Projeto Olímpico de Programação

Seletiva POP

21 de junho de 2019

(Este caderno contém 7 problemas)

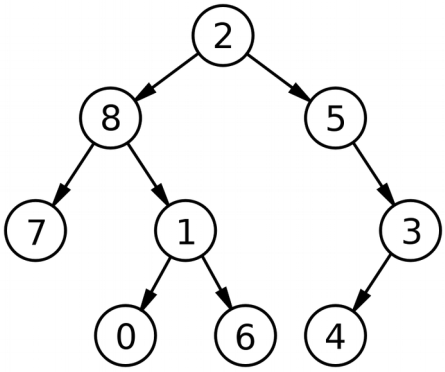
A PROVA TERÁ DURAÇÃO DE **DUAS HORAS E MEIA**

LEIA ATENTAMENTE AS INSTRUÇÕES ABAIXO ANTES DE INICIAR A PROVA

- A prova deve ser realizada individualmente;
- Observem o nome do arquivo que deve ser enviado para cada problema;
- Cada questão tem um tempo limite para execução;
- É permitido consultar material impresso durante a prova;
- Não é permitida a consulta de qualquer material online;
- Este caderno de tarefas é composto de 13 páginas;
- Verifique se o caderno está completo.

Problema A – Aposta à altura
Nome do Programa: aposta.(c|cpp|py|java)
Tempo: 2 segundos

Em uma árvore, a profundidade de um nó é a distância desse nó até a raiz. A altura da árvore é a maior profundidade existente nela. Na figura abaixo, por exemplo, a altura da árvore é 3, que é a profundidade dos nós 0, 4 e 6.



João e Maria, sem nada para fazer, escreveram um programa para gerar árvores aleatoriamente e estão apostando quem descobre mais rápido a altura da árvore. Para saber quem acertou, precisam da altura correta, mas não confiam um no outro para escrever um programa para isso. Para garantir que nenhum dos dois trapaceie, escreva um programa para determinar essa altura.

Entrada

A entrada é composta por um único caso de teste. A primeira linha contém dois inteiros **N** e **R** ($1 \leq N \leq 10^5$, $0 \leq R < N$), representando a quantidade de nós e a raiz da árvore, respectivamente. Os nós são números inteiros entre 0 e N – 1. As próximas N – 1 linhas contém dois nós **X** e **Y** ($0 \leq X, Y < N$), indicando que há uma aresta que conecta ambos.

Saída

A saída deve conter um único inteiro representando a altura da árvore.

Casos de Teste

Entrada	Saída
9 2 2 5 5 3 4 3 8 1 1 6 0 1 7 8 8 2	3

Entrada	Saída
6 0 3 0 2 3 1 2 4 2 5 0	3

Problema B - Bonacci
Nome do Programa: bonacci.(c|cpp|py|java)
Tempo: 2 segundos

Bonacci era um famoso jardineiro italiano que viveu na mesma época que um famoso matemático cujo nome é bem parecido, mas não me recordo bem. Certa vez, em um sonho, o jovem Bonacci imaginou uma sequência de números: 0 1 1 2 3 5 8 13 21 ... e julgou que esta seria uma sequência mágica, tal sua beleza. Segue a definição matemática da sequência:

B(x) = 0, se x ≤ 0,

B(x) = 1, se x = 1,

B(x) = [B(x – 1) + B(x – 2)] % 99, se x ≥ 2

Observe que % 99 significa o resto da divisão por 99.

Uma vez que o jovem Bonacci é apenas um jardineiro, ele te pediu para escrever um programa que, dado um número inteiro, calculasse o número de Bonacci correspondente.

Entrada

A entrada apenas um número inteiro **x** ($0 \leq$ **x** ≤ 70000).

Saída

A saída deve conter os números **x** e **B(x)**, separados por um espaço.

Casos de Teste

Entrada	**Saída**
0	0 0

Entrada	**Saída**
12	12 45

Entrada	**Saída**
50	50 55

Problema C – Cuadrado quer dizer quadrado em espanhol
Nome do Programa: cuadrado.(c|cpp|py|java)
Tempo: 2 segundos

O pequetitito Jamez Gonçales é um chico que ama aqueles desafios do tipo quadrado mágico e Sudoku. Para quem não está familiarizado, são aqueles desafios nos quais se deseja posicionar números em um tabuleiro tal que todas as somas verticais e horizontais sejam iguais.

No último Navidad, Jamez ganhou um livro com 500 páginas cheias deste tipo de desafio. Mas conseguiu resolver todos em apenas 2 horas (!). Entediado, o garoto resolveu descobrir se separando números inteiros dígito por dígito seria possível montar um quadrado mágico. Por exemplo, o número 3113 forma um quadrado mágico? Sim, pois todas as somas horizontais e verticais são 4:

3 1

1 3

Seguindo o mesmo raciocínio, Jamez descobriu que o número 1234 não forma um quadrado mágico.

Entrada

A estrada possui exatamente um número inteiro **x** com quatro dígitos, isto é, $1000 \leq \mathbf{x} \leq 9999$.

Saída

Caso o número da entrada possa formar um quadrado mágico, o programa deve imprimir OK. Caso contrário o programa deve imprimir FAIL.

Casos de Teste

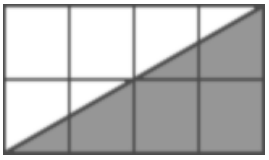
Entrada	Saída
3113	OK

Entrada	Saída
1234	FAIL

Entrada	**Saída**
4071	FAIL

Problema D - Diagonal
Nome do Programa: diagonal.(c|cpp|py|java)
Tempo: 2 segundos

Diego Nau é um sujeito excêntrico, obcecado por retângulos e diagonais. Em todas as suas casas (ele é muito rico), todos os cômodos são retangulares e os pisos possuem duas cores, uma de cada lado da diagonal. Observe o exemplo seguinte.



Considere que o piso de um cômodo possui tamanho **M**×**N** e é formado por peças quadradas de tamanho 1×1. Para que cada lado da diagonal tenha uma cor, algumas dessas peças precisão ser cortadas (aquelas pelas quais passa a diagonal). No exemplo acima, quatro peças de cada cor precisariam ser cortadas.

Diego Nau quer fazer um novo projeto e precisa saber quantas peças de cada cor precisarão ser cortadas, pois esta é a parte mais demorada e cara. Como os valores são muito grandes, ele não consegue calcular e pediu sua ajuda.

Entrada

A entrada é composta por uma única linha contendo dois inteiros **M** e **N** ($1 \leq \mathbf{M}, \mathbf{N} \leq 10^{18}$) representando as dimensões do cômodo.

Saída

A saída deve conter um único inteiro indicando quantas peças de cada cor precisarão ser cortadas.

Casos de Teste

Entrada	Saída
4 2	4

Entrada	Saída
3 5	7

Problema E – Escapada não trivial
Nome do Programa: escapada.(c|cpp|py|java)
Tempo: 2 segundos

Sempre que vê uma moto, Timerinho, filho de Timério, fica com muito medo e vai correndo para casa. No entanto, ele percebeu que não adianta fugir pelo menor caminho, pois é muito fácil de ser assaltado.

Então Timerinho resolveu inovar. Como escapar pelo menor caminho não é efetivo, vai começar a escapar pelo segundo menor caminho numa tentativa de despistar os possíveis assaltantes.

O problema é que ele não conhece direito os segundos menores caminhos para sua casa. Mas você pode ajudá-lo dizendo apenas o tamanho do segundo caminho, dados um grafo da cidade e as posições de Timerinho e da sua casa.

Note que o comprimento do segundo menor caminho pode ser igual ao do primeiro, se houverem caminhos diferentes com o mesmo comprimento. O caminho também pode passar pelo mesmo lugar mais de uma vez e até mesmo percorrer um mesmo trecho várias vezes.

Entrada

A entrada é composta por um único caso de teste. A primeira linha contém dois inteiros **V** e **A** indicando a quantidade de vértices e arestas, respectivamente, do grafo ($1 \leq \mathbf{V} \leq 10^2$ e $1 \leq \mathbf{A} \leq 10^4$).

Seguem-se **A** linhas. Cada uma delas contém três inteiros **X**, **Y** e **Z** ($0 \leq \mathbf{X}, \mathbf{Y} < \mathbf{V}$ e $0 \leq \mathbf{Z} \leq 10^4$), indicando que existe um caminho de **X** até **Y** (nesta direção) com distância **Z**.

A última linha da entrada contém dois inteiros **T** e **C** ($0 \leq \mathbf{T}, \mathbf{C} < V$) representando a posição inicial de Timerinho e a posição de sua casa, respectivamente.

Saída

A saída deve conter um único inteiro indicando o comprimento do segundo menor caminho entre a posição de Timerinho e sua casa. Garante-se que sempre existirá um resultado válido.

Casos de Teste

Entrada	Saída
3 4 2 0 3 0 2 1 0 1 7 1 2 6 0 2	5

Entrada	Saída
4 7 2 0 9 3 3 10 3 0 14 1 0 1 0 1 8 3 1 4 1 2 16 3 0	14

Entrada	Saída
2 2 1 0 2 0 1 4 1 0	8

Problema F – Fundo do poço
Nome do Programa: fundo.(c|cpp|py|java)
Tempo: 2 segundos

Não é segredo para ninguém que os programas de auditório japoneses são os melhores do mundo. Isso se dá, principalmente, por conta das muitas loucuras que eles são capazes de aprontar com os participantes. Em especial, no ano passado foi lançado um novo quadro chamado “Fundo do poço” (FdP). No Fdp, é um participante vê rapidamente um mapa e pode mandar um trator (!!!) para cavar um (e apenas um) ponto do mapa. Todos os tesouros encontrados são dados ao participante.

Depois de assistir várias vezes ao programa, Tobuco Toburo (TT), muito esperto, resolveu comprar um programa para auxiliar os participantes nesta tarefa. Sua missão é escrever este programa.

Entrada

A primeira linha da entrada contém um inteiro **N** ($2 \leq \mathbf{N} \leq 100$), representando o tamanho do mapa. As próximas **N** linhas apresentam uma matriz **N**×**N** que representa o mapa mostrado ao participante.

Cada célula da matriz é um número inteiro $\mathbf{x_{ij}}$ ($0 \leq \mathbf{x_{ij}} \leq 1000$) que diz a quantidade de prêmios que existe nesta posição do mapa. O que TT percebeu, contudo, é que sempre que a esta quantidade é maior que 100, o programa paga apenas 100, nada mais.

Saída

Um buraco cavado pelo trator consegue capturar os prêmios de um quadrado de tamanho 2×2, isto é, com duas células adjacentes na vertical e na horizontal. A saída do programa deve ser a maior quantidade de prêmio que o participante consegue obter.

Casos de Teste

Entrada	Saída
3 41 120 34 64 70 28 114 116 115	334

Entrada	**Saída**
4 0 1 1 0 0 1 1 0 0 0 0 0 2 1 0 3	4

Problema G – Gameplay comentado
Nome do Programa: gameplay.(c|cpp|py|java)
Tempo: 2 segundos

Nesta era da Internet e dos jogos digitais, uma forma bem interessante de melhorar a sua capacidade em certos jogos é assistindo a replays (gameplays) comentados. Certa vez, ZurubuGuda, um famoso streamer de gameplays do aclamado videogame Z-caixa, fez uma promoção para quem acertasse todos os botões que ele apertou durante um jogo que ele transmitiu online. Foi um fracasso (!!!). Todo mundo acertou e quando o prêmio foi dividido, cada um ganhou R$ 0,02. ZurubuGuda, desta vez, tornou o desafio mais difícil. Além de acertar os botões em sequencia, ele dá certos comandos para mudar os botões. Todos sabem que os botões do Z-Caixa são **a**, **b**, **x** e **y**. Quando ZurubuGuda dá um comando flip, **a** → **b**, **b** → **x**, **x** → **y** e **y** → **a**. Dada a sequência de botões, de comandos flip, e de perguntas, você consegue escrever um programa que vença o novo desafio de ZurubuGuda?

Entrada

A entrada contém apenas um caso de teste. A primeira linha da entrada tem um inteiro **N** ($1 \leq$ **N** ≤ 20000) indicando o número de botões que ZurubuGuda pressionou. A próxima linha possui uma string com **N** letras representando estes botões. As próximas **N** linhas apresentam **N** operações no formato **op start end**. Caso **op** seja ‘q’, o programa deve imprimir o botão que mais se repetiu entre os índices (inclusive) **start** e **end** ($0 \leq$ **start** $\leq$ **end** $\leq$ **N** - 1). Em caso de empate **a** tem prioridade sobre **b**, **b** sobre **x**, e **x** sobre **y**. Caso **op** seja ‘f’, é realizada a operação de flip em todos os botões de **start** até **end** (inclusive).

Saída

Para cada operação ‘q’, o programa deve imprimir os valores **start**, **end**, o botão que mais se repetiu e o número de vezes que esse botão aparece.

Casos de Teste

Entrada	Saída
5 ayaya q 1 2 q 0 1 f 2 3 q 1 4 q 1 1	1 2 a 1 0 1 a 1 1 4 a 2 1 1 y 1

Entrada	Saída
2 ab f 0 1 q 0 1	0 1 b 1