Trabajo práctico 4

Fecha límite de entrega: Viernes 2 de diciembre, hasta las 17:00 hs.
Fecha estimada de devolución: Dos semanas después de la fecha en la que es entregado el TP (haremos todo lo posible para que sea una semana)
Única fecha de reentrega: Dos semanas después de devueltas las correcciones, hasta las 17:00.

Este trabajo práctico consta de 4 problemas. Para aprobar el mismo se requiere aprobar todos los problemas.
Los requisitos de aprobación del mismo así como los criterios de corrección están detallados en las pautas de aprobación que pueden encontrar en el repositorio de la materia.

En este trabajo práctico se evaluarán contenidos de algoritmos y problemas sobre grafos. Los problemas que serán evaluados en el mismo serán sobre:

- Combinatoria y Problemas de Probabilidades.
- Geometría.
- Aritmética.

Problema A: Armando el polígono

Nos acaban de avisar que nuestro polígono simple favorito fue descompuesto en triángulos. Una descomposición en triángulos de un polígono de N lados es un conjunto de $N-2$ triángulos cuyos vértices son también vértices del polígono original, tal que dos triángulos cualesquiera no tienen un punto estrictamente interior en común, y de manera tal que la unión de todos los triángulos produce el polígono original.

Alarmados por la situación, nos piden que reconstruyamos el polígono lo antes posible para poder devolverle la tranquilidad a la comunidad de Triángulos Argentinos y Polígonos (TAP).

La gente del TAP nos provee de la siguiente información: Para cada uno de los $N-2$ triángulos (cuyos vértices tienen coordenadas enteras), nos darán las coordenadas de estos vértices. Es nuestra tarea reconstruir el polígono original, dando los N vértices del mismo ordenados en sentido horario.

El algoritmo debe tener una complejidad temporal $\mathbf{O}(N \log N)$.

Entrada

La entrada de cada caso de prueba constará en cada caso de varias líneas.

La primera línea tendrá un entero N con la cantidad de vértices (y lados) del polígono original.

Las siguientes $N-2$ líneas constarán de 6 enteros cada una, describiendo las coordenadas de los tres vértices de cada triángulo de la descomposición.

La entrada contará con el siguiente formato:

```
N
X11 Y11 X12 Y12 X13 Y13
X21 Y21 X22 Y22 X23 Y23
...
X(N-2)1 Y(N-2)1 X(N-2)2 Y(N-2)2 X(N-2)3 Y(N-2)3
```

Siendo Xij e Yij las coordenadas x e y respectivamente del j-ésimo punto del i-ésimo triángulo.

Salida

La salida debe constar de una única línea con $2N$ enteros que indiquen las coordenadas de los N vértices del polígono ordenados en sentido horario, empezando por el de menor coordenada X, y en caso de haber más de uno, desempatando por el de menor coordenada Y con el siguiente formato:

```
X1 Y1 X2 Y2 ... XN YN
```

Entrada de ejemplo 1	**Salida para la entrada de ejemplo 1**
5 0 0 10 9 10 0 10 9 0 9 0 0 10 9 0 9 5 13	0 0 0 9 5 13 10 9 10 0

Entrada de ejemplo 2	**Salida para la entrada de ejemplo 2**
3 0 1 1 1 1 0	0 1 1 1 1 0

Entrada de ejemplo 3	**Salida para la entrada de ejemplo 3**
10 -1 -2 2 -2 2 -1 -1 -2 0 -1 -2 3 -2 3 2 3 1 2 0 -1 2 1 1 2 -1 -2 2 -1 0 -1 2 1 0 -1 2 0 2 3 2 2 1 2 0 -1 -2 3 1 2	-2 3 2 3 2 2 1 2 2 1 2 0 0 -1 2 -1 2 -2 -1 -2

(la salida del ejemplo 3 está en dos líneas sólo para que entre en el pdf pero va en una sóla línea).

Pesos mínimo y máximo: 7 y 10.

Cotas recomendadas para testear: Se recomienda testear el problema con valores de $N \leq 10^5$.

Problema B: Bien ordenado

La semana pasada anunciaron el lanzamiento de BigTalk, un nuevo lenguaje de programación en el cual existen pocas instrucciones, una de ellas muy curiosa: Dado un arreglo de N números, y un número $i \leq N$, el lenguaje permuta aleatoriamente los últimos $N - i$ números del arreglo, siendo todas las posibles permutaciones equiprobables.

Los fanáticos de este tipo de lenguajes nos dicen que es el lenguaje del futuro. Nosotros, como estamos convencidos de que esto no es así, queremos demostrar que no están en lo cierto. Para eso, queremos calcular cuánto se tarda en ordenar un arreglo de enteros.

Para esto, tenemos que escribir un programa que reciba un arreglo de enteros, y nos diga cuánto tardaríamos en ordenarlo en BigTalk si el programa hace lo siguiente:

1. Comenzamos con $i = 0$ y el arreglo A de tamaño N dado en la entrada.
2. Permutamos al azar el arreglo $A[i..N)$.
3. Mientras que i sea menor a N y el menor elemento de $A[i..N)$ sea $A[i]$ incrementamos i en 1.
4. Si $i < N$ volvemos al paso 2.

Lo único importante es saber cuántas veces pasamos por el paso 2, ya que es el más costoso, es decir, cuántas veces permutamos un sufijo del arreglo A al azar.

Como este valor no es siempre el mismo (al haber azar presente), lo que se nos pide es calcular la esperanza del mismo.

El algoritmo debe tener una complejidad temporal $\mathbf{O(N^2)}$ siendo N la cantidad de elementos de la entrada.

Entrada

La entrada de cada caso de prueba constará de dos líneas.

La primera línea constará de un entero N indicando la cantidad de números del arreglo y la segunda línea contará con los N números del arreglo.

La entrada contará con el siguiente formato:

```
N
A1 A2 ... AN
```

Salida

La salida debe constar de un número E (la esperanza de la cantidad de veces que ejecutamos el paso 2 de nuestro algoritmo) con una precisión de 6 dígitos decimales, con el siguiente formato:

```
E
```

Entrada de ejemplo 1	Salida para la entrada de ejemplo 1
1 1	1.000000

Entrada de ejemplo 2	Salida para la entrada de ejemplo 2
2 1 2	2.000000

Entrada de ejemplo 3	Salida para la entrada de ejemplo 3
3 2 3 1	4.000000

Entrada de ejemplo 4	**Salida para la entrada de ejemplo 4**
6 15 16 4 8 42 23	16.000000

Entrada de ejemplo 5	**Salida para la entrada de ejemplo 5**
10 1 1 1 1 1 1 1 1 1 1	1.000000

Entrada de ejemplo 6	**Salida para la entrada de ejemplo 6**
6 1 2 2 2 2 3	8.083333

Pesos mínimo y máximo: 7 y 10.

Cotas recomendadas para testear: Se recomienda testear el problema con valores de $N \leq 10^3$, $0 \leq A_i \leq 10^5$.

Problema C: Construyendo murallas

El reino de Nlogonia ha entrado en guerra con la República de Banania y su rey ha encomendado la construcción de una muralla que proteja al reino de potenciales ataques de sus enemigos.

El rey sabe que es posible que no pueda salvar a todo el reino del devastador ataque de sus enemigos, y por eso nos ha pedido nuestra ayuda. Existen dentro del reino lugares históricos que le interesa preservar al rey (como esculturas, edificios o plazas que son muy importantes para su pueblo). A su vez, existen en el reino edificios que el rey sospecha que funcionan como puntos de reunión de soldados de Banania, y prefiere que queden fuera de la región encerrada por la muralla.

Para saber si es viable encomendarle a sus arquitectos la construcción de esta muralla, el rey nos pide, indicandonos las coordenadas de todos los puntos históricos de la ciudad, y de los edificios del enemigo, que calculemos cuál es la mayor cantidad de lugares históricos que pueden quedar dentro de la región encerrada por la muralla, dejando del lado de afuera a todos los edificios enemigos.

Su objetivo es escribir un algoritmo que calcule el mayor número de lugares históricos que pueden quedar dentro del área rodeada por la muralla, si el único requisito es que esta forme un polígono convexo y que no contenga ningún edificio enemigo.

El algoritmo debe tener una complejidad temporal polinomial (único requisito de complejidad para la aprobación del ejercicio), y deberá tener una complejidad $\mathbf{O}(N^5)$ para obtener el puntaje máximo donde $N = H + E$ siendo H la cantidad de lugares históricos y E la cantidad de edificios del enemigo. Existen soluciones con complejidad temporal $\mathbf{O}(N^4)$, y aunque no es necesario alcanzar esta complejidad para obtener el puntaje máximo, se valorará más una solución $\mathbf{O}(N^4)$ que una $\mathbf{O}(N^5)$ (en caso de no estar perfecto el ejercicio conseguir la mejor complejidad puede mejorar la nota).

Entrada

La entrada de cada caso de prueba constará en cada caso de varias líneas.

La primera línea constará de dos enteros H y E que indican la cantidad de lugares históricos del reino y de edificios enemigos.

Las siguientes H líneas constarán de dos enteros cada una, que representarán las coordenadas x e y de cada lugar histórico de la ciudad.

Luego seguirán E líneas indicando las coordenadas de los edificios enemigos con el mismo formato.

Se puede asumir que no existirán en la entrada tres puntos alineados.

La entrada contará con el siguiente formato:

```
H E
xh1 yh1
xh2 yh2
...
xhH yhH
xe1 ye1
xe2 ye2
...
xeE yeE
```

Siendo xhi e yhi las coordenadas x e y respectivamente del i-ésimo lugar histórico, y xei e yei las coordenadas del i-ésimo edificio enemigo.

Salida

La salida debe constar de una línea con la respuesta R del problema con el siguiente formato:

```
R
```

Entrada de ejemplo 1	Salida para la entrada de ejemplo 1
4 1 0 0 0 10 10 0 10 10 100 120	4

Entrada de ejemplo 2	Salida para la entrada de ejemplo 2
4 1 0 0 0 10 10 0 10 10 3 4	3

Entrada de ejemplo 3	Salida para la entrada de ejemplo 3
4 2 0 0 0 10 10 0 10 10 3 2 6 7	2

Entrada de ejemplo 4	Salida para la entrada de ejemplo 4
1 1 0 0 1 1	1

Entrada de ejemplo 5	Salida para la entrada de ejemplo 5
5 2 0 0 0 10 10 0 10 10 3 4 5 6 2 7	4

Pesos mínimo y máximo: 6 y 10.

Cotas recomendadas para testear: $N \leq 40$, $|X_i|, |Y_i| \leq 2^{15}$.

Problema D: Difícil de convencer

El Torneo Argentino de Programación 2017 tendrá un nuevo formato: Todos los equipos competiran contra todos los demás equipos en competencias de a dos equipos.

Para probar este formato, hemos hecho algunas simulaciones para ver sus resultados. En cada competencia hay un sólo posible resultado: uno de los dos equipos resulta ganador, y el otro pierde la competencia.

Tenemos dos copias de los resultados de la última simulación, que fueron anotadas por dos personas distintas. Cada una de estas copias están anotadas en una planilla distinta. Diego y Daniela, los encargados de anotar los resultados, utilizaron distintos números para identificar a los equipos, y sorprendentemente los resultados anotados por ambos fueron los mismos.

Diego, no cree que esto sea posible, y piensa que Daniela hizo trampa. Por eso, queremos demostrarle que esto sí es posible.

Un renombre de los equipos es una permutación P de los números entre 1 y N donde al equipo identificado por el número i pasamos a identificarlo con el número $P(i)$.

Para convencer a Diego le pedimos que nos de una permutación P, para la cual le vamos a decir cuántos de los posibles resultados cumplen con que al renombrar los equipos de acuerdo a la permutación P, estos no se modifican. La condición para esto, es que si el equipo i le gana al equipo j, entonces el equipo $P(i)$ le gane al equipo $P(j)$.

Su tarea es, dada la permutación P que nos pasa Diego, contar la cantidad de posibles resultados que cumplen con esta propiedad. Como este número puede ser muy grande, deben dar la respuesta módulo $10^9 + 7$ (1.000.000.007).

El algoritmo debe tener una complejidad temporal $\mathbf{O}(N \log N)$ donde la permutación de la entrada es una permutación de los números enteros entre 1 y N.

Entrada

La entrada de cada caso de prueba constará en cada caso de dos líneas.

La primera línea constará de un entero N indicando la cantidad de números de la permutación. La segunda línea constará de los N números enteros entre 1 y N inclusive (en cualquier orden) siendo el i-ésimo de estos números el número $P(i)$ donde P es la permutación descripta en el enunciado.

La entrada contará con el siguiente formato:

```
N
P1 P2 ... PN
```

Salida

La salida debe constar de un entero con la respuesta al problema, con el siguiente formato:

```
R
```

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

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

Pesos mínimo y máximo: 6 y 9.

Cotas recomendadas para testear: Se recomienda testear el problema con valores de $N \leq 10^5$.