NCPC 2016

October 8, 2016

Problems

A Artwork
B Bless You Autocorrect!
C Card Hand Sorting
D Daydreaming Stockbroker
E Exponial
F Fleecing the Raffle
G Game Rank
H Highest Tower
I Interception
J Jumbled Compass
K Keeping the Dogs Apart

Do not open before the contest has started.

Advice, hints, and general information

- Your submissions will be run multiple times, on several different input files. If your submission is incorrect, the error message you get will be the error exhibited on the first input file on which you failed. E.g., if your instance is prone to crash but also incorrect, your submission may be judged as either "wrong answer" or "run time error", depending on which is discovered first.

- For problems with floating point output, we only require that your output is correct up to some error tolerance.

  For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that

  - If the correct answer is 0.05, any answer between 0.0499 and $.0501$ will be accepted.
  - If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.

  Any reasonable format for floating point numbers is acceptable. For instance, "17.000000", "0.17e2", and "17" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Problem A
Artwork
Problem ID: artwork
Time limit: 4 seconds

A template for an artwork is a white grid of $n \times m$ squares. The artwork will be created by painting q horizontal and vertical black strokes. A stroke starts from square (x_1, y_1), ends at square (x_2, y_2) ($x_1 = x_2$ or $y_1 = y_2$) and changes the color of all squares (x, y) to black where $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$.

The beauty of an artwork is the number of regions in the grid. Each region consists of one or more white squares that are connected to each other using a path of white squares in the grid, walking horizontally or vertically but not diagonally. The initial beauty of the artwork is 1. Your task is to calculate the beauty after each new stroke. Figure A.1 illustrates how the beauty of the artwork varies in Sample Input 1.

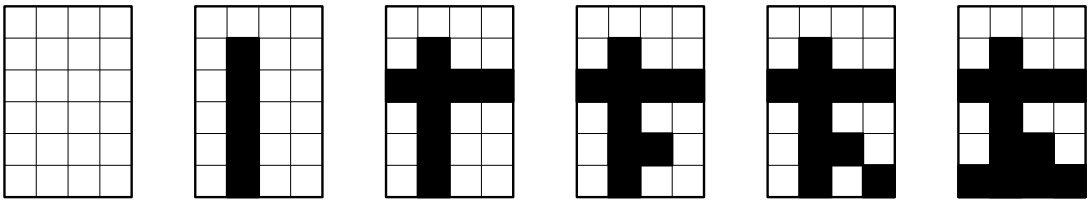


Figure A.1: Illustration of Sample Input 1.

Input

The first line of input contains three integers n, m and q ($1 \leq n, m \leq 1000$, $1 \leq q \leq 10^4$).

Then follow q lines that describe the strokes. Each line consists of four integers x_1, y_1, x_2 and y_2 ($1 \leq x_1 \leq x_2 \leq n$, $1 \leq y_1 \leq y_2 \leq m$). Either $x_1 = x_2$ or $y_1 = y_2$ (or both).

Output

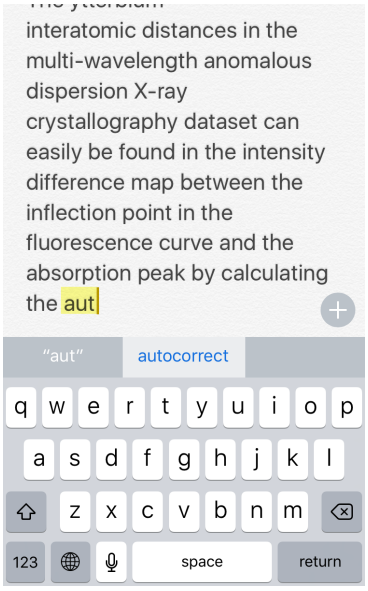
For each of the q strokes, output a line containing the beauty of the artwork after the stroke.

Sample Input 1	Sample Output 1
4 6 5 2 2 2 6 1 3 4 3 2 5 3 5 4 6 4 6 1 6 4 6	1 3 3 4 3

This page is intentionally left (almost) blank.

Problem B
Bless You Autocorrect!
Problem ID: autocorrect
Time limit: 3 seconds

Typing on phones can be tedious. It is easy to make typing mistakes, which is why most phones come with an autocorrect feature. Autocorrect not only fixes common typos, but also suggests how to finish the word while you type it. Jenny has recently been pondering how she can use this feature to her advantage, so that she can send a particular message with the minimum amount of typing.



cc by-sa NCPC2016

The autocorrect feature on Jenny's phone works like this: the phone has an internal dictionary of words sorted by their frequency in the English language. Whenever a word is being typed, autocorrect suggests the most common word (if any) starting with all the letters typed so far. By pressing tab, the word being typed is completed with the autocorrect suggestion. Autocorrect can only be used after the first character of a word has been typed – it is not possible to press tab before having typed anything. If no dictionary word starts with the letters typed so far, pressing tab has no effect.

Jenny has recently noticed that it is sometimes possible to use autocorrect to her advantage even when it is not suggesting the correct word, by deleting the end of the autocorrected word. For instance, to type the word "autocorrelation", Jenny starts typing "aut", which then autocorrects to "autocorrect" (because it is such a common word these days!) when pressing tab. By deleting the last two characters ("ct") and then typing the six letters "lation", the whole word can be typed using only 3 ("aut") + 1 (tab) + 2 (backspace twice) + 6 ("lation") = 12 keystrokes, 3 fewer than typing "autocorrelation" without using autocorrect.

Given the dictionary on the phone and the words Jenny wants to type, output the minimum number of keystrokes required to type each word. The only keys Jenny can use are the letter keys, tab and backspace.

Input

The first line of input contains two positive integers n ($1 \leq n \leq 10^5$), the number of words in the dictionary, and m ($1 \leq m \leq 10^5$), the number of words to type. Then follow n lines with one word per line, sorted in decreasing order of how common the word is (the first word is the most common). No word appears twice in the dictionary. Then follow m lines, containing the words to type.

The dictionary and the words to type only use lower case letters '`a`'-'`z`'. The total size of the input file is at most 1 MB.

Output

For each word to type, output a line containing the minimum number of keystrokes required to type the corresponding word.

Sample Input 1

```
5 5
austria
autocorrect
program
programming
computer
autocorrelation
programming
competition
zyx
austria
```

Sample Output 1

```
12
4
11
3
2
```

Sample Input 2

```
5 3
yogurt
you
blessing
auto
correct
bless
you
autocorrect
```

Sample Output 2

```
5
3
9
```

Problem C
Card Hand Sorting
Problem ID: cardhand
Time limit: 1 second

When dealt cards in the card game Plump it is a good idea to start by sorting the cards in hand by suit and rank. The different suits should be grouped and the ranks should be sorted within each suit. But the order of the suits does not matter and within each suit, the cards may be sorted in either ascending or descending order on rank. It is allowed for some suits to be sorted in ascending order and others in descending order.

Sorting is done by moving one card at a time from its current position to a new position in the hand, at the start, end, or in between two adjacent cards. What is the smallest number of moves required to sort a given hand of cards?

Sample Input 2 (cc by-sa NCPC 2016)

Input

The first line of input contains an integer n ($1 \leq n \leq 52$), the number of cards in the hand. The second line contains n pairwise distinct space-separated cards, each represented by two characters. The first character of a card represents the rank and is either a digit from `2` to `9` or one of the letters `T`, `J`, `Q`, `K`, and `A` representing Ten, Jack, Queen, King and Ace, respectively, given here in increasing order. The second character of a card is from the set {`s`, `h`, `d`, `c`} representing the suits spades ♠, hearts ♡, diamonds ♢, and clubs ♣.

Output

Output the minimum number of card moves required to sort the hand as described above.

Sample Input 1	Sample Output 1
4 2h Th 8c Qh	1

Sample Input 2	Sample Output 2
7 9d As 2s Qd 2c Jd 8h	2

Sample Input 3	Sample Output 3
4 2h 3h 9c 8c	0

This page is intentionally left (almost) blank.

Problem D
Daydreaming Stockbroker
Problem ID: stockbroker
Time limit: 1 second

Gina Reed, the famous stockbroker, is having a slow day at work, and between rounds of solitaire she is daydreaming. Foretelling the future is hard, but imagine if you could just go back in time and use your knowledge of stock price history in order to maximize your profits!

Photo by liz west on flickr, cc by

Now Gina starts to wonder: if she were to go back in time a few days and bring a measly $\$100$ with her, how much money could she make by just buying and selling stock in Rollercoaster Inc. (the most volatile stock in existence) at the right times? Would she earn enough to retire comfortably in a mansion on Tenerife?

Note that Gina can not buy fractional shares, she must buy whole shares in Rollercoaster Inc. The total number of shares in Rollercoaster Inc. is $100\,000$, so Gina can not own more than $100\,000$ shares at any time. In Gina's daydream, the world is nice and simple: there are no fees for buying and selling stocks, stock prices change only once per day, and her trading does not influence the valuation of the stock.

Input

The first line of input contains an integer d ($1 \leq d \leq 365$), the number of days that Gina goes back in time in her daydream. Then follow d lines, the i'th of which contains an integer p_i ($1 \leq p_i \leq 500$) giving the price at which Gina can buy or sell stock in Rollercoaster Inc. on day i. Days are ordered from oldest to newest.

Output

Output the maximum possible amount of money Gina can have on the last day. Note that the answer may exceed 2^{32}.

Sample Input 1	Sample Output 1
6 100 200 100 150 125 300	650

This page is intentionally left (almost) blank.

Problem E
Exponial
Problem ID: exponial
Time limit: 1 second

Everybody loves big numbers (if you do not, you might want to stop reading at this point). There are many ways of constructing really big numbers known to humankind, for instance:

- Exponentiation: $42^{2016} = \underbrace{42 \cdot 42 \cdot \ldots \cdot 42}_{\text{2016 times}}$.
- Factorials: $2016! = 2016 \cdot 2015 \cdot \ldots \cdot 2 \cdot 1$.

Illustration of $\mathrm{exponial}(3)$ (not to scale), Picture by C.M. de Talleyrand-Périgord via Wikimedia Commons

In this problem we look at their lesser-known love-child the *exponial*, which is an operation defined for all positive integers n as

$$\mathrm{exponial}(n) = n^{(n-1)^{(n-2)^{\cdots^{2^1}}}}.$$

For example, $\mathrm{exponial}(1) = 1$ and $\mathrm{exponial}(5) = 5^{4^{3^{2^1}}} \approx 6.206 \cdot 10^{183230}$ which is already pretty big. Note that exponentiation is right-associative: $a^{b^c} = a^{(b^c)}$.

Since the exponials are really big, they can be a bit unwieldy to work with. Therefore we would like you to write a program which computes $\mathrm{exponial}(n) \bmod m$ (the remainder of $\mathrm{exponial}(n)$ when dividing by m).

Input

The input consists of two integers n ($1 \leq n \leq 10^9$) and m ($1 \leq m \leq 10^9$).

Output

Output a single integer, the value of $\mathrm{exponial}(n) \bmod m$.

Sample Input 1	Sample Output 1
2 42	2

Sample Input 2	Sample Output 2
5 123456789	16317634

Sample Input 3	Sample Output 3
94 265	39

This page is intentionally left (almost) blank.

Problem F
Fleecing the Raffle
Problem ID: raffle
Time limit: 2 seconds

A tremendously exciting raffle is being held, with some tremendously exciting prizes being given out. All you have to do to have a chance of being a winner is to put a piece of paper with your name on it in the raffle box. The lucky winners of the p prizes are decided by drawing p names from the box. When a piece of paper with a name has been drawn it is not put back into the box – each person can win at most one prize.

The Raffle (Raffling for the Goose) by William Sidney Mount, public domain

Naturally, it is against the raffle rules to put your name in the box more than once. However, it is only cheating if you are actually caught, and since not even the raffle organizers want to spend time checking all the names in the box, the only way you can get caught is if your name ends up being drawn for more than one of the prizes. This means that cheating and placing your name more than once can sometimes increase your chances of winning a prize.

You know the number of names in the raffle box placed by other people, and the number of prizes that will be given out. By carefully choosing how many times to add your own name to the box, how large can you make your chances of winning a prize (i.e., the probability that your name is drawn exactly once)?

Input

The input consists of a single line containing two integers n and p ($2 \leq p \leq n \leq 10^6$), where n is the number of names in the raffle box excluding yours, and p is the number of prizes that will be given away.

Output

Output a single line containing the maximum possible probability of winning a prize, accurate up to an absolute error of 10^{-6}.

Sample Input 1	Sample Output 1
3 2	0.6

Sample Input 2	Sample Output 2
23 5	0.45049857550

This page is intentionally left (almost) blank.

Problem G
Game Rank
Problem ID: gamerank
Time limit: 1 second

The gaming company Sandstorm is developing an online two player game. You have been asked to implement the ranking system. All players have a rank determining their playing strength which gets updated after every game played. There are 25 regular ranks, and an extra rank, "Legend", above that. The ranks are numbered in decreasing order, 25 being the lowest rank, 1 the second highest rank, and Legend the highest rank.

Picture by Gonkasth on DeviantArt, cc by-nd

Each rank has a certain number of "stars" that one needs to gain before advancing to the next rank. If a player wins a game, she gains a star. If before the game the player was on rank 6-25, and this was the third or more consecutive win, she gains an additional bonus star for that win. When she has all the stars for her rank (see list below) and gains another star, she will instead gain one rank and have one star on the new rank.

For instance, if before a winning game the player had all the stars on her current rank, she will after the game have gained one rank and have 1 or 2 stars (depending on whether she got a bonus star) on the new rank. If on the other hand she had all stars except one on a rank, and won a game that also gave her a bonus star, she would gain one rank and have 1 star on the new rank.

If a player on rank 1-20 loses a game, she loses a star. If a player has zero stars on a rank and loses a star, she will lose a rank and have all stars minus one on the rank below. However, one can never drop below rank 20 (losing a game at rank 20 with no stars will have no effect).

If a player reaches the Legend rank, she will stay legend no matter how many losses she incurs afterwards.

The number of stars on each rank are as follows:

- Rank 25-21: 2 stars
- Rank 20-16: 3 stars
- Rank 15-11: 4 stars
- Rank 10-1: 5 stars

A player starts at rank 25 with no stars. Given the match history of a player, what is her rank at the end of the sequence of matches?

Input

The input consists of a single line describing the sequence of matches. Each character corresponds to one game; '`W`' represents a win and '`L`' a loss. The length of the line is between 1 and 10 000 characters (inclusive).

Output

Output a single line containing a rank after having played the given sequence of games; either an integer between 1 and 25 or "Legend".

Sample Input 1	Sample Output 1
WW	25

Sample Input 2	Sample Output 2
WWW	24

Sample Input 3	Sample Output 3
WWWW	23

Sample Input 4	Sample Output 4
WLWLWLWL	24

Sample Input 5	Sample Output 5
WWWWWWWWWLLWW	19

Sample Input 6	Sample Output 6
WWWWWWWWWLWWL	18

Problem H
Highest Tower
Problem ID: tower
Time limit: 7 seconds

Photo by Matt Schilder on flickr, cc by-sa

Oni loved to build tall towers of blocks. Her parents were not as amused though. They were on the verge of going crazy over that annoying loud noise whenever a tower fell to the ground, not to mention having to pick up blocks from the floor all the time. Oni's mother one day had an idea. Instead of building the tower out of physical blocks, why couldn't Oni construct a picture of a tower using two-dimensional rectangles that she montaged on a board on the wall? Oni's mother cut out rectangles of various sizes and colors, drew a horizontal line representing the ground at the bottom of the board, and explained the rules of the game to Oni: every rectangle must be placed immediately above another rectangle or the ground line. For every rectangle you can choose which of its two orientations to use. I.e., if a rectangle has sides of length s and t, you can either have a side of length s horizontally or a side of length t horizontally. You may place exactly one rectangle immediately above another one if its horizontal side is *strictly* smaller than the horizontal side of the rectangle beneath. Exactly one rectangle must be placed on the ground line. Now try to build as tall a tower as possible!

Oni's mother took extra care to make sure that it was indeed possible to use all rectangles in a tower in order not to discourage Oni. But of course Oni quickly lost interest anyway and returned to her physical blocks. After all, what is the point of building a tower if you cannot feel the suspense before the inevitable collapse? Her father on the other hand got interested by his wife's puzzle as he realized this is not a kids' game.

Input

The first line of input contains an integer n ($1 \leq n \leq 250\,000$), the number of rectangles. Then follow n lines, each containing two integers s and t ($1 \leq s \leq t \leq 10^9$ nm), the dimensions of a rectangle.

You may safely assume that there is a way to build a tower using all n rectangles.

Output

Output a single line containing the height in nm of the tallest possible tower using all the rectangles while having the horizontal side lengths strictly decreasing from bottom to top.

Sample Input 1	Sample Output 1
3 50000 160000 50000 100000 50000 100000	200000

This page is intentionally left (almost) blank.

Problem I
Interception
Problem ID: interception
Time limit: 8 seconds

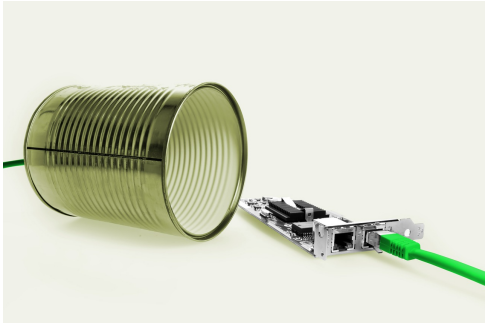


Photo by Michael Schwarzenberger, public domain

You are helping your local Neighborhood Surveillance Association setting up a surveillance system on your street in order to identify the culprits behind some recent unfortunate dog doo incidents.

The system in question consists of listening devices placed on the tin-can phone lines of the street (the idea being that sooner or later the dog doo malefactors will mess up and discuss their dirty deeds over the phone). Each listening device is placed on one of the phone lines connecting the houses, and will intercept all calls that are routed along that phone line.

The phone line network looks as follows. One side of the street has odd-numbered houses, and the other side has even-numbered houses. The numbers are lined up so that odd-numbered house i is directly across the street from even-numbered house $i + 1$. On both sides of the street, there are direct phone lines between adjacent pairs of houses. In addition there are two direct phone lines connecting the two sides of the street. Both of these two phone lines connect a house with the house directly across from it.

Calls between two houses on the same side of the street are routed along the direct path between the two houses, without going to the other side of the street. For calls between two houses on opposite sides of the street, we have information about which of the two direct connections across the street the call will use.

From previous ventures, we do know which people on the street are on speaking terms with each other. We would like to place listening devices in such a way that we can intercept all phone calls. In other words, for every pair of people that are on speaking terms with each other, there must be a listening device on the route between their houses.

Your job is to find a way to do this using the minimum possible number of listening devices (since they are kind of expensive and we burnt most of our budget on the barbecue last week).

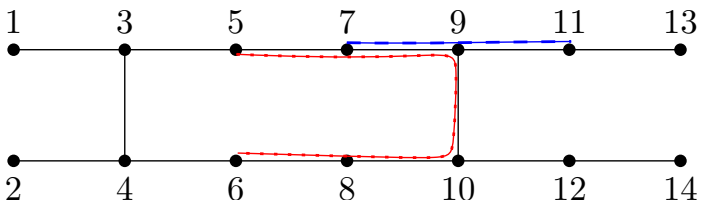


Figure I.1: Illustration of the phone line network in Sample Input 1. There are 14 houses, 7 on each side, and the two direct connections across the street connect house 3 with house 4, and house 9 with house 10. The people in houses 7 and 11 are on speaking terms, and calls between them are routed via house 9. The people in houses 5 and 6 are also on speaking terms, and calls between them are routed via houses 7, 9, 10, and 8. Sample Input 2 is similar, but there the calls between 5 and 6 are instead routed via the direct connection between houses 3 and 4.

Input

The first line of input consists of four integers n, m, c_1, and c_2 ($4 \leq n \leq 250\,000$, $1 \leq m \leq 500\,000$, $1 \leq c_1 < c_2 < n$), where n is even and denotes the number of houses on the street, m is the number of pairs of people that are on speaking terms with each other, and finally c_1 and c_2 are odd, denoting that the two wires crossing the street go between houses c_1 and $c_1 + 1$, and between houses c_2 and $c_2 + 1$.

Then follow m lines, each beginning with two distinct integers a and b between 1 and n (inclusive), indicating that the person in house a is on speaking terms with the person in house b. If the houses are on opposite sides of the street, they are followed by an integer $c \in \{c_1, c_2\}$ indicating that calls between a and b are routed along the direct connection between c and $c + 1$. Each pair $\{a, b\}$ appears at most once in the input.

Output

Output one line with an integer ℓ, the smallest possible number of listening devices needed. Then output ℓ lines, each containing a pair of integers, describing between which pairs of houses listening devices should be placed (of course, each of these pairs must be directly connected by a phone line). If there is more than one optimal solution, you may output any one of them.

Sample Input 1

```
14 2 3 9
7 11
5 6 9
```

Sample Output 1

```
1
7 9
```

Sample Input 2

```
14 2 3 9
7 11
5 6 3
```

Sample Output 2

```
2
3 4
7 9
```

Problem J
Jumbled Compass
Problem ID: compass
Time limit: 1 second

Jonas is developing the JUxtaPhone and is tasked with animating the compass needle. The API is simple: the compass needle is currently in some direction (between 0 and 359 degrees, with north being 0, east being 90), and is being animated by giving the degrees to spin it. If the needle is pointing north, and you give the compass an input of 90, it will spin clockwise (positive numbers mean clockwise direction) to stop at east, whereas an input of -45 would spin it counterclockwise to stop at north west.

cc-by NCPC 2016

The compass gives the current direction the phone is pointing and Jonas' task is to animate the needle taking the *shortest path* from the current needle direction to the correct direction. Many `ifs`, moduli, and even an arctan later, he is still not convinced his `minimumDistance` function is correct; he calls you on the phone.

Input

The first line of input contains an integer n_1 ($0 \leq n_1 \leq 359$), the current direction of the needle. The second line of input contains an integer n_2 ($0 \leq n_2 \leq 359$), the correct direction of the needle.

Output

Output the change in direction that would make the needle spin the shortest distance from n_1 to n_2. A positive change indicates spinning the needle clockwise, and a negative change indicates spinning the needle counter-clockwise. If the two input numbers are diametrically opposed, the needle should travel clockwise. I.e., in this case, output 180 rather than -180.

Sample Input 1	Sample Output 1
315 45	90

Sample Input 2	Sample Output 2
180 270	90

Sample Input 3	Sample Output 3
45 270	-135

This page is intentionally left (almost) blank.

Problem K
Keeping the Dogs Apart
Problem ID: dogs
Time limit: 6 seconds

Picture by Cartman0052007 via Wikimedia Commons, cc by

Despite the unfortunate incident last summer, which resulted in ten little puppies, you have been tasked with taking care of your neighbors' dogs again. Shadow and Lydia may be very cute mutts, but this year you have strict instructions to walk them one by one. However, you have other things to do during the summer than walking dogs! Like playing fetch and solving programming problems! It seems terribly inefficient to walk the dogs one at a time.

Shadow and Lydia have a particular walk they each prefer and know by heart. If you just let them out, they will follow their favorite walk, eventually ending up in their respective doghouses. Problem solved!

Sadly, you realize that if you just let both dogs out at the same time and let them do their walks on their own, they might get too close to each other. If they get too close, they will leave their favorite walk to "have some fun" and you are not sure you can find good homes for any more puppies. To ensure this does not happen, you need to calculate the minimum distance between the dogs when they are out walking on their own.

Both dogs start at the same time and keep exactly the same pace. Immediately after a dog arrives at its doghouse it stays inside and goes to sleep, so we no longer need to worry about the distance to the other dog, even though the other dog may still walk for a while longer. Note that a dog is still awake at the exact moment of entering its house and falls asleep immediately after entering.

Input

The first line of input consists of an integer n ($2 \leq n \leq 100\,000$), the number of points describing the walk of Shadow. The next n lines contain 2 integers each, giving the x and y coordinates of Shadow's walk. Two consecutive points in the walk always differ in at least one coordinate. All coordinates are non-negative and at most $10\,000$. Similarly, the next line contains an integer m ($2 \leq m \leq 100\,000$), the number of points describing the walk of Lydia. The next m lines describe its walk in the same format as for Shadow.

Output

Output the minimum distance between the two dogs during their walks. The numbers should be accurate to an absolute or relative error of at most 10^{-4}.

Sample Input 1	Sample Output 1
2 0 0 10 0 2 30 0 15 0	10

Sample Input 2	Sample Output 2
5 10 0 10 8 2 8 2 0 10 0 9 0 8 4 8 4 12 0 12 0 8 4 8 4 12 0 12 0 8	1.414213562373

NCPC 2016
Presentation of solutions

The Jury

2016-10-08

J — Jumbled Compass

Problem

Simple problem with solutions by the jury in all languages available in the contest.

Some solution (guess the language)

```
solve(N1, N2) :-
    (N2-N1 >  180 -> Ans is N2-N1-360;
     N2-N1 > -180 -> Ans is N2-N1;
                     Ans is N2-N1+360),
    write(Ans), nl.
```

Statistics: 535 submissions, 266 accepted, first after 00:03

G - Game Rank

Problem

Simulate ranking system of some vaguely familiar game.

Solution

1. Read and understand the rules.
2. Keep track of current rank, current number of stars and current number of consecutive wins.
3. Update accordingly.
4. Don't try to be clever.

Statistics: 960 submissions, 218 accepted, first after 00:17

D - Daydreaming Stockbroker

Problem

Play the stock market when knowing the future.

Solution (guess the language)

```
fscanf(STDIN, "%d", $days);
$money = 100;
$prev = 1<<30;
for ($i = 0; $i < $days; ++$i) {
    fscanf(STDIN, "%d", $cur);
    if ($cur > $prev)
        $money += min(floor($money/$prev), 100000)*($cur-$prev);
    $prev = $cur;
}
echo $money;
```

Statistics: 740 submissions, 183 accepted, first after 00:12

F - Fleecing the Raffle

Problem

When drawing p items out of $n + x$ items, what is probability that *exactly one* out of the first x items is drawn?

What is the maximum such probability over all x?

Solution

1. Probability is
$$\frac{\binom{x}{1} \cdot \binom{n}{p-1}}{\binom{n+x}{p}} = \{\text{...some calculations...}\} = \frac{x \cdot p}{n+1} \cdot \prod_{i=2}^{x} \frac{n-p+i}{n+i}$$
2. When going from $x-1$ to x, probability changes by factor
$$\frac{x}{x-1} \cdot \frac{n-p+x}{n+x}$$
3. Some calculus $\Rightarrow$ increase if $x < \frac{n}{p-1}$, decrease otherwise
$\Rightarrow$ max happens at $x = \lfloor n/(p-1) \rfloor$.

F - Fleecing the Raffle

Problem

When drawing p items out of $n + x$ items, what is probability that *exactly one* out of the first x items is drawn?

What is the maximum such probability over all x?

Solution

Linear $O(n/p)$ time solution:

```
int n, p;
scanf("%d%d", &n, &p);
int x = n/(p-1);
double res = double(x*p) / (n+1);
for (int i = 2; i <= x; ++i)
    res *= double(n-p+i) / (n+i);
printf("%.9lf\n", res);
```

F - Fleecing the Raffle

Problem

When drawing p items out of $n + x$ items, what is probability that *exactly one* out of the first x items is drawn?

What is the maximum such probability over all x?

Solution

Constant time solution:

```
int n, p;
scanf("%d%d", &n, &p);
int x = n++/(p-1);
printf("%.9lf\n", x*p*exp(lgamma(n-p+x)-lgamma(n-p+1)
                          -lgamma(n+x)+lgamma(n));
```

(But in order to do this in languages that don't provide full ISO C support, one may have to implement the Γ function oneself)

Statistics: 349 submissions, 68 accepted, first after 00:17

C - Card Hand Sorting

Problem

What is minimum number of cards to move to get list of cards in some form of order?

Solution

1. Try all $4! = 24$ possible ways of ordering the 4 suits.
2. Try all $2^4 = 16$ possible ways of choosing ascending/descending order within suits.
3. Now we have a fixed total order on the cards.
4. Maximum number of cards that can remain in place is length of longest increasing subsequence with respect to the chosen ordering.

Statistics: 77 submissions, 23 accepted, first after 00:29

A - Artwork

Problem

Fill horizontal and vertical blocks of squares in an initially empty grid, and output the number of unfilled connected components after each operation.

Solution

1. The problem can be modelled as a graph where each node represents a square in the grid, and two nodes are connected if the squares belong to the same component.
2. Each operation can be divided into modifications of single squares in the grid.
3. The process can be simulated efficiently using a union-find structure when all operations are done in the reverse order.

Statistics: 134 submissions, 17 accepted, first after 01:14

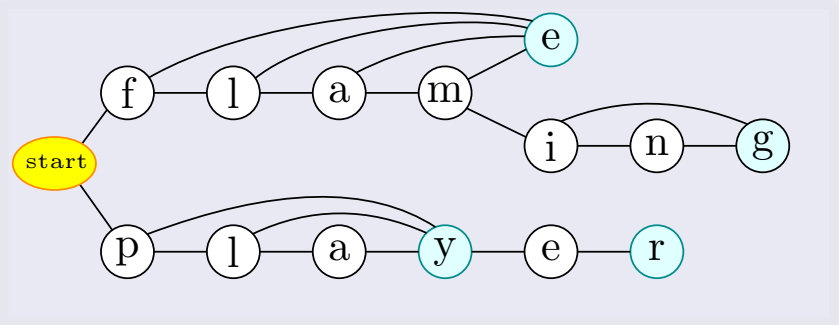
B - Bless You Autocorrect!

Problem

Type a word using autocorrect.

Solution

1. Realisation: may need multiple autocorrects for a single word
2. Build trie of dictionary, plus shortcut edges for autocorrects



Graph for dictionary “flame”, “flaming”, “play”, “player”

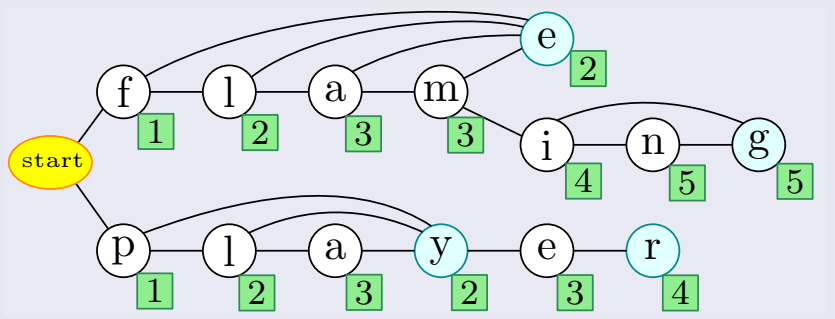
B - Bless You Autocorrect!

Problem

Type a word using autocorrect.

Solution

1. Realisation: may need multiple autocorrects for a single word
2. Build trie of dictionary, plus shortcut edges for autocorrects



3. Find shortest distance to each node with BFS
4. To type word w, find node corresponding to longest prefix of w
5. Answer for w is distance to node + #remaining letters
6. Time complexity: linear in size of input.

Statistics: 70 submissions, 12 accepted, first after 00:43

K - Keeping the Dogs Apart

Problem

Two dogs move around along straight line segments, what is the closest they get to each other?

Solution 1

1. Split the walks into intervals during which the two dogs don't switch line segments.
2. Movement is relative: the two dogs walking from P to $P + \Delta P$ and from Q to $Q + \Delta Q$ is *equivalent* to one standing still at P and other moving from Q to $Q + \Delta Q - \Delta P$
3. Closest distance in each such interval boils down to distance between point and line segment, basic geometric primitive.
4. Time complexity: $O(n)$.

K - Keeping the Dogs Apart

Problem

Two dogs move around along straight line segments, what is the closest they get to each other?

Solution 2 (more or less the same but different perspective)

1. Split the walks into intervals during which the two dogs don't switch line segments.
2. In an interval where dogs walk from P to $P + \Delta P$ and Q to $Q + \Delta Q$, square dist. after fraction $t \in [0, 1]$ of the time is

$$\|P - Q + t(\Delta P - \Delta Q)\|_2^2 = \|P - Q\|_2^2 + 2t\langle P - Q, \Delta P - \Delta Q\rangle + t^2\|\Delta P - \Delta Q\|_2^2$$

3. Minimum happens at $t = \frac{-\langle P-Q, \Delta P-\Delta Q\rangle}{\|\Delta P-\Delta Q\|_2^2}$ (basic calculus)
   Truncate to $t \in [0, 1]$, be careful with $\Delta P = \Delta Q$.
4. Time complexity: $O(n)$.

Statistics: 95 submissions, 11 accepted, first after 01:47

E - Exponial

Problem

Compute $f(n) \bmod m$, where $f(1) = 1$, $f(n) = n^{f(n-1)}$.

Solution

1. If $n \leq 5$: just compute $f(n-1)$ and then $n^{f(n-1)} \bmod m$ with modular exponentiation.
2. If $n > 5$: ??????????

Lemma

For all n and m, and $e \geq log_2(m)$ it holds that

$$n^e \bmod m = n^{\phi(m)+e \bmod \phi(m)} \bmod m.$$

($\phi(m) =$ Euler's totient function.)

Proof: ugly and does not fit on slide. (Boils down to Chinese Remainder Theorem and ϕ being multiplicative.)

E - Exponial

Problem

Compute $f(n) \bmod m$, where $f(1) = 1$, $f(n) = n^{f(n-1)}$.

Solution

1. If $n \leq 5$: just compute $f(n-1)$ and then $n^{f(n-1)} \bmod m$ with modular exponentiation.
2. If $n > 5$: compute $z = f(n-1) \bmod \phi(m)$ recursively. The lemma then says that $f(n) \bmod m = n^{\phi(m)+z} \bmod m$
3. Time complexity:
   - each recursive call dominated by time to compute $\phi(m)$: $O(\sqrt{m})$ using naive factorization.
   - recursing until n becomes ≤ 5 hopelessly slow...
   - ...but we can stop when we reach $m = 1$!

     Lemma: $\phi(\phi(\cdots\phi(m)))$ reaches 1 after $O(\log m)$ iterations
     Proof: cute but does not fit on slide.

Statistics: 47 submissions, 3 accepted, first after 00:45

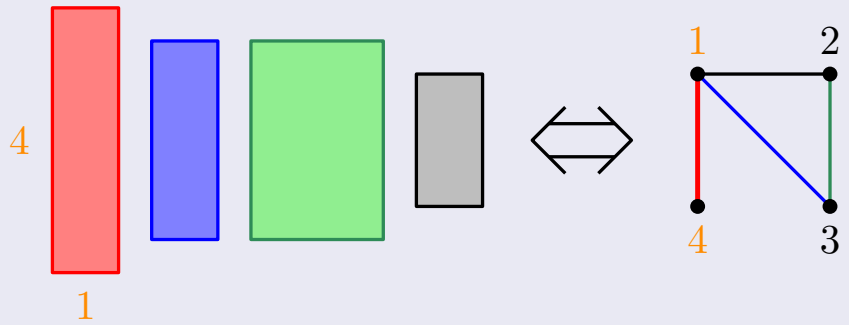
H - Highest Tower

Problem

Given a set of rectangles, build a tower of maximum height.

Solution

1. Make a graph: vertices = lengths, edges = given rectangles.



2. Encode orientation of a rectangle as direction of an edge.

3. Orientation of rectangles gives a valid tower if no width occurs more than once $\Leftrightarrow$ all nodes have outdegree ≤ 1

H - Highest Tower

Problem **Reformulated**

Given an undirected graph, direct edges so that each node has at most one out-going edge and maximize $\sum_{v \in V} \text{value}(v) \cdot \text{indeg}(v)$

Solution

1. In connected component with v vertices and e edges, average out-degree is e/v, so we must have $e \leq v$
   Each component is a tree, or a tree plus one edge.
2. **Case 1:** $e = v$ **(tree plus one edge)**: each node must get out-degree exactly 1 so $\text{indeg}(v) = \deg(v) - 1$.
3. **Case 2:** $e = v - 1$ **(tree)**: one node will have out-degree 0, the rest out-degree 1. Let node with highest value get out-degree 0 in order to maximize height.
4. Time complexity: $O(n)$ (assuming $O(1)$ dictionary lookup).

Statistics: 38 submissions, 3 accepted, first after 02:22

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1

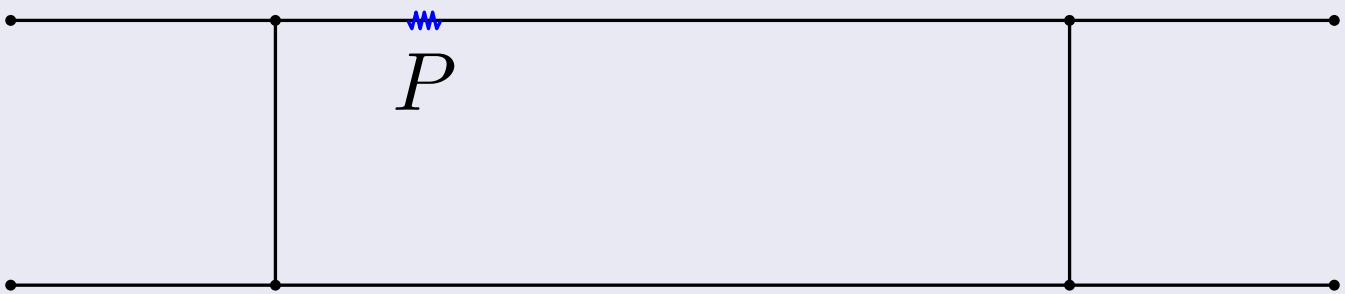
1. Try all 4 possible ways of using the two crossings.
2. **Case 1, use both crossings**: problem decomposes into two separate problems on a line, simple greedy.
3. **Case 2, use one crossing**: a bit of work, better to skip and then revisit with ideas from the harder Case 3.
4. **Case 3, don't use crossings**: main challenge to handle.
   - In order to improve on Case 1, can use at most 1 extra device for the sides.
   - One side must use minimum number of crossings.

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1



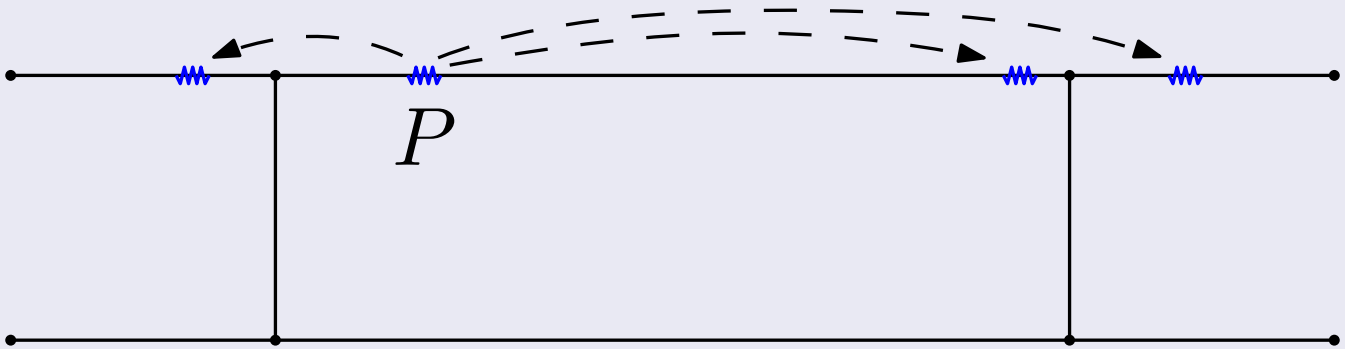
Guess first position P to use after first crossing, $O(n)$ choices.

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1



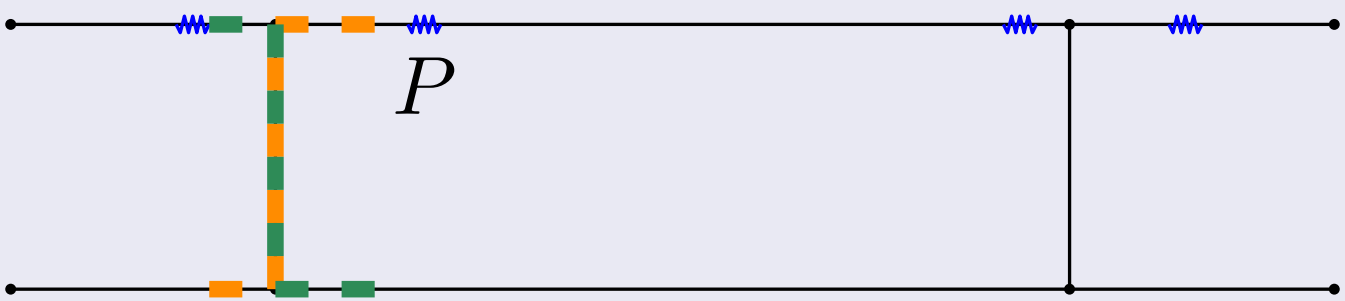
Try to squeeze the rest as close as possible to the crossing, $O(1)$ choices given P.

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1



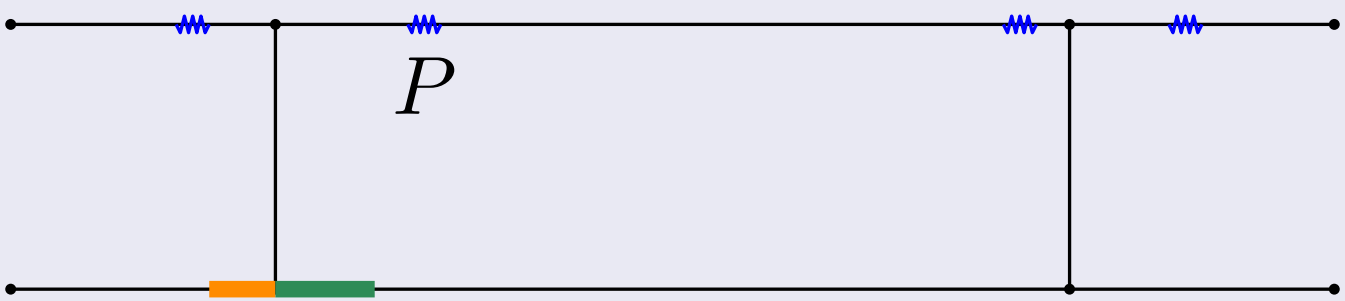
When one side is decided, the positions next to the crossings determine which crossing calls remain uncovered.

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1



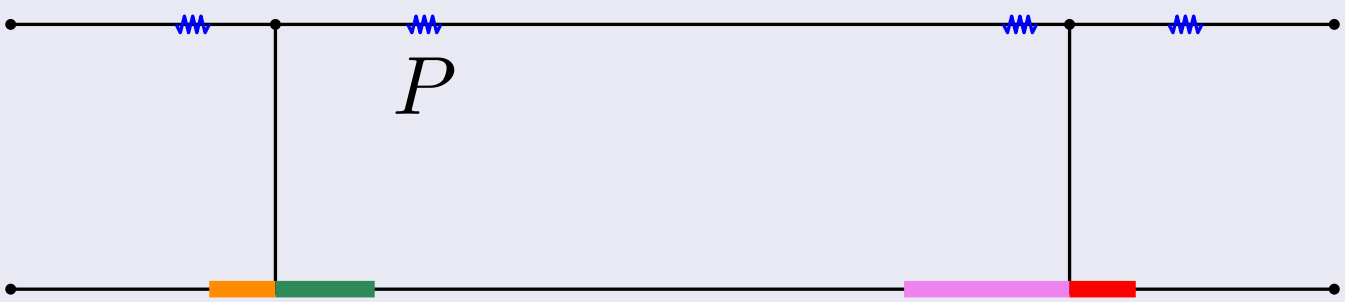
When one side is decided, the positions next to the crossings determine which crossing calls remain uncovered.

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1



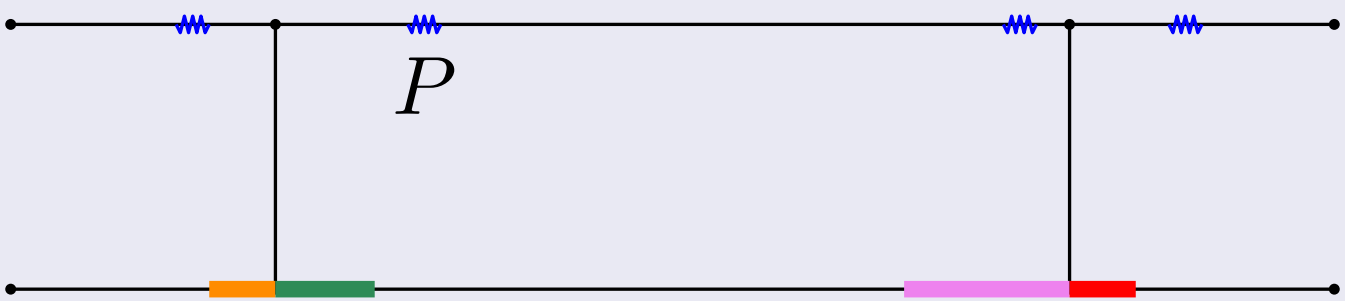
Now want optimal solution to the other side with up to 4 extra intervals added – there are $O(1)$ choices to try for the key positions.

I - Interception

Problem

Given large set of paths in large graph with very special structure, find minimum set of edges that hit all paths.

Solution 1



...one horrible implementation and 20 bugs later: success!

Time complexity: $O((n+m)\log n)$ (though can be made linear at cost of making implementation even more horrible)

I - Interception

If you prefer to remain sane and don't want to take the mildly masochistic path of solution 1...

Solution 2

1. Cover all paths contained in one of the four "tails" optimally, get first device as close to beginning as possible. *Greedy.*
2. Add one extra device to some tails to cut off all paths from the tail to the rest of the graph. *Only* 16 *possibilities; try them all.*
3. Graph and remaining paths now truncated to a circle. *Can be solved with dynamic programming.*
4. Time complexity: $O(n + m)$ (assuming bucket sort or similar)

Statistics: 7 submissions, 0 accepted, first after N/A

Random numbers

296 teams with 722 contestants.

3045 submissions, 802 accepted (26%)[1]

34 number of seconds before end that last accepted submission was submitted.

482 number of lines of code used in total by the shortest **jury** solutions to solve the entire problem set. (154 of those lines for I **Interception**)

[1] These numbers only count submissions up to the first accepted solution on each problem for each team.

Random facts

- All but one of the problems have **near-linear solutions**

  Exception: E (**Exponial**). Basic solution $O(\sqrt{m} \log m)$, input size $O(\log m)$. Very unlikely to have near-linear time solution.

  (Asymptotically, F (**Raffle**) is probably also an exception.)
- Required precision for K (**Dogs**) changed from 10^{-6} to 10^{-4} just a few days before the contest. Many solutions have poor precision when answer is ≈ 0. Problem meant to be easy, not a floating point trap. Unfortunately this change was insufficient, some teams were still tripped up by precision issues.
- I (**Interception**) had the largest number of test cases (125), largest amount of test data ($\approx$ 325 MB), and largest number of intentionally incorrect judge solutions (41).
- The jury wrote Python solutions for almost all problems. Exceptions: C (**Card Hand Sorting**) for no good reason, and I (**Interception**) because painful.