2ª Maratona Doméstica de Programação da UDESC

Prova Principal

Joinville, 13 de junho de 2014

Codinome: Padrão FIFA

Sevidor BOCA:

`http://10.20.107.207/`
(acesso interno)

`http://200.19.107.207/`
(acesso externo)

Organização e Realização:

Claudio Cesar de Sá, Lucas Hermann Negri (coordenação técnica), Ricardo Oliveira (UFPr), Fernando Deeke Sasse, Alexandre Gonçalves Silva, Roberto Silvio Ubertino Rosso Jr., Rogério Eduardo da Silva

Lembretes:

- Aos *javaneiros*: **o nome da classe deve ser o mesmo nome do arquivo a ser submetido**. Ex: **classe petrus**, nome do arquivo **petrus.java**;
- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova;
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão;
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos;
- Utilize o *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos;
- A interface KDE também está disponível nas máquinas Linux, que pode ser utilizada em vez da Unity. Para isto, basta dar *logout*, e selecionar a interface KDE. Usuário e senha: *udesc*

Patrocinador e Agradecimentos

- Linx – Patrocinador oficial do ano de 2014;
- Realização: DCC/UDESC;
- Aos bolsistas pelo empenho nos treinos e no projeto;
- Alguns, muitos outros anônimos.

Conteúdo

1 Problema A: Almirante

`Arquivo: almirante.[c|cpp|java]` `Tempo limite: 5 s`

Ivanna Vega Bomtempo é uma famosa almirante na história boliviana e é bastante conhecida por sua participação na épica batalha naval de 1614 contra o Paraguai. Bomtempo pessoalmente comandou um navio e chefiou navios de guerra aliados durante a batalha naval.

Na época de Bomtempo, a teoria dos grafos recém havia sido inventada e ela a usou para ter vantagem no planejamento de suas batalhas. Pontos de controle no mar eram representados por vértices, e possíveis passagens entre pontos de controle eram representados por arestas direcionadas. Dados dois pontos de controle quaisquer W_1 e W_2 , existe no máximo uma passagem $W_1 \rightarrow W_2$. Cada aresta direcionada é anotada com o número de canhões que precisam ser disparados a fim de permitir de forma segura que um navio siga pelo caminho representado pela aresta, afundando quaisquer navios inimigos que ele encontre pelo caminho.

Uma das táticas de maior sucesso de Bomtempo foi a *Manobra Bomtempo*. Aqui, dois navios começam no mesmo ponto de controle, se separam e abrem caminho lutando com os navios inimigos, se juntando novamente em um ponto de encontro. A manobra dita que dois navios de guerra tomem rotas disjuntas, significando que eles não podem visitar os mesmos pontos de controle (com exceção dos pontos de partida e destino), ou ainda que usem a mesma passagem durante a batalha.

A almirante Bomtempo era conhecida por não gostar de desperdiçar dinheiro; e no período de guerra do século XVII, isso significava disparar o mínimo possível, as caríssimas balas de canhão dos navios de guerra.

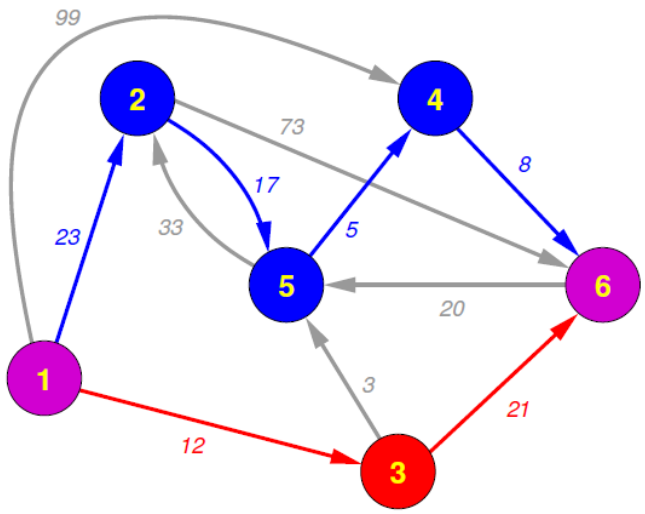

Figura 1: Um momento da manobra de Bomtempo, visualizado como um grafo. Dois navios (vermelho e azul) se movem a partir de um ponto inicial (1) até um ponto de encontro (6). A rota do navio vermelho é $1 \rightarrow 3 \rightarrow 6$ (disparou 33 balas de canhão durante o percurso); a rota do navio azul é $1 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 6$ (disparando 53 balas). No total, 86 balas de canhão foram disparadas durante a manobra. Com exceção para os estados inicial e final, nenhum vértice ou aresta foi visitado pelos dois navios.

Entrada

Para cada caso de teste, a entrada consiste de:

- Uma linha contendo dois inteiros v ($3 \leq v \leq 1000$) e e ($3 \leq e \leq 10000$), que são o número de pontos de controle e de passagens, respectivamente
- A seguir, encontram-se e linhas: para cada passagem, uma linha contendo três inteiros:
  1. a_i ($1 \leq a_i \leq v$), que é o ponto de origem de uma passagem, o qual é representado por um dos pontos de controle;
  2. b_i ($1 \leq b_i \leq v$) e ($a_i \neq b_i$), o ponto destino de uma passagem, o qual é representado por um dos pontos de controle. Todas as passagens são direcionadas (origem $\rightarrow$ destino);
  3. c_i ($1 \leq c_i \leq 100$), representando o número de balas de canhão que precisam ser disparadas durante a travessia da passagem.

O ponto de partida é 1 e o ponto de destino é v. Sempre irão existir duas rotas distintas a partir do ponto 1 até o ponto v.

Saída

Para cada caso de teste, a saída consiste de um único inteiro positivo: a menor quantidade possível de balas de canhão disparadas conjuntamente pelos dois navios até chegarem ao ponto de destino.

Exemplo de Entrada	**Exemplo de Saída**
6 11 1 2 23 1 3 12 1 4 99 2 5 17 2 6 73 3 5 3 3 6 21 4 6 8 5 2 33 5 4 5 6 5 20	86

2 Problema B: Bollywood

`Arquivo: bollywood.[c|cpp|java]` `Tempo limite: 5 s`

Seu amigo Rajesh foi contratado por uma empresa ligada à Bollywood. Rajesh está gostando muito do emprego, já que tem a prerrogativa de assistir em primeira mão os novos lançamentos de Bollywood. Porém, uma coisa o incomoda: seu chefe pediu para que ele contabilize os gastos totais de cada filme produzido, somando o gasto dos itens que compõem cada filme em questão.

Sabendo que você é um bom programador, Rajesh pediu que você escreva um programa que leia as listas de gastos e calcule o gasto total em cada filme produzido.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste corresponde aos gastos de um filme e é iniciado por uma linha contendo o número de itens ($N, 1 \leq N \leq 1.000$) a serem contabilizados. As próximas N linhas contêm cada uma a descrição e o preço do i-ésimo item, sendo que a descrição é uma palavra com 1 até 20 caracteres (pode conter "_", mas não espaços) e o valor é um número inteiro entre 10.000. A entrada termina quando $N = 0$, caso que não deve ser processado.

Saída

Para cada caso de teste, imprima uma linha contendo o gasto total do filme.

Exemplo de Entrada	**Exemplo de Saída**
3 alimentacao 1000 dubles 500 efeitos_especiais 700 2 explosoes 10000 propaganda 5000 0	2200 15000

3 Problema C: Caso das Arestas

`Arquivo: caso_arestas.[c|cpp|java]` `Tempo limite: 5 s`

Em teoria dos grafos, um *casamento* (*matching*) ou *conjunto de arestas independentes* em um grafo $G = (V, E)$ é um conjunto de arestas $M \subseteq E$ tal que não há nenhuma aresta no casamento de M que compartilhe um vétice em comum.

Em 2012, talvez você tenha ouvido falar que o Banco da Suécia concedeu um prêmio em ciências econômicas, informalmente conhecido como prêmio Nobel em Economia para Alvin E. Roth e Lloyd S. Shapley. Dentre outras coisas, o algoritmo que eles criaram encontrava o casamento de certos critérios em grafos bi-partidos. Uma vez que você também já ouviu falar que casamentos em *grafos cíclicos* tem suas aplicações na química (cadeias de carbono), sua cabeça *viajou em círculos*, lembrando a copa do mundo de 2014: a bola é redonda com vários gomos coloridos, os estádios são ovais bem como a distribuição da iluminação dentro destes, etc.

Um grafo cíclico, C_n, $n \geq 3$, é um simples grafo não-direcionado, sobre o conjunto de vértices $\{1, ..., n\}$, com um conjunto de arestas $E(C_n) = \{\{a, b\} \mid |a - b| \equiv 1 \bmod n\}$. Ele é regular-2, e contém n arestas. Os grafos C_3, C_4, C_5, e C_6 são retratados na figura 2.

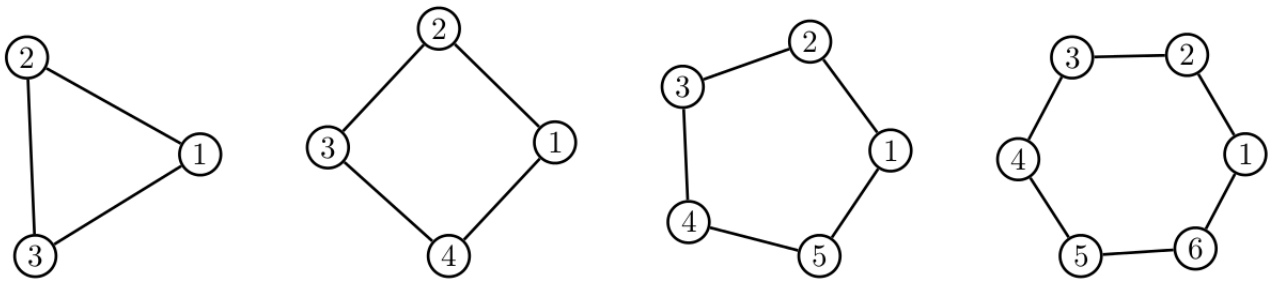

Figura 2: Grafos C_3, C_4, C_5 , e C_6

Assim, seu primeiro passo em direção a fama de se tornar um Prêmio Nobel e estar habilitado em calcular o número de casamentos em um grafo cíclico C_n. Para ilustrar este conceito, veja a figura 3 que apresenta os sete casamentos de um grafo cíclico C_4.

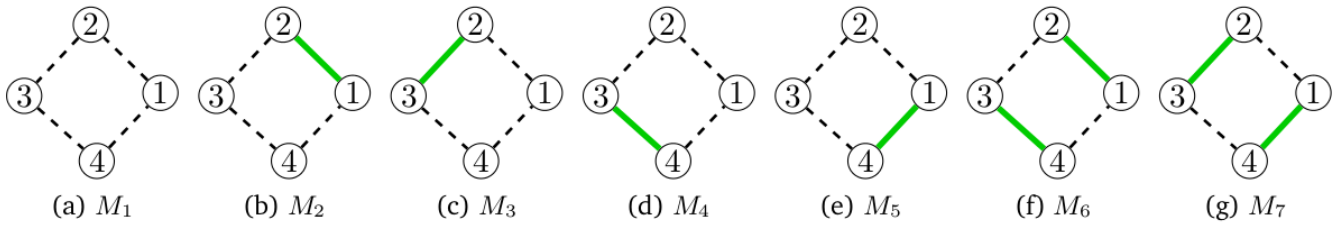

Figura 3: Os casamentos de C_4

Obserse que as arestas que fazem parte destes respectivos casamentos estão pintados de verde, enquanto que as demais arestas são tracejadas e estão fora do casamento. Isto é: $M_1 = \emptyset$, $M_2 = \{\{2, 1\}\}$, $M_3 = \{\{3, 2\}\}$, $M_4 = \{\{4, 3\}\}$, $M_5 = \{\{1, 4\}\}$, $M_6 = \{\{2, 1\}, \{4, 3\}\}$, e $M_7 = \{\{3, 2\}, \{1, 4\}\}$.

Entrada

Para cada caso de teste, você vai ler uma simples linha contendo um inteiro positivo: n, com $3 \leq n \leq 10000$.

Saída

Para caso de teste, imprima um linha contendo o número de casamentos em C_n.

Exemplo de Entrada	**Exemplo de Saída**
3 4 5 100	4 7 11 792070839848372253127

4 Problema D: Relógio Digital

`Arquivo: relogio_digital.[c|cpp|java]` `Tempo limite: 5 s`

Equipamentos eletrônicos comumente usam segmentos de 7 elementos para mostrar números. Destes 7 elementos 3 são horizontais e 4 verticais. Cada segmento pode ser independentemente ligado ou desligado, de modo que números de 0 a 9 podem ser representados, ver figura 4.

Figura 4: Os padrões de 7-segmentos de dígitos decimais.

Para mostrar mais de um dígito ao mesmo tempo, múltiplos elementos de 7 segmentos podem ser colocados um ao lado do outro. Neste caso, cada segmento de cada elemento pode ser independentemente ligado ou desligado. Por exemplo, relógios de alarme digitais tipicamente usam 4 elementos de 7 segmentos para mostrar o tempo: dois dígitos para a hora (00 a 23) e dois dígitos para os minutos (00 a 59).

Figura 5: Alguns exemplos de uso

Suponha que você encontrou um velho relógio despertador, algo como os da figura 5. Infelizmente ele parece não funcionar muito bem: aparentemente ele não mostra sempre a hora correta. Você suspeita que isso é devido à fiação defeituosa no display de 7 segmentos.

Como resultado das falhas alguns dos segmentos podem não estar funcionando: eles nunca acendem, não importa a hora que o relógio tente colocar no seu display. Por outro lado, os segmentos que se acendem algumas vezes parecem estar funcionando corretamente. Portanto, você supõe que cada um dos 28 segmentos do relógio ou está completamente quebrado (nunca acende) ou funcionando perfeitamente (se acendem precisamente como devem).

Problema

Você quer saber que horas são, usando este relógio. Você tem observado o relógio por algum tempo, escrevendo os padrões de dígitos no seu display a cada minuto. (Note que é possível que o display permaneça inalterado por alguns minutos. Quando isso acontece, você simplesmente escreve o mesmo padrão várias vezes).

Sua tarefa é determinar qual o tempo que o relógio está tentando mostrar quando você escreve o primeiro padrão da sequência. A resposta deve ser consistente com todas as observações do display do relógio, para o primeiro minuto e para os subsequentes, sob a suposição de que cada segmento do display ou está completamente quebrado (nunca acende) ou está perfeito.

Podem haver múltiplas respostas possíveis. Neste caso, você deve fazer uma lista de todas as possíveis respostas, em ordem crescente de tempo.

Ainda é possível que o relógio esteja realmente quebrado em alguma outra parte, além dos segmentos defeituosos. Neste caso, pode ocorrer uma possível resposta que seja consistente com todas as informações.

Entrada

Em cada caso, há um linha de entrada contendo os seguintes itens

- Um inteiro positivo N ($1 \leq N \leq 50$), o número de minutos de observação do relógio.
- N itens, cada um representando um padrão de dígitos que foi observado no relógio. Cada padrão é formatado como dois dígitos decimais, seguinte por um símbolo ":", seguido por mais dois dígitos decimais. Os padrões são listados na ordem em que eles são vistos no relógio.

É teoricamente possível que que um segmento defeituoso de 7 dígitos mostre uma forma que não corresponda a qualquer um dos dígitos de 0 a 9. No entando, por alguma razão misteriosa, isso jamais aconteceu durante o tempo em que você observou o relógio.

Saída

Para cada caso-teste, dê uma linha de saída.

Se há ao menos uma resposta possível, imprima uma lista de todas as possíveis respostas separadas por espaços. Cada possível resposta deve ser um tempo de relógio de 24 horas válido, formatado em dois dígitos (00 a 23), seguido por uma caractere ":", seguido por dois dígitos (00 a 59). A lista de possíveis respostas deve ser impressa em ordem crescente de tempo.

Se não há uma resposta possível, imprima a palavra '`NENHUM`'.

Exemplo de Entrada	**Exemplo de Saída**

`1 88:88`
`2 23:25 23:26`
`3 71:57 71:57 71:07`
`1 00:00`
	`NENHUM`
	`23:25`
	`00:58 03:58 07:58 08:58`
	`00:00 00:08 08:00 08:08`

5 Problema E: Escolha de Bares

`Arquivo: escolha.[c|cpp|java]` `Tempo limite: 5 s`

Desde os meados do século XX, sociólogos têm estudado o fascinante tema do comportamento relacionado ao consumo de chopp, entre a população de estudantes das universidades públicas brasileiras. Um dos maiores sucessos foi o desenvolvimento de um modelo muito preciso que descreve o intrincado ritual da *escolha de bares*, que pode ser observado nos finais de tarde (logo após o final das aulas).

Estudantes escolhem os bares através de votação. O que torna isso interessante do ponto de vista sociológico, é a combinação de comportamentos dominantes por parte de alguns estudantes, com comportamentos controlados pela pressão social por parte dos companheiros combinado ainda com aleatoriedade por parte dos restantes.

Especificamente, o modelo descreve o ritual de escolha de bares da seguinte forma:

- Os estudantes dominantes, os quais possuem uma predominante preferência por determinados bares, irão anunciar (gritar) seus votos. Vários estudantes dominantes poderão inclusive escolher o mesmo bar.
- A seguir, os demais estudantes votam, um por um. Estes estudantes não-dominantes votam probabilisticamente e estão sujeitos à pressão de seus companheiros; a probabilidade deles votarem para um bar específico é igual ao número de votos já recebidos por aquele bar até o momento, dividido pelo número de votos já computados até o momento.
- Finalmente, quando todos os votos já estiverem sido computados, o bar com a maior quantidade de votos é o escolhido. Se vários bares obtiverem a mesma quantidade de votos, um deles é escolhido aleatoriamente, com igual probabilidade para todos.

Por exemplo, em um momento em particular com sete estudantes, os cinco estudantes dominantes iniciam a votação declarando seus votos para três bares diferentes. Os três estudantes dominantes escolheram o primeiro bar; o quarto dominante escolheu o segundo bar e o último estudante dominante votou no terceiro bar. Disto resulta a seguinte contagem inicial da votação $(3, 1, 1)$.

Depois disso, os dois estudantes restantes (não-dominantes) começam a votar, um após o outro. O primeiro deles irá votar ou no primeiro bar com probabilidade de $\frac{3}{5}$, ou no segundo bar (com probabilidade de $\frac{1}{5}$), ou ainda no terceiro bar com probabilidade de $\frac{1}{5}$. Hipoteticamente digamos que ele escolheu o bar número três, e agora a votação está em $(3, 1, 2)$.

Finalmente, a última estudante irá escolher ou o bar número um (probabilidade de $\frac{3}{6} = \frac{1}{2}$), ou o segundo bar com probabilidade de $\frac{1}{6}$, ou ainda o terceiro com probabilidade de $\frac{2}{6} = \frac{1}{3}$. Ela também escolheu o bar número três.

Isso deixa a votação empatada $(3, 1, 3)$: entre os bares um e três. Este empate é resolvido ao se jogar uma moeda (probabilidade $\frac{1}{2}$). E por fim, o bar número um é o escolhido para eles irem beber naquela noite após as aulas.

Problema

Você é um sociólogo observando este ritual descrito acima. Após os estudantes dominantes terem votado, você gostaria de saber quais as chances de cada bar ser o escolhido daquela noite.

Entrada

Para cada caso de teste, a entrada consiste de duas linhas:

- uma linha contendo dois números inteiros positivos: n denotando o número de bares da votação ($n \leq 5$); e k, o número total de estudantes votantes, incluindo-se aqui os dominantes e os não-dominantes ($k \leq 50$).
- uma linha contendo n números inteiros positivos $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, denotando a contagem de votos logo após todos os estudantes dominantes terem votado.

E ainda, é garantido que $k \geq \Sigma_{i=1}^{n} \alpha_i$.

Saída

Para cada caso de teste, escreva n linhas de saída contendo a probabilidade do *i-ésimo* bar ser o escolhido ($1 \leq i \leq n$), ordenados de forma ascendente pelo valor i.

Cada linha deverá ser da forma: `bar i:  percentagem %`, onde *percentagem* é um número real com duas casas decimais.

Um espaço separa o número percentual e o subsequente sinal %.

Exemplo de Entrada	**Exemplo de Saída**

`3 7`
`3 1 1`
	`bar 1: 93.33 %`
	`bar 2: 3.33 %`
	`bar 3: 3.33 %`

6 Problema F: Fibra Ótica

`Arquivo: fibra.[c|cpp|java]` `Tempo limite: 5 s`

As fibras óticas são largamente utilizadas na sociedade moderna. Seu principal uso está na comunicação, sendo a base para a internet. Além disto, as fibras óticas também podem ser utilizadas como transdutores.

Grande parte do progresso que tornou as fibras óticas adequadas para a comunicação a longas distâncias foi realizado por Sir Charles Kuen Kao, cientista chinês que ganhou o Prêmio Nobel de física em 2009 justamente por suas contribuições na área de fibras óticas. Sir Charles Kao decidiu não patentear as suas descobertas para propiciar um avanço rápido da nova tecnologia, o que permitiu o avanço que todos usufruímos hoje.

NiHao é um estudante de computação e está fazendo iniciação científica em um laboratório de ótica. O orientador de NiHao pediu para que ele escreva um programa que calcule o atraso aproximado em um sinal que se propaga em uma fibra ótica monomodo, seguindo a seguinte equação aproximada:

$$D = \frac{Ln_{eff}}{c}, \tag{1}$$

onde D é o atraso, L é o comprimento da fibra, n_{eff} é o índice de refração efetivo do modo de propagação da fibra e c é a velocidade da luz no vácuo, aproximada aqui por $3 \times 10^8 m/s$.

NiHao está muito ocupado com as disciplinas do semestre, e pediu para que você o ajude a escrever o programa que seu orientador requisitou!

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é descrito em uma linha composta por dois números reais, separados por um espaço, sendo que o primeiro valor corresponde ao índice de refração efetivo $n_{eff}, 1 \leq n_{eff} \leq 2$ e o segundo ao comprimento da fibra $L, 1 \leq L \leq 1000$, em quilômetros.

Saída

Para cada caso de teste, imprima uma linha contendo o tempo de atraso de transmissão em nanossegundos, arredondado para exatamente duas casas decimais. Adicione o sufixo *ns* como visto no exemplo de saída.

Exemplo de Entrada	**Exemplo de Saída**

`1 1`
`2 1`
`1 10`
	`3.33 ns`
	`6.67 ns`
	`33.33 ns`

7 Problema G: The Castle Guards

`Arquivo: guards.[c|cpp|java]` `Tempo limite: 5 s`

The royal castles in Molvania follow the design of king Sane, first of his dynasty. He ruled by divide and conquer. Therefore, all castles are built according to a hierarchical pattern based on interconnected buildings. A building consists of *halls* and *corridors* that connect halls.

Initially, a castle consists of only one building (the *main building*). When its population grows, the castle is extended as follows: A new peripheral building is constructed, attached to one of the existing buildings. Like any other building, the new building also consists of halls and corridors. An additional corridor is created to connect a hall in the existing building to a hall in the new building. That corridor is the only way to access the new building.

The number of halls in a building is at most 10.

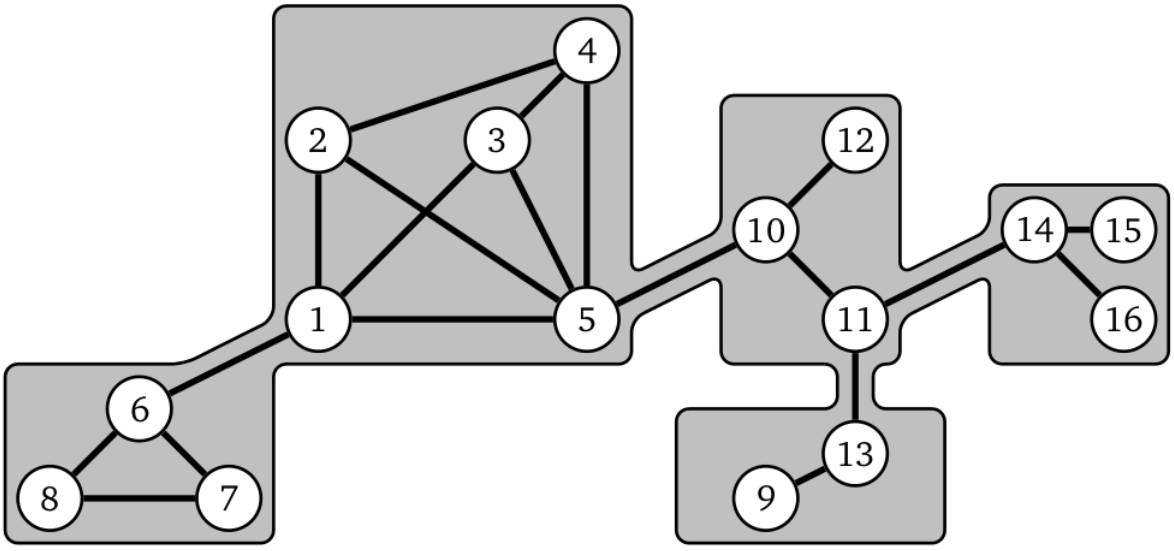

Figura 6: The castle layout of the example provided above

In times of turmoil, the king monitors all corridors by strategically placing guards in halls. He asks you to determine the least number of guards required to monitor all corridors in the castle (as he wants to keep his personal guard as large as possible). Note that since the last fire, there are no doors in the castle, so we can safely assume that a guard placed in a hall can monitor all connecting corridors.

Input

The input contains a number of castle descriptions. Within a castle, each hall is identified by a unique number between 1 and 10000. Each castle is recursively defined, starting with a description of the main building:

1. A line containing three integers, representing the number of halls in this building ($2 \leq n \leq 10$), the number of corridors in this building ($1 \leq m \leq 45$), and the number of peripheral buildings that were later attached to this building ($0 \leq w \leq 10$).

2. For each of the m corridors:

- A line containing two integers (each ≤ 10000), representing the two halls connected by this corridor. Both halls are located inside the current building.

3. For each of the w peripheral buildings:

   - A line containing two integers, describing the corridor that leads to this peripheral building. The first integer represents a hall in the current building, while the second integer represents a hall in the peripheral building.
   - The structure of the peripheral building and any newer buildings that were later attached to it, described by repeating rules 1 to 3.

The castle is fully connected: any hall is directly or indirectly reachable from any other hall. Corridors with the same start and end hall do not exist, and for every two halls there is at most one corridor between them.

Output

For each castle, print a single line containing a positive integer: the minimum number of guards to place in halls such that all corridors in the castle are monitored.

Example

Input	Output
5 8 2 1 2 2 4 3 4 1 3 1 5 2 5 3 5 4 5 1 6 3 3 0 6 7 7 8 8 6 5 10 3 2 2 10 11 10 12 11 13 2 1 0 13 9 11 14 3 2 0 14 15 14 16	8

8 Problema I: Ídolos

`Arquivo: idolos.[c|cpp|java]` `Tempo limite: 20 s`

Pedrão está competindo na fase preliminar de um show de talentos chamado Marathons Business–Show, e quem avançar para a próxima rodada, vai competir junto com o seu ídolo lá em São Paulo, na capital. Neste show de talentos, cada um dos participantes tem 10 minutos para impressionar os juízes. Depois de todos os candidatos terem se apresentado, cada um dos juízes vai dar dois votos distintos. Um voto pode ser a favor de um candidato (ou seja, este candidato deve avançar a fase seguinte) ou contra um candidato (o que significa este candidato não deve avançar).

O número de competidores/candidatos que avançam para a próxima rodada não é conhecido antecipadamente; se houver apenas candidatos muito ruins, então é possível que ninguém vá avançar, ou se todo mundo é incrível, então todo mundo pode avançar.

Pedrão tem medo de que os juízes não apreciem os seus talentos de dança gauchesca, e quer usar *seu outro talento* para avançar para a próxima rodada: *hacking* (os juízes descuidados, não sabiam que Pedrão era *Ninja*). Outro descuido, e Pedrão teve acesso ao sistema do júri, e Pedrão é capaz de substituir o processo de contagem normal de votos, e pré-selecionar exatamente quais competidores devem avançar para a próxima rodada. O único problema é que ele tem que ter o cuidado para não levantar suspeitas.

Cada juiz espera que pelo menos um de seus próprios dois votos corresponda ao resultado do concurso. Se o resultado contrariar os dois votos de um dado juiz, o juiz vai levantar suspeita de fraude. Por exemplo, suponha que o juiz ClaudiusNP deu um voto a favor de Pedrita e um voto contra Suzan. Se Suzan avançar e Pedrita não, o juiz ClaudiusNP ficará em dúvida e poderá descobrir a adulteração no sistema feita por Pedrão.

Uma vez que os talentos de programação de Pedrão são limitados (caso contrário, ele não precisaria *hackear* o sistema de contagem de votos), ele precisa de você para fazer um programa que descubra se existe um conjunto de candidatos, no qual ele se inclui, que ele possa optar por avançar para a próxima rodada, *hacking* o sistema do juiz, de tal modo que não alarme nenhum dos juizes.

Entrada

Para cada caso de teste, a entrada é como se segue:

- Uma linha contendo dois inteiros positivos: o número de competidores n ($2 \leq n < 1000$) e o número de juízes m ($1 \leq m < 2000$).
- m linhas contendo os votos de cada juiz. Cada uma destas linhas contém dois inteiros: os números a ($1 \leq |a| \leq n$), e b ($1 \leq |b| \leq n$), os dois votos deste juiz ($|a| \neq |b|$).
  - Um voto $x < 0$ significa que o voto é contra o avanço do competidor $|x|$.
  - Um voto $x > 0$ significa que o voto é a favor do competidor $|x|$.

Os competidores são numerados de $1...n$. Pedrão é concorrente 1.

Saída

Para cada caso de teste, imprima uma linha de saída contendo a palavra "`SIM`" se houver um conjunto de competidores que avança para a próxima fase, o qual inclui Pedrão, e não faça

alarde a nenhum dos juízes. Se não existe tal conjunto de competidores, a linha deve conter "NAO"(é "NAO" sem o "~" na letra A).

Exemplo de Entrada	**Exemplo de Saída**
4 3 1 2 -2 -3 2 4 2 4 1 2 1 -2 -1 2 -1 -2 2 4 -1 2 1 2 -1 -2 1 -2	SIM NAO NAO

9 Problema J: Juntando as Peças

`Arquivo: juntando.[c|cpp|java]` `Tempo limite: 5 s`

Tyranno e SaurusRex estão construindo um robô para uma disciplina do curso e descobriram que eles precisam encaixar duas peças de Lego em uma abertura.

A abertura tem x centímetros de largura e a soma dos comprimentos das duas peças tem de ser precisamente igual à largura da abertura, ou então o robô vai quebrar/bugar durante a demonstração do projeto, com consequências catastróficas nas notas destes dois estudantes.

Felizmente, Tyranno e SaurusRex foram capazes de infiltrar-se no laboratório de física, tarde da noite para medir os comprimentos das peças de Lego restantes, com muita precisão. Agora eles só precisam selecionar duas peças que se encaixam perfeitamente na abertura.

Entrada

Para cada caso de teste, você tem:

- uma linha contendo um inteiro positivo: x, que indica a largura da abertura em centímetros, com $1 \leq x \leq 20$.
- uma linha que contém um número inteiro não-negativo n, que define o número peças de Lego acessíveis a Tyranno e SaurusRex, com $0 \leq n \leq 1000000$.
- n linhas contendo inteiros positivos, definindo os comprimentos das peças de Lego em nanometros. Tyranno e SaurusRex disseram que nenhuma peça de Lego é maior *que* 10 centímetros, ou 100000000 nanômetros.

  A entrada termina com o final de arquivo (EOF).

Saída

Para cada caso de teste, escreva uma linha que contendo a palavra "`PERIGO`"se não existirem duas peças de Lego que se ajustem precisamente na abertura, ou "`SIM` l_1 l_2", com $l_1 \leq l_2$, se essas duas peças de comprimentos l_1 e l_2 existirem.

No caso de existirem múltiplas soluções, uma solução de maximizada da diferença de tamanho $\mid l_1 - l_2 \mid$ deve ser impressa.

Exemplo de Entrada	**Exemplo de Saída**
1 4 9999998 1 2 9999999 1 0	SIM 1 9999999 PERIGO

10 Problema K: Mensagens Cifradas

`Arquivo: cifras.[c|cpp|java]` `Tempo limite: 5 s`

Alice e Bob adoram trocar mensagens, mas não gostam quando outras pessoas leem as mensagens deles. Seu amigo Charles está bastante interessado no que Alice e Bob escrevem um para o outro, mas como eles agora estão criptografando suas mensagens, ele não consegue mais lê-las, mas mesmo assim ele continua sendo capaz de interceptar as mensagens criptografadas (chamadas de "*texto cifrado*").

Recentemente, não apenas Charles conseguiu interceptar uma mensagem cifrada, mas também conseguiu acessar a mensagem original que gerou a cifra (chamada de "*texto original*"). Para ajudá-lo a descriptar futuras mensagens entre Alice e Bob, foi solicitado a você que escreva um programa que o ajude a monitorar a chave de encriptação.

Charles te disse que ele sabe que Alice e Bob estão usando um encriptador por bloco de transposição. Isto significa que para cada bloco de k caracteres na mensagem, estes caracteres são reordenados em uma das $k!$ possíveis permutações durante a encriptação. A chave correspondente à permutação mostrada na Figura 7 seria alguma representação de $(123456) \rightarrow (541362)$. Como a sua tarefa se resume a apenas contar as (possíveis) chaves, a real chave de encriptação não é relevante.

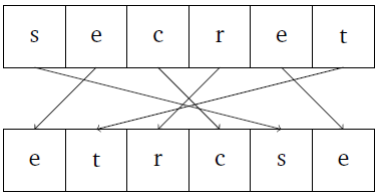

Figura 7: Exemplo de Transposição

Felizmente, Charles sabe o tamanho do bloco k, e também sabe que o *texto original* e o *texto cifrado* que ele interceptou, consistem de um ou mais blocos de tamanho k (isto é, não existem blocos incompletos) os quais foram todos encriptados usando a mesma chave.

Dado o texto original M e o texto cifrado C que Charles interceptou, seu programa deve computar o número de possíveis chaves de encriptação.

Entrada

Para cada caso de teste, a entrada contém tres linhas:

- Uma linha contendo um número inteiro positivo k, que é o tamanho do bloco ($k \geq 1$)
- Uma linha contendo M, o texto original ($1 \leq \mid M \mid \leq 100, \mid M \mid$ é múltiplo de k)
- Uma linha contendo C, o texto cifrado ($\mid C \mid = \mid M \mid$)

Ambos os textos (original e cifrado) consistem apenas de letras minúsculas.

Saída

Para cada caso de teste, imprima uma linha contendo o número de possíveis chaves de encriptação de tamanho k. Este número não será maior do que $2^{63} - 1$. Caso seja impossível obter-se M a partir de C por uma cifra de transposição de bloco de tamanho k, imprima '`0`' (o número zero).

Exemplo de Entrada	**Exemplo de Saída**
4 treewood ertedowo 1 nwerc ncrew 6 secret etrcse 1 impossibru youdontsay	1 0 2 0