de x é zero*
3	$x^3 + x + 1$	**1011**	*Nesse caso o coeficiente de x^2 é zero*

Dicas

1) **Tabela da verdade do XOR:**

A	B	A XOR B
1	1	**0**
1	0	**1**
0	1	**1**
0	0	**0**

2) **Exemplos de XOR bit a bit com A e B com 3 bits:**

A	B	A XOR B
100	001	**101**
111	000	**111**
101	111	**010**
010	111	**101**

3) **Tabela de exemplos de deslocamento para esquerda e para direita:**

Operando (em binário)	Quantidade de deslocamento (em decimal)	Esquerda <-	Direita ->
00000001	2	00000100	00000000
00000011	3	00011000	00000000
00001000	1	00010000	00000100
00100000	5	10000000000	000000001

4) **Transformar a entrada binária em lista**
>> mensagem = list(input())

5) **Saber o tamanho de uma lista**
>> tamanho_da_lista = len(lista)

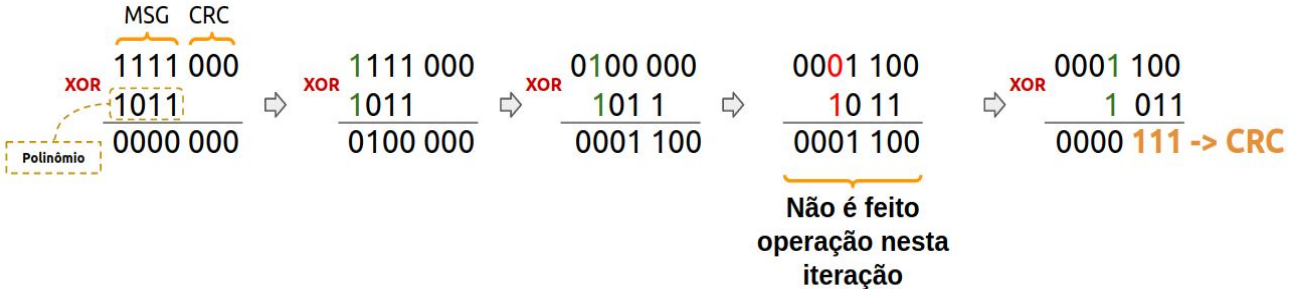
Algoritmo com Exemplo

Exemplo de geração de código para uma mensagem de 4 bits:
Mensagem: 1010 (em binário)
Polinômio: $x^3 + x + 1$ (equivalente a 1011)

- o polinômio é operado com o operador binário **XOR** a partir do bit mais significativo (ou seja, o que está mais à esquerda)

Algoritmo:

- Deslocar a mensagem N-1 vezes para a esquerda (sendo N o grau do polinômio).
- Depois, para cada bit **da mensagem**, começando do mais significativo (ou seja, o que está mais à esquerda) até o menos significativo (mais à direita), é feita uma análise. Caso o bit da mensagem for '1', faz-se um **XOR** entre o polinômio e a mensagem **a partir desse bit analisado** (note, no exemplo, que o bit está alinhado com o polinômio). O resultado obtido será utilizado na próxima iteração no papel da mensagem. Caso o bit seja '0', não se faz nenhuma operação.
- O algoritmo termina quando todos os bits **da mensagem** forem analisados.
- Feito isso o código estará nos **N-1 bits menos significativos** (da direita para esquerda).



Tarefa

Fazer um programa para gerar o código CRC de N-1 bits (sendo N a quantidade bits do polinômio) para uma mensagem de M bits. Ou seja, dada uma entrada composta por uma mensagem e um polinômio, o programa deve gerar o código CRC correspondente.

- Obs 1: As entradas serão binárias.
- Obs 2: Usar listas para manipulação de bits.

O programa deve ser em python e deve abranger os seguintes tópicos:

- ❑ Variáveis e tipos básicos;
- ❑ Expressões Lógicas e Relacionais
- ❑ Comandos Condicionais
- ❑ Listas;
- ❑ Comandos de Repetição;
- ❑ Entrada e saídas de dados.

Entrada

A entrada consiste em um número binário de M bits representando a mensagem e um polinômio de N bits.

Saída

O seu programa deve gerar como saída o valor em binário do CRC gerado (N-1 bits, sendo N o número de bits do polinômio).

Exemplos

A grafia da saída abaixo deve ser seguida rigorosamente por seu programa.
Obs .: A primeira linha é a mensagem e a segunda linha é o polinômio. Ambos em binário.

Exemplo dado anteriormente:

Entrada

```
1111
1011
```

Saída

```
111
```

Exemplo 1:

Entrada

```
1110
1100
```

Saída

```
100
```

Exemplo 2:

Entrada

```
1000
1011
```

Saída

```
101
```

Exemplo 3:

Entrada

```
10100
1010
```

Saída

```
000
```

Exemplo 3:

Entrada

```
1010
11010
```

Saída

```
0010
```

Critérios específicos

Os seguintes critérios específicos sobre o envio e implementação devem ser satisfeitos.

i. Submeter no *SuSy* os arquivos:
- ⇒ **`lab05.py`**: Arquivo onde deverá ser implementada a tarefa.

Observações gerais

No decorrer do semestre haverá 3 tipos de tarefas no *SuSy* (descritas logo abaixo). As tarefas possuirão os mesmos casos de testes abertos e fechados, no entanto o número de submissões permitidas e prazos são diferentes. As seguintes tarefas estão disponíveis no *SuSy*:

- ❏ **lab05-AmbienteDeTeste**: Esta tarefa serve para testar seu programa no *SuSy* antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém os arquivos submetidos aqui **não serão corrigidos**.
- ❏ **lab05-Entrega**: Esta tarefa tem limite de uma **única** submissão e serve para entregar a **versão final** dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa e submeta aqui apenas quando não for mais fazer alterações no seu programa.

- **lab05-ForaDoPrazo**: Esta tarefa tem limite de uma **única** submissão e serve para entregar a versão final fora prazo estabelecido para o laboratório. Esta tarefa irá substituir a nota obtida na tarefa **lab05-Entrega** apenas se o aluno tiver realizado as correções sugeridas no *feedback* ou caso não tenha enviado anteriormente na tarefa **lab05-Entrega**.

Avaliação

Este laboratório será avaliado da seguinte maneira: se o seu programa apresentar resposta correta para todos os casos de teste do *SuSy*, código de qualidade e comentários no código explicando o que está sendo feito, então é nota 10; caso contrário pontos serão descontados da nota final.

Testando seu programa

Para testar se a solução do seu programa está correta, basta seguir o exemplo abaixo.

```
python lab05.py < arq01.in > arq01.out
diff arq01.out arq01.res
```

O `arq01.in` é a entrada e `arq01.res` é a saída esperada, ambos disponíveis no *SuSy*. O `arq01.out` é a saída gerada pelo seu programa. Após o prazo, os casos de teste fechados serão liberados e podem ser baixados e testados da mesma forma que os testes abertos.