12ª Maratona de Programação

29 de junho de 2013

Caderno de Problemas

Regras: ACM *International Collegiate Programming Contest*

Brasil – **Desafio CTBC**

(Este caderno contém 10 problemas; as páginas estão numeradas de 1 a 12)

Participantes:

CESUC
FAZU
FEIT-UEMG
FPU
IFTM Uberaba
IFTM-Uberlândia
Pitágoras
UNIFEI
UFU
UNIPAC Araguari
UNIPAM
UNITRI
UNIUBE Uberaba
UNIUBE Uberlândia
University of Southern California

Problema A
Necessidade de outro Planeta

Arquivo fonte: planeta.c, planeta.cpp ou planeta.java

Em um planeta distante, pertencente a uma galáxia mais distante ainda, moram os CAPs. Eles são altamente dependentes de uma estrela semelhante ao nosso sol. Para que eles sobrevivam, constantemente eles medem a temperatura desta estrela. Como esta super estrela é milhões de vezes mais poderosa do que o sol, ela alcança temperaturas elevadíssimas. Sabe-se que esta estrela tem sua temperatura aumentada constantemente durante um período de tempo, e após um determinado evento (desconhecido), a temperatura começa a cair constantemente durante um período de tempo. Isso se repete todos os dias, e se alguma anomalia ocorrer, os CAPs devem ficar em alerta.

Qualquer mudança nesse comportamento deve ser detectada rapidamente. Então, diariamente, uma equipe especializada no assunto faz medições da temperatura desta estrela em intervalos muito pequenos e armazena as informações em uma espécie de planilha. Após a coleta das informações, deve-se verificar se existem temperaturas repetidas. Se encontrar alguma temperatura repetida, um alarme deve ser emitido o mais rápido possível, pois algo estranho poderá ocorrer.

Sua tarefa é fazer um algoritmo que ajude os CAPs a encontrar as anomalias o mais rapidamente possível.

Entrada
A entrada é formada de vários casos de teste. Cada caso é formado por duas linhas. A primeira linha contém o número de leituras a serem realizadas ($1 \leq N \leq 10^7$), seguido de um espaço em branco separando o número de leituras do valor máximo esperado das temperaturas t ($10^3 \leq t \leq 10^9$). A segunda linha contém as leituras realizadas, sendo que cada temperatura é separada por um espaço em branco (inclusive após a última leitura). A última linha contém o número zero indicando o fim das entradas.

Saída
Para cada caso de teste, deverá ser gerada apenas uma linha contendo um número 1 (um) ou o número 0 (zero), indicando alarmar ou não (1 alarma, 0 normal).

Exemplo de Entrada
```
5 1000
234 560 980 981 999
10 5000
234 560 980 981 999 1560 2360 4599 4599 5000
6 5000
5000 4000 3233 2360 1556 500
0
```

Exemplo de Saída
```
0
1
0
```

Problema B
Criptografia por Bases

arquivo: base.c , base.cpp, base.java

Natal é um cientista da computação apaixonado por números. Recentemente lhe ocorreu uma idéia inovadora: é possível aplicar a teoria de conversão de bases na área de criptografia.

Todas as bases podem ser representadas por um alfabeto de símbolos em ordem crescente, na qual o primeiro símbolo equivale ao zero. Por exemplo: o sistema decimal pode ser representado por “0123456789”, o binário por “012”, a hexadecimal por “0123456789ABCDEF” e assim por diante.

Dessa maneira, poderíamos utilizar qualquer alfabeto desde que a representação obedeça as regras citadas acima. Uma estranha base poderia ser composta pelo alfabeto “oF8”, onde o = 0 e o < F < 8, dessa maneira os números de um a dez da base decimal seriam convertidos para F, 8, Fo, FF, F8, 8o, 8F, 88, Foo, FoF.

O processo de criptografia de uma mensagem necessita de duas bases, a de criptografia (utilizada na mensagem criptografada) e a de apresentação (utilizada para reconstrução da mensagem original).

Você foi convidado a participar desse ambicioso projeto, que pode revolucionar a computação contemporânea. Para isso será necessário desenvolver o algoritmo de “descriptografia”. Dada uma mensagem, a base de criptografia e a base de apresentação, o algoritmo deverá ser capaz de descobrir a mensagem original.

Entrada

A primeira linha de código é um inteiro N (1 ≤ N ≤ 100) que indica o número de casos de teste. Cada uma das N linhas posteriores terá 3 valores separados por espaço: ***mensagem base_criptografia base_apresentação***, onde 1 ≤ número de símbolos na mensagem ≤ 40, 2 ≤ número de símbolos em base_criptografia ≤ 100 e 2 ≤ número de símbolos em base_apresentação ≤ 100. Cada base é definida com um alfabeto de símbolos dispostos em ordem crescente. Não há símbolos repetidos na representação de nenhuma das bases. Os símbolos podem ser: números (0-9), letras (a-zA-Z) ou um dos caracteres a seguir `!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

Saída

Para cada caso de teste deverá ser impressa uma linha de código contendo "`Caso #x: `", onde x é o número do caso de teste, seguido da mensagem descriptografada.

Exemplo de Entrada

```
3
9 0123456789 oF8
Foo oF8 0123456789
13 0123456789abcdef 01
```

Exemplo de Saída

```
Caso #1: Foo
Caso #2: 9
Caso #3: 10011
```

Problema C
IP TV

Arquivo fonte: iptv.c, iptv.cpp ou iptv.java

Um consórcio de fornecedores europeus de Internet gerencia uma grande rede de backbone, com links diretos (conexões) entre um grande número de cidades europeias. A ligação entre um par de cidades é bidirecional. A transmissão de uma mensagem em um link tem um determinado custo associado. Como é comum na Internet, é possível utilizar uma sequência (sem limites) das ligações diretas indiretamente para transferir dados entre qualquer par de cidades.

Para transmitir programas de televisão, utilizando este backbone, é necessário enviar continuamente dados para todos os nós da rede. Para ajudar a minimizar os custos, é necessário selecionar as ligações de rede que serão utilizados para a transmissão de dados. O conjunto de ligações selecionado deve ser conectado e inclui todos os nós da rede.

Para ajudar o consórcio a gerenciar sua rede, você foi convidado para criar um programa que calcula o custo mínimo para a transmissão de dados para todas as cidades do backbone.

Dado um conjunto de links de rede, calcular o custo de transmissão mínima para alcançar todos os nós.

Entrada

A entrada consiste em vários casos de teste. A primeira linha da entrada contém o número de casos de teste. Há uma linha em branco antes de cada conjunto de dados. A primeira linha de cada conjunto de dados contém M, um número inteiro positivo, não superior a 2000, com o número de cidades que têm conexões de rede. A segunda linha contém um número inteiro N, não superior a 50.000, com o número de ligações existentes. Cada uma das N linhas seguintes contém a representação de um link. Cada linha contém duas strings e um inteiro, separados por espaços vazios, B E C, onde B e E são os nomes das cidades das extremidades do link de rede, com não mais de 8 caracteres, e C é um número inteiro positivo, não superior a 30 , representando o custo de transmissão no link.

Saída

A saída é constituída por uma única linha, que contém um número inteiro com o mínimo custo de transmissão para o envio de dados para todas as cidades. Imprima uma linha em branco entre os conjuntos de dados.

Exemplo de Entrada

```
1

4
5
lisboa londres 6
lisboa paris 5
londres paris 1
londres berlin 2
paris berlin 10
```

Exemplo da Saída

```
8
```

Problema D
Abelha

Arquivo fonte: abelha.c, abelha.cpp ou abelha.java

Na África há uma espécie muito especial de abelhas. Todos os anos, as abelhas fêmeas dessas espécies dão à luz a uma abelha macho, enquanto que as abelhas macho dão a luz a uma abelha macho e uma fêmea, e então eles morrem!

Agora os cientistas descobriram acidentalmente uma "abelha fêmea mágica" de tais espécies especiais sendo que ela é imortal, mas ainda capaz de dar à luz uma vez por ano, como todas as outras abelhas fêmeas. Os cientistas gostariam de saber quantas abelhas existirão depois de N anos. Escreva um programa para ajudá-los a encontrar o número de abelhas macho e o número total de todas as abelhas, após N anos.

Entrada
Cada linha da entrada contém um inteiro N ($N \geq 0$). A entrada finaliza quando N = -1 (neste caso, não deve ser processado).

Saída
Cada linha da saída deverá ter dois números, o primeiro é o número de abelhas macho após N anos, e o segundo representa o total de abelhas após N anos. (Os dois números não deverão exceder 2^{32}).

Exemplo de entrada
```
1
3
-1
```

Exemplo de saída
```
1 2
4 7
```

Problema E
Copa das Confederações

Arquivo fonte: copa.c, copa.cpp ou copa.java

Durante a Copa das Confederações no Brasil, a seleção do Taiti resolveu aproveitar para fazer turismo: ela vai visitar cada uma das cidades-sede da copa. Porém, como os taitianos não possuem muitos recursos, eles vão utilizar a malha rodoviária do país para fazer o passeio, ou seja, vão de ônibus.

Podemos considerar que sempre existirá uma estrada ligando diretamente qualquer uma das cidades-sede. Entretanto, devido algumas chuvas, determinadas estradas acabaram sendo destruídas e, consequentemente, será impossível a seleção do Taiti utilizá-las em seu passeio. Sua missão é ajudar o motorista, identificando quantas possibilidades de caminho o ônibus tem para passar por cada cidade exatamente uma vez.

Como você é esperto, rapidamente abstraiu o problema acima. É fácil perceber que as cidades-sede e as estradas do Brasil formam um grafo completo e sua missão é encontrar o número de ciclos Hamiltonianos que não utilizam as arestas (estradas) bloqueadas.

Considere, então, que dado um grafo completo não-direcionado com N nós, numerados de 1 a N, e K arestas proibidas (estradas bloqueadas pelas chuvas), você deve encontrar o número de ciclos Hamiltonianos no grafo que não utilizam nenhuma das K arestas. Um ciclo Hamiltoniano é um ciclo que visita cada vértice exatamente uma vez. Um ciclo que contém as mesmas arestas é contado apenas uma vez. Por exemplo, os ciclos `1 2 3 4 1` e `1 4 3 2 1` e `2 3 4 1 2` são todos iguais, mas `1 3 2 4 1` é diferente.

Entrada

A primeira linha da entrada contém um inteiro T ($1 \leq T \leq 10$), representando a quantidade de casos de teste. A primeira linha de cada caso de teste contém dois inteiros, N ($3 \leq N \leq 300$) e K ($0 \leq K \leq 15$). As próximas K linhas contêm dois inteiros cada, representando os vértices de uma aresta proibida. Não haverá arestas repetidas nem arestas cujos dois vértices são iguais (auto-arestas).

Saída

Para cada caso de teste, imprima uma linha contendo o número de ciclos Hamiltonianos H que não incluem nenhuma das K arestas. Imprima sua resposta H utilizando módulo 9901, ou seja, H mod 9901.

Exemplo de entrada

```
2
4 1
1 2
8 4
1 2
2 3
4 5
5 6
```

Exemplo de saída

```
1
660
```

Problema F
Falha de Disco

Arquivo fonte: disco.c, disco.cpp ou disco.java

Merge sort é um clássico algoritmo de ordenação. Ele divide o array de entrada em duas metades, ordena cada uma recursivamente, e, então, faz o *"merge"* das duas metades ordenadas.

Neste problema, o *merge sort* é usado para ordenar um array de inteiros em ordem ascendente. O seguinte pseudocódigo reproduz o comportamento exato do algoritmo:

```
function merge_sort(arr):
    n = arr.length()
    if n <= 1:
        return arr
    // arr é indexado de 0 até n-1, inclusive
    mid = floor(n/2)

    first_half = merge_sort(arr[0..mid-1])
    second_half = merge_sort(arr[mid..n-1])
    return merge(first_half, second_half)

function merge(arr1, arr2):
    result = []
    while arr1.length() > 0 and arr2.length() > 0:
        if arr1[0] < arr2[0]:
            print '1' // para debug
            result.append(arr1[0])
            arr1.remove_first()
        else:
            print '2' // para debug
            result.append(arr2[0])
            arr2.remove_first()

    result.append(arr1)
    result.append(arr2)
    return result
```

Uma importante permutação de inteiros de 1 a N foi perdida devido uma falha de disco. Felizmente, a sequência foi ordenada pelo algoritmo acima e a sequência de *debug* de 1s e 2s foi gravada num disco diferente.

Dado o tamanho N da sequência original, e a sequência de *debug*, sua missão é: recupere a sequência original de inteiros.

Entrada
A primeira linha da entrada contém um inteiro T ($5 \leq T \leq 20$), representando a quantidade de casos de teste. Cada caso de teste é composto por duas linhas: a primeira linha contém o tamanho da sequência original, N ($2 \leq N \leq 10000$). A segunda linha contém uma string de 1s e 2s: a sequência de debug produzida pelo *merge sort* durante a ordenação da sequência original.

Saída
Para evitar a impressão da sequência original inteira, imprima um inteiro representando o *"checksum"* da sequência original, calculado pelo seguinte algoritmo:

```
function checksum(arr):
    result = 1
    for i=0 to arr.length()-1:
        result = (31 * result + arr[i]) mod 1000003
    return result
```

Exemplo de entrada

```
5
2
1
2
2
4
12212
5
1122211
10
12122121211122121212
```

Exemplo de saída

```
994
1024
987041
570316
940812
```

Problema G
Loop do Brasil

Arquivo fonte: loop.c, loop.cpp ou loop.java

Alex está empolgado com as manifestações pelo país. Antes de sair para as ruas hoje, ele precisa de sua ajuda: dada uma mochila cheia de pedaços de corda, Alex quer construir o maior loop de corda com cores alternadas, para fazer bonito na manifestação.

A mochila contém P pedaços de corda e cada pedaço é Verde (V) ou Amarelo (A). A ideia de Alex é alternar entre as cores e, devido esse requisito, não necessariamente será possível utilizar todos os pedaços. Se existirem pedaços de uma cor apenas, não será possível fazer o loop e a resposta deve ser 0 (zero).

O tamanho de cada pedaço é dado em centímetros e cada nó do loop consome 1 centímetro do tamanho do loop. Em outras palavras, um nó consome metade de um centímetro de cada pedaço que ele conecta.

Note que pedaços com tamanho 1, se utilizados no loop, serão reduzidos a um par de nós, resultando em um pedaço de tamanho 0. Isto é permitido e o pedaço conta como utilizado.

Entrada

A primeira linha da entrada indica o número de casos de teste, N ($N \leq 50$). Para cada caso de teste seguinte, haverá:

a) uma linha contendo o valor P ($1 \leq P \leq 1000$), o número de pedaços de corda na mochila;

b) uma linha contendo uma lista de P valores, separados por espaço. Cada valor T ($1 \leq T \leq 100$) indica o tamanho do pedaço em centímetros, seguido pelo letra V ou A para indicar a cor do pedaço.

Saída

Para cada caso de teste, imprima o tamanho máximo do loop que pode ser gerado com os dados pedaços.

Exemplo de entrada

```
4
1
5V
4
6A 1V 7A 3V
7
5V 4A 3A 2A 5A 4A 3A
2
20V 20A
```

Exemplo de saída

```
0
13
8
38
```

Problema H
Reserva Florestal

arquivo: reserva.c , reserva.cpp, reserva.java

A Amazônia é uma floresta latifoliada úmida com cerca de sete milhões de quilômetros quadrados. Ela representa mais da metade das florestas tropicais remanescentes no planeta e compreende a maior biodiversidade em uma floresta tropical no mundo.

Tamanha riqueza deve ser preservada, e para isso Fernanda, uma bióloga contratada pelo governo, decidiu analisar as áreas nas quais são encontradas todas as espécies de animais ameaçados de extinção, de forma a escolher a melhor área para a construção de uma reserva florestal. Contudo, o governo afirmou que só será construída uma reserva caso seja possível abranger todos os animais ameaçados de extinção.

Como os cálculos dessas áreas são complexos, Fernanda pediu que seu namorado Caio desenvolvesse um programa para a solução desse problema. Caio é um ótimo programador, mas não gosta de geometria, por isso ele lhe pediu ajuda.

Entrada

A primeira linha contém um inteiro não negativo, N ($0 \leq N \leq 100000$), que indica o número de animais ameaçados de extinção (onde o valor N = 0 indica o final da entrada). Seguem-se N linhas, cada uma contendo quatro números inteiros X, Y, U e V ($-10000 \leq X,Y,U,V \leq 10000$) que descrevem uma região: o par X, Y representa a coordenada do canto superior esquerdo e o par U, V representa a coordenada do canto inferior direito de um retângulo na qual o animal pode ser encontrado. A entrada é finalizada com uma linha contendo o número 0.

Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste n”, onde n é numerado a partir de 1. A segunda linha deve conter as coordenadas do retângulo de interseção encontrado pelo seu programa, no mesmo formato utilizado na entrada. Caso a interseção seja vazia, a segunda linha deve conter a expressão “nenhum”. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Entrada

```
3
0 6 8 1
1 5 6 3
2 4 9 0
3
0 4 4 0
3 1 7 -3
6 4 10 0
0
```

Exemplo de Saída

```
Teste 1
2 4 6 3

Teste 2
nenhum
```

Problema I
Armstrong

arquivo: armstrong.c , armstrong.cpp, armstrong.java

Dado um intervalo de números inteiros positivos, você deve encontrar aqueles números que podem ser escritos como a soma de cada um de seus dígitos individuais elevados ao número de seus dígitos.

Um número *N é conhecido como um número Armstrong de ordem n (n sendo o número de dígitos*) se

$abed..... = a^n + b^n + e^n + d^n + ... = N$

Exemplo:

$370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370$

$9474 = 9^4 + 4^4 + 7^4 + 4^4 = 6561 + 256 + 2401 + 256 = 9474$

Entrada

A entrada contém vários casos de testes. A primeira linha da entrada contem um número T ($1 \leq T \leq 1000$) que representa o número de casos de testes. As próximas T linhas contêm dois números separados por um espaço, A1 e A2, onde A1 ≤ A2 ($0 \leq A1 \leq A2 \leq 2^{31}$) , correspondendo ao inicio e fim do intervalo fechado onde a busca será efetuada.

Saída

Para cada caso de teste, deverá ser gerada uma única linha. A linha conterá os números de Armstrong encontrados separados por um espaço (considere um espaço após o último elemento no final da linha), ou caso não seja encontrado nenhum número, deverá ser escrita a palavra “Nenhum”.

Exemplo de Entrada

```
2
1 9
100 110
```

Exemplo de Saída

```
1 2 3 4 5 6 7 8 9
Nenhum
```

Problema J
Gauss

arquivo: gauss.c , gauss.cpp, gauss.java

Em uma determinada escola, o professor Abacaxi, apelidado carinhosamente por seus alunos, gostava de incentivar seus alunos com algumas brincadeiras relacionadas a matemática. Uma delas era verificar qual o aluno que somava mais rapidamente uma dada sequência de números. Ele verificou que na média os alunos levavam o mesmo tempo com exceção do Joãozinho que fazia rapidamente o problema.

Intrigado com isso, o professor Abacaxi perguntou ao Joãozinho o que ele fazia para somar tão rápido. Então, para surpresa do professor, Joãozinho não quis contar.

Mediante tal situação, o professor Abacaxi propôs um desafio que você ajudará a resolver.
Dado um intervalo fechado de números inteiros positivos, você deve calcular a soma dos números neste intervalo.

Entrada
A entrada contém vários casos de testes. A primeira linha da entrada contem um número N ($1 \leq N \leq 100000$) que representa o número de casos de testes. A segunda linha contém dois números, G1 e G2, onde G1 ≤ G2 (1 ≤ G1 ≤ G2 ≤ 10000000) , correspondendo ao inicio e fim do intervalo.

Saída
Para cada caso de teste, deverá ser gerada uma única linha contendo a soma dos números.

Exemplo de Entrada
```
2
1 70
41 150
```

Exemplo de Saída
```
2485
10505
```