

icpc international collegiate programming contest

ICPC Asia Pacific & Southeast Regionals 2019

ICPC Asia Seoul
Regional Contest

Official Problem Set



icpc.foundation



icpc global
programming
tools sponsor



Problem Set

Please check that you have 12 problems that are spanned across 24 pages in total (including the front page and this page).

- | | |
|--------------------------|-----------|
| A. Fire on Field | (2 pages) |
| B. Gene Tree | (2 pages) |
| C. Islands | (2 pages) |
| D. Ladder Game | (2 pages) |
| E. Network Vulnerability | (2 pages) |
| F. Quadrilaterals | (2 pages) |
| G. Same Color | (1 page) |
| H. Strike Zone | (2 pages) |
| I. Thread Knots | (2 pages) |
| J. Triangulation | (2 pages) |
| K. Washer | (1 page) |
| L. What's Mine is Mine | (2 pages) |

The memory limits for the twelve problems are all the same, 512MB.

Problem A

Fire on Field

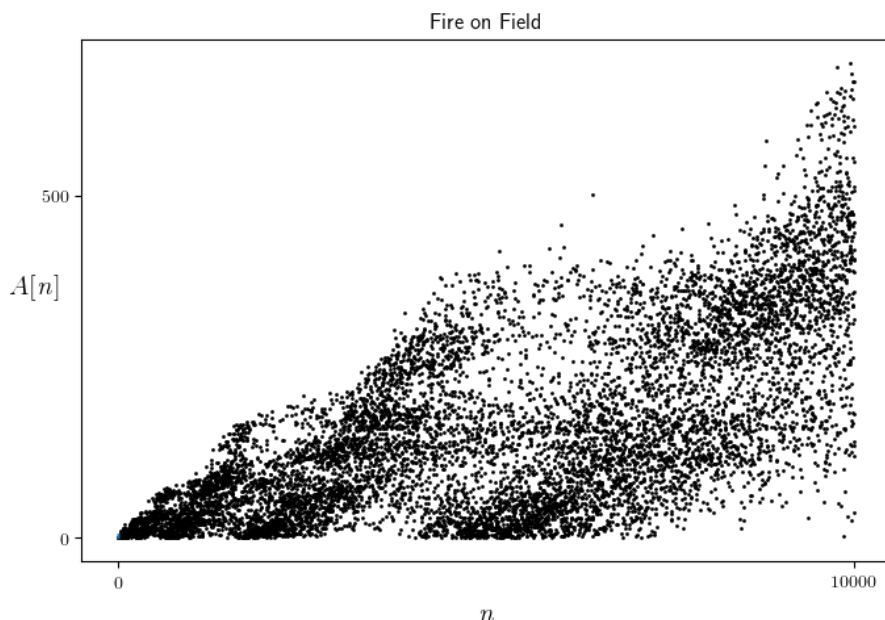
Time Limit: 1 Second

We define A as a sequence of positive integers like the following.

Let $A[0] = 1, A[1] = 1$. For a positive integer $i \geq 2$, $A[i]$ is the smallest positive integer under the condition that no three terms $A[i - 2k]$, $A[i - k]$, and $A[i]$ form an arithmetic progression for any integer $k > 0$ such that $i - 2k \geq 0$, that is, $A[i] - A[i - k] \neq A[i - k] - A[i - 2k]$.

The sequence is uniquely determined like the following sequence: $A[0] = 1, A[1] = 1, A[2] = 2, A[3] = 1, A[4] = 1, A[5] = 2, A[6] = 2, A[7] = 4, A[8] = 4$, etc. The sequence element $A[2]$ cannot be 1 since otherwise $A[0] = 1, A[1] = 1, A[2] = 1$ form an arithmetic progression; here $i = 2$ and $k = 1$. If $A[2]$ is any integer larger than one, then the condition is satisfied. Therefore, $A[2]$ should be 2 which is the smallest positive integer among the possible ones. Similarly, it is easy to know that $A[3] = 1$. The sequence element $A[4]$ cannot be 3 since otherwise $A[4] - A[4 - 2] = A[4 - 2] - A[4 - 2 \times 2]$; here $i = 4$ and $k = 2$. Other natural numbers like 1, 2 and 4 are also possible for $A[4]$, but the smallest one is 1. Therefore, $A[4] = 1$.

This sequence is called “fire on field” or “forest fire” since the scatter plot of the sequence looks like spreading fire on a field. See the figure below.



Given a non-negative integer n , write a program to output $A[n]$.

Input

Your program is to read from standard input. The input consists of one line containing one non-negative integer n ($0 \leq n \leq 1,000$).

Output

Your program is to write to standard output. Print exactly one line. The line should contain $A[n]$.

The following shows sample inputs and outputs for three test cases.

Sample Input 1	Output for the Sample Input 1
5	2
Sample Input 2	Output for the Sample Input 2
8	4
Sample Input 3	Output for the Sample Input 3
100	4

Problem B

Gene Tree

Time Limit: 1 Second

A *gene tree* is a tree showing the evolution of various genes or biological species. A gene tree represents the relatedness of specific genes stored at the leaf nodes without assumption about their ancestry. Leaf nodes represent genes, called *taxa*, and internal nodes represent putative ancestral taxa. Each edge in the tree is associated with a positive integer, *phylogenetic length*, which quantifies the evolutionary distance between two nodes of the edge. For example, the left figure below shows a gene tree with six leaf nodes, which approximates the relation among six taxa, and the right one shows a gene tree with four taxa.

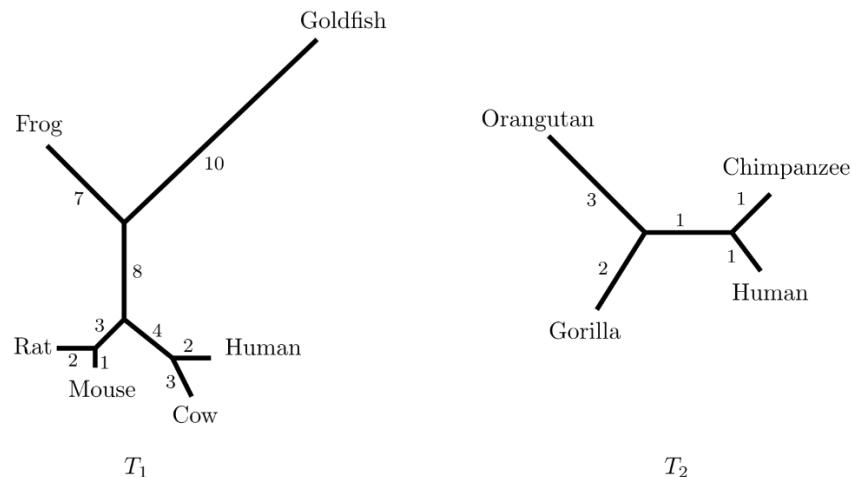


Figure B.1: Unrooted gene trees T_1 and T_2 .

Like the trees T_1 and T_2 above, gene trees are modeled as unrooted trees where each internal node (non-leaf node) has degree three. A path-length between two leaf nodes is the sum of the phylogenetic lengths of the edges along the unique path between them. In T_1 , the path-length between Human and Cow is $2 + 3 = 5$ and the path-length between Human and Goldfish is $2 + 4 + 8 + 10 = 24$. These lengths indicate that Human is much closer to Cow than to Goldfish genetically. From T_2 , we can guess that the primate closest to Human is Chimpanzee.

Researchers are interested in measuring the distance between genes in the tree. A famous distance measure is the sum of *squared* path-lengths of all unordered leaf pairs. More precisely, such a distance $d(T)$ is defined as follows:

$$d(T) = \sum_{\text{unordered pair } (u,v)} p_{u,v}^2$$

where $p_{u,v}$ is a path-length between two leaf nodes u and v in T . Note that $d(T)$ is the sum of the squared path-lengths $p_{u,v}^2$ over all unordered leaf pairs u and v in T . For the gene tree T_2 in Figure B.1, there are six paths over all unordered leaf pairs, (Human, Chimpanzee), (Human, Gorilla), (Human, Orangutan), (Chimpanzee, Gorilla), (Chimpanzee, Orangutan), and (Gorilla, Orangutan). The sum of squared path-lengths is $2^2 + 4^2 + 5^2 + 4^2 + 5^2 + 5^2 = 111$, so $d(T_2) = 111$.

Given an unrooted gene tree T , write a program to output $d(T)$.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($4 \leq n \leq 100,000$), where n is the number of nodes of the input gene tree T . Then T has $n - 1$ edges. The nodes of T are numbered from 1 to n . The following $n - 1$ lines represent $n - 1$ edges of T , where each line contains three non-negative integers a , b , and l ($1 \leq a \neq b \leq n$, $1 \leq l \leq 50$) where two nodes a and b form an edge with phylogenetic length l .

Output

Your program is to write to standard output. Print exactly one line. The line should contain one positive integer $d(T)$.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
4 1 4 1 4 3 1 2 4 1	12

Sample Input 2	Output for the Sample Input 2
6 1 5 1 5 2 1 5 6 1 6 4 3 6 3 2	111

Sample Input 3	Output for the Sample Input 3
10 1 2 10 10 2 7 3 2 8 3 9 3 9 8 2 7 9 1 6 4 3 4 5 2 3 4 4	4709



Problem C

Islands

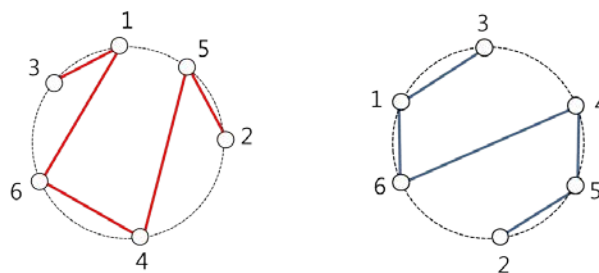
Time Limit: 1 Second

There are twin islands with convex coastlines in the southern sea of the Korean Peninsula. There are n villages on the coastline in each island. All villages are numbered from 1 to n . These numbers have been randomly assigned to villages regardless of their locations. A sequence of the village numbers in clockwise direction on the coastline of an island is called a *coastline sequence*. Considering a village as a point, the closed polyline connecting all villages in order of a coastline sequence forms the boundary of a convex polygon.

At a meeting of village representatives from both islands, they decided to construct a road inside each island satisfying the following three conditions.

1. Each road passes through all villages of each island exactly once.
2. Each road is not self-intersecting.
3. Two road sequences of both islands are same, where a *road sequence* is a sequence of the village numbers along the road from the start to the end.

For example, suppose that there are six villages in each island and that two coastline sequences of two islands are (1, 5, 2, 4, 6, 3) and (3, 4, 5, 2, 6, 1), respectively. Then two roads with a road sequence (3, 1, 6, 4, 5, 2) satisfy the above conditions. See the figure below. If two coastline sequences are (1, 2, 3, 4, 5) and (1, 3, 5, 2, 4), there is no road sequence which satisfies the above conditions.



Given two coastline sequences of two islands, write a program to find a road sequence satisfying the above conditions if it exists.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($5 \leq n \leq 100,000$), where n is the number of villages in each island. The villages are numbered from 1 to n . In the following two lines, each line contains n integers which represent a coastline sequence of each island.

Output

Your program is to write to standard output. Print exactly one line. The line should contain n integers which represent a road sequence satisfying above conditions if it exists, otherwise -1 . If there are one or more solutions, then print an arbitrary one.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
6 1 5 2 4 6 3 3 4 5 2 6 1	3 1 6 4 5 2
Sample Input 2	Output for the Sample Input 2
5 1 2 3 4 5 1 3 5 2 4	-1

Problem D

Ladder Game

Time Limit: 1 Second

Consider a situation that we need to distribute n gifts to n people fairly and randomly. For this purpose one ancient technique is popular in Asia, and is usually used to represent a random permutation. Chinese call it Ghost Leg ($\square \square \square$), Japanese Buddha Lots (あみだくじ) and Korean Ladder Game (사다리타기). We first start with some terms. This ladder consists of several vertical poles and horizontal bars connecting two adjacent vertical poles. From the top of each vertical pole, a path is traced through the ladder using the following three steps:

1. When tracing a vertical pole, continue downwards until an end of the first bar is reached, then continue along the bar.
2. When tracing a bar, continue along it until the end of the bar is reached, then continue down the vertical pole.
3. Repeat Step 1 and Step 2 until the bottom of a vertical pole is reached.

Figure D.1(a) shows a ladder L with three vertical poles and three bars. The vertical poles are numbered with $1, \dots, n$ from left to right. The paths of tracing these three vertical poles are shown in (b), (c), and (d). The input $(1, 2, 3)$ is permuted finally to $\pi_L = (3, 2, 1)$ by the ladder L . Note that we do not allow the case that two immediately adjacent horizontal bars meet at some point (like w in Figure D.1(e)) because there is no unique way in tracing at the point.

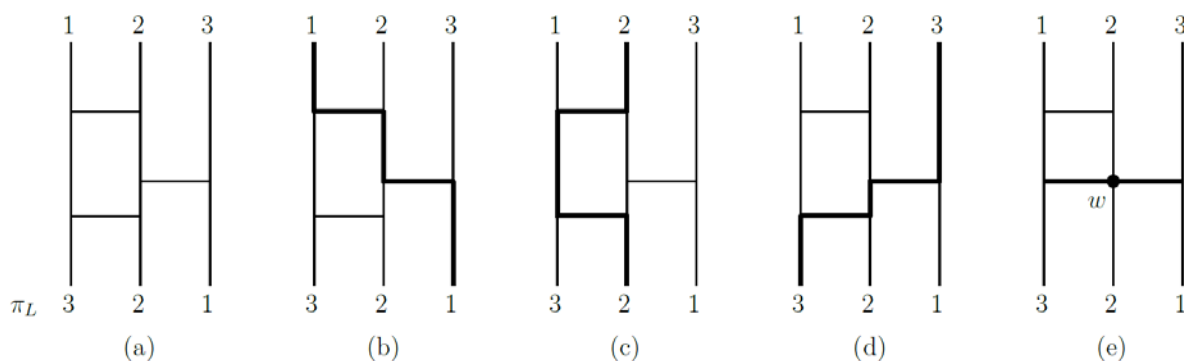


Figure D.1: Tracing the ladder.

You are given a ladder L which achieves a permutation π_L . If the permutation π_L is unchanged after removing a bar, then the bar is said to be *redundant* for π_L . Your task is to find all redundant bars for π_L . This is equivalent to constructing a *minimal* ladder by removing all redundant bars from L . In the minimal ladder of L , the removal of any bar from the ladder gives a different permutation from π_L .

Input

Your program is to read from standard input. The input starts with a line containing one integers n ($3 \leq n \leq 50$), where n is the number of vertical poles. The vertical poles are ordered from left to right. Let $d_{i,j}$ denote the depth of the j -th bar between the i -th and $(i + 1)$ -th poles from the top, which is an integer between 1 and 1,000. In the following $n - 1$ lines, the i -th line contains the sequence of depths $d_{i,j}$, where $1 \leq i < n, j \geq 1$

and $d_{i,j} < d_{i,j+1}$. Notice that this depth sequence is ended with zero (0), which is not a depth but just a marker to indicate the end of the sequence. See Figure D.2 for illustration.

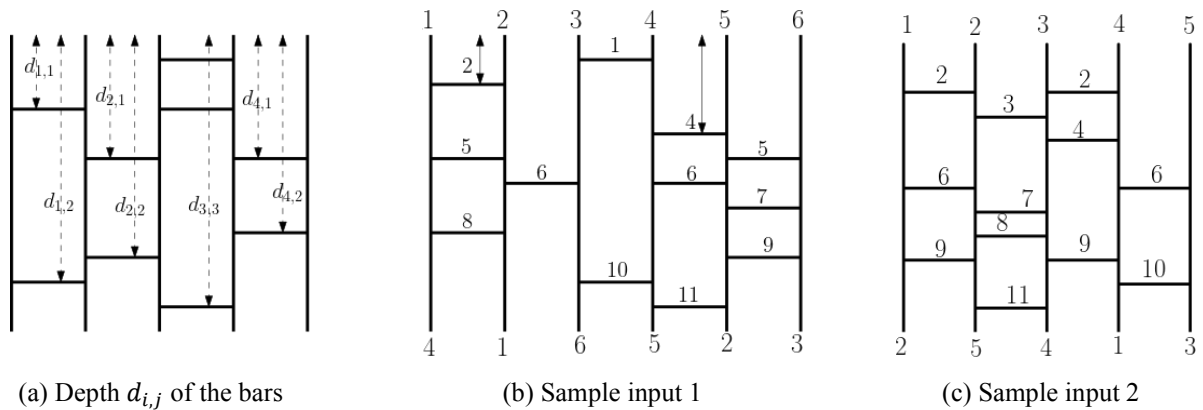


Figure D.2: Depths of the bars between two adjacent poles in the ladder.

Output

Your program is to write to standard output. Print a set of the bars to be remained in a minimal ladder for π_L . The first line should contain k , the number of the bars remained in the minimal ladder. Then, each line of the following k lines should contain two indices i and j of $d_{i,j}$ of the bar in the minimal ladder. Note that the minimal ladder is not unique.

The following shows sample input and output for two test cases in Figure D.2(b) and Figure D.2(c).

Sample Input 1

```
6
2 5 8 0
6 0
1 10 0
4 6 11 0
5 7 9 0
```

Output for the Sample Input 1

```
8
3 1
4 1
5 1
2 1
4 2
1 3
3 2
4 3
```

Sample Input 2

```
5
2 6 9 0
3 7 8 11 0
2 4 9 0
6 10 0
```

Output for the Sample Input 2

```
6
1 1
3 1
2 1
4 1
3 3
2 4
```

Problem E

Network Vulnerability

Time Limit: 3 Seconds

Network vulnerability refers to how much the network performance reduces in various cases of disruptions, such as natural disasters, element failures, or adversarial attacks. A network is frequently represented as a graph, in which the vertices and edges correspond to nodes and links, respectively. So the terms network and graph are used interchangeably here. The connectivity, which is defined to be the minimum number of vertices whose deletion results in a disconnected graph or a one-vertex graph, is recognized as the most important measure to evaluate the vulnerability of a network.

Nonetheless, the connectivity only partly reflects the resistance of a graph to the deletion of vertices. Accordingly, there have been many studies proposing other metrics to account for the network vulnerability, among which the toughness and the scattering number appear to be the most popular. The underlying notion of the toughness and scattering number is the maximum number of connected components resulting from removing k vertices from a graph. Let $c_k(G)$ denote the maximum number of connected components obtained by removing exactly k vertices from a graph G . It is unfortunate that given a graph G and a positive integer k , the problem of determining $c_k(G)$ is computationally intractable.

For some graph classes like interval graphs, however, $c_k(G)$ can be computed in polynomial time. An interval graph is the intersection graph of a family of (closed) intervals on the real line, where two vertices are connected with an edge if and only if their corresponding intervals intersect. The family is usually called an interval representation for the graph. See Figure E.1 for an example of an interval graph and its interval representation.

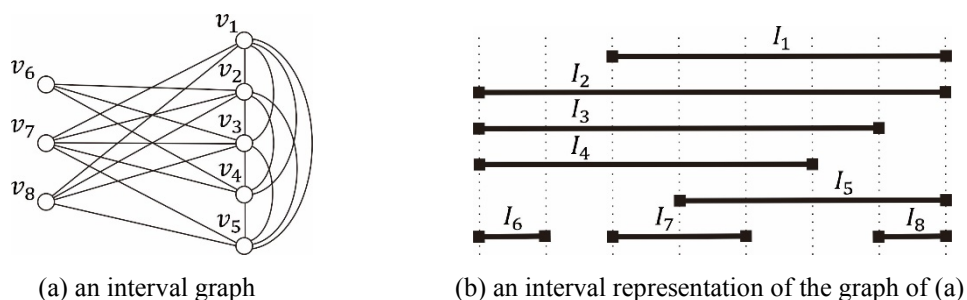


Figure E.1: An interval graph and its interval representation.

Given an interval representation of an interval graph G with n vertices, your job is to write an efficient running program for computing $c_k(G)$ for all k included in $\{0, 1, \dots, n-1\}$. If the interval representation shown in Figure E.1(b), for example, is given, then $c_0(G)$, $c_1(G)$, $c_2(G)$, $c_3(G)$, $c_4(G)$, $c_5(G)$, $c_6(G)$, $c_7(G)$ are 1, 1, 1, 2, 2, 3, 2, 1, respectively.

Input

Your program is to read from standard input. The first line contains a positive integer n representing the number of intervals, where $n \leq 2,000$. In the following n lines, each contains a pair of left and right endpoints of an interval. You may assume that the endpoints, left or right, of intervals are integers between $-100,000,000$ and $100,000,000$, inclusive.

Output

Your program is to write to standard output. Print exactly one line which contains the sequence $c_0(G)$, $c_1(G)$, ..., $c_{n-1}(G)$ of n numbers separated by a single space.

The following shows sample input and output for four test cases.

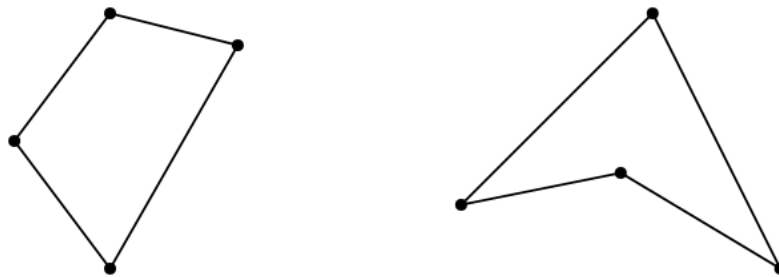
Sample Input 1	Output for the Sample Input 1
8 3 8 1 8 1 7 1 6 4 8 1 2 3 5 7 8	1 1 1 2 2 3 2 1
Sample Input 2	Output for the Sample Input 2
3 -2 -2 -2 7 -2 7	1 1 1
Sample Input 3	Output for the Sample Input 3
4 -1 -1 3 4 5 6 7 8	4 3 2 1
Sample Input 4	Output for the Sample Input 4
5 1 2 2 3 3 4 6 7 7 8	2 3 3 2 1

Problem F

Quadrilaterals

Time Limit: 1.7 Seconds

A *quadrilateral* is a polygon with exactly four distinct corners and four distinct sides, without any crossing between its sides. A quadrilateral is called *convex* if all the inner angles at its corners are less than 180 degrees, or called *non-convex*, otherwise. See the illustration below for a convex quadrilateral (left) and a non-convex quadrilateral (right).



In a test problem, you are given a set P of n points in the plane, no three of which are collinear, and asked to find as many quadrilaterals as possible by connecting four points from P , while each point in P can be used limitlessly many times and those quadrilaterals you find may freely overlap each other. You will earn different credits for each quadrilateral you find, depending on its shape and area. In principle, you earn more credits for convex quadrilaterals and for quadrilaterals with minimum area.

More precisely, the rules for credits are as follows, where a denotes the minimum over the areas of all possible quadrilaterals formed by connecting four points in P :

- For each distinct convex quadrilateral with area exactly a , you earn 4 credits.
- For each distinct non-convex quadrilateral with area exactly a , you earn 3 credits.
- For each distinct convex quadrilateral with area strictly larger than a , you earn 2 credits.
- For each distinct non-convex quadrilateral with area strictly larger than a , you earn 1 credit.

Note that two quadrilaterals are distinct unless the corners and sides of one are exactly the same to the other's, and that there may be two or more quadrilaterals of the minimum area a .

You of course want to find all possible quadrilaterals and earn the maximum possible total credits. Given a set P of n points in the plane, write a program that outputs the maximum possible total credits you can earn when you find all possible quadrilaterals from the set P .

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($4 \leq n \leq 1,000$), where n denotes the number of points in the set P . In the following n lines, each line consists of two integers, ranging -10^9 to 10^9 , representing the coordinates of a point in P . There are no three points in P that are collinear.

Output

Your program is to write to standard output. Print exactly one line consisting of a single integer that represents the maximum possible total credits you can earn from the set P .

The following shows sample input and output for four test cases.

Sample Input 1	Output for the Sample Input 1
4 0 0 1 0 0 1 1 1	4

Sample Input 2	Output for the Sample Input 2
4 0 0 10 0 5 10 5 8	5

Sample Input 3	Output for the Sample Input 3
4 0 0 10 0 5 10 5 3	7

Sample Input 4	Output for the Sample Input 4
5 0 0 0 5 5 0 5 5 4 2	14



Problem G

Same Color

Time Limit: 0.5 Seconds

There are n distinct points with m colors on the line, where $m \leq n$. Let S be the set of those points. We want to select a non-empty subset $C \subseteq S$ that satisfies the following:

For every point p in $S - C$, not belonging to C , the closest point of p among points in C has the same color as p . Of course, if there are more than one closest point of p in C , then it is sufficient that one of them has the same color as p .

For example, there are six points labeled by 1 to 6 with two colors in Figure G.1. Points 4 and 5 are red and the others blue. The set $\{2, 4, 6\}$ satisfies the above property. But the set $\{2, 4\}$ does not satisfy the property, because the closest point of point 6 in $\{2, 4\}$ is point 4, which is different from the color of point 6. In fact, the set $\{2, 4, 6\}$ is a minimum cardinality subset to satisfy the property.

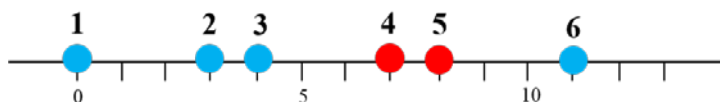


Figure G.1: Six colored points on a line

Given n distinct points on the line and m their colors, write a program to find a non-empty subset C of S with minimum cardinality satisfying the above property and to output the minimum cardinality.

Input

Your program is to read from standard input. The input starts with a line containing two integers, m and n ($1 \leq m \leq n \leq 100,000$), where m is the number of colors and n is the number of points. The points are numbered 1 to n from left to right on the line, and the colors are numbered 1 to m . The second line contains a sequence of sorted n integers in increasing manner, where the i -th number is the coordinate of the point i . The coordinates x of points satisfy $0 \leq x \leq 10^9$ and are all distinct. The third line contains a sequence of n integers, where the i -th number is the color of the point i , which is between 1 and m .

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum cardinality of a non-empty subset C of S to satisfy the above property.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
2 6 0 3 4 7 8 11 1 1 1 2 2 1	3
Sample Input 2	Output for the Sample Input 2
2 6 0 3 4 7 8 11 1 2 1 2 2 1	5

Problem H

Strike Zone

Time Limit: 1 Second

The strike zone in baseball is the volume of space which a baseball must pass through in order to be called a *strike*, if the batter does not swing. A baseball that misses the strike zone is called a *ball*, if the batter does not swing. Figure H.1 shows the locations of baseballs at plate which were captured by a ball tracking device during a baseball match. Each blue point was called a strike and each red point was called a ball during the match. This may motivate us to define a rectangular region that represents the strike zone of the match, by analyzing such a ball tracking data of the match.

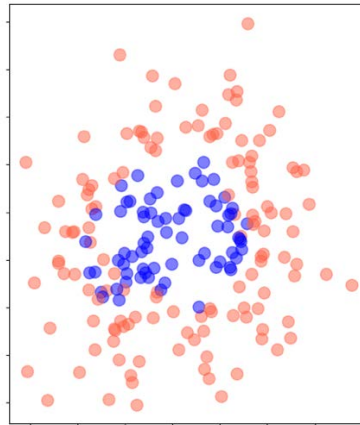


Figure H.1: The locations of baseballs at plate during a baseball match.
Blue points were called strikes and red points were called balls.

In this problem, you are given two sets, P_1 and P_2 , of points in the plane and two positive constants c_1 and c_2 . You are asked to find an axis-parallel rectangle R that maximizes the evaluation function $\text{eval}(R) = c_1 \times s - c_2 \times b$, where s is the number of points in $P_1 \cap R$ and b is the number of points in $P_2 \cap R$.

Input

Your program is to read from standard input. The input starts with a line containing an integer n_1 ($1 \leq n_1 \leq 1,000$), where n_1 denotes the number of points in P_1 . In the following n_1 lines, each line consists of two integers, ranging -10^9 to 10^9 , representing the coordinates of a point in P_1 . The next line contains an integer n_2 ($1 \leq n_2 \leq 1,000$), where n_2 denotes the number of points in P_2 . In the following n_2 lines, each line consists of two integers, ranging -10^9 to 10^9 , representing the coordinates of a point in P_2 . There are no two points in $P_1 \cup P_2$ that share the same x or y coordinate. Then the next line consists of two integers, c_1 and c_2 , ranging 1 to 10,000.

Output

Your program is to write to standard output. Print exactly one line consisting of one integer that is $\text{eval}(R)$, where R is an axis-parallel rectangle with the maximum possible eval value for P_1 and P_2 with respect to c_1 and c_2 .

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
2 -1 -1 4 4 2 0 0 2 2 5 2	6

Sample Input 2	Output for the Sample Input 2
3 0 5 3 3 8 -1 3 1 4 6 0 7 1 3 2	4

Problem I

Thread Knots

Time Limit: 1 Second

There are n threads on the x -axis. The length of the i -th thread, say T_i , is denoted by l_i and the location of its starting point by x_i , both being integers. We want to make a knot in each thread. The location of the knot must also be an integer. The knot can be made at any point in the thread and it is assumed that the length of the thread is not reduced by the knot. You can assume that no thread is totally contained by another, which means that there are no two threads T_i and T_j ($i \neq j$) such that $x_j \leq x_i$ and $x_i + l_i \leq x_j + l_j$.

We want to determine the location of the knot for each thread in order to make the distance between the closest two neighboring knots as big as possible.

For example, the figures below show the locations of the knots for six threads. The location of a knot is denoted by a point. All the threads actually lie on the x -axis, however, they are drawn separately to distinguish each other. In Figure I.1, the distance between the closest two knots is 20. However, if we make the knot for T_2 at different location as shown in Figure I.2, the distance between the closest two knots becomes 25, which is what this problem requests.

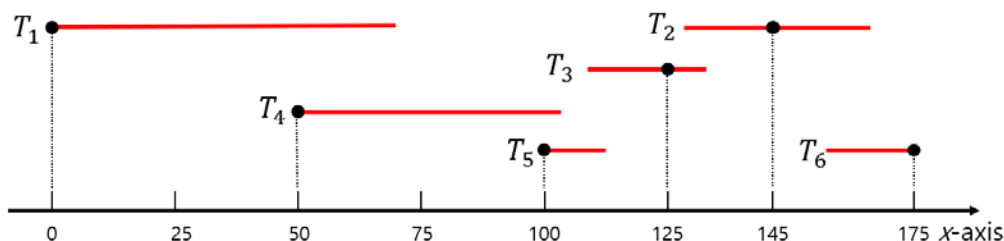


Figure I.1: An example of knots for 6 threads.

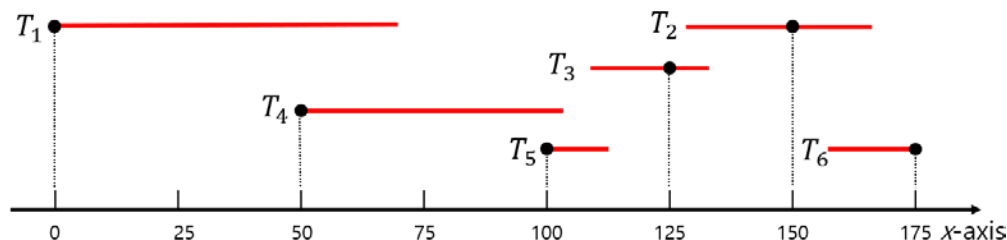


Figure I.2: Another example with different location of knot for T_2 .

Given information about the n threads, write a program that calculates the maximum distance between two closest knots.

Input

Your program is to read from standard input. The input starts with a line containing one integer, n ($2 \leq n \leq 100,000$), where n is the number of threads. In the following n lines, the i -th line contains two integers x_i ($0 \leq x_i \leq 10^9$) and l_i ($1 \leq l_i \leq 10^9$), where x_i and l_i denote the location of the starting point and the length of the i -th thread, respectively. You can assume that no thread is totally contained by another, which means that there are no two threads T_i and T_j ($i \neq j$) such that $x_j \leq x_i$ and $x_i + l_i \leq x_j + l_j$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain an integer which is the maximum distance between two closest knots.

The following shows sample input and output for three test cases.

Sample Input 1

```
6
0 67
127 36
110 23
50 51
100 12
158 17
```

Output for the Sample Input 1

```
25
```

Sample Input 2

```
6
0 40
10 55
45 28
90 40
83 30
120 30
```

Output for the Sample Input 2

```
30
```

Sample Input 3

```
3
0 20
40 10
100 20
```

Output for the Sample Input 3

```
50
```

Problem J

Triangulation

Time Limit: 1 Second

A regular n -sided polygon P can be partitioned into $n - 2$ triangles by adding non-crossing line segments connecting two vertices of P . For example, a square can be partitioned into two triangles, a regular pentagon can be partitioned into three triangles, and a regular hexagon can be partitioned into four triangles. The resulting set of triangles is called a *triangulation* of P . There exist two or more triangulations of P if n is greater than three.

Once a triangulation T of P is chosen, the distance between two triangles a and b in T is defined to be the number of hops crossing the borders of two adjacent triangles when you travel from a to b . Note that you must stay inside the polygon P at any time during this travel, that is, it is not allowed to hop crossing the border of P .

For example, the distance of a and d in the triangulation shown in Figure J.1 is three since the triangles, a , b , c , and d , should be visited to travel from a to d , and you have to hop three times crossing borders between triangles.

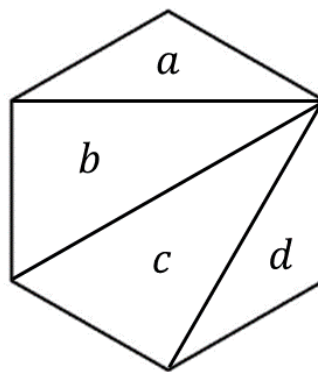


Figure J.1 A triangulation of a regular hexagon

The diameter of a triangulation T is the maximum of the distances between all pairs of triangles in T . Write a program to find a triangulation of a regular n -sided polygon P whose diameter is the minimum over all possible triangulations and print its diameter.

Input

Your program is to read from standard input. The input starts with a line containing n ($3 \leq n \leq 1,000,000$), where n is the number of sides of the regular n -sided polygon.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum diameter of the triangulations of a regular n -sided polygon.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
3	0

Sample Input 2	Output for the Sample Input 2
4	1

Sample Input 3	Output for the Sample Input 3
6	2



Problem K Washer

Time Limit: 1 Second

You have n clothes and a washer. The washer is large enough to wash all clothes at once. However, we should worry about the color transfer: if we put clothes of different colors in the washer, the dye from one may stain another. Precisely, let r_i, g_i, b_i denote the amount of red, green, blue color of the i -th clothes. When n clothes are washed together, the *color transfer* c is defined by

$$c = \sum_{i=1}^n (r_i - r)^2 + (g_i - g)^2 + (b_i - b)^2$$

where r, g , and b are the averages of r_i, g_i , and b_i , respectively. The i -th clothes with r_i, g_i , and b_i is defined as a point (r_i, g_i, b_i) in 3-dimensional RGB space. You can assume that no three points (clothes) are on a same line and no four points (clothes) are on a same plane in RGB space.

The washer consumes a lot of electricity, and you have to partition n clothes into at most k groups, and run the washer for each group. The total color transfer is the sum of color transfers from each run. Given the color information of n clothes and k , write a program to calculate the minimum total color transfer.

Input

Your program is to read from standard input. The first line contains two integers n ($1 \leq n \leq 100$) and k ($1 \leq k \leq 2$). In the following n lines, the i -th line contains three integers r_i, g_i, b_i ($0 \leq r_i, g_i, b_i \leq 1,000$).

Output

Your program is to write to standard output. Print exactly one line containing the minimum total color transfer, rounded to the sixth decimal point.

Sample Input 1

```
2 1
36 16 85
74 87 38
```

Output for the Sample Input 1

```
4347.000000
```

Sample Input 2

```
1 2
12 26 90
```

Output for the Sample Input 2

```
0.000000
```

Sample Input 3

```
3 2
93 50 26
40 0 77
99 10 29
```

Output for the Sample Input 3

```
822.500000
```

Problem L

What's Mine is Mine

Time Limit: 0.5 Seconds

The hot new video game “Mining Simulator” has just been released. In the game, rare earth mineral ores appear at certain times and you can mine them when they appear. After mining, you can later turn in the minerals for money. The quantity of mineral available during an appearance is proportional to the duration of the appearance and the price per unit of each mineral is decided beforehand.

The game contains a geological sensor that determines times when mineral ores will appear during a given day and at the beginning of each day you have a price list for each mineral. Assuming you mine out the mineral in exactly the amount of time that it is available, you cannot partially mine minerals (you must either not mine any of a given occurrence or mine all of it) and you can only mine one ore occurrence at a time.

Given a list of the prices of m minerals and n ore occurrences during a day, write a program to output the maximum amount of money you can earn from mining on that day. The mineral amount is the appearance time (end time – start time). You can mine an ore right after finishing the previous mining. In other words, one mined-ore's end time can be same as another mined-ore's start time. In the case depicted in Figure L.1, if you choose the mineral of type 1 that appears at time 2 and disappears at time 5, then the mineral amount is $5 - 2 = 3$ and you earn $3 \times 2 = 6$. Next, if you choose the mineral of type 2 that appears at time 7 and disappears at time 11, then the mineral amount is $11 - 7 = 4$ and you earn $4 \times 3 = 12$. Therefore, you earn 18 in total.

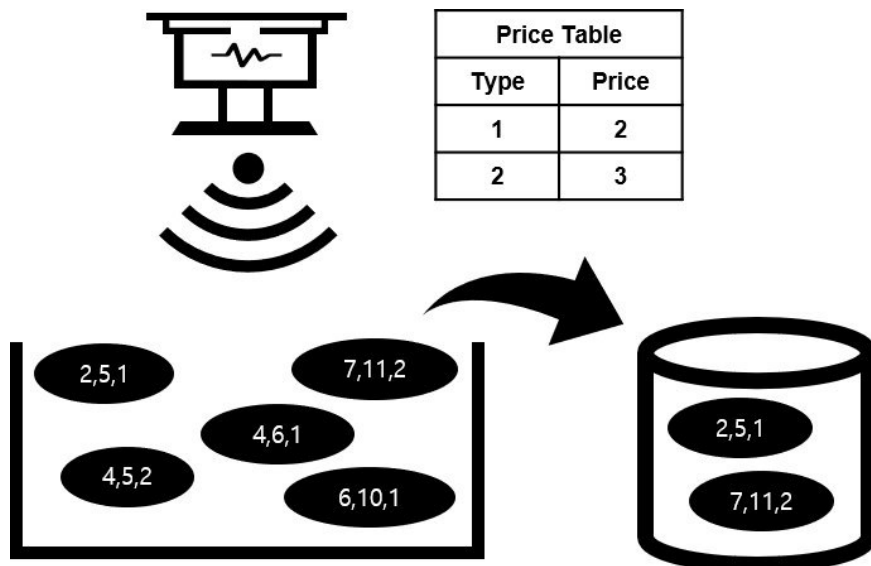


Figure L.1: A mining example. For each mineral (s, e, t) , s is the start time, e is the end time and t is the mineral type. Therefore, the mineral amount is $e - s$ and the possible earning is $(e - s) \times t$'s price.

Input

Your program is to read from standard input. The input starts with a line containing two integers, m and n ($1 \leq m \leq 100$, $1 \leq n \leq 10,000$), where m is the number of types of minerals and n is the number of ore occurrences during the day. The mineral types are labeled from 1 to m . The following m lines contain a single

integer corresponding to the price of one unit of the i -th mineral type on that day (the price is between 1 and 10,000). The following n lines represent ore occurrences. Each line contains three integers, s , e , t where s is the start time, e is the end time and t is the mineral type in each ore occurrence with $0 < s < e < 15,000$ and $1 \leq t \leq m$. The amount of mineral at each occurrence is $e - s$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the maximum amount of money that can be earned on that day.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
<pre> 2 5 2 3 2 5 1 4 5 2 4 6 1 7 11 2 6 10 1 </pre>	<pre> 18 </pre>
Sample Input 2	Output for the Sample Input 2
<pre> 3 5 2 3 1 1 4 1 3 6 3 5 8 2 7 10 1 9 12 2 </pre>	<pre> 24 </pre>
Sample Input 3	Output for the Sample Input 3
<pre> 5 7 1 2 3 4 5 1 5 2 3 8 1 2 4 3 3 9 2 4 10 5 7 11 4 5 7 3 </pre>	<pre> 36 </pre>