IX - Maratona POP Superior - Le Dicário!

Olá pessoal, este é o Le Dicário! O objetivo deste documento é fazer com que a maratona POP seja, antes de um momento de competição, um momento de aprendizagem.

Deste modo, preparamos um conjunto de dicas sobre os problemas da prova, de modo a destravar algum competidor que esteja com problemas, ou mesmo criar a possibilidade de resolução de mais questões. Então vamos lá!

Ordem de dificuldade das questões: K C A H B E F G D J I

As dicas que seguem são dadas em ordem crescente da dificuldade das questões. Isto é, segundo o nosso julgamento, as questões mais fáceis da prova aparecem aqui antes das mais difíceis.

Dicas gerais sobre Entrada e Saída

Use as funções padrão de litura da sua linguagem favorita. Preparamos exemplos de leitura na questão X do aquecimento. Considere copiá-las.

Evite imprimir qualquer coisa que não seja pedido pelo problema. Por exemplo, nunca imprima algo como “digite um número inteiro:”, a não ser que o problema peça exatamente isso.

Dica sobre a questão K

Bom, se vocês estão enfrentando dificuldades com este problema, por favor, entrem em contato com a organização, eles irão até aí te ajudar! =-)

Dica sobre a questão C

Considere olhar cada letra da *string* passada, partindo da posição p=0 até p=N-1 (sendo N o número de letras da *string*). Caso p % 2 seja 0, a carta foi para Eduardo Caso p % 2 == 1, a carta foi para Mônica.

Dica sobre a questão A

O problema consiste em verificar quantas linhas do texto tem a palavra MARIQUINHA. Note que MA-RIQUINHA não conta, e se o nome aparece duas vezes na mesma linha, também não conta.

Toda linguagem possui funções para verificar se um texto contem determinada *string*. Mesmo sem estas funções, basta varrer o texto desde o início e encadear condições procurando por 'M', 'A', 'R', 'I', ..., 'H', 'A'.

Dica sobre a questão H

Após a leitura do nome da pessoa, basta usar dois laços for, comparar as letras, e verificar a menor distância entre letras iguais.

Dica sobre a questão B

Bom, você deve ler duas *strings:* A e B. Dá para trocar cada dígito por 9, transformar em inteiro, e verificar se foi a melhor troca possível.

Dica sobre a questão E

O teste para verificar se um número é primo é bem simples: veja se o número só é divisível por ele mesmo e por 1. Existem testes mais elaborados, mas este deve ser suficiente dado o tamanho das entradas. Observe que se um número é primo, você deve somar os seus dígitos e este número resultante também deve ser primo.

Mas não para por aí, se o resultante for primo, você deve somar seus dígitos e verificar se o resultante também é primo. Isto deve seguir até que o resultante tenha apenas um dígito. Que tal um algoritmo recursivo? Se não quiser, pode fazer uma função testa_primo() e um laço:

ENQUANTO numero_digitos(string) > 1 **FAÇA** ....

Caso o tempo não tenha sido suficiente, considere salvar a quantidade de primo-primo-primo-primos de 1 até 1000. Depois basta subtrair quando a questão pedir.

Dica sobre a questão F

Esta é uma questão clássica: a mochila 0-1 sem limite de repetição. Existe um algoritmo bem conhecido de programação dinâmica que resolve este problema. Caso não conheça o algoritmo, pense no seguinte:

Resolva o problema considerando a capacidade máxima =1 (fácil né). Agora use esta solução para resolver o problema com capacidade máxima 2 (ou coloco um item com capacidade 2, ou um com capacidade 1 e o melhor possível (que já encontrei) com capacidade 1.

Agora use esta solução para resolver o problema com capacidade máxima 3 (ou coloco um item com capacidade 3, ou um com capacidade 2 e o melhor possível (que já encontrei) com capacidade 1, ou um com capacidade 1 e o melhor possível (que já encontrei) com capacidade 2. E assim em diante...

Dica sobre a questão G

Considere os caminhos até o pai de todos. Depois basta comparar estes caminhos até encontrar um ancestral comum (ou um ancestral diferente, depende de onde você começou a comparação).

Ah, e como guardar a árvore de parentesco? Considere um vetor P (array) de tamanho 5000. O valor P[x] é -1 caso este número não tenha aparecido; P[x]=x caso não se saiba quem é o pai de x e P[x]=y caso y seja o pai de x. Desta forma, você consegue o caminho de parentesco de qualquer bactéria e depois resta compará-los.

Dica sobre a questão D

Já considerou força bruta? ;-) Combinação de 5, 2 a 2, é apenas 10. Assim como combinação de 5, 3 a 3, é apenas 10.

Dica sobre a questão J

Uma questão bem interessante, diga-se de passagem. Pode-se pensar que é mais fácil resolvê-la usando Cálculo (Fórmula do comprimento do arco). Contudo, acredito que métodos numéricos façam muito mais sentido.

Dica sobre a questão I

Onde está o seu caderno de Geometria Computacional? Convex Hull, Área do Polígono, Intersecção entre Polígonos... =-)

Boa sorte!