UNIVERSIDADE FEDERAL DE ITAJUBÁ

ECOX21 - MARATONA DE PROGRAMAÇÃO I

Profs. Edmilson Marmo e Luiz Olmes

Warmup: Aula 10

06/11/2024

Regras

1. Há 5 problemas que devem ser resolvidos no tempo estipulado.
2. Os dados de entrada devem ser lidos a partir da entrada padrão.
3. Os dados de saída devem ser escritos na saída padrão.
4. Quando uma linha contém vários valores, eles estarão separados por um único espaço. Não há espaços extras na entrada/saída. Não há linhas em branco na entrada.
5. A entrada e a saída usam o alfabeto latino. Não haverá letras com til, acentos, tremas ou outros sinais diacríticos.
6. Todas as linhas da entrada e da saída, incluindo a última contêm o caractere de fim de linha.
7. O código fonte de cada solução deve ser enviado pelo Boca: `boca.unifei.edu.br`

Ambiente de Testes

A correção das soluções enviadas é realizada no sistema operacional Red Hat Enterprise Linux, versão 8.6 (Ootpa), usando os seguintes compiladores/interpretadores:

```
C: gcc versão 8.5.0 20210514 (Red Hat 8.5.0-10)
C++: g++ versão 8.5.0 20210514 (Red Hat 8.5.0-10)
Java: openjdk versão 1.8.0_342
Python: python3 versão 3.6.8
```

Limites

```
Memória (C, C++, Python): 1GB
Memória (Java): 1GB + 100MB stack
Tamanho máximo do código fonte: 100KB
Tamanho máximo do arquivo executável: 1MB
```

Códigos que extrapolem os limites permitidos receberão *Runtime Error* como resposta.

Comandos de Compilação

```
C: gcc -g -O2 -std=gnu11 -static -lm
C++: g++ -g -O2 -std=gnu++17 -static -lm
Java: javac
```

Códigos C/C++

- O programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- Para entradas grandes, objetos `iostream` podem ser lentos, devido a questões de sincronização de buffer com a biblioteca `stdio`. Recomenda-se desabilitar este mecanismo de sincronização em programas que empreagam `std::cin` e `std::cout` através dos seguintes comandos:

  ```
  std::ios::sync_with_stdio(0);
  std::cin.tie(0);
  ```

  Note que, neste caso, deve-se evitar usar `printf` e `scanf` no mesmo programa, pois a separação dos buffers pode levar a resultados inesperados.

Códigos Java

- O programa não deve estar encapsulado em um `package`.
- Para cada problema, o arquivo `.java` e a `public class` devem ter o mesmo nome `basename` mostrado no Boca.
- Comando de execução: `java -Xms1024m -Xmx1024m -Xss100m`

Códigos Python

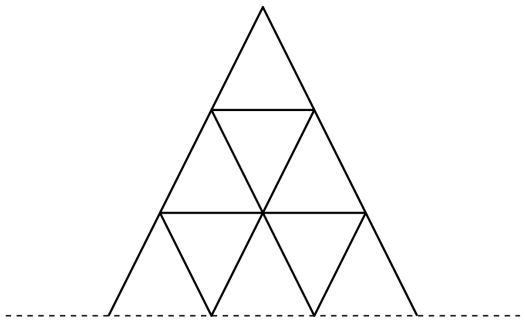
- Apenas Python 3 é suportado. Python 3 não é compatível com Python 2.
- **Atenção**: não é garantido que soluções escritas em Python executarão dentro do tempo limite especificado para cada problema.
- Comando de execução: `python3`

Problema $\mathcal{A}$

HOUSE OF CARDS

Timelimit: 1s

Enquanto seu pai assiste ao seriado “House of Cards”, Luís resolveu construir sua “torre de cartas”: ele começa a estrutura alinhando N pares de cartas, onde cada par forma um V invertido. Em seguida, ele coloca uma carta, no sentido horizontal, apoiada nos topos de cada par de Vs invertidos adjacentes. Em cima das cartas horizontais ele constrói um novo nível da torre, com um par de cartas a menos do que o nível anterior, e prossegue até chegar ao topo, que contém um único par de cartas. A figura seguinte ilustra uma torre com três níveis, que foi construída usando exatamente 15 cartas (a linha pontilhada representa o chão, que não tem cartas):



Luís deseja construir a maior torre de cartas possível, tendo à sua disposição C cartas. Ajude o garoto escrevendo um programa que receba o valor de C e compute o maior valor possível para N de modo que ele possa construir uma torre completa. Por exemplo, se ele tem $C = 10$ cartas, ele conseguirá montar uma torre com $N = 2$ pares de cartas na base (usando um total de 7 cartas, sobrando portanto 3 cartas), mas ele não tem o suficiente para uma torre com $N = 3$ (a qual, conforme visto na figura, demanda um total de 15 cartas).

Entrada

A entrada contém vários casos de teste. Cada caso de teste é representado por uma única linha, contendo o valor de C ($2 \leq C \leq 10^{18}$). A entrada termina com final de arquivo (`EOF`).

Saída

Para cada caso de teste imprima, em uma linha, o maior valor possível para N de modo que Luís possa construir uma torre completa.

Exemplos

Entrada	*Saída*
10 15 1234567890	2 3 28688

Problema $\mathcal{B}$

FIBRA ÓTICA

Timelimit: 1s

A Unifei está se modernizando cada vez mais. A mais recente conquista de nossa universidade foi a aprovação de um projeto que melhora o seu link de comunicação. Com isso, a velocidade da internet no campus tende a aumentar significativamente, pois todos os institutos e departamentos serão interconectados com fibra ótica. Cada par de localidades conectadas por uma fibra é considerado um ramo da rede da Unifei. Inicialmente, o pessoal da DTI especificou que haveria um ramo entre quase todas as localidades da Unifei, justamente para gerar redundância nos caminhos de transmissão e evitar acúmulo de tráfego de dados em certas conexões. Porém, o pessoal da Pró-Reitoria de Administração não concordou. O custo para conectar todas as localidades pode ultrapassar o orçamento disponível. Assim, o projeto deve considerar conexões indiretas entre os institutos, isto é, a conexão pode passar por ramos intermediários. Dessa maneira, a Reitoria conta com a sua ajuda para escrever um programa que calcule o menor custo da rede de fibras de forma que haja conexão entre todas as localidades.

Entrada

A primeira linha da entrada contém dois inteiros N e M, indicando, respectivamente, a quantidade de localidades e a quantidade de ramos existentes no projeto ($1 \leq N \leq 500; 1 \leq M \leq 10^7$). Cada localidade é numerada de 1 a N. As M linhas seguintes contém, cada uma, três inteiros U, V e C, indicando que a implantação de um ramo entre as localidades U e V tem custo C ($1 \leq C \leq 500$).

Saída

Seu programa deve imprimir uma linha contendo o menor custo para implantação da rede.

Exemplos

Entrada	*Saída*
4 6 1 2 1 1 3 10 1 4 1 2 3 1 2 4 10 3 4 1	3

Entrada	*Saída*
5 6 1 2 15 1 3 10 2 3 1 3 4 3 2 4 5 4 5 20	34

Problema C

VIRA CARTAS

Timelimit: 1s

José possui um baralho com N cartas. Cada carta possui dois números, um em cada face. Estas cartas são colocadas em uma mesa, com um dos lados virados para cima. Dados dois valores inteiros i e j, com $i \leq j$, uma operação `troca(i, j)` consiste em virar todas as cartas da posição i até a posição j, inclusive. Por exemplo, seja a sequência:

```
valor da face virada para cima:  31 02 45 03 08 01 32 10 04 27 12 07 07 09 63 47
valor da face virada para baixo: 01 12 06 04 97 02 87 10 03 09 55 56 11 90 03 08
```

A operação `troca(5, 11)` resultaria na sequência abaixo:

```
valor da face virada para cima:  31 02 45 03 97 02 87 10 03 09 55 07 07 09 63 47
valor da face virada para baixo: 01 12 06 04 08 01 32 10 04 27 12 56 11 90 03 08
```

O problema de José é que a sequência de cartas pode ser muito grande e podem ser feitas muitas operações de troca. Ele precisa saber a sequência dos números que estarão virados para cima ao final de todas as operações. Você pode ajudá-lo?

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois inteiros N e T ($1 \leq N, T \leq 10^5$) indicando, respectivamente, a quantidade de cartas e a quantidade de operações de troca. A segunda linha contém N inteiros v ($0 \leq v \leq 10^9$), indicando os números virados para cima inicialmente. A terceira linha contém N inteiros v, indicando os números virados para baixo inicialmente. As T linhas seguintes contém, cada uma, dois inteiros I e J, indicando os limites de uma operação de troca ($1 \leq I \leq J \leq N$). Valores $N = T = 0$ indicam o fim da entrada.

Saída

Para cada caso de teste, imprima uma linha contendo N inteiros separados por um único espaço, representando os números que estarão virados para cima após todas as operações.

Exemplos

Entrada	*Saída*
16 1 31 2 45 3 8 1 32 10 4 27 12 7 7 9 63 47 1 12 6 4 97 2 87 10 3 9 55 56 11 90 3 8 5 11 10 5 7 88 23 44 1 67 73 2 9 11 4 55 1 1 3 74 82 9 8 37 1 3 5 10 2 6 5 9 1 7 0 0	31 2 45 3 97 2 87 10 3 9 55 7 7 9 63 47 7 55 1 44 1 67 82 2 9 37

Problema $\mathcal{D}$

SALDO DE GOLS

Timelimit: 1s

Hipólito é um torcedor fanático. Coleciona flâmulas, bandeiras, recortes de jornal, figurinhas de jogadores, camisetas e tudo o mais que se refira a seu time preferido. Quando ganhou um computador de presente em uma festa, resolveu montar um banco de dados com os resultados de todos os jogos de seu time ocorridos desde a sua fundação, em 1911. Depois de inseridos os dados, Hipólito começou a ficar curioso sobre estatísticas de desempenho do time. Por exemplo, ele deseja saber qual foi o período em que o seu time acumulou o maior saldo de gols. Como Hipólito tem o computador há muito pouco tempo, não sabe programar muito bem, e precisa de sua ajuda.

Para realizar a tarefa, é dada uma lista, numerada seqüencialmente a partir de 1, com os resultados de todos os jogos do time (primeira partida: 3×0, segunda partida: 1×2, terceira partida: $0 \times 5 \ldots$). A tarefa é escrever um programa que determine em qual período o time conseguiu acumular o maior saldo de gols. Um período é definido pelos números de seqüência de duas partidas, A e B, onde $A \leq B$. O saldo de gols acumulado entre A e B é dado pela soma dos gols marcados pelo time em todas as partidas realizadas entre A e B (incluindo as mesmas) menos a soma dos gols marcados pelos times adversários no período. Se houver mais de um período com o mesmo saldo de gols, é escolhido o maior período. Se ainda assim houver mais de uma solução possíve a primeira é a escolhida.

Entrada

O programa deve ler vários conjuntos de teste. A primeira linha de um conjunto de teste contém um inteiro não negativo, N ($0 \leq N \leq 10000$), que indica o número de partidas realizadas pelo time (o valor $N = 0$ indica o final da entrada). Seguem-se N linhas, cada uma contendo um par de números inteiros não negativos X e Y ($0 \leq X, Y \leq 50$) que representam o resultado da partida: X são os gols a favor e Y os gols contra o time de Hipólito. As partidas são numeradas sequencialmente a partir de 1, na ordem em que aparecem na entrada.

Saída

Para cada conjunto de teste da entrada seu programa deve produzir uma única linha de saída. A linha deve conter um par de inteiros I e J que indicam respectivamente a primeira e última partidas do melhor período, conforme determinado pelo seu programa, exceto quando o saldo de gols do melhor período for menor ou igual a zero; neste caso a linha deve conter a expressão “nenhum”.

Exemplos

Entrada	*Saída*
2 2 3 7 1 3 0 2 0 3 0 4 0	2 2 nenhum

Problema $\mathcal{E}$

NÚMEROS ALEGRES

Timelimit: 1s

Como você já está estudando para a Maratona de Programação há algum tempo, você está familiarizado com os Números Alegres.

Assim, neste problema, você deve... o que? Você não sabe o que são Números Alegres? Eles também são conhecidos como Números Niven – isso te soa familiar? Não?

Certo... este conceito é simples. Vamos à sua explicação. Um Número Alegre é aquele que é divisível pela soma de seus dígitos. Por exemplo, 24 é um Número Alegre: a soma de seus dígitos é 2 + 4 = 6, e 24 é divisível por 6. Outro exemplo é o número 156: 1 + 5 + 6 = 12, e 156 é divisível por 12. Já o número 157 não é um Número Alegre, pois 1 + 5 + 7 = 13, e 157 não é divisível por 13, pois sobra resto.

Agora que você já sabe, vamos começar tudo novamente. Neste problema, dado um número N, você deve encontrar o menor Número Alegre que seja maior ou igual a N.

Entrada

A entrada contém vários casos de teste. A única linha de um caso de teste possui um inteiro N, tal que $1 \leq N \leq 10^9$. A entrada termina com um valor $N = 0$.

Saída

Para cada instância da entrada, imprima uma linha com o valor do menor Número Alegre maior ou igual a N.

Exemplos

Entrada	*Saída*
24 25 987654321 0	24 27 987654330