ACM-ICPC SouthWestern Europe Regional Contest 2017

Paris, 26 November 2017

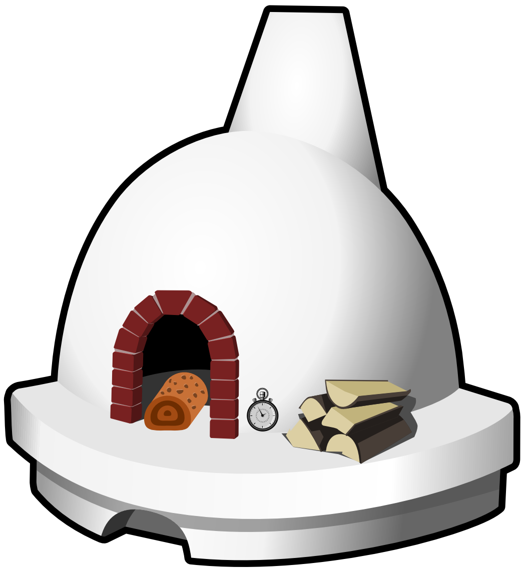
Judges and Problem Setters

- Mehdi Bouaziz (Facebook)
- Christoph Dürr (CNRS, LIP6)
- Jean-Christophe Filliâtre (CNRS, LRI)
- Thomas Huet (Google)
- Ioana Ileana (Université Paris-Descartes)
- Louis Jachiet (École normale supérieure)
- Jacques-Henri Jourdan (CNRS, LRI)
- Silviu Maniu (Université Paris-Sud)
- Raphaël Marinier (Google)
- Pierre Senellart (École normale supérieure), *chief judge*
- Samuel Tardieu (Télécom ParisTech)

This problem set consists of 11 problems, on 23 pages.

This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" license.

A: Cakey McCakeFace



Cakey McCakeFace's signature pastry, the Unknowable Cake, is baked daily in their Paris facility. The make-or-break trick for this cake is the cooking time, which is a very well-kept secret. Eve, the well-known spy, wants to steal this secret, and your job is to help her.

Cakes are cooked in a single huge oven that has exactly one front and one back door. The uncooked cakes are inserted through the front door. After the exact and very secret cooking time has passed, the cakes exit the oven through the back door. Only one cake can go through the front or back door at any given time.

Eve has secretly installed detectors at the front and back of the oven. They record a signal every time a cake passes through the doors. A cake will therefore trigger the entry detector at some time t when it goes through the front door, and then trigger the exit detector at time exactly $t + \mathit{cooking\_time}$ when it goes through the back door (all cakes at Cakey McCakeFace are always perfectly cooked).

After a few days, she receives two sets of timestamps (in ms) corresponding to entry and exit detectors. Unfortunately, the detectors are faulty: they are sometimes triggered when no cake has passed, or they may fail to be triggered when a cake passes. Eve noticed that she could make a good guess of the secret *cooking_time* by finding the time difference that maximizes the number of correspondences of entry and exit detection times. Help Eve compute this.

Input

- **Line 1:** the number N of times the entry detector was triggered.
- **Line 2:** the number M of times the exit detector was triggered.
- **Line 3:** the N integer timestamps at which the entry detector was triggered, sorted in ascending order, with no repetition, space-separated.
- **Line 4:** the M integer timestamps at which the exit detector was triggered, sorted in ascending order, with no repetition, space-separated.

Limits

- $1 \leqslant N, M \leqslant 2\,000$;
- each timestamp is between 0 and 1 000 000 000 (inclusive).

Output

A single integer: your best guess of the secret *cooking_time*, the (positive or zero) time difference that maximizes the number of correspondences of entry and exit detection times. If multiple such values exist, the smallest one should be returned.

Sample Input

```
5
5
0 10 12 20 30
1 5 17 27 50
```

Sample Output

```
5
```

B: Table

Famous chef Clémentine Debœuf is buying new tables for her high-class restaurant. She has decided to go for the latest trend, a large model with numerous, wide, yet delicate ornaments. However, she needs to make sure that these decorations do not hamper the perfectly tuned ballet of dishes.

Ornaments reduce the amount of flat table surface where waiters can safely place dishes. Clémentine wants to make sure that all dishes can be put somewhere on the table in a sufficiently large safe area, that is, without overlapping any ornament. Given the dimensions and locations of all ornamental areas, Clémentine asks you to tell her what impact these ornaments have on the locations available for dishes.

The table is a rectangle of width X and length Y, given in millimeters. On top of it, N ornaments are placed. Each of them is located at fixed table coordinates and shaped like a rectangle whose sides are parallel to the sides of the table. Of course, none of these areas overlap each other, but they can come into contact.

In Clémentine's restaurant, all D dishes are rectangular and placed with their sides parallel to the sides of the table, in a predetermined orientation. Waiters have millimetric precision: They will place dishes at integer millimetric coordinates with their sides parallel to those of the table. Dishes should not overlap any ornamental area (but they can touch its edges). Given a list of dishes described by their dimensions, your task is to tell, for each of them, the number of (integer) locations where it can be safely placed on the table. Note: only one dish is served at a time on the table; this means that you do not need to worry about the fact that dishes could overlap, you can count the locations for each dish independently from others.

Input

The input comprises several lines, each consisting of integers separated with single spaces:

- The first line consists of the integers X, Y, N, and D;
- Each of the N following lines contains the coordinates of an ornament as four integers x, x', y, and y', with $0 \leqslant x < x' \leqslant X$ and $0 \leqslant y < y' \leqslant Y$, describing an ornament spanning between the point (x, y) and the point (x', y');
- The D following lines contain the width x and length y of dishes as two integers with $0 < x \leqslant X$ and $0 < y \leqslant Y$.

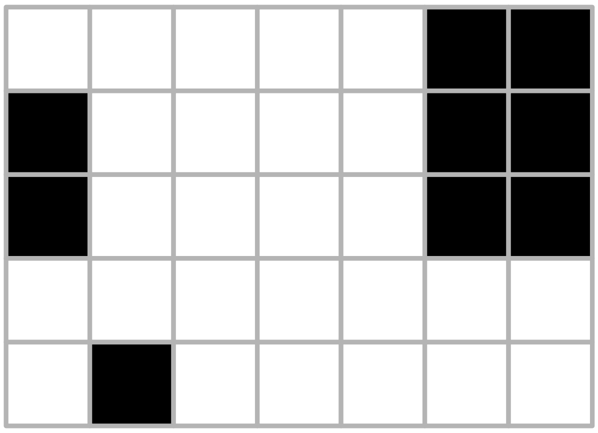
Limits

- $1 \leqslant X, Y \leqslant 2\,000$;
- $0 \leqslant N \leqslant 1\,000\,000$;
- $1 \leqslant D \leqslant 100\,000$.

Output

D lines containing the number of valid integer locations for each dish.

Example



Sample Input

```
7 5 3 9
1 2 0 1
5 7 2 5
0 1 2 4
7 1
3 5
5 3
2 2
3 3
4 4
4 5
6 2
1 1
```

Sample Output

```
1
1
0
13
5
1
0
0
26
```

C: Macarons

Pierre is famous for his macarons. He makes round macarons, stored in square boxes of size 1×1, and oval-shaped macarons, stored in rectangular boxes of size 1×2 (or, rotated, in rectangular boxes of size 2×1). For the purpose of a buffet, Pierre wishes to tile a rectangular table of size $N \times M$ with the two kinds of macarons, meaning that the table must be completely full, with no empty space left. The width N of the table is small, for the guest to be able to grab the macarons easily, and the length M of the table is large, to accommodate a huge number of guests. To keep the table pretty, the orientation of macarons should always be aligned with the sides of the table.

Pierre wishes to know how many ways there are to tile the table. Can you help him?

Input

The input consists of the following integers:

- the value of N, an integer, on the first line;
- the value of M, an integer, on the second line.

Limits

The input satisfies $1 \leqslant N \leqslant 8$ and $1 \leqslant M \leqslant 10^{18}$.

Output

The output should consist of the total number of tilings, given modulo 10^9, on a single line.

Sample Input

```
2
2
```

Sample Output

```
7
```

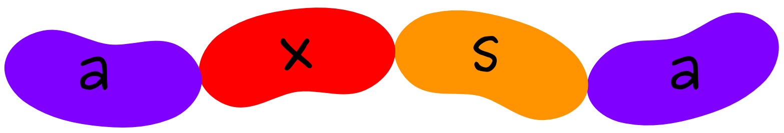
Sample Input

```
2
4
```

Sample Output

```
71
```

D: Candy Chain



A Candy Chain is a sequence of individual candies. Candies come in 26 different flavors identified by the lowercase letters *a* to *z*. Margot has a particularly fancy Candy Chain displayed in her shop.

After school, children will come to her and buy portions of the Candy Chain. Every child has different preferences. For example there is a child who likes portions with flavors *ababi*, and is willing to pay 3 euros for it. Another child likes portions with flavors *axsa* and is willing to pay 5 euros for it.

Margot can extract portions of the Candy Chain and sell them to the children. When she extracts a portion, she joins the remaining left and right parts of the Candy Chain, and can then continue serving additional portions, or decide to stop.

The same sequence of flavors can be sold multiple times to the same child (as long as Margot is able to extract multiple instances of it from her Candy Chain). Margot never throws away candies if she cannot sell them. She can reverse candy portions while selling them (e.g., *axsa* and *asxa* are equivalent). Margot does not have to serve all children, and she does not have to serve the children in any particular order.

Your task is to help Margot compute the maximum amount she can earn from her Candy Chain.

Input

- Line 1 represents Margot's Candy Chain as a non-empty string of characters.
- Line 2 consists of the number of children, an integer C.
- The C following lines represent the preferences of each child as two items separated by a space: his or her preferred candy portion, represented by a non-empty string of characters; and what he or she is willing to pay, represented by an integer P_i.

Limits

- Margot's Candy Chain, as well as the candy portion of every child, is formed of at most 50 candies;
- $1 \leqslant C \leqslant 200$;
- for all $1 \leqslant i \leqslant C$, $0 \leqslant P_i \leqslant 1\,000\,000$;
- all strings are formed of the lowercase characters `a` to `z` only.

Output

A single integer: the maximum amount Margot can earn.

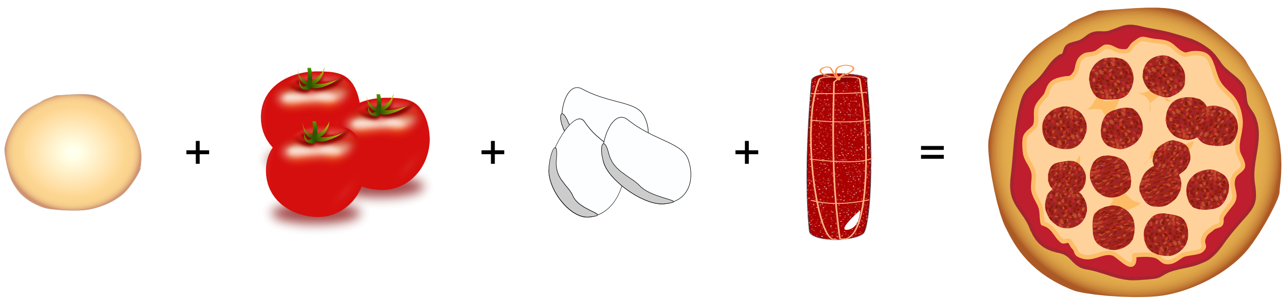
Sample Input

```
acmicpcxxxacmzacmzacmzmca
5
icpc 5
cpci 1
acm 2
acmacm 10
xxx 0
```

Sample Output

```
21
```

E: Ingredients



The chef of a restaurant aspiring for a Michelin star wants to display a selection of her signature dishes for inspectors. For this, she has allocated a maximum budget B for the cumulated cost, and she wants to maximize the cumulated prestige of the dishes that she is showing to the inspectors.

To measure the prestige of her dishes, the chef maintains a list of recipes, along with their costs and ingredients. For each recipe, a derived dish is obtained from a base dish by adding an ingredient. The recipe mentions two extra pieces of information: the cost of applying the recipe, on top of the cost of the base dish, and the prestige the recipe adds to the prestige of the base dish. The chef measures the prestige by her own units, called "prestige units."

For example, a recipe list for making pizza looks like:

- `pizza_tomato pizza_base tomato 1 2`
- `pizza_classic pizza_tomato cheese 5 5`

Here, `pizza_base` is an *elementary dish*, a dish with no associated recipe, a dish so simple that its cost is negligible (set to 0) and its prestige also 0. The chef can obtain the *derived dish* `pizza_tomato` by adding the *ingredient* `tomato` to the *base dish* `pizza_base`, for a cost of 1 euro and a gain of 2 prestige units. A `pizza_classic` is obtained from a `pizza_tomato` by adding `cheese`, for an added cost of 5, and a prestige of 5 added to the prestige of the base dish; this means the total cost of `pizza_classic` is 6 and its total prestige is 7.

A signature dish selection could for instance include both a `pizza_tomato` and a `pizza_classic`. Such a selection would have cumulated total prestige of 9, and cumulated total cost of 7.

Armed with the list of recipes and a budget B, the chef wants to provide a signature dish selection to Michelin inspectors so that the cumulated total prestige of the dishes is maximized, keeping their cumulated total cost at most B.

Important Notes

- No dish can appear twice in the signature dish selection.
- Any dish that does not appear as a derived dish in any recipe is considered to be an elementary dish, with cost 0 and prestige 0.
- A dish can appear more than once as a resulting dish in the recipe list; if there is more than one way to obtain a dish, the one yielding the smallest total cost is always chosen; if the total costs are equal, the one yielding the highest total prestige should be chosen.
- The recipes are such that no dish D can be obtained by adding one or more ingredients to D itself.

Input

- The first line consists of the budget B, an integer.
- The second line consists of the number N of recipes, an integer.
- Each of the following N lines describes a recipe, as the following elements separated by single spaces: the derived dish name (a string); the base dish name (a string); the added ingredient (a string); the added price (an integer); the added prestige (an integer).

Limits

- $0 \leqslant B \leqslant 10\,000$;
- $0 \leqslant N \leqslant 1\,000\,000$;
- there can be at most 10 000 different dishes (elementary or derived);
- costs and prestiges in recipes are between 1 and 10 000 (inclusive);
- strings contain at most 20 ASCII characters (letters, digits, and '_' only).

Output

The output should consist of two lines, each with a single integer. On the first line: the maximal cumulated prestige within the budget. On the second line: the minimal cumulated cost corresponding to the maximal cumulated prestige, necessarily less than or equal to the budget.

Sample Input

```
15
6
pizza_tomato pizza_base tomato 1 2
pizza_cheese pizza_base cheese 5 10
pizza_classic pizza_tomato cheese 5 5
pizza_classic pizza_cheese tomato 1 2
pizza_salami pizza_classic salami 7 6
pizza_spicy pizza_tomato chili 3 1
```

Sample Output

```
25
15
```

F: Shattered Cake

A rectangular cake is transported via a truck to a restaurant. On the way to the destination, the truck hits a pothole, which shatters the cake in N perfectly rectangular pieces of width w_i and length l_i, for $1 \leqslant i \leqslant N$.

At the destination, the damage is assessed, and the customer decides to order a replacement cake of the same dimensions. Unfortunately, the original order form was incompletely filled and only the width W of the cake is known. The restaurant asks for your help to find out the length L of the cake. Fortunately, all pieces of the shattered cake have been kept.

Input

The input consists of the following integers:

- on the first line, the width W of the cake;
- on the second line, the number N of shattered pieces;
- on each of the next N lines, the width w_i and length l_i of each piece.

Limits

- $1 \leqslant N \leqslant 5\,000\,000$;
- $1 \leqslant W, L \leqslant 10\,000$;
- for each $1 \leqslant i \leqslant N$, $1 \leqslant w_i, l_i \leqslant 10\,000$.

Output

The output should be the integer L.

Sample Input

```
4
7
2 3
1 4
1 2
1 2
2 2
2 2
2 1
```

Sample Output

```
6
```

G: Cordon Bleu

A Parisian entrepreneur has just opened a new restaurant “Au bon cordon bleu”, named after a famous French recipe. However, no one has any idea of which wine would be appropriate with such a dish. The entrepreneur plans to sample many different wines in order to build the wine menu.

The various bottles of wine he plans to taste can be obtained from different wine merchants located in or around Paris. Being a very sensitive product, high-quality wine can only be transported by highly trained couriers on motorbikes. Therefore, those couriers are very expensive.

A courier can be used for the transportation of several wine bottles, but can transport only one bottle at a time. All couriers get paid at the same fixed rate: one euro per kilometer. The distance function used is the Manhattan distance (also known as taxicab metric) of every individual segment of the trip: the distance from a point (x_1, y_1) to a point (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

A courier in charge of transporting a single wine bottle will get paid as many euros as the sum of the following two (Manhattan) distances: from her base to the wine merchant place, and from the wine merchant place to the restaurant.

Consider a more complex example: a courier in charge of transporting two wine bottles, one after the other. The amount paid will be the sum of the following distances: from his base to the location of the first bottle, then to the restaurant, then to the location of the second bottle, then to the restaurant.

Help the entrepreneur minimize the costs of hiring the couriers. Given a set of Cartesian coordinates corresponding to available couriers, a set of Cartesian coordinates corresponding to the locations of the precious wine bottles they need to collect, and the location of the restaurant, compute the smallest number of kilometers that the couriers will be paid for. There is no obligation to use all available couriers, and bottles can be collected in an arbitrary order.

Input

The input comprises several lines, each consisting of integers separated with single spaces:

- The first line consists of the number N of wine bottles to collect and the number M of available couriers.
- The N following lines consist of the coordinates of each bottle as two integers x and y.
- The M following lines consist of the coordinates of each courier’s base as two integers x and y.
- The last line contains the coordinates of the restaurant as two integers x and y.

Limits

- $1 \leqslant N \leqslant 1\,000$;
- $1 \leqslant M \leqslant 1\,000$;
- all coordinates (x, y) verify $-1\,000 \leqslant x \leqslant 1\,000$ and $-1\,000 \leqslant y \leqslant 1\,000$.

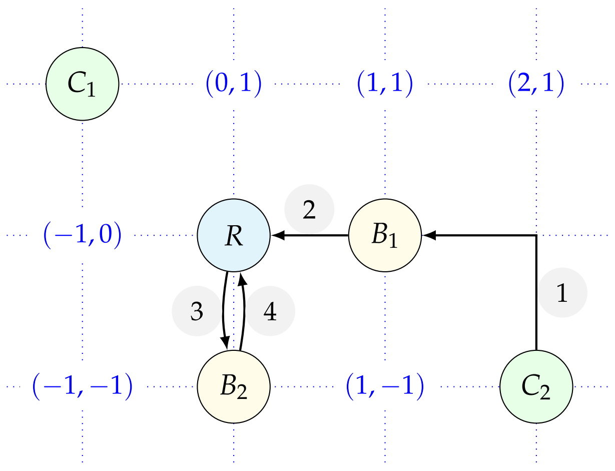
Output

A single integer: the smallest number of euros that needs to be paid to collect all bottles.

Notes

There might be more than one item at the same initial location. For example, it would be possible for two bottles, ten couriers, and the restaurant to share the same starting position.

Example



On this example, only one courier (C_2) is used to retrieve the two bottles B_1 and B_2, and bring them to the restaurant R by performing the moves labeled 1 to 4 in succession. The total number of kilometers is 5. This is one of the optimal solutions.

Sample Input

```
2 2
1 0
0 -1
-1 1
2 -1
0 0
```

Sample Output

```
5
```

H: Kabobs

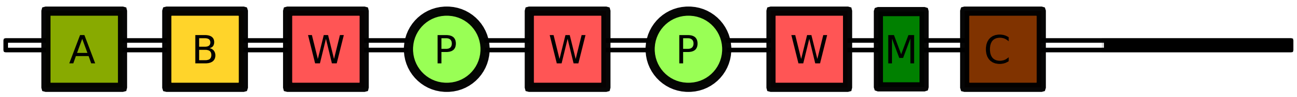
Anna wants to open a marvelous restaurant, "Candy Mountain", serving only candy *kabobs*: sticks on which one puts various pieces of food, to be eaten from the tip to the base.

Like other kabob enthusiasts, when Anna eats a pattern of consecutive types of food in a kabob, she expects to eat another pattern later in the kabob. For instance, once she eats a piece of apple immediately followed by a piece of banana, she expects to have a leaf of mint immediately followed by chocolate in the remaining part of the kabob. She is happy if she finds this mint-chocolate pattern anywhere in the remaining kabob pieces.

Here is a kabob that Anna likes:

```
Apple-Banana-Watermelon-Plum-Watermelon-Plum-Watermelon-Mint-Chocolate
```

or, graphically:



Anna has written down her perfect set of kabob patterns for the new restaurant, but she worries that these rules would give too many potential kabob types. This set of patterns is written down as a *ruleset*, where each rule is of the form "b implies e afterwards", where b and e are non-empty sequences of characters, representing food pieces. A rule of the form $b > e$ with $b = b_1 \dots b_k$ and $e = e_1 \dots e_l$ means that, if the pattern b is encountered in the kabob, then the kabob should also contain e at some point after. Each b_{i+1} must immediately follow b_i to trigger the rule and similarly each e_{j+1} must immediately follow e_j to satisfy it, but b_k and e_1 do not need to be consecutive. No food piece can appear more than once in a rule (i.e., there is no i, j such that $e_j = b_i$ and no $i \neq j$ with $b_i = b_j$ or $e_i = e_j$) but a food piece can appear in several rules.

Note that if there are several occurrences of the word b in a kabob, they all need to be followed by an e. This can be a single e, as long as this e appears after all the b.

In a ruleset, rules are separated by '`|`' and are of the form $u > v$ meaning that each pattern u implies a pattern v afterwards. u and v are words composed of alphanumerical characters and no character can appear twice in a rule. For instance, the ruleset "`AB>X|R>A|T>B`" describes three rules:

- for each `AB` there must be an `X` afterwards;
- for each `R` there must be an `A` afterwards; and
- for each `T` there must be a `B` afterwards.

Using this ruleset, the kabobs `SBSB`, `REA`, `ABX`, `BA`, `ABXBA`, `RRA`, `TBTB`, and `RTABX` are valid; but `RAT`, `TAB`, and `ABXAB` are not.

Anna asks you how many kabobs of a given size are compatible with her ruleset.

Input

- The first line contains an integer K, the size of all kabobs, followed with a space, and a non-empty string S of alphanumerical characters ('`A`' to '`Z`', '`a`' to '`z`', and '`0`' to '`9`'), representing the elements that can be used in a kabob (no characters appear twice in S).
- The second line contains a non-empty string R, which contains no spaces and represents a ruleset as described above. The patterns in R are composed of characters from S only.

Limits

The input is such that $1 \leqslant K \leqslant 500$ and $3 \leqslant |R| \leqslant 60$.

Output

One integer: the number of kabobs of length K satisfying all the rules in R, modulo 10 000 000.

Sample Input

```
4 ABC
A>B|B>C|CB>A
```

Sample Output

```
9
```

I: Burglary

The giant Candy-Bar Wall in the Royal Kitchen is used to store… well, candy bars. These are placed in jars on N equal-length shelves. The shelves are positioned in a vertical progression from the ground up, and perfectly aligned horizontally on their rightmost and leftmost edges. Each shelf is partitioned into M equal-sized slots, which can either be empty or occupied by a jar. A jar contains between 1 and 9 candy bars.

Every shelf is connected to the one directly below (or to the floor, for the bottommost shelf) by one or more vertical ladders. A ladder connects a slot to the corresponding slot on the shelf below, or to the floor. There is at most one ladder directly under any given slot.

The topmost shelf does not contain any jars, but just above it there is a huge open window leading to the Royal Kitchen rooftop. Mini-Fierce-And-Hungry Bandit has surprisingly managed to get into the Royal Kitchen via this top window and now plans a massive candy-bar burglary. More precisely, he intends to do a fun-and-profit "round trip" across the wall, that is:

- start from the topmost shelf;
- move down 0 or more shelves, possibly until reaching the floor;
- and then move up again, to the top window through which he will gracefully exit;

while of course grabbing as many candy bars as possible during this trip.

The major issue the bandit faces however is that he cannot enter a slot containing a candy jar more than once, since this would trigger a dreadful Candy Alarm.

Your mission: help the bandit carefully plan his round trip across the Candy-Bar Wall so as to maximize the number of candy bars he grabs, without triggering any alarm.

Input

The first line consists of two integers: N (number of shelves) and M (number of slots on a shelf), separated by a space. The following $2N$ lines each consist of strings of length M depicting, in an ASCII art manner, the shelf and ladder configuration:

- Line $2k$, for $1 \leqslant k \leqslant N$, comprises the characters '`-`', '`1`', …, '`9`', where a digit x represents a jar containing x candy bars, and '`-`' denotes an empty slot.
- Line $2k+1$, for $1 \leqslant k \leqslant N$, comprises the characters '`.`' and '`|`', where '`|`' represents a ladder, and '`.`' means empty wall space.

Limits

- $1 \leqslant N \leqslant 1\,000$;
- $1 \leqslant M \leqslant 5\,000$;
- there are between 1 and 10 ladders directly under any shelf.

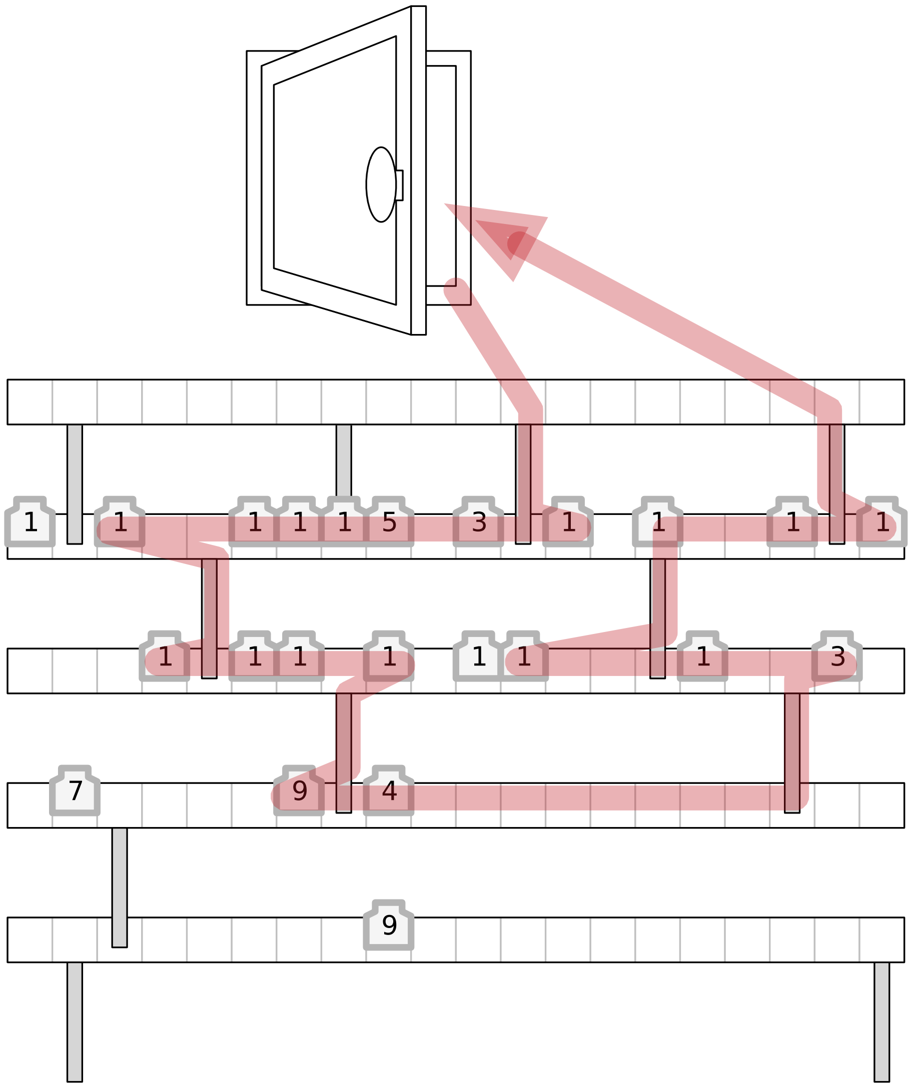
Output

The output consists of an integer on a single line, representing the maximum number of candy bars the bandit can collect without triggering any alarm.

Notes

- Slots corresponding to ladder endpoints may contain jars.
- A slot may be entered either by walking (left to right or right to left) on a shelf, or by reaching the endpoint of a ladder. Slots corresponding to a ladder's endpoints are necessarily entered if the ladder is used.
- There are no candy jars on the topmost shelf and on the floor.

Example



Sample Input

```
5 20
--------------------
.|.....|...|......|.
1-1--1115-3-1-1--1-1
....|.........|.....
---1-11-1-11---1--3-
.......|.........|..
-7----9-4-----------
..|.................
--------9-----------
.|.................|
```

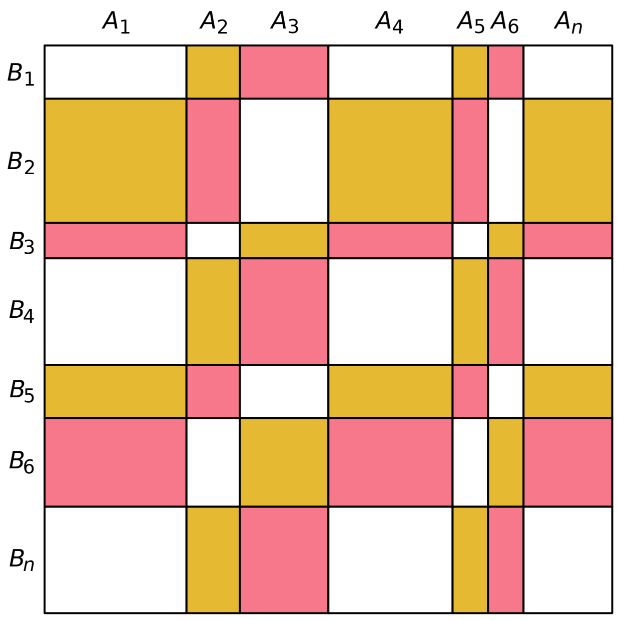
Sample Output

```
38
```

J: Frosting on the Cake

Iskander the Baker is decorating a huge cake, covering the rectangular surface of the cake with frosting. For this purpose, he mixes frosting sugar with lemon juice and food coloring, in order to produce three kinds of frosting: yellow, pink, and white. These colors are identified by the numbers 0 for yellow, 1 for pink, and 2 for white.

To obtain a nice pattern, he partitions the cake surface into vertical stripes of width $A_1, A_2, \ldots, A_n$ centimeters, and horizontal stripes of height $B_1, B_2, \ldots, B_n$ centimeters, for some positive integer n. These stripes split the cake surface into $n \times n$ rectangles. The intersection of vertical stripe i and horizontal stripe j has color number $(i + j) \bmod 3$ for all $1 \leqslant i, j \leqslant n$. To prepare the frosting, Iskander wants to know the total surface in square centimeters to be colored for each of the three colors, and asks for your help.



Input

The input consists of the following integers:

- on the first line: the integer n,
- on the second line: the values of $A_1, \ldots, A_n$, n integers separated with single spaces,
- on the third line: the values of $B_1, \ldots, B_n$, n integers separated with single spaces.

Limits

The input satisfies $3 \leqslant n \leqslant 100\,000$ and $1 \leqslant A_1, \ldots, A_n, B_1, \ldots, B_n \leqslant 10\,000$.

Output

The output should consist of three integers separated with single spaces, representing the total area for each color 0, 1, and 2.

Sample Input

```
3
1 1 1
1 1 1
```

Sample Output

```
3 3 3
```

Sample Input

```
7
6 2 4 5 1 1 4
2 5 1 4 2 3 4
```

Sample Output

```
155 131 197
```

K: Blowing Candles

As Jacques-Édouard really likes birthday cakes, he celebrates his birthday every hour, instead of every year. His friends ordered him a round cake from a famous pastry shop, and placed candles on its top surface. The number of candles equals the age of Jacques-Édouard in hours. As a result, there is a huge amount of candles burning on the top of the cake. Jacques-Édouard wants to blow all the candles out in one single breath.

You can think of the flames of the candles as being points in the same plane, all within a disk of radius R (in nanometers) centered at the origin. On that same plane, the air blown by Jacques-Édouard follows a trajectory that can be described by a straight strip of width W, which comprises the area between two parallel lines at distance W, the lines themselves being included in that area. What is the minimum width W such that Jacques-Édouard can blow all the candles out if he chooses the best orientation to blow?

Input

The first line consists of the integers N and R, separated with a space, where N is Jacques-Édouard's age in hours. Then N lines follow, each of them consisting of the two integer coordinates x_i and y_i of the ith candle in nanometers, separated with a space.

Limits

- $3 \leqslant N \leqslant 2 \cdot 10^5$;
- $10 \leqslant R \leqslant 2 \cdot 10^8$;
- for $1 \leqslant i \leqslant N$, $x_i^2 + y_i^2 \leqslant R^2$;
- all points have distinct coordinates.

Output

Print the value W as a floating point number. An additive or multiplicative error of 10^{-5} is tolerated: if y is the answer, any number either within $[y - 10^{-5}; y + 10^{-5}]$ or within $[(1 - 10^{-5})y; (1 + 10^{-5})y]$ is accepted.

Sample Input

```
3 10
0 0
10 0
0 10
```

Sample Output

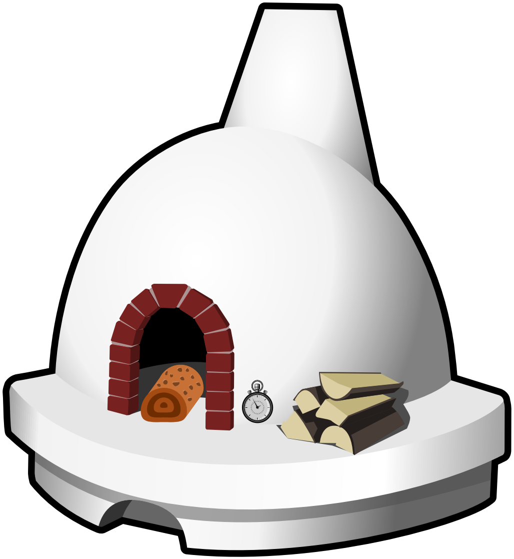
```
7.0710678118654755
```

Problem Analysis Session

SWERC judges

November 30, 2017

A - Cakey McCakeFace



A - Cakey McCakeFace

Algorithm

Iterate over the two sets and count the occurences of the differences with a hash map.

Complexity

$O(n^2)$ (time and space)

Python Solution

```
def solve(A, B):  # A, B are list(int).
    C = collections.Counter(b - a for b in B for a in A
                                  if b - a >= 0)
    occ, negative_offset = max(((C[k], -k) for k in C),
                                    default=(0,0))
    return -negative_offset
```

A - Cakey McCakeFace

Other algorithm in $O(n^2 \log(n))$

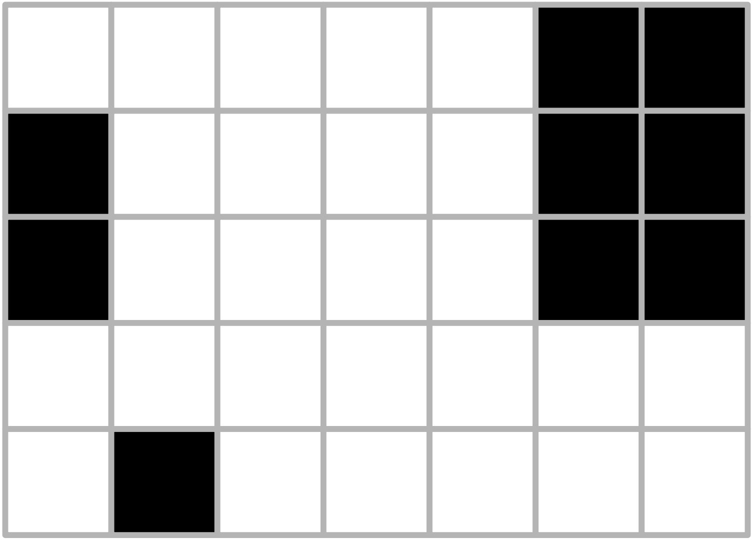
- Maintain a heap containing, for each elemt of the second set, the smallest time shift for matching an element of the first set.
- Iterate over time shifts stored in the heap, update the heap as we go.

$O(n^2 \log(n))$ in time, $O(n)$ in space.

Running time

- In practice, $\sim 5x$ faster than $O(n^2)$ solution naively implemented until memory becomes an issue.
- Cause: CPU stalled on main memory latency (a few tens of ns).

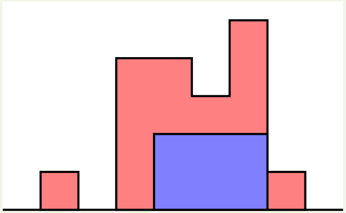
B - Table



B - Table

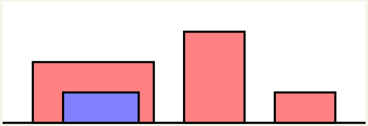
First simplifications

- Lots of ornaments: this is **just a bitmap**.
- Lots of queries ⇒ We **compute all the results**.
- Fix the low y coordinate of counted rectangles, then accumlate.



Simplified version

The **free** area contains only **separated rectangles**:

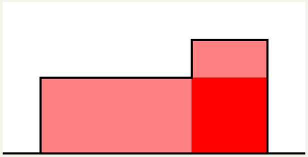


Cumulative array

- $+1$ at the size of every red rectangle
- Sum twice on x, once on y

B - Table

What if rectangles intersect?



Count -1 for intersection

Solution of the full problem in $O(X \times Y + D)$

- Enumerate all the maximal free rectangles
  - Use classical algorithm: “largest rectangle of zeros”
- Use a cumulative array
  - Count $+1$ for each maximal rectangle, and intersections negatively

C - Macarons

C - Macarons

The problem

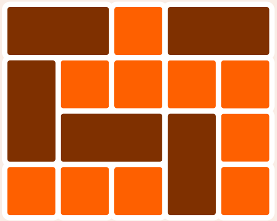
tiling a $N \times M$ grid with monominos and dominos

Homage

to Pierre Hermé, of course

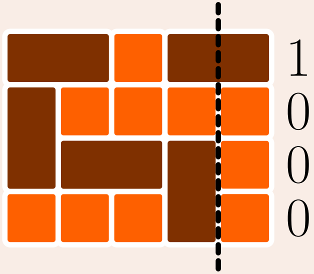
Example

one of the 120,465 solutions for $N = 4$ and $M = 5$



C - Macarons

Transition



Transition matrix

$T[i][j]$ is the number of columns with left mask i and right mask j

Solution

the number of path of length M from 0 to 0, that is

$$T^M[0][0]$$

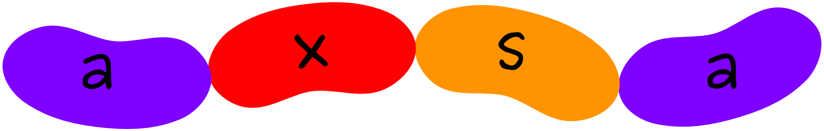
Algorithmic techniques

- Fast exponentiation
- Matrix multiplication
- Modulo arithmetic

Complexity

- matrix has size $2^N \times 2^N$
- one multiplication costs $(2^N)^3$
- overall complexity is $(2^N)^3 \times \log(M)$

D - Candy Chain



D - Candy Chain

Key idea

Dynamic programming: Compute $\mathbf{F(i, j, require\_full\_consumption, p, k)}$, the maximum score of selling the Candy Chain range $[\mathbf{i}, \mathbf{j})$ given:

- Prefix $[\mathbf{0}, \mathbf{k})$ of child's portion $\mathbf{p}$ was already produced from prefix $[\mathbf{0}, \mathbf{i})$ of the Candy Chain.
- Full consumption of range $[\mathbf{i}, \mathbf{j})$ is required depending on **require_full_consumption** (boolean).

D - Candy Chain

Computing **F**

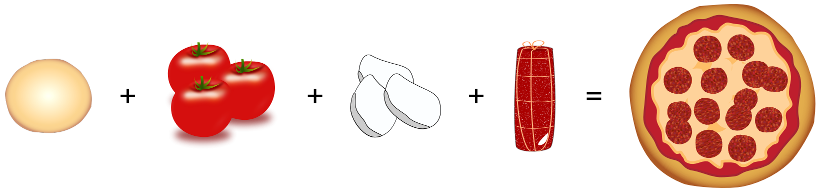
At state $\mathbf{i}, \mathbf{j}, \mathbf{require\_full\_consumption}, \mathbf{p}, \mathbf{k}$ we can:

- Make immediate progress on the current child portion **p** (if $\mathbf{candy\_chain[i]} == \mathbf{portions[p][k]}$) using $\mathbf{F(i+1, j, require\_full\_consumption, p, k+1)}$
- For $\mathbf{m} \in [\mathbf{i+1}, \mathbf{j}]$, try to skip $\mathbf{candy\_chain[i..m)}$ for the current child portion:
  - Maximize score for the skipped range $[\mathbf{i}, \mathbf{m})$ using: $\mathbf{F(i, m, -1, true)}$ (require full consumption of this range, no child portion already consumed)
  - Continue current child portion **p** after the skipped range with: $\mathbf{F(m, j, p, require\_full\_consumption)}$

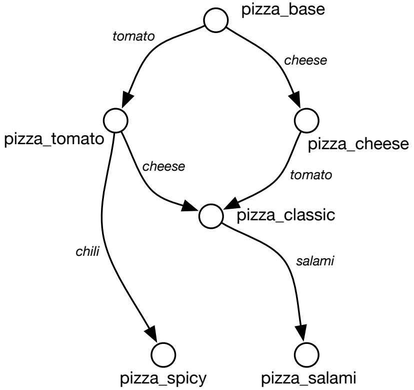
Complexity

$O(N^4 \times W)$ in time, $O(N^3 \times W)$ in space.

E - Ingredients



E – Ingredients

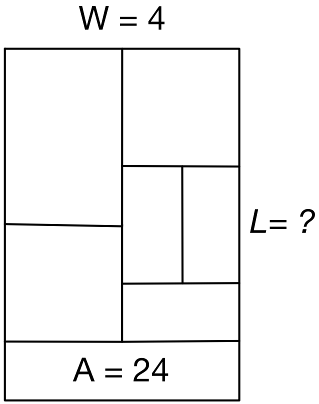


The solution combines *shortest paths* and *0/1 knapsack* algorithms:

1. the recipes form a DAG: compute first the topological sort of the recipe graph, and then compute in $O(N)$ time the dish costs;
2. dynamic program for the knapsack problem in $O(NB)$, using the costs and prestiges.

F - Shattered Cake

F – Shattered Cake



We know that we have all the pieces of the cake and they cannot be rotated, so we simply have to divide the *total area* by the given width W:

$$L = \frac{\sum_{1 \leqslant i \leqslant N} w_i \cdot l_i}{W}.$$

G - Cordon Bleu

G - Cordon bleu

Fitting a known problem

1. If every courier could handle exactly one bottle, we could solve a maximum bipartite matching problem of minimum weight (*a.k.a* assignment problem).
2. By introducing $N_b - 1$ additional virtual couriers starting from the restaurant, we can represent extra fares by a courier.
3. We can now match every bottle with an exclusive courier.

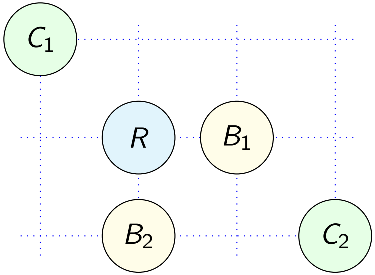
Solving the assignment problem

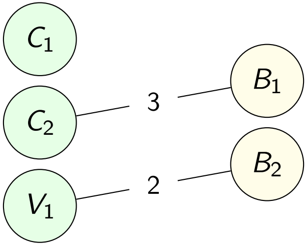
1. The matching can be computed in $O(n^3)$ using the Kuhn-Munkres algorithm (*a.k.a* the Hungarian method).

G - Cordon bleu

Example

One courier (out of two) will take care of delivering both bottles. One virtual courier V_1 is introduced at R.

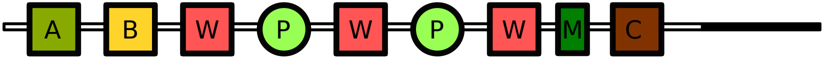




	C_1	C_2	V_1
B_1	4	**3**	2
B_2	4	3	**2**

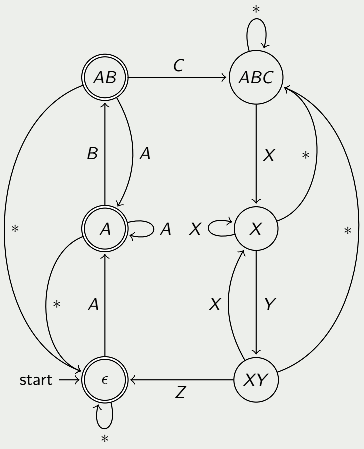
$\Rightarrow$ Total cost is 5

H - Kabobs



H - Kabobs

Automaton for rule $ABC > XYZ$



Algorithmic techniques

- Remove inaccessible states
- Use a default transition
- Counting paths of given size

Complexity

O(Size of the automaton $\times$ #steps)

too much?

H - Kabobs

Number of states

Each automaton for a rule has *rulesize* states thus:

$$\#states \leq avg(rulesize)^{\#rules}$$

States are determined by pending rules and prefix read:

$$\#states \leq 2^{\#rules} * \#letters$$

Number of transitions

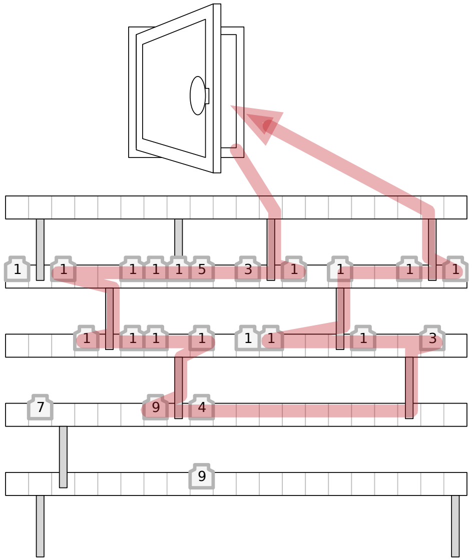
A state of the product automaton $(s_1, \ldots, s_r)$ has a rule named c when at least one the states s_i has a rule named c thus

$$\frac{\#trans}{\#states} \leq 1 + 2 \times r$$

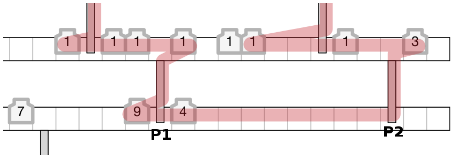
How many exactly?

Combining the above bounds gives us $\#trans < 3 \times 10^6$ and we can even lower this bound and pass easily!

I - Burglary



I - Burglary

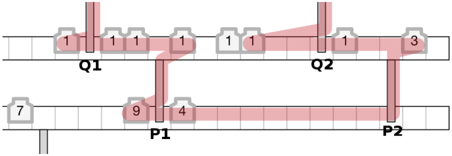


Solution sketch

Shelves: 0 (topmost) to N (floor). Slots: 0 to $M-1$. L = max ladders.
Max(T) = max candy grabbed for a trip with lowest reached shelf = T.
Result = $\max_{\mathbf{1<=T<=N}} \mathbf{Max(T)}$
With P_1 and P_2 "up" ladder endpoints:
$\mathbf{Max(T)} = \max_{\mathbf{P_1,P_2}}(\mathbf{MaxUp(T,P_1,P_2)} + \mathbf{Grabbed(T,P1,P2)})$

- $MaxUp(T,P_1,P_2)$ = max candy grabbed on $0,\ldots,T-1$ when reaching ("downwards") T by P_1 and leaving ("upwards") T by P_2
- $Grabbed(T,P_1,P_2)$ = all candy from P_1 to P_2 + potential "safely reachable" side candy (left and/or right).

I - Burglary



Key idea / dynamic programming

Shelves: 0 (topmost) to N (floor). Slots: 0 to $M-1$. $L =$ max ladders.
Idea: Compute $\mathbf{MaxUp}(\mathbf{T}, \mathbf{P}_1, \mathbf{P}_2)$ reccursively based on $\mathbf{MaxUp}(\mathbf{T}-1, \mathbf{Q}_1, \mathbf{Q}_2)$, $\mathbf{Grabbed}(\mathbf{T}-1, \mathbf{P}_1, \mathbf{Q}_1)$, $\mathbf{Grabbed}(\mathbf{T}-1, \mathbf{P}_2, \mathbf{Q}_2)$.

- Consider all Q_1, $Q_2 =$ "up" ladder endpoints for $T-1$
- Discard configs with jars in the intersection (not safe); avoid counting "middle" side candy twice.

Time complexity for all shelves: $\mathbf{O}(\mathbf{N} * \mathbf{L}^4 * \mathbf{Compl\_Grabbed})$

I - Burglary

Essential observation

$\mathbf{Grabbed(T, P_1, P_2)}$ **can be computed in constant time** for any (T, P_1, P_2) if one precomputes for all slots on all shelves:

- closest jar position left and right
- partial sums $SumLeft[T, P]$=sum of all candy on T left to P.

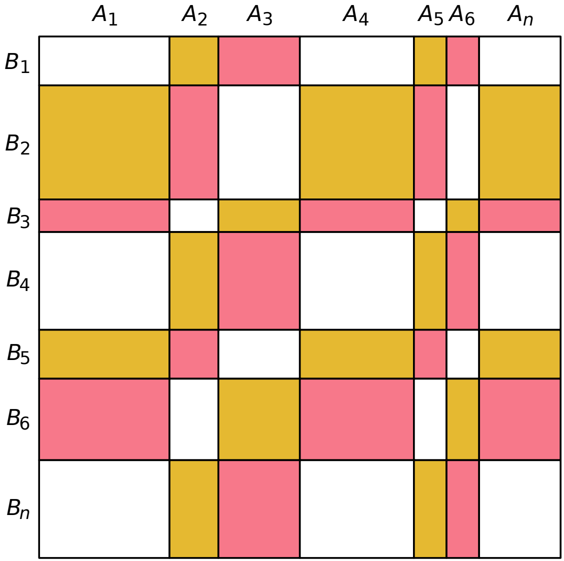
Precomputation: $\mathbf{O(N * M)}$

And so...

Overall complexity $\mathbf{= O(N * M + N * L^4)}$.

- Intersection tests + side candy = slight headache
- "Smaller" optims possible such as exploiting symmmetry, keeping only two rows for *MaxUp*...
- Tests may not be exhaustive but the Bandit is happy!

J - Frosting on the Cake



J - Frosting on the Cake

Key observation

Permuting columns or rows preserve the total area of each color. Hence we can reduce to a 3 by 3 grid, the dimensions are given by the sum of the entry lengths of same base 3 modulo.

Python Solution

```
def read_ints(): return [int(x) for x in input().split()]

def cat(l): return tuple(sum(l[n::3]) for n in [1, 2, 0])

input() # n
A = cat(read_ints())
B = cat(read_ints())
print("{} {} {}".format(B[2]*A[0]+B[0]*A[2]+B[1]*A[1],
                        B[2]*A[1]+B[0]*A[0]+B[1]*A[2],
                        B[2]*A[2]+B[0]*A[1]+B[1]*A[0]))
```

K - Blowing Candles

K - Blowing Candles

Key observation

The narrowest strip touches 3 points of the convex hull, 2 of them being consecutive on the hull

Algorithm

- Compute and restrict to convex hull in $O(n \log n)$
- Loop over all consecutive point pairs (a, b)
- Maintain a point c being furthest from (a, b) in $O(n)$ amortized time.