Departamento de Ingeniería en Sistemas de Información
Facultad Regional Santa Fe
Universidad Tecnológica Nacional

UTN SANTA FE

Competencia de Programación

TecnoMate

Nivel 2
Paradigmas de Programación

31 de Octubre de 2014

Problema 1: Al azar (21168. TAP2014A)

Los juegos de cartas son muchos y muy variados, y su origen se remonta a tiempos ancestrales. A veces puede resultar sorprendente que sigan siendo capaces de proveernos entretenimiento después de siglos de ser jugados con las mismas reglas, pero entonces debemos comprender que cada partida es esencialmente distinta de todas las demás que se han jugado en la historia de la humanidad, dada la gran cantidad de posibles formas de ordenar las cartas antes de iniciarla. En efecto, pocos juegos resultan entretenidos si utilizamos las cartas siempre en el mismo orden, o si existe una correlación entre cartas sucesivas que nos permita predecir el orden en el que se encuentran. Esta es la razón por la cual se suele mezclar las cartas antes de empezar cada juego, y por esto mismo les pedimos ahora que hagan un programa para controlar que una secuencia de cartas ha sido bien mezclada.

Para simplificar el problema, vamos a concentrarnos solamente en las barajas de *cartas españolas*, que consisten en 48 cartas distintas. Cada carta está identificada por un *valor*, que es un número del 1 al 12, y por un *palo*, que puede ser "bastos", "copas", "espadas" u "oros". Ahora bien, como no queremos simplificar excesivamente su tarea, vamos a tener en cuenta que no todos los juegos utilizan las 48 cartas de la baraja. Dada una secuencia de N cartas, decimos que está bien mezclada si no hay en ella dos cartas sucesivas que comparten el mismo valor o el mismo palo. Caso contrario, decimos que ha sido mal mezclada. ¿Pueden ayudarnos a decidir si una secuencia esta bien mezclada?

Entrada:

La primera línea contiene un número entero T que representa el número de casos de prueba ($1 \leq T \leq 200$). A continuación siguen T casos de prueba.

La primera línea de cada caso de prueba contiene un entero N, que representa la cantidad de cartas que se utilizan en el juego que vamos a considerar ($2 \leq N \leq 48$). Cada una de las siguientes N líneas contiene la descripción de una carta de la secuencia que queremos analizar, dada por un entero V que representa su valor ($1 \leq V \leq 12$) y un caracter P que representa su palo: "b" para bastos; "c" para copas; "e" para espadas y "o" para oros. Todas las cartas dadas son distintas, y se dan en la entrada en el mismo orden en el que aparecen en la secuencia.

Salida:

Para cada caso de prueba, imprimir en la salida una línea conteniendo un caracter que representa si la secuencia de cartas dada en la entrada ha sido bien mezclada o no. El caracter debe ser "B" en caso de que esté bien mezclada y "M" en caso de que esté mal mezclada.

Entrada de ejemplo	Salida para la entrada de ejemplo
4 4 1 b 2 c 3 e 4 o 3 1 b 2 b 3 c 3 1 b 1 c 2 e 32 5 c 2 b 4 e 3 o 12 b 1 c 7 e 6 c 12 e 4 o 1 b 6 o 3 e 12 o 11 e 12 c 5 o 10 b 9 o 3 c 4 b 11 c 8 e 9 c 1 e 4 c 8 b 2 o 6 b 9 e 7 b 5 e	B M M B

Problema 2: Base-3 balanceada (21169. TAP2014B)

A lo largo de la historia se han desarrollado sistemas de numeración muy diversos. Algunos, como el de números romanos, se han abandonado casi por completo en la actualidad por ser poco convenientes. Otros sistemas aún más exóticos son utilizados sólo para ciertas aplicaciones, como por ejemplo el sistema factorádico en el contexto de la numeración de permutaciones. En este problema vamos a considerar un sistema de numeración conocido como *base-3 balanceada*, que surge naturalmente en el análisis de diversos problemas matemáticos relacionados con las balanzas de platillos.

El sistema de base-3 balanceada es similar al sistema decimal o base-10 al que estamos acostumbrados, que es un sistema *posicional*. Los sistemas posicionales tienen *dígitos* cuyo orden relativo determina qué potencia de la *base* los acompaña. Por ejemplo, en base-10 tenemos

$$123 = \mathbf{1} \times 10^2 + \mathbf{2} \times 10^1 + \mathbf{3} \times 10^0.$$

En los sistemas posicionales estándar, los dígitos permitidos son todos los números del 0 al $B - 1$, donde B es la base del sistema utilizado. Entonces, el número 123 en base-10 queda escrito en base-3 estándar como "11120", dado que

$$\mathbf{1} \times 3^4 + \mathbf{1} \times 3^3 + \mathbf{1} \times 3^2 + \mathbf{2} \times 3^1 + \mathbf{0} \times 3^0 = 123.$$

La base-3 balanceada difiere de la base-3 estándar solamente en el hecho de que los dígitos permitidos son el 0, el 1 y el −1, que vamos a escribir como "**0**", "**+**" y "**-**" respectivamente. De este modo, el número 123 en base-10 queda escrito como "**+----0**" en base-3 balanceada, porque

$$\mathbf{1} \times 3^5 + (\mathbf{-1}) \times 3^4 + (\mathbf{-1}) \times 3^3 + (\mathbf{-1}) \times 3^2 + (\mathbf{-1}) \times 3^1 + \mathbf{0} \times 3^0 = 123.$$

Como la conversión de números en base-10 a base-3 balanceada es un proceso mecánico y algo tedioso, requerimos un programa que lo realice por nosotros. ¿Pueden ayudarnos?

Entrada:

La primera línea contiene un número entero T que representa el número de casos de prueba ($1 \leq T \leq 200$). A continuación siguen T casos de prueba.

Cada caso de prueba consiste en una única línea con un entero positivo N, el número en base-10 que deseamos escribir en base-3 balanceada ($1 \leq N \leq 1000$).

Salida:

Para cada caso de prueba, imprimir en la salida una línea conteniendo una cadena formada únicamente por los caracteres "0", "+" y "-" y que no empiece con "0", que representa los dígitos del número N en base-3 balanceada. La restricción de no comenzar la secuencia con "0" asegura que la representación es única.

Entrada de ejemplo	**Salida para la entrada de ejemplo**
2	+----0

123 729	+000000

Problema 3: Boca de urna (16276. TAP2013B)

Justo en este momento se están desarrollando las elecciones presidenciales en Nlogonia. Para que un candidato gane en primera vuelta debe obtener más votos que cada uno de los otros candidatos. Pero con eso no alcanza: además, debe obtener al menos el 45% de todos los votos, o al menos el 40% de todos los votos y al menos un 10 % más de votos que cada uno de los otros candidatos. Si ningún candidato gana en primera vuelta, se realiza una nueva elección en segunda vuelta.

Benicio es un periodista político de Nlogonia que siempre quiere tener la primicia. Por eso recolectó información de las encuestas de boca de urna, y quiere saber si de acuerdo a esos datos algún candidato gana en primera vuelta o, por el contrario, hay segunda vuelta. Benicio necesita decidir esto con urgencia antes de que alguien le saque la primicia. ¿Pueden ayudarlo?

Entrada:

La primera línea contiene un entero N que indica la cantidad de candidatos ($2 \leq N \leq 10$). La segunda línea contiene N enteros V_i que representan las cantidades de votos obtenidos por cada uno de los candidatos ($0 \leq V_i \leq 1000$ para i = 1, 2, . . . , N). Al menos un candidato obtuvo al menos un voto y no hay dos candidatos con la misma cantidad de votos.

Salida:

Imprimir en la salida una línea conteniendo un dígito que representa si hay o no ganador en primera vuelta. Si hay ganador en primera vuelta el dígito debe ser "1"; caso contrario (es decir, si hay segunda vuelta) el dígito debe ser "2".

Entrada de ejemplo	**Salida para la entrada de ejemplo**
2 60 40	1
3 16 28 21	1
3 42 23 35	2
3 297 302 401	2

Problema 4: Cartones Credecrecientes (11732. CREDECRE)

Para una variante de la tómbola, se quiere confeccionar e imprimir cartones del siguiente tipo:

T1

1	2	3
6	5	4
7	8	9

T2

1	2	4
7	6	5
8	9	10

T3

1	2	5
8	7	6
9	10	11

En la matriz, se deben ubicar los números en 1..*N*, los cuales pueden aparecer una vez como máximo -en cada cartón-. Por ejemplo, T1, es un cartón válido cuando *N* = 9.

T2 es un ejemplo de un cartón válido cuando *N* = 10, y T3 es un ejemplo de un cartón válido cuando *N* = 11.

Los tableros válidos cumplen con las siguientes condiciones:

- Los elementos de la fila 1 tienen que estar ordenados en forma creciente.
- Los elementos de la fila 2 tienen que estar ordenados en forma decreciente.
- Los elementos de la fila 3 tienen que estar ordenados en forma creciente.
- El menor de los elementos de la fila 2 tiene que ser mayor que el mayor de la fila 1.
- El menor de los elementos de la fila 3 tiene que ser mayor que el mayor de la fila 2.

Los cartones siempre son de 3x3, pero el número *N* puede variar. Se está organizando un evento multitudinario y se quiere conocer la cantidad de cartones válidos para diferentes valores de *N*. Ud. debe codificar un programa que lea valores para *N*, ingresados por teclado, e informe la cantidad de cartones válidos diferentes que se pueden confeccionar para dicho valor de *N*.

La entrada de datos termina cuando ingresa *N* = 0, los valores de *N* están en el rango 0..24.

Entrada de ejemplo	**Salida para la entrada de ejemplo**
4 9 11 10 24 0	0 1 55 10 1307504

PROBLEMA 5: Pista Iluminada (98. DFLOOR)

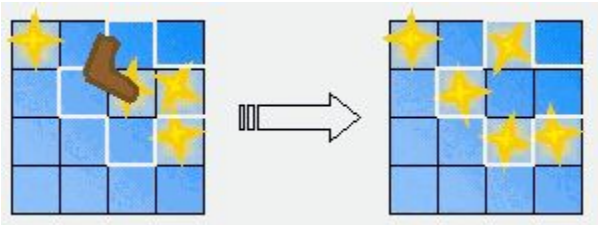
Hace poco tiempo viste un videoclip en el cual un cantante bailaba en una pista con baldosas coloridas iluminadas desde abajo. Por cada paso que el cantante daba en una baldosa, la misma cambiaba su estado, es decir se prendía o apagaba. Además, las cuatro baldosas que la rodeaban también cambiaban su estado.

En este desafío se te pide que obtengas un programa simple que permita decidir si es posible que el cantante encienda las luces de todas las baldosas, suponiendo que bailará pisando en las baldosas adecuadas.

La pista tiene forma rectangular, y consiste en una grilla de baldosas. Al principio, algunas de las baldosas ya se encuentran encendidas.

Tu programa puede apagar temporalmente algunas baldosas, si considera que esto es necesario para lograr su objetivo. Al pisar en una baldosa cambia el estado de la misma así como el estado de las cuatro baldosas vecinas, es decir, aquellas ubicadas directamente arriba, abajo, a la izquierda y a la derecha. Por supuesto, para las baldosas ubicadas en los bordes de la pista, el número de baldosas vecinas puede ser solamente dos o tres.

A continuación se presenta un ejemplo:



Si el cantante pisa en la baldosa indicada por el zapato marrón, todas las baldosas ubicadas en el área blanca cambian su estado. La pista de baile resultante se muestra a la derecha.

Puedes asumir que el cantante es suficientemente atlético para saltar desde cualquier baldosa hasta cualquier otra, aún si la baldosa destino se encuentra en el extremo opuesto de la pista.

Entrada:

Hay varios casos de prueba. La primera línea de cada caso contiene dos números enteros x e y, indicando el ancho y alto de la pista de baile. Estos números están separados por un espacio y satisfacen la relación $3 <= x,y <= 15$.

Las siguientes y líneas con x caracteres cada una describen el estado inicial de encendido/apagado de las baldosas. Un cero significa "la baldosa está apagada" y un uno significa "la baldosa está encendida".

La entrada termina con 0 0.

Salida:

Para cada caso de prueba tu programa deberá indicar el número de pasos necesarios para encender todas las luces, seguido de ese mismo número de líneas, cada una con

dos números *i* y *j* separados por un espacio. Cada línea instruye al cantante que debe pisar la baldosa *i*-ésima de la fila *j*-ésima. Comenzando con la situación del archivo de entrada y ejecutando todas las instrucciones del archivo de salida, todas las baldosas de la pista deben quedar encendidas.

Si existe más de una solución, tu programa deberá proporcionar cualquiera de ellas.

Si, por otra parte, no existen soluciones, tu programa deberá indicar el número "-1".

Entrada de ejemplo	**Salida para la entrada de ejemplo**
4 3 0111 1010 1000 0 0	3 1 2 1 3 4 3

Problema 6: Jurado despistado (21177. TAP2014J)

Este problema fue agregado a último momento reemplazando a otro que el jurado no pudo terminar de preparar a tiempo. No vamos a contarles mucho sobre el problema original porque pensamos usarlo para el torneo del año que viene. Sólo mencionaremos que se llama "Jugando con listas" y trata sobre listas de números enteros. La entrada del problema consistía en L listas de enteros dadas en la entrada de la siguiente manera:

- una línea con un entero N_1 indicando la cantidad de números de la primera lista;
- N_1 líneas con un entero cada una representando los números de la primera lista;
- una línea con un entero N_2 indicando la cantidad de números de la segunda lista;
- N_2 líneas con un entero cada una representando los números de la segunda lista;
- ...
- una línea con un entero N_L indicando la cantidad de números de la última lista;
- N_L líneas con un entero cada una representando los números de la última lista.

En un lamentable accidente, el jurado Joaquín borro la primera línea de cada archivo de entrada (la que contiene N_1). Necesitamos su ayuda para restaurar los archivos de entrada y usar el problema el año que viene. Dado el archivo sin la primera línea, les pedimos que nos digan cuales son los posibles valores de N_1 tales que, agregando una línea con ese valor, el archivo resultante es una entrada válida según la descripción dada arriba.

Entrada:

La primera línea contiene un entero T ($1 \leq T \leq 10^5$) que indica la cantidad de líneas que quedaron en el archivo de entrada.
Cada una de las T líneas siguientes contiene un entero R_i que representa el número que quedó en la i-ésima línea del archivo de entrada ($1 \leq R_i \leq 10^5$ para $i = 1, 2, \ldots, T$).

Salida:

Imprimir en la salida una línea para cada valor posible valor de N_1. Imprimir todas las posibles respuestas en orden creciente.

Entrada de ejemplo	Salida para la entrada de ejemplo
5 3 1 2 4 5	2 5

(sigue en reverso)

Entrada de ejemplo	Salida para la entrada de ejemplo
10 1 2 3 4 5 6 7 8 9 10	1 4 10