OBI2013

Caderno de Tarefas

Modalidade **Programação** • Nível **2**, Fase **2**

6 de abril de 2013

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python devem ser arquivos com sufixo *.py*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em Pascal: *readln, read, writeln, write*;
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,print,write*
- Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

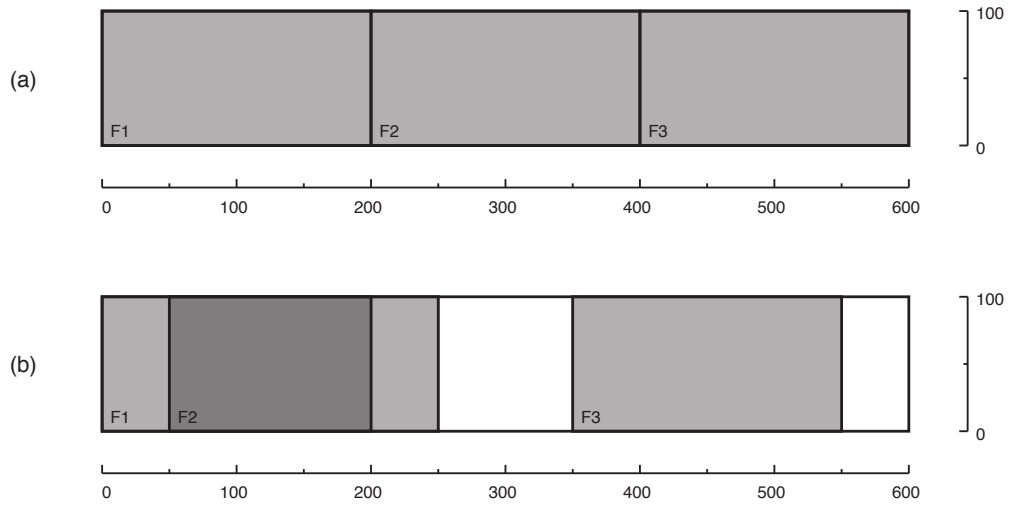
Janela

Nome do arquivo fonte: `janela.c`, `janela.cpp`, `janela.pas`, `janela.java`, *ou* `janela.py`

A sala de aulas utilizada para os cursos da OBI tem uma grande janela, composta de três folhas de vidro. A janela tem um metro de altura por seis metros de comprimento. Cada folha da janela tem um metro de altura e dois metros de comprimento. As folhas deslizam sobre trilhos, ao longo do comprimento da janela, de forma que é possível controlar a abertura da janela, para circulação de ar.

Dadas as posições das três folhas da janela, deseja-se determinar qual a área da janela que está aberta, em centímetros quadrados.

A figura abaixo ilustra duas configurações das folhas da janela. Na figura, os cantos inferiores esquerdos de cada folha são indicados por `F1`,`F2` e `F3`. Na configuração (a) a janela está totalmente fechada, e portanto o total da área aberta é igual a zero. Na configuração (b) há duas aberturas, e o total de área aberta é igual a $(100 \times 100) + (50 \times 100) = 15.000\ cm^2$.



Dadas as posições das três folhas da janela, escreva um programa que calcule a área da janela que está aberta, em centímetros quadrados.

Entrada

A primeira e única linha da entrada contém três inteiros F_1, F_2 e F_3, indicando as posições das três folhas. A posição de cada folha é dada pela distância, em centímetros, da extremidade esquerda da janela até a extremidade esquerda da folha.

Saída

Seu programa deve imprimir uma única linha, contendo um único inteiro, a área aberta da janela em centímetros quadrados.

Restrições

- $0 \leq F_1, F_2, F_3 \leq 400$.

Exemplos

Entrada	**Saída**
0 200 400	0

Entrada	**Saída**
0 50 350	15000

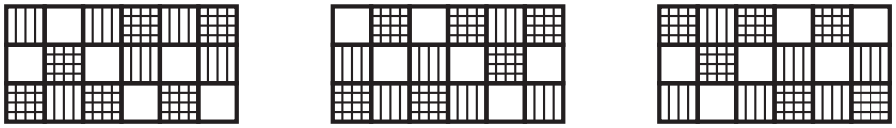
Entrada	**Saída**
344 344 344	40000

Cachecol da Vovó Vitória

Nome do arquivo fonte: `cachecol.c`, `cachecol.cpp`, `cachecol.pas`, `cachecol.java`, *ou* `cachecol.py`

Vovó Vitória possui muitos netinhos; como toda boa avó, ela se preocupa constantemente com a saúde de seus netos, e quer garantir que eles estejam sempre bem agasalhados o tempo todo.

Vovó Vitória dispõe de um saco com vários retalhos quadrados de mesmo tamanho, em três cores diferentes, e quer usá-los para costurar cachecóis para seus netos. Ela quer que cada cachecol tenha três retalhos de largura por N de comprimento e, além disso, retalhos adjacentes devem ter cores diferentes. Por exemplo, a figura abaixo mostra três cachecóis que Vovó Vitória pode costurar.



Vovó Vitória tem muitos netos, e quer fazer um cachecol diferente para cada um deles, mas ela não sabe de quantas formas ela pode arrumar os retalhos para formar cachecóis diferentes. Por isso, ela pediu para você escrever um programa que determina quantos cachecóis diferentes ela pode costurar.

Entrada

A entrada consiste de uma única linha contendo um único inteiro N, indicando o número de retalhos no comprimento do cachecol.

Saída

Imprima uma única linha contendo um único número inteiro, indicando o número de cachecóis distintos que a Vovó Vitória pode costurar. Como este número pode ser muito grande, imprima o resto que este número deixa quando dividido por 1.000.000.007 ($10^9 + 7$).

Restrições

- $1 \leq N \leq 10^{18}$

Exemplos

Entrada	**Saída**
1	12

Entrada	**Saída**
2	54

Entrada	**Saída**
4	1122

Famílias de Troia

Nome do arquivo fonte: `troia.c`, `troia.cpp`, `troia.pas`, `troia.java`, *ou* `troia.py`

A Guerra de Troia pode ter sido um grande conflito bélico entre gregos e troianos, possivelmente ocorrido entre 1300 a.C. e 1200 a.C. (fim da Idade do Bronze no Mediterrâneo). Recentemente foram encontradas inscrições numa caverna a respeito de sobreviventes. Após um trabalho árduo, arqueólogos descobritam que as incrições descreviam relações de parentesco numa certa população. Cada item da inscrição indicavam duas pessoas que pertenciam a uma mesma família. Seu problema é determinar quantas famílias distintas existem.

Entrada

O arquivo de entrada consiste de $M + 1$ linhas. A primeira linha do arquivo de entrada contém um inteiro positivo N, que indica o número de elementos da comunidade, numerados de 1 a N. As demais M linhas do arquivo de entrada contêm, cada uma, dois inteiros. Cada inteiro identifica um elemento da comunidade. Cada linha indica que os dois indivíduos pertencem a uma mesma família.

Saída

A saída deve conter apenas uma linha contendo um único inteiro, que é o número de famílias.

Restrições

- $1 \leq N \leq 5 \times 10^4$
- $1 \leq M \leq 10^5$

Exemplos

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

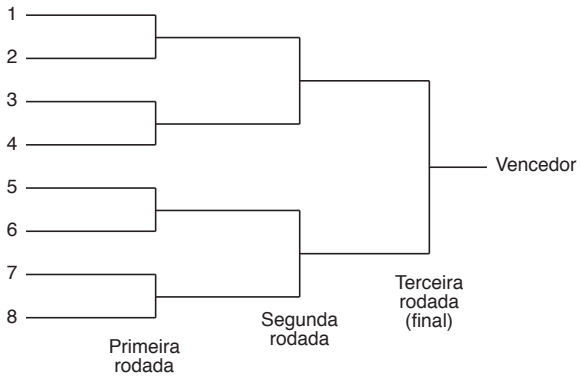
Entrada	**Saída**
8 10 1 2 2 3 3 6 6 5 5 4 4 3 6 7 7 8 8 1 1 5	1

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

Torneio

Nome do arquivo fonte: `torneio.c`, `torneio.cpp`, `torneio.pas`, `torneio.java`, *ou* `torneio.py`

Juquinha foi convidado para participar do prestigiado torneio de tênis de Rolando Barros, na Nlogônia. O torneio é composto de N rodadas no estilo mata-mata: todo jogador que perde uma partida é eliminado do torneio, e o vencedor desta partida avança para a próxima rodada. Como o número de jogadores ativos cai pela metade a cada rodada, é necessário que o número de jogadores participantes seja uma potência de 2.



Os jogadores são inicialmente dispostos na chave por sorteio. Em uma disposição é atribuido a cada jogador um valor de 1 a 2^N, que corresponde a sua posição na chave do torneio. Jogadores vencedores avançam para a direita da chave, e disputam com o vencedor da sub-chave vizinha. Na imagem acima, caso os jogadores das posições 1 e 3 vençam suas partidas na primeira rodada, estes se enfrentarão na segunda rodada.

Juquinha não quer perder a chance de tornar-se um jogador mundialmente famoso, e para isso contratou você para ajudá-lo em suas análises estatísticas. Ele atribuiu a cada jogador um coeficiente de habilidade H_i, e sabe que se dois jogadores disputarem uma partida, aquele com maior coeficiente de habilidade certamente será o vencedor. Seu papel é calcular quantas disposições iniciais dos jogadores forçam Juquinha perder na K-ésima rodada (ou vencer o torneio, caso $K = N + 1$). Duas disposições são consideradas distintas se para algum jogador foi atribuido um valor diferente nas duas disposições.

Entrada

A primeira linha contém dois inteiros N e K. Cada uma das próximas 2^N linhas seguintes contêm um único inteiro representando o coeficiente de habilidade de um jogador. O coeficiente de Juquinha é representado pelo primeiro desses inteiros.

Saída

Seu programa deve imprimir uma única linha contendo um único inteiro indicando o número de disposições iniciais que forçam Juquinha a perder na K-ésima rodada (ou ganhar o torneio, se $K = N + 1$). Como este número pode ser muito grande, imprima o resto que este número deixa quando dividido por 1.000.000.007 ($10^9 + 7$).

Restrições

- $1 \leq N \leq 16$
- $1 \leq K \leq N + 1$
- $0 \leq$ coeficiente de habilidade de um jogador $\leq 10^9$
- Não existem dois jogadores diferentes com a mesma habilidade.

Informações sobre a pontuação

- Em um conjunto de casos de testes que totaliza 30 pontos, $N \leq 3$

Exemplos

Entrada	**Saída**
2 2 3 4 2 1	16

Entrada	**Saída**
1 2 7 5	2