Universidade Federal do Pampa
Análise e Projeto de Algoritmos - Engenharia de Software
Trabalho 2
Entrega: 28 de novembro de 2019

Regras para a execução do trabalho:

1. No dia **28/11/2019** às **23:55** encerra-se o prazo para a entrega do trabalho prático. Trabalho atrasados não serão considerados.
2. O trabalho será realizado **individualmente**.
3. O aluno, até o prazo final de entrega, deverá fazer o envio **via moodle** de:
   (a) arquivos de código-fonte do programa (documentado) utilizado para analisar a complexidade;
   (b) instruções de como compilar e/ou executar a aplicação;
   (c) relatório sobre a atividade em formato de artigo SBC (até 8 páginas) contendo a descrição do algoritmo e sua análise de complexidade de tempo.

Sobre a avaliação do trabalho:

1. A nota será composta pela qualidade técnica da solução desenvolvida. A nota é condicionada a apresentação do trabalho.

Datas:

1. **Apresentação do enunciado do trabalho**: 30/10/2019
2. **Entrega do trabalho**: 28/11/2019 até às 23:55 (via moodle).
3. **Apresentação do trabalho**: 02 e 05 de dezembro, em aula.

Definição:

Você deve selecionar um problema abaixo e realizar o seguinte:

- Implementação do problema utilizando método de força bruta.
- Implementação do problema utilizando um dos paradigmas de programação apresentados em aula: divisão e conquista, método guloso ou programação dinâmica.
- Análise de complexidade de tempo das duas implementações.
- A implementação pode ser realizada em qualquer linguagem.

1 Problemas

1.1 Expressões Booleanas

João fica intrigado com o fato de o valor avaliado nas expressões booleanas poder variar dependendo da ordem da avaliação!

Por exemplo, considere o exemplo: *1 xor 1 and 0*

Podemos ter duas interpretações:

- $((1\ xor\ 1)\ and\ 0) =¿ (0\ and\ 0) =¿ 0$
- $(1\ xor\ (1\ and\ 0) =¿ (1\ xor\ 0) =¿ 1$

Agora, João está mais interessado em encontrar o número de possíveis parênteses diferentes, de modo que o resultado da computação seja **R**.

Input: A primeira linha do arquivo de entrada contém duas strings separadas por espaço. A primeira string indica os literais da expressão booleana ***S***, a segunda string indica os operadores. A próxima linha indica um número inteiro ***q***, ou seja, o número de *queries* a ser computadas. Cada uma das próximas ***q*** linhas contém dois inteiros separados por espaço l e r e uma sequência ***R***, que é *true* ou *false*.

Output: Para cada *query*, a saída consistira de uma nova linha representando o número de maneiras pelas quais a expressão booleana da substring $[l,r]$ pode existir para que o resultado seja ***R***.

Definições do tamanho do problema: $1 \leq |S| \leq 300$; $1 \leq q \leq 90000$; $1 \leq l \leq r \leq |S|$

Notas sobre a implementação:

- Nenhuma ordem de precedência é considerada. A expressão deve estar totalmente entre parênteses.
- Dois parênteses são considerados diferentes se a sequência produzida for diferente.
- A string de operações deve conter apenas caracteres 'a', 'o' e 'x', representando and, or e xor, respectivamente.

Exemplo de entrada:

110 ax
3
1 1 true
1 2 false
2 3 false

Exemplo de Saída:

1
0
0

1.2 Maximizando a Soma

Dado um *array* A de N números e um *array* P inicialmente vazio, você pode executar 4 tipos de operações:

- Pegar o último número do *array* A, remove-lo e adicionar à P.
- Pegar os dois últimos números do *array* A (se o tamanho for maior que 1), remove-los e adicionar o produto deles à P.
- Inverter o *array* A, pegar o último número do *array* A, remover e adiciona-lo à P.
- Inverter o *array* A, pegar os dois últimos números do *array* A (se o tamanho for maior que 1), remova-os e adicione o seu produto à P.

Observe que após cada operação, o *array* A se torna menor e talvez revertido. Você precisa continuar adicionando elementos ao *array* P até que o *array* A tenha zero elementos. Você precisa maximizar a soma de todos os valores no *array* P.

Entrada: A primeira linha contém um único número inteiro N. A próxima linha contém N números inteiros separados por espaço, indicando o *array* A.

Saída: Emitir a máxima soma de todos os elementos do *array* P que podem ser obtidos executando as 4 operações listadas acima.

Considerar: $1 \leq N \leq 10^3$; $1 \leq A_i \leq 10^6$

Exemplo de entrada:

5
1 4 2 3 5

Exemplo de saída:

24

Explicação:

- Pode-se obter 24 através das operações nas ordens [2, 3, 2].
- Após a primeira operação (2): A = [1, 4, 2], P = [15].
- Após a segunda operação (3): A = [2, 4], P = [15, 1].
- Após a terceira operação (2): A = [], P = [15, 1, 8].
- Soma total: 15 + 1 + 8 = 24.

1.3 Particionamento de Cadeias Binárias

Você recebe uma sequência binária de 0s e 1s.

Sua tarefa é calcular o número de maneiras para que uma sequência possa ser particionada, satisfazendo as seguintes restrições:

- O comprimento de cada partição deve estar no formato não decrescente. Portanto, o comprimento da partição anterior deve ser menor ou igual ao comprimento da partição atual.
- O número de bits definidos em cada partição deve estar no formato não decrescente. Portanto, o número de 1s disponíveis na partição anterior deve ser menor ou igual ao número de 1s disponíveis na partição atual.
- Não deve haver zeros disponíveis à esquerda em cada partição.

Por exemplo, a cadeia *101100* contém *10|1100*, *101100*, como as duas partições válidas, enquanto que *10|1|100*, *1|0|1100* como algumas das partições inválidas.

Nota: A cadeia como um todo é uma partição válida.

Imprima o número de maneiras possíveis.

Entrada: A primeira e única linha contém e representa o comprimento da cadeia e a cadeia binária.

Saída: Contém o número de maneiras que a cadeia binária pode ser particionada.

Considerar: $n \leq 5000$

Exemplo de Entrada: 6 101110

Exemplo de Saída: 3

1.4 A força de um *array*

Vamos considerar um *array* A. O algoritmo a seguir calcula sua força:

Encontre todos os blocos contínuos onde todos os elementos de A são iguais. Calcular a soma dos comprimentos ao quadrado desses blocos. Por exemplo, se A = *2, 3, 3, 1, 1, 1, 4, 2, 2*, sua força será $1^2 + 2^2 + 3^2 + 1^2 + 2^2 = 19$

Podemos reordenar alguns elementos e obter um *array* com maior força. No exemplo acima, podemos mover o primeiro elemento de A para o final e obter o *array* A = *3, 3, 1, 1, 1, 4, 2, 2, 2* com força $2^2 + 3^2 + 1^2 + 3^2 = 23$.

Você recebe um *array*. Qual é a força máxima que você pode obter ao reordenar alguns de seus elementos?

Entrada: A primeira linha contém T inteiro, indicando o número de casos de teste. As seguintes T linhas contêm 4 números inteiros cada: A[0], A[1], N, MOD.

O *array* A de N elementos é gerado da seguinte maneira:

A[0] e A[1] são dados

A[i] = (A [i-1] + A [i-2]) módulo MOD para 1 ¡i ¡N.

Saída: Para cada caso de teste, imprima um número inteiro em linha separada respondendo a pergunta anterior.

Considere:

$1 \leq T \leq 100$

$0 \leq A[0], A[1] \leq 10^6$

$2 \leq N \leq 10^6$

$\max(A[0], A[1]) \leq MOD \leq 10^6$

Exemplo de Entrada:

2

0 1 6 4

1 1 4 2

Exemplo de Saída:

12

10

Exmplicação:

Caso de teste 1: A = 0, 1, 1, 2, 3, 1. A maior força possível será com a reordenação 2, 0, 1, 1, 1, 3

Caso de teste 2: A = 1, 1, 0, 1. A maior força possível será com a reordenação 1, 1, 1, 0

1.5 A última chance de Dexter

Dexter lhe dá uma última chance de sobreviver. Ele lhe dá um problema para resolver. Se você resolver o problema corretamente, ele o deixará ir, caso contrário, ele o matará.

Você recebe **N** números inteiros a_1, a_2, a_N. Considere um hiper-espaço N-dimensional. Seja (x_1, x_2, ..., x_N) um ponto neste hiper-espaço e todos os valores x_i para i petencente a [1,N] são inteiros. Agora, Dexter dá a você um conjunto que contém todos os pontos tal que $0 \leq x_i \leq a_i$ para i pertencendo a [1,N]. Encontre o número de maneiras para selecionar dois pontos A e B deste conjunto, tal que o ponto médio de A e B também recai sobre este conjunto.

Nota: Os dois pontos podem ser os mesmos.

Entrada: A primeira linha contém um único inteiro N, representando o número de dimensões do hiper-espaço. A segunda linha contém N inteiros, o i^{th} representando a_i, conforme definido no problema.

Saída: Contém um único inteiro que responde a pergunta do problema.

Considere:

$1 \leq \mathrm{N} \leq 10^5$

$0 \leq a_i \leq 10^9$

Exemplo de Entrada:

2

1 2

Exemplo de Saída:

10

Explicação: O conjunto contém os pontos: {(0,0), (0,1), (0,2), (1,0), (1,1), (1,2)}. Os pares de pontos (A,B), cujo ponto médio que recaem no conjunto são:

A=(0,0), B=(0,0), ponto médio = (0,0)
A=(0,0), B=(0,2), ponto médio = (0,1)
A=(0,1), B=(0,1), ponto médio = (0,1)
A=(0,2), B=(0,2), ponto médio = (0,2)
A=(0,2), B=(0,0), ponto médio = (0,1)
A=(1,0), B=(1,0), ponto médio = (1,0)
A=(1,0), B=(1,2), ponto médio = (1,1)
A=(1,1), B=(1,1), ponto médio = (1,1)
A=(1,2), B=(1,2), ponto médio = (1,2)
A=(1,2), B=(1,0), ponto médio = (1,1)

1.6 Onde está a bolinha de gude?

João e Julia adoram brincar com bolinhas de gude. Eles têm muitas bolinhas de gude com números escritos nelas. No começo, João colocava as bolinhas uma após a outra em ordem crescente dos números escritos nelas. Então Julia pedia a João que encontrasse a primeira bola de gude com um certo número. Ela contaria 1 ... 2 ... 3. João recebe um ponto para a resposta correta, e Julia recebe o ponto se João falha. Após um número fixo de tentativas, o jogo termina e o jogador com o máximo de pontos vence. Hoje é sua chance de jogar como João. Sendo o aluno esperto, você estaria aceitando o favor de um computador. Mas não subestime Julia, ela havia escrito um programa para acompanhar quanto tempo você está demorando para dar todas as respostas. Então agora você precisa escrever um programa, que o ajudará em seu papel como João.

Entrada: Pode haver vários casos de teste. O número total de casos de teste é menor que 65. Cada caso de teste consiste em começar com 2 números inteiros: N o número de bolas de gude e Q o número de consultas que Julia faria. Nas próximas N linhas contém os números escritos nas N bolinhas de gude. Esses números da bolinha de gude não virão em qualquer ordem específica. As seguintes Q linhas terão Q consultas. Tenha certeza de que nenhum dos números de entrada são maiores que 10000 e nenhum deles é negativo. A entrada é finalizada por um caso de teste em que $N = 0$ e $Q = 0$.

Saída: Para cada caso de teste, imprima o número de série do caso. Para cada uma das consultas, imprima uma linha de saída. O formato desta linha dependerá se o número da consulta está escrito ou não em qualquer uma das bolinhas de gude. Os dois formatos diferentes são descritos abaixo:

- *'x encontrado em y'*, se a primeira bolda de gude com o número x foi encontrada na posição y. As posições são numeradas de *1, 2,. . . , N*.
- *'x não encontrado*', se a bolinha de gude com o número x não estiver presente.

Exemplo de Entrada:

```
4 1
2
3
5
1
5
5 2
1
3
3
3
1
2
3
0 0
```

Exemplo de Saída:

```
CASO# 1:
5 encontrado em 4
CASO# 2:
2 não encontrado
3 encontrado em 3
```

1.7 O Pão do Chefe

O Chef tem um pão longo de comprimento 1. Ele quer cortá-lo em tantos pães quanto puder. Mas ele quer aderir à seguinte regra: A qualquer momento, o comprimento do pão mais longo que ele possui não pode ser maior que o comprimento do pão mais curto, vezes um fator constante. Toda vez, ele só pode cortar exatamente um pão em dois mais curtos.

Entrada: Um número de ponto flutuante, $1 \leq k \leq 1999$, significando o fator constante. O número terá no máximo 3 dígitos após o ponto decimal.

Saída: Primeiro, você deve gerar um número n, o número máximo possível de pães para o valor determinado do fator constante. Em seguida, você deve apresentar qualquer prova de que esse número de pães seja de fato possível: *n-1* descrições de corte, usando a seguinte notação.

Em cada etapa, você imprime dois números: primeiro, o índice do pão que deseja cortar em duas partes; segundo, o comprimento do pão recém-criado (cortado do original). Supõe-se que o pão inicial tenha o índice 0. Cada pão recém-criado receberá o menor número inteiro livre possível (portanto, no i-ésimo passo, este será i). Cada vez, o tamanho do pão original será diminuído pelo tamanho do pão recém-criado.

Exemplo de Entrada:

1.5

Exemplo de Saída:

4
0 0.4
0 0.3
1 0.2