Problem A. Access Points

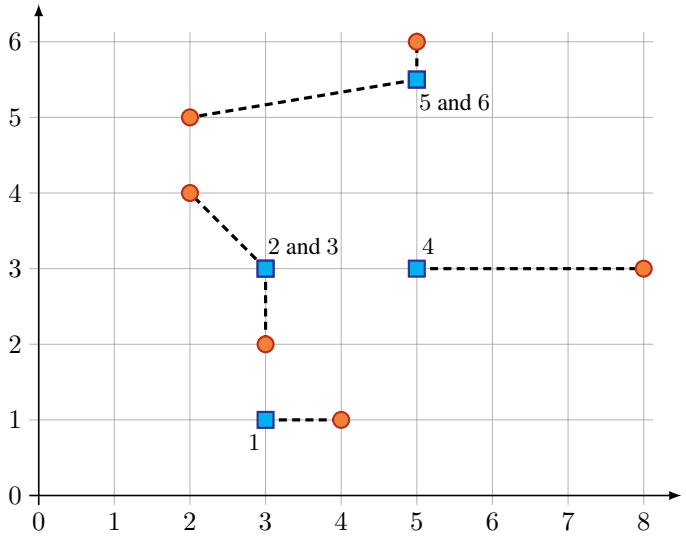
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A well-known programming contest is considering a new way to position its teams. For the contest all n teams have to be assigned some position (x, y) in an infinitely-large gym hall. To keep a good overview of the teams the following strategy is chosen:

All teams have been assigned a unique integer ID in the range $[1, n]$. Any two teams with IDs i and j, where $i < j$, must be placed at positions (x_i, y_i), (x_j, y_j), such that $x_i \leq x_j$ and $y_i \leq y_j$.

Unfortunately, someone already assigned the (fixed) internet access point for each team. The access points are quite big and only have one port, so access points for different teams are located at different positions. Every team must be connected to its designated access point by a direct UTP cable. The cost of a UTP cable of length ℓ is ℓ^2.

Find a placement for all teams, such that their respective order along both axes is maintained and the total cost of the required UTP cables is minimised. As the judges are not too worried about privacy, they are fine with two (or more) teams being placed at the exact same location or being arbitrarily close together. See the figure for an example.



Input

- One line with one integer n ($1 \leq n \leq 10^5$), the number of teams.
- n lines, the ith of which contains two integers s_i, t_i ($1 \leq s_i, t_i \leq 10^6$), the location of the internet access point of team i.

Output

Output the minimum total cost of all UTP cables required to connect the teams to their access points in an optimal legal layout.

Your answer should have an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
6 11 6 23 7 24 11 24 32 27 38 42 42	0.000000000
6 4 1 2 4 3 2 8 3 5 6 2 5	22.500000000

Problem B. Brexit Negotiations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

As we all know, Brexit negotiations are on their way—but we still do not know whether they will actually finish in time.

The negotiations will take place topic-by-topic. To organise the negotiations in the most effective way, the topics will all be discussed and finalised in separate meetings, one meeting at a time.

This system exists partly because there are (non-cyclic) dependencies between some topics: for example, one cannot have a meaningful talk about tariffs before deciding upon the customs union. The EU can decide on any order in which to negotiate the topics, as long as the mentioned dependencies are respected and all topics are covered.

Each of the topics will be discussed at length using every available piece of data, including key results from past meetings. At the start of each meeting, the delegates will take one extra minute for each of the meetings that has already happened by that point, even unrelated ones, to recap the discussions and understand how their conclusions were reached. See the figure for an example.

Nobody likes long meetings. The EU would like you to help order the meetings in a way such that the longest meeting takes as little time as possible.

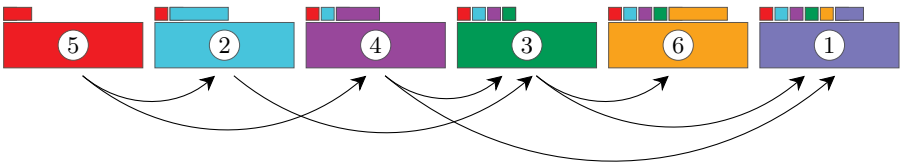


Illustration of how time is spent in each meeting in a solution to Sample Input 2.

Input

The input consists of:

- One line containing an integer n ($1 \leq n \leq 4 \cdot 10^5$), the number of topics to be discussed. The topics are numbered from 1 to n.
- n lines, describing the negotiation topics.

  The ith such line starts with two integers e_i and d_i ($1 \leq e_i \leq 10^6$, $0 \leq d_i < n$), the number of minutes needed to reach a conclusion on topic i and the number of other specific topics that must be dealt with before topic i can be discussed.

  The remainder of the line has d_i distinct integers $b_{i,1}, \ldots, b_{i,d_i}$ ($1 \leq b_{i,j} \leq n$ and $b_{i,j} \neq i$ for each j), the list of topics that need to be completed before topic i.

It is guaranteed that there are no cycles in the topic dependencies, and that the sum of d_i over all topics is at most $4 \cdot 10^5$.

Output

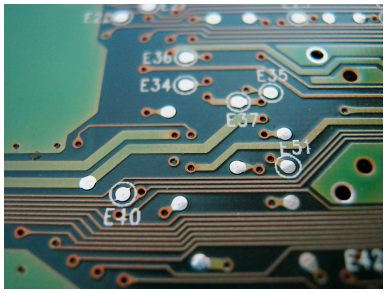
Output the minimum possible length of the longest of all meetings, if meetings are arranged optimally according to the above rules.

Examples

standard input	standard output
3 10 0 10 0 10 0	12
6 2 2 4 3 4 1 5 1 2 2 4 3 1 5 2 0 4 1 3	8

Problem C. Circuit Board Design

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes



You have been hired at the Nano Wiring Efficient Route Company (NWERC) to help with the design of their new circuit boards. The circuits themselves have already been designed, and your task is to come up with a way to print them onto the blank boards that the company has bought.

More specifically, each circuit design consists of a number of connection points with some connections between them such that the resulting graph is connected and does not have any cycles (i.e., the graph is a tree).

You are free to place the connection points anywhere on the circuit board and solder the connections between them so that no two connections intersect (except at the connection points). The boards you ordered are fairly large, so there is no danger of running out of space. You can solder so precisely that connections and connection points can be considered infinitesimal.

This would all be very easy, however your boss persists that each connection needs to be a straight line of length exactly 1 mm (this is, so he says, to make sure the electrons do not have to travel around corners, which would be detrimental to the efficiency of the design).

You soon realise that battling with him will be unsuccessful. Your quickest way out of this is to etch a new design according to his specifications.

Input

The input consists of:

- One line with one integer n ($2 \leq n \leq 1\,000$), the number of connection points. The points are numbered from 1 to n.
- $n-1$ lines, each with two integers a and b ($1 \leq a, b \leq n$), describing a connection between a and b.

It is guaranteed that these edges describe a valid tree.

Output

Output n lines, the ith of which contains two real numbers x_i, y_i, the coordinates of point i. To make the production feasible, the following restrictions apply:

- The distance between each pair of points should be at least 10^{-4}.
- The length of each edge should be 1, up to an absolute error of at most 10^{-6}.
- Edges that are not incident to the same vertex should be at least a distance 10^{-6} apart.

- The coordinates may not exceed an absolute value of 3 000.

If there are multiple valid solutions, you may output any one of them.

Examples

standard input	standard output
5 1 2 1 3 1 4 1 5	0.0000000 0.0000000 1.0000000 0.0000000 0.7071068 0.7071068 0.0000000 1.0000000 -0.7071068 0.7071068
5 2 1 3 1 2 4 2 5	0.0000000 0.0000000 1.0000000 0.0000000 -0.7071068 0.7071068 2.0000000 0.0000000 1.7071068 0.7071068

Problem D. Date Pickup

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Richard and Janet are going on their first date. Richard has offered to meet her at home with his bicycle and Janet tells him she will call when she is ready in 10 to 20 minutes. But Richard is an impatient person; while he could wait at home for Janet's signal, he might also leave early and travel around the neighbourhood for a bit, in order to minimise the time it takes him to reach her once she calls. Due to his impatience, once Richard is on his bicycle, he does not want to ride any slower than the legal speed limit, stop at intersections, or wait outside Janet's house (but he does not mind passing by Janet's house and returning to it later).

Given the directed graph representing the neighbourhood around Richard's and Janet's houses, Richard wants to devise a route around the neighbourhood (after an optional waiting period at his own house) which minimises the time that Janet has to wait in the worst case. He can travel for as long as he likes and visit each intersection as many times as he likes.

Janet will call Richard as soon as she is ready, and at that point, Richard will take the shortest path to her that he can. Richard does not know exactly when Janet will be ready, but he knows it will be in somewhere between a and b minutes (not necessarily at a whole minute).

If Richard is passing through an intersection at the exact same instant Janet calls, the call is considered to happen before he chooses what to do at the intersection. For example, if he is passing by Janet's house at the moment she calls, he can immediately stop there and she does not have to wait for him at all.

It could happen that Janet never has to wait for w minutes, but that she might have to wait for $w - \epsilon$ minutes for arbitrarily small $\epsilon > 0$, if she calls Richard at some inopportune moment (say, nanoseconds after he has left an intersection). In this case, we still define the worst-case waiting time to be w.

Input

The input consists of:

- One line with two integers a, b ($0 \leq a \leq b \leq 10^{12}$), indicating that Janet will be ready in at least a minutes and at most b minutes.
- One line with two integers n, m ($2 \leq n \leq m \leq 10^5$), the number of intersections and the number of roads in the neighbourhood. The intersections are numbered from 1 to n.
- m lines, each with three integers u, v and t ($1 \leq u, v \leq n$, $1 \leq t \leq 10^6$), indicating that there is a one-way road from intersection u to intersection v, and that it takes Richard exactly t minutes to travel along this road.

Richard's house is at intersection 1 and Janet's house is at intersection n. It is guaranteed that it is possible to travel from Richard's house to Janet's house, and that it is possible to exit each intersection through at least one road, even if that road just loops back to the same intersection.

Output

Output the time Janet has to wait in the worst case assuming she will be ready in at least a minutes and at most b minutes and Richard plans his route optimally.

It can be shown that the worst-case waiting time is always an integer.

Examples

standard input	standard output
10 20 3 5 1 3 7 2 1 1 2 3 2 2 3 5 3 2 4	6
4 10 5 7 1 4 6 4 5 5 4 5 3 5 5 30 1 2 1 2 3 1 3 2 1	5

Problem E. Equality Control

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

In programming contest circles, one of the most important roles is that of the Chief Equality Officer (CEO). This person is responsible for making sure that every team has an equal chance of winning the contest. Since last year's NWERC the current CEO, Gregor, has thought at length about how to make the contest even more fair and equal.

His answer is to introduce a new programming language as the only one allowed for submissions. This way, no team will be disadvantaged by not having mastered any of the allowed languages. This language is called Balloon, short for Building A Long List Of Ordinary Numbers. Its only data type is the list of integers. To keep the language fast, it contains only four instructions:

- `[`x_1`,...,`x_n`]` is the constructor for lists. It returns the integers inside the brackets in their given order.
- `concat(<Expr`$_1$`>,<Expr`$_2$`>)` returns a list of all the integers returned when evaluating the expression `<Expr`$_1$`>` followed by all of the integers returned when evaluating `<Expr`$_2$`>`.
- `shuffle(<Expr>)` returns a list of all the integers returned by `<Expr>`, rearranged according to a uniformly random permutation, i.e., each permutation of the elements is used with equal probability.
- `sorted(<Expr>)` returns a list of all the integers returned by `<Expr>`, rearranged into non-decreasing order.

As an example, consider the first expression of Sample Input 1. The two shuffle exressions both take the list `[1,2]` as input and return one of the lists `[1,2]` and `[2,1]`, each with probability 0.5 (independently of each other). The outer concat operator takes the two returned lists as its input and returns their concatenation. I.e., it returns one of the lists `[1,2,1,2]`, `[1,2,2,1]`, `[2,1,1,2]`, and `[2,1,2,1]`, each with probability 0.25.

Naturally, we cannot use byte-by-byte output comparison any more when teams submit their solutions in Balloon, as its output is probabilistic. The judge server instead has to check whether a submitted program is equivalent to the sample solution created by the judges. Two programs are equivalent if for any list L of integers, both programs have an equal probability of returning L.

It is your task to determine whether two given Balloon programs are equivalent.

Input

The input consists of:

- One line containing a string `A`, the first program.
- One line containing a string `B`, the second program.

Each program is a syntactically valid Balloon program with between 3 and 10^6 characters, and contains neither spacing nor empty lists (i.e., the strings " " or "`[]`" do not occur in the input).

Each integer in each program is greater than 0 and less than 10^9.

Output

If the two programs are equivalent, output "`equal`", otherwise output "`not equal`".

Examples

standard input	standard output
concat(shuffle([1,2]),shuffle([1,2])) shuffle([1,2,1,2])	not equal
sorted(concat([3,2,1],[4,5,6])) [1,2,3,4,5,6]	equal
concat(sorted([4,3,2,1]),shuffle([1])) concat(concat([1,2,3],shuffle([4])),sorted([1]))	equal

Problem F. Fastest Speedrun

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

The classic video game "Prince of Python" comprises n levels, numbered from 1 to n. You are going to speedrun this game by finishing all of the levels as fast as possible, and you can beat them in any order that you want.

You enter each level equipped with one of $n + 1$ magical items. In the beginning, you only have item 0 in your inventory. Once you beat a level, you get to keep the item numbered the same as that level. For example, on finishing level 5, you obtain a mighty Gauntlet of 5 Fingers you may equip thereafter instead of the less-acclaimed Sword of 0 Damage you always start out with.

Beating a level can take different amounts of time depending on which item you take into the level with you. Higher-numbered items are more powerful, so if playing by the rules it is always at least as fast to finish the level with a higher-numbered item as with a lower-numbered item.

However, each level also has a shortcut left in by the developers. The shortcut for a level can be accessed by applying a specific item in an unconventional way. By doing so you can finish the level as fast as, or even faster than if you had used any of the other items.

How long will it take you to beat all of the levels of the game?

Input

The input consists of:

- One line containing an integer n ($1 \leq n \leq 2\,500$), the number of levels.
- n lines, describing the levels.

  The ith such line starts with two integers x_i and s_i ($0 \leq x_i \leq n$, $1 \leq s_i \leq 10^9$), the shortcut item for level i and the completion time for level i when using the shortcut.

  The remainder of the line has $n+1$ integers $a_{i,0}, \ldots, a_{i,n}$ ($10^9 \geq a_{i,0} \geq a_{i,1} \geq \ldots \geq a_{i,n} \geq s_i$), where $a_{i,j}$ is the completion time for level i when playing by the rules using item j.

Output

Output the minimum time it takes to beat, in any order, all of the levels in the game.

Examples

standard input	standard output
3 1 1 40 30 20 10 3 1 95 95 95 10 2 1 95 50 30 20	91
4 4 4 5 5 5 5 5 4 4 5 5 5 5 5 4 4 5 5 5 5 5 4 4 5 5 5 5 5	17

Problem G. Game Design

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Carol enjoys playing with wooden games. The objective of the game that fascinates her the most is to tilt a maze, made out of 1 cm-by-1 cm blocks, in any of the four cardinal directions to move a ball to a hole in the centre at $(0, 0)$. As shown in the figure, once the 1 cm wide ball starts moving, it keeps going until either it runs into a wooden block, or it falls into the hole—whichever comes first.

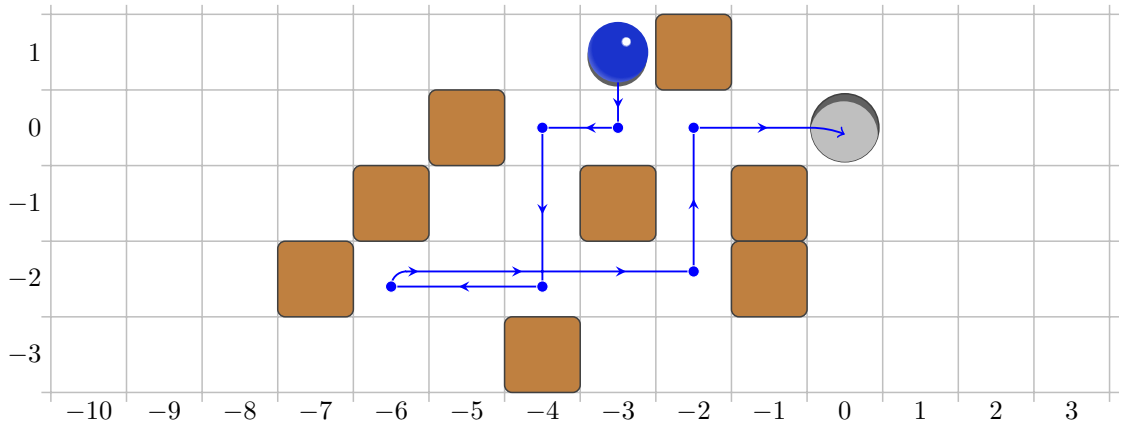


Illustration of Sample Output 1.

Carol is interested in designing a maze of her own, and like any good game designer she already has a fixed solution in mind. This is given as a sequence of tilt moves which must all be followed in order. If any move causes nothing to happen, for example, because the ball is up against a block in that direction or already in the hole, the solution does not count.

The ball can be placed anywhere to start. Carol will take care of adding a square border of blocks covering the rows and columns $10^9 + 1$ cells away from the centre in each direction.

Design a board which can be won by applying her sequence of moves.

Input

The input consists of:

- One line with a string s consisting of only the characters "`LRUD`" ($1 \leq |s| \leq 20$), the sequence of moves. These characters correspond to the directions $-x, +x, +y, -y$ respectively. No two consecutive characters in s are the same.

Output

If it is possible to create a maze with the given solution, first output x and y, the integer coordinates for the ball to start at. Then on the next line, output n, the number of blocks to use. On each of the remaining n lines, output s and t, the integer coordinates of a block.

Otherwise, output "`impossible`".

You may use at most $n \leq 10^4$ blocks. All coordinates used must be at most 10^9 in absolute value. No coordinate pair may be the same as the centre or any other coordinate pair. If there are multiple valid solutions, you may output any one of them.

Examples

standard input	standard output
DLDLRUR	-3 1 8 -1 -1 -1 -2 -2 1 -3 -1 -5 0 -6 -1 -7 -2 -4 -3
LRLRLRLRULD	1 1 5 2 1 2 0 -1 1 -1 0 -1 1000000000
LRLR	impossible

Problem H. Hard Drive

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Pia is getting ready for her flight to the NWERC 2018 in Eindhoven. As she is packing her hard drive, she remembers the airline's ridiculous weight restrictions, which may pose a problem. You see, the hard drive is essentially a string of ones and zeros, and its weight depends on the number of "bit changes" in it: for any two adjacent bits storing two different values, the hard drive gets slightly heavier, so Pia cannot just store arbitrary information on it.

To make matters worse, the drive is so old that some bits are already broken and will always store zeros. The first bit will never be broken, but the last bit will always be.

Pia decides to treat this situation as a challenge: she is now trying to modify the information on the hard drive so that it has exactly the maximum number of bit changes permitted by the airline. However, the broken bits make this harder than expected, so she needs your help.

Find a bit pattern which can be stored on the hard drive and has exactly the desired number of bit changes.

Input

The input consists of:

- One line with three integers n, c, and b ($2 \leq n \leq 5 \cdot 10^5$, $1 \leq c, b \leq n-1$), the size of the hard drive in bits, the desired amount of bit changes, and the number of broken bits. The positions on the hard drive are numbered from 1 to n.
- One line with b integers $z_1, \ldots, z_b$ ($2 \leq z_1 < z_2 < \ldots < z_b = n$), the positions of the broken bits on the hard drive.

Output

Output a bit string of length n, representing Pia's hard drive and containing exactly c bit changes. If there are multiple valid solutions, you may output any one of them. It is guaranteed that at least one solution exists.

Examples

standard input	standard output
5 2 3 2 3 5	00010
7 4 2 2 7	0010110

Problem I. Inflation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

For NWERC 2018, the organisers have done something rather special with the balloons. Instead of buying balloons of equal size, they bought one balloon of every integer size from 1 up to n. A balloon of size s has a capacity of s decilitres.

To avoid inflating the balloons by hand, the organisers also bought n helium gas canisters. Each canister can only be used to inflate one balloon, and must be emptied completely into that balloon (it is not possible to disconnect a canister from a balloon before the canister has been fully used).

Unfortunately the gas canisters were bought at a garage sale, and may contain differing amounts of helium. Some may even be empty! To make the best of this challenging situation, the canisters will have to be paired with the balloons smartly.

The organisers want to assign all of the gas canisters to separate balloons, such that the balloon that is inflated the least (relative to its capacity) still contains the maximum possible fraction of helium inside. What is the maximum such (minimum) fraction that is possible?

Balloons filled beyond their capacity will explode. Explosions are upsetting and must be avoided.

Input

The input consists of:

- One line with the integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of balloons and gas canisters.
- One line with n integers $c_1, \ldots, c_n$ ($0 \leq c_i \leq n$ for each i), the amounts of helium in the gas canisters, in decilitres.

Output

If it is possible to fill all the balloons without any exploding, output the maximum fraction f such that every balloon can be filled to at least f of its capacity. Otherwise, output "-1".

Your answer should have an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
6 6 1 3 2 2 3	0.6
2 2 2	-1
5 4 0 2 1 2	0

Problem J. Jinxed Betting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

For example, suppose as in Sample Input 1 that Julia initially has three points, and there are two other bettors with three and two points respectively. For the first match, she will make the same bet as the other bettor with three points. If this person bets on the losing team and the other bets on the winning team all players have an equal score of three points. If for the next match the other two persons bet on different teams, Julia places the bet on her favourite, which of course may lose. Thus after two more matches she might lose the lead.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Output

Output the number of matches for which Julia is guaranteed to stay in the lead.

Examples

standard input	standard output
3 3 3 2	1
5 8 4 3 5 2	6

Problem K. Kleptography

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

John likes simple ciphers. He had been using the "Caesar" cipher to encrypt his diary until recently, when he learned a hard lesson about its strength by catching his sister Mary browsing through the diary without any problems.

Rapidly searching for an alternative, John found a solution: the famous "Autokey" cipher. He uses a version that takes the 26 lower-case letters '`a`'–'`z`' and internally translates them in alphabetical order to the numbers 0 to 25.

The encryption key k begins with a secret prefix of n letters. Each of the remaining letters of the key is copied from the letters of the plaintext a, so that $k_{n+i} = a_i$ for $i \geq 1$. Encryption of the plaintext a to the ciphertext b follows the formula $b_i = a_i + k_i \bmod 26$.

Mary is not easily discouraged. She was able to get a peek at the last n letters John typed into his diary on the family computer before he noticed her, quickly encrypted the text document with a click, and left. This could be her chance.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 30$, $n + 1 \leq m \leq 100$), where n is the length of the keyword as well as the number of letters Mary saw, and m is the length of the text.
- One line with n lower-case letters, the last n letters of the plaintext.
- One line with m lower-case letters, the whole ciphertext.

Output

Output the plaintext of John's diary.

Examples

standard input	standard output
5 16 again pirpumsemoystoal	marywasnosyagain
1 12 d fzvfkdocukfu	shortkeyword

NWERC 2018 presentation of solutions

NWERC 2018 Jury

- **Per Austrin**
  KTH Royal Institute of Technology
- **Gregor Behnke**
  Ulm University
- **Jeroen Bransen**
  Chordify
- **Alexander Dietsch**
  FAU Erlangen-Nürnberg
- **Arthur van Goethem**
  Eindhoven University of Technology
- **Bjarki Ágúst Guðmundsson**
  Syndis
- **Irina Kostitsyna**
  Eindhoven University of Technology
- **Stefan Kraus**
  FAU Erlangen-Nürnberg
- **Robin Lee**
  Google
- **Simon Lindholm**
  KTH Royal Institute of Technology
- **Lukáš Poláček**
  Innovatrics
- **Alexander Raß**
  FAU Erlangen-Nürnberg
- **Philipp Reger**
  FAU Erlangen-Nürnberg
- **Tobias Werth**
  Google
- **Paul Wild**
  FAU Erlangen-Nürnberg

Big thanks to our test solvers

- **Eduard Kalinicenko**
  Palantir
- **Timon Knigge**
  Google
- **Ragnar Groot Koerkamp**
  Google
- **Jan Kuipers**
  bol.com
- **Dominik Paulus**
  Google
- **Tobias Polzer**
  Google

I: Inflation

Problem Author: Arthur van Goethem

Problem

Assign gas canisters to balloons to maximize $\min(\frac{c_i}{i})$, such that $\forall i. \frac{c_i}{i} \leq 1$

Solution

1. Sort c
2. If $\exists i. \frac{c_i}{i} > 1$, print `impossible` and return
3. Print $\min(\frac{c_i}{i})$

Statistics: 146 submissions, 118 accepted

I: Inflation

Problem Author: Arthur van Goethem

Jury: behind the scenes

`--- November 17, 2018 ---`

- **Robin**: Good, we're all set for the contest.
- *Per has entered the room*
- *Per has successfully challenged* `jeroen.java`
- *Per has successfully challenged* `rgl.java`
- *Per has successfully challenged* `tobi.kt`
- **Jeroen**: What?!
- **Per**: They all use `*CENSORED*` which is `*CENSORED*`.
- **Simon**: Let's not include those cases.
- **Others**: Indeed, because we are so nice.

K: Kleptography

Problem Author: Alexander Dietsch

Problem

Given ciphertext $b_1 b_2 \cdots b_m$ encrypted with the Autokey cipher and last n letters $a_{m-n+1} \cdots a_m$ of plaintext, recover entire plaintext a.

Autokey cipher recap:

- Key $k_1 k_2 \cdots k_n$, gets padded with plaintext ($k_{n+i} = a_i$).
- Plaintext a encrypted by $b_i = (a_i + k_i) \bmod 26$.

Solution

1. For all $i \leq m - n$, we have $a_i = k_{i+n} = (b_{i+n} - a_{i+n}) \bmod 26$.
2. Compute for i from $m - n$ down to 1.
3. Complexity $O(m)$.

Statistics: 126 submissions, 117 accepted

H: Hard Drive

Problem Author: Stefan Kraus

Problem

Create a bitstring of length n such that:

- The number of bit changes is c
- The bits at given positions z_i are all 0

Solution

1. If c is even, start with a 0, otherwise a 1
2. Greedily alternate 0 and 1 where possible, as long as changes are needed

Statistics: 227 submissions, 117 accepted

B: Brexit Negotiations

Problem Author: Gregor Behnke

Problem

Given a DAG with vertex weights $e(i)$, find a topological ordering π that minimizes

$$\max_i (e(\pi(i)) + i).$$

Solution 1 more helpfully

- Basic idea: last vertex should have as small $e(i)$ as possible.
- Adapt standard topological sort.
  - Build schedule from end.
  - Keep adding currently available topic with smallest $e(i)$.
  - Maintain available topics in priority queue.
- Time complexity $O(n \log n + m)$.

Solution 2

- Ideally would like to sort vertices by decreasing value of $e(i)$, but that might violate the precedence constraints.
- Assign potential $p(i) = -e(i)$ for each vertex.
- Propagate potentials: for each predecessor j of i, must have $p(j) \leq p(i) - 1$.
- Sorting by propagated potentials gives a valid and optimal ordering.

Statistics: 212 submissions, 72 accepted

B: Brexit Negotiations

Problem Author: Gregor Behnke

Credit: BBC.com

G: Game Design

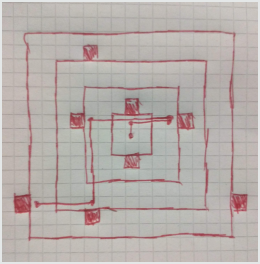
Problem Author: Alexander Dietsch, Robin Lee & Tobias Werth

Problem

Given a sequence of UP/DOWN/LEFT/RIGHT instructions, construct a sliding ball maze with the given instructions as a solution.

Solution

1. Start at centre
2. Each time we turn ± 90 degrees, extend bounding box of maze by 2, and add blocks in both directions at edge of bounding box.
3. At end, shift maze so that we end at $(0, 0)$ rather than starting there.
4. Time complexity $O(n)$.



G: Game Design

Problem Author: Alexander Dietsch, Robin Lee & Tobias Werth

Solution 2

1. Brute force: for each instruction try making it 1 step, 2 steps, etc and recursively solve the rest.
2. To make it fast enough, good to figure out when answer is `impossible`. This happens if and only if input ends with LRL, RLR, or UDU, DUD.

Solution 3

1. Randomly place blocks (make each cell a block with probability 5%).
2. Run walk from $(0, 0)$, if all steps can be executed and we end up in a position not visited earlier in the walk then this gives a solution.

Statistics: 121 submissions, 44 accepted

J: Jinxed Betting

Problem Author: Alexander Raß

Problem

Betting tournament:

- Each round we bet between two options, each correct bet gets a point.
- Julia starts in the lead, has terrible luck but compensates by copying the majority bet from the runners-up.
- For how many rounds will she stay in the lead?

(Not a) Solution

1. In a round with t runners-ups, worst thing that can happen is that their bets are split evenly $\lceil t/2 \rceil$.
2. Naive simulation: $\Theta(r \cdot n)$ where r is number of rounds, which can be as large as 10^{16}.
3. Mildly better simulation: keep track of Julia's lead over the others instead of their scores. $\Theta(r)$ time instead, still a year or so too slow.

J: Jinxed Betting

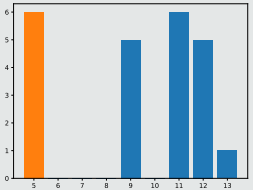
Problem Author: Alexander Raß

Further Analysis

x axis: number of points behind J.

y axis: number of bettors

leftmost bar: group of runners-up



1. In next $1 + \lfloor \log_2 t \rfloor$ rounds, each of the t runners-up catch up in all but one round, remaining bettors catch up in all rounds.
2. If the t runners-up initially have a lead of d over the next group of bettors, this pattern repeats d times, then the two groups are joined and the number of runners-up grows.
3. Keep running this sped-up simulation until J. no longer in the lead.
4. Complexity $O(n \log n)$ for sorting the scores, then $O(n)$ arithmetic operations.

Statistics: 231 submissions, 34 accepted

C: Circuit Design

Problem Author: Paul Wild

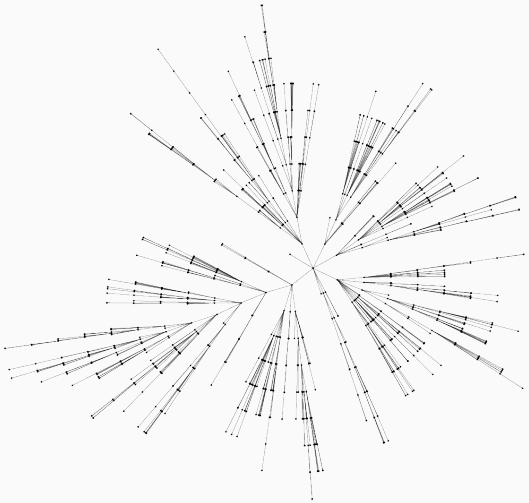
Problem

Given a tree, assign each point a position on a plane such that:

- The distance between any two connected points is (approx.) 1
- Points are not too close together
- Edges do not intersect
- $-3000 \leq x, y \leq 3000$

C: Circuit Design

Problem Author: Paul Wild



C: Circuit Design

Problem Author: Paul Wild

Solution

1. Pick arbitrary root and place at origin
2. Draw points in region between two angles
3. Split region into subregions for children proportional to their width
4. Alternative: Instead of angle, use x-coordinates between -1 and 1, and increase y

Pitfalls

1. Print enough digits (9 could be too few)

Statistics: 190 submissions, 48 accepted

A: Access Points

Problem Author: Arthur van Goethem

Problem

Given list of n points $p_1, \ldots, p_n$ in $\mathbb{R}^2$, find n points $q_1, \ldots, q_n$ such that q_i is componentwise smaller than q_{i+1}, and $\sum_{i=1}^{n} \|q_i - p_i\|_2^2$ is minimized.

Solution

- $\|q_i - p_i\|_2^2 = (x(p_i) - x(q_i))^2 + (y(p_i) - y(q_i))^2$
- x and y coordinates do not interact, solve them separately and add up the answers.
- Problem reformulated:
  given sequence $a_1, \ldots, a_n \in \mathbb{R}$, find $x_1 \leq x_2 \leq \ldots \leq x_n$ such that $\sum (x_i - a_i)^2$ is minimized.

A: Access Points

Problem Author: Arthur van Goethem

The 1D case

- Add item by item.
- When adding x_i, it wants to go to position a_i.
- If a_i smaller than position of previous item, x_i ends up in same position as previous item and pushes it (and possibly more items) towards the left.
- View x_i together with previous item as a new "meta-item".
  (Previous item may already have been a meta-item, so these can get larger and larger.)
- Where does meta-item consisting of items $x_j, \ldots, x_i$ want to go?
  $\sum_{k=j}^{i}(x - a_k)^2$ is minimized by $x = \text{avg}(a_j, \ldots, a_i) = \frac{1}{i-j+1}\sum_{k=j}^{i} a_k$.
- Keep number of items and $\sum a_k$ for each meta-item to quickly be able to add more things to it.
- While positions of the last two meta-items out of order, merge them.
- Time complexity $O(n)$.

Statistics: 26 submissions, 13 accepted

E: Equality Control

Problem Author: Robin Lee

Problem

Consider a programming language whose expressions consist of

- elementary lists: $[x_1, \ldots, x_n]$
- concating two lists: $\mathit{concat}(E_1, E_2)$
- random reordering: $\mathit{shuffle}(E)$
- sorting: $\mathit{sorted}(E)$

Question: Given two expressions, are the distributions the same?

(Actual) Problems

Explicit computation of the distribution is not possible due to size.

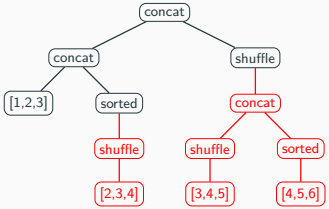
Monte-Carlo sampling also does not work.

E: Equality Control

Problem Author: Robin Lee

Algebra

- Operations below *sorted* or *shuffle* are irrelevant!
- Compact expressions to sequences of lists (sorts can be executed directly) and *shuffle*s.
- Expressions are identical iff sequences are



Pitfalls

- *shuffle*$([1,1,1]) = [1,1,1]$
- Adjacent lists must be joined

E: Equality Control

Problem Author: Robin Lee

Sampling - possibility 2

Take two samples per expression, i.e., $s_1^{E_1}, s_2^{E_1}$ and $s_1^{E_2}, s_2^{E_2}$ with:

1. s_1 : $\mathit{shuffle} = \mathit{sorted}$
2. s_2 : $\mathit{shuffle} = \mathit{reverse} \circ \mathit{sorted}$

If $s_1^{E_1} = s_1^{E_2}$ and $s_2^{E_1} = s_2^{E_2}$ then `equal` else `not equal`.

Statistics: 127 submissions, 20 accepted

D: Date Pickup

Problem Author: Bjarki Ágúst Guðmundsson

Problem

Given a directed graph, find a walk from vertex s minimizing the maximum shortest distance to vertex t during time $[a, b]$.

Solution

1. Distances will be needed... compute distance from s to all vertices, and from all vertices to t, using two runs of Dijkstra's algorithm.
2. Binary search on the answer.
3. Now just need to check if given delay δ is possible.

D: Date Pickup

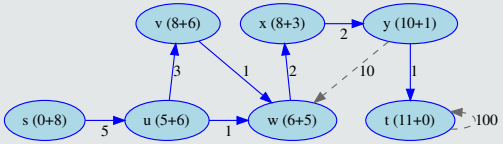
Problem Author: Bjarki Ágúst Guðmundsson

Checking if delay $\leq \delta$ possible

1. Mark vertices u as "good" if $d(s,u) + d(u,t) \leq a + \delta$.
   (If u does not satisfy this, then any route through u will give delay $> \delta$ if signal comes at time a.)
2. Propagate: if u is good and $u \xrightarrow{\ell} v$ with $\ell + d(v,t) \leq \delta$, then mark v and edge as good too.
   (For such edges we get delay $\leq \delta$ if signal comes when traversing the edge.)
3. If subgraph of good edges has a cycle: delay δ is possible.
   (Can just cycle around indefinitely until signal comes.)
4. Otherwise, subgraph is a DAG. Use dynprog to compute longest time we can stay in the subgraph. If this is $\geq b$, delay δ is possible.
5. Complexity $O(n)$ for the check.

Checking if delay $\leq \delta$ possible, example

- $\delta = 10$:
  1. vertex v not initially good ($d(s, v) + d(v, t) = 8 + 6 > a + \delta$).
  2. propagate: edge $u \to v$ and v marked as good.
  3. no cycle, compute longest paths
  4. can stay at s until time $a + \delta - d(s, t) = 4$
  5. propagating $\implies$ can arrive at t at time 18 at the latest
  6. delay 10 *not possible* (but would be possible if $b \leq 18$)



$a = 2, b = 20$.

Legend: "u ($d(s, u) + d(u, t)$)"

D: Date Pickup

Problem Author: Bjarki Ágúst Guðmundsson

Checking if delay $\leq \delta$ possible

1. Mark vertices u as "good" if $d(s, u) + d(u, t) \leq a + \delta$.
   (If u does not satisfy this, then any route through u will give delay $> \delta$ if signal comes at time a.)
2. Propagate: if u is good and $u \xrightarrow{\ell} v$ with $\ell + d(v, t) \leq \delta$, then mark v and edge as good too.
   (For such edges we get delay $\leq \delta$ if signal comes when traversing the edge.)
3. If subgraph of good edges has a cycle: delay δ is possible.
   (Can just cycle around indefinitely until signal comes.)
4. Otherwise, subgraph is a DAG. Use dynprog to compute longest time we can stay in the subgraph. If this is $\geq b$, delay δ is possible.
5. Complexity $O(n)$ for the check.

Statistics: 13 submissions, 1 accepted

F: Fastest Speedrun

Problem Author: Simon Lindholm

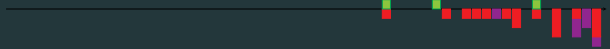
Problem

Beat a game as fast as possibly by solving the levels in an optimal order using special items.

Solution

General idea: smart preprocessing + DP

1. Each level should be solved, so add s_i to answer and subtract from all $a_{i,j}$
2. Now special item can always be used for free
3. The special items form a set of cycles with trees pointed at them
4. For each cycle, find the cheapest $a_{i,n}$, add it to the answer and subtract from cycle
5. Now if we get the highest item, we can solve all the rest for free
6. Now use DP (or Dijkstra) to find the cheapest way of obtaining that
7. $O(n^2)$



Solution 2

- Problem can be reduced to *Minimum directed spanning tree*
- Use algorithm from TCR (Edmonds' algorithm)
- Make sure it is bug-free and runs in $O(n^2)$

Statistics: 25 submissions, 3 accepted

Language stats