培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE

第一屆 · 2016

2016 年 8 月 22 日 (星期一)

香港培正中學

題解

題號		名稱	作者	難易度
A		Shuttle Bus	潘力行	★★★☆☆
B		Salt Trading	董鑫泉	★★☆☆☆
C		Annoying Mathematics	潘力行	★★★★★
D		Archery	黃敏恆	★☆☆☆☆
E		Bacteria Experiment	韓文軒	★★★★☆
F		Anniversaries	董鑫泉	★☆☆☆☆
G		Monorail	黃敏恆	★★★★★
H		Pokemon GO	黃敏恆	★★★★☆
I		RNG	董鑫泉	★★☆☆☆
J		Posters	董鑫泉	★★☆☆☆
K		Lattice Points	董鑫泉	★★★☆☆
L		Textbook Game	黃敏恆	★★☆☆☆

難易度是指作者團隊認為能在比賽中解決該題的隊伍比例

由 1 星至 5 星為: >75%, 50-75%, 25-50%, 5-25%, < 5%

Problem A – Shuttle Bus

To solve this task, we can first consider a simpler version of this problem :

1. Consider a 2 x N grid with NO obstacle.
2. We fix our starting point and ending point where the starting point can be either the top-left corner or the bottom-left corner of the grid whereas the ending point can be either the top-right corner or the bottom-right corner of the grid.

We should determine whether there exist a path such that every cells are visited for exactly once.

For example:
N = 7, starting = bottom-left, ending = bottom-right.
Output : NO

>	v	>	v	>	v	
^S	>	^	>	^	>	E

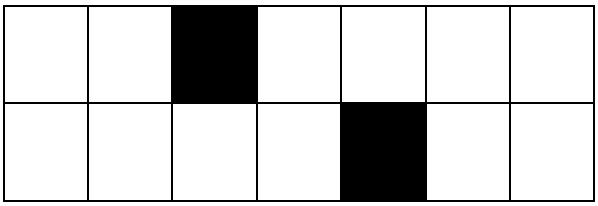
N = 7, starting = bottom-left, ending = top-right
Output : YES

>	v	>	v	>	v	E
^S	>	^	>	^	>	^

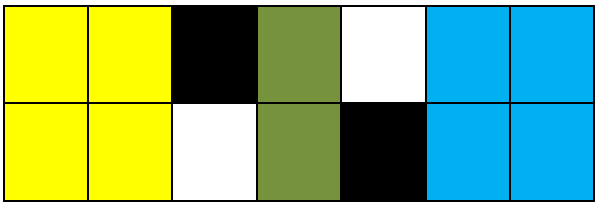
How to solve the above simpler version?
It is obviously that :

if (N is odd) and (row_of_starting NOT equal to row_of_ending) then YES
if (N is even) and (row_of_starting equal to row_of_ending) then YES
otherwise NO

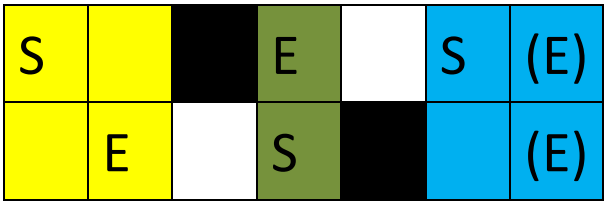
Then we back to our original problem, a grid with blocks.



We can divide it to some sub-grid without block

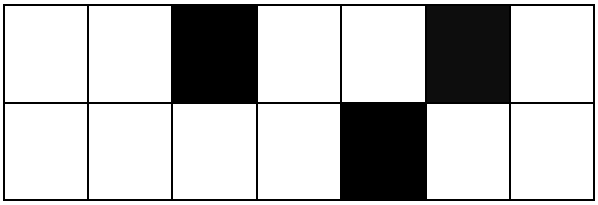


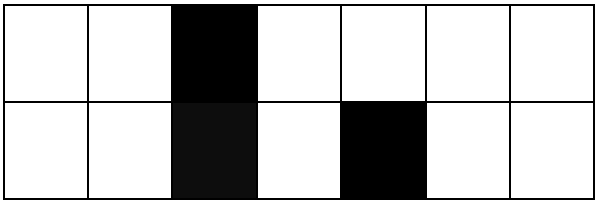
Also, we can determine the starting point and ending point of each sub-grid by the block.



Then the problem is simplified to our previous and simpler version ☺.

But wait, we still need some handling. What if the blocks divide the grid into two or more unconnected sub-grids.



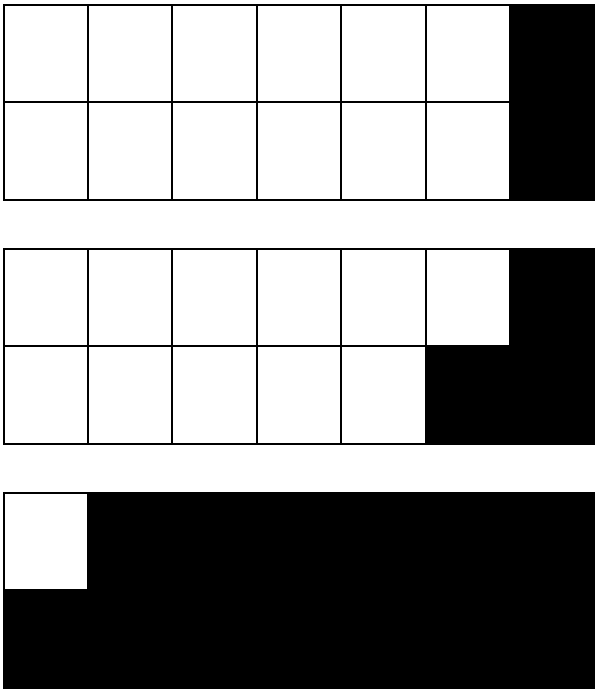


It is obviously that for 2 blocks i, j:

if (row[i] not equal to row[j]) and (abs(col[i] – col[j]) <= 1) then

they divide the grid into two or more unconnected sub-grids, therefore NO……?

The above statement is correct in general. However, there are some tricky case in our test data :



Therefore, we should also check that whether there is at least one empty cell in the two unconnected sub-grids by count the blocks in two sides.

Problem B — Salt Trading

First, I hope you like the story :) After going through the long problem statement, and doing a bit of mathematical reasoning, you will find that the problem basically requires you to output `Yes` if $\frac{1}{A} + \frac{1}{B} + \frac{1}{C} < 1$ and `No` if otherwise.

Why is this true? For the answer to be `Yes`, we need to find *non-negative* real numbers x, y, and z, such that

$$\begin{cases} x + y + z = 100 \\ min(Ax, By, Cz) > 100 \end{cases}$$

Without loss of generality, assume that $Ax = min(Ax, By, Cz)$. Then RB can save some salt by replacing y by $y' = \frac{Ax}{B}$ and z by $z' = \frac{Ax}{C}$. Note that $y' \leq y$ and $z' \leq z$. Now the equations become

$$\begin{cases} \frac{Ax}{A} + \frac{Ax}{B} + \frac{Ax}{C} \leq 100 \\ Ax > 100 \end{cases}$$

For $Ax(\frac{1}{A} + \frac{1}{B} + \frac{1}{C}) \leq 100$, we must have $\frac{1}{A} + \frac{1}{B} + \frac{1}{C} < 1$. Done.

Not yet. If you look at the scoreboard and see 70 wrong submissions, you know that this task isn't all about maths. Something tricky, maybe?

Indeed. If you read A, B, C as floating point numbers and check directly, you may get `Wrong Answer` on cases where $\frac{1}{A} + \frac{1}{B} + \frac{1}{C} = 1$ and the computer rounds the value of $\frac{1}{A} + \frac{1}{B} + \frac{1}{C}$ down. For example, $A = 2.00$, $B = 3.00$, $C = 6.00$ is such a case.

Solution 1. Read the numbers as fractions. Since A, B, and C are given with *exactly* two decimal places, $100A$, $100B$, and $100C$ are integers. Let A_{num} denote the numerator of A (written as a fraction) and A_{den} denote the denominator of A. Similar for B and C. We need to check if $\frac{A_{den}}{A_{num}} + \frac{B_{den}}{B_{num}} + \frac{C_{den}}{C_{num}} < 1$. Multiply both sides by $A_{num}B_{num}C_{num}$ we get a condition which can be handled by computers without error.

Remember to use `long long` (`int64` for Pascal) to handle the integers though.

Solution 2. Instead of checking if $\frac{1}{A} + \frac{1}{B} + \frac{1}{C} < 1$, we check if $\frac{1}{A} + \frac{1}{B} + \frac{1}{C} < 1 - \epsilon$, where ϵ is a small positive value. The use of ϵ (epsilon) is common in handling floating point problems with discrete (e.g. Yes/No) answers. In this problem, we need $\epsilon <$ (roughly) 2×10^{-11}. The worst case is $A = 1.01$, $B = 166.69$, $C = 256.29$, where $\frac{1}{A} + \frac{1}{B} + \frac{1}{C} \approx 0.99999999997682398$ (you should output `Yes` but the output will be `No` if ϵ is too large).

To conclude, to successfully solve this problem, you need to:

- Unwrap the lengthy problem statement to discover the task requirement,
- Use mathematical reasoning to solve the problem, and
- Be familiar with floating point manipulation to avoid precision error.

Problem C – Annoying Mathematics

We need to choose K distinct integers among 1 – R such that LCM of them equal to N in this task. We can first ignore K and find what is the minimum number of integers needed (denote it as Min_K) in order to make the LCM equal to N.

For example :

N	R	Min_K	Which Min_K integers
12	4	2	3 4
12	13	1	12
30	5	3	2 3 5
30	10	2	3 10
13	12	-1	Impossible

How to find Min_K and find which Min_K integers have a LCM equal to N? We can solve it by exhaustion.

Firstly, we can factorize N and time the same factor together. We will get a set of integers and we denote it as S.
Observation: To find which Min_K integers is needed, there exist a solution such that only the integers in S and the product of the integers in S are used.

For example, Let N = 60 = 2^2 x 3 x 5 = 4 x 3 x 5. So, S = {3, 4, 5}. Therefore, to find Min_K and which Min_K integers in needed, we can only consider {3, 4, 5, 3 x 4, 3 x 5, 4 x 5, 3 x 4 x 5} = {3, 4, 5, 12, 15, 20, 60}.

With this observation, we can come up with an exhaustion solution by permuting the integers in S and times the integers from the front to the end greedily. Which mean, the product of 2 integers does not exceed R, we times them together.

N = 420, S = {3, 4, 5, 7}, R = 21

Permutation of S	After times greedily	Explanation
3 4 5 7	12 5 7	3 x 4 = 12 <= R (times together) 12 x 5 = 60 > R 5 x 7 = 35 > R
3 4 7 5	12 7 5	Omit
3 5 4 7	15 4 7	Omit
3 5 7 4	15 7 4	omit
3 7 4 5	21 20	3 x 7 = 21 <= R (times together) 21 x 4 = 84 > R 4 x 5 = 20 <= R (times together)
….		

We can keep the optimal solution (Which is {21, 20} in the above case) during permuting S and at last we can find Min_K and find which Min_K integers have a LCM equal to N.

But wait, the original problem is to output K integers but not Min_K integers.

If Min_K > K, it is impossible to output K integers from 1 to R such that LCM of them equal to N.
If Min_K <= K, we can output any factors (which is smaller than R) of N to our solution as it will not change the LCM of the set.

Time complexity analysis:

As N <= 10^9 and 2 x 3 x 5 x 7 x 11 x 13 x 17 x 19 x 23 x 29 = 6469693230 > 10^9. Which mean, the set S contain at most 9 integers.

Therefore, time complexity = O(|S|! * |S|) where |S| is the number of integers in S which is at most 9 for finding Min_K.

On the other hand, the time complexity of factorize N is O(sqrt(N)).
Therefore, total time complexity is O(|S|! * |S| + sqrt(N)).

Problem D – Archery

Background

I always wanted to propose an easy task, and I thought archery scoring could test contestants' data processing and implementation skills. (Note: I was the Chairperson of HKUST Archery Club)

At first I used Alice and Bob as the characters when writing the problem statement, but then I suddenly recalled the TV programme. After a quick search from Wikipedia and MBC website, I found that GFRIEND participated in the archery event of 2016 Idol Star Athletics Championships. The photo was obtained from `http://imbbs.imbc.com/view.mbc?list_id=6513765&page=4&bid=2016new_photo`.

Implementation

Maintain 6 variables: `ascore`, `a10`, `ax`, `bscore`, `b10` and `bx`. Read the scores as strings. One way to check the string s is:

- If the length of s is 2, it is a 10.
  `length(s) = 2` / `strlen(s) == 2` / `s.length() == 2`
- Else if the string is `X`, it is an X.
  `s = 'X'` / `s[0] == 'X'` / `strcmp(s, "X") == 0` / `s == "X"`
- Else, the score is `ord(s[0]) - 48` / `s[0] - '0'`

Finally,

- If `ascore` $>$ `bscore`, output `Yuju`
- Else if `ascore` $<$ `bscore`, output `Yerin`
- Else if `a10` $>$ `b10`, output `Yuju`
- Else if `a10` $<$ `b10`, output `Yerin`
- Else if `ax` $>$ `bx`, output `Yuju`
- Else if `ax` $<$ `bx`, output `Yerin`
- Else, output `Shoot-off`

Problem E

First we take a look on the *Pólya urn model*: consider an urn with n blue balls and m red balls. At every turn, a ball is drawn randomly from the urn and its color is observed. Then it is then returned in the urn, with an additional ball of the same color is added to the urn. This process is then repeated.

Intuitively, if the urn has more blue balls at the beginning, it is highly likely that the blue balls will dominate the urn as time progress, and the bias will get larger and larger. It is 'richer got richer' in some sense.

Now lets return to the problem. At first glance, it seems like maximum degree is a good feature to separate the two seeds; however, this feature has a drawback that its effectiveness diminish as time progress. Instead, lets consider the edges. Every edge connects two subtrees. If we take the size of one of the subtree to be 'blue balls' and the other subtree to be 'red balls', this is precisely the Pólya urn model. That is, if one side of the subtree is larger than the other side, it is likely that it will be much larger as time progress. In other words, edges that are unbalanced will tend to be more unbalanced.

We can see that the L seed is more unbalanced, and the R seed is more balanced. There are many ways to capture the balancedness, and one solution that will work is

$$\sum_{e \in T} |L_e|^2 |R_e|^2$$

where L_e and R_e are the subtrees of edge e respectively. To find the best separation point for this value, we can perform experiments locally during the contest.

Problem F — Anniversaries

This problem is easy in that it is straightforward and you don't need to do *any* thinking. Literally.

Here is what you need to do:

```
Set current date to Day 0
Set answer to 0
Repeat 2048 times
  Increment current date by 1
  if current date is special
    add answer by 1
end
Output answer
```

In order to make life easier, you can declare an array `int daysinmonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}`. Then, you can avoid writing a bunch of messy `if`s.

Problem G – Monorail

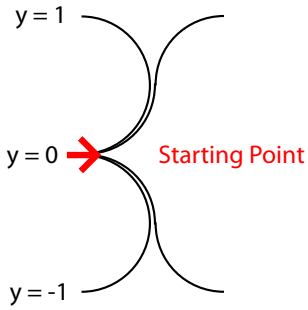
Impossible condition

There is no solution when any of the following is true:

When the number of curved tracks M is odd. After odd number of curved tracks, the train would have turned 90 or 270 degrees, not 360 degrees.

When the number of straight tracks N is odd. Since M is even, $N + M$ would be odd if N is odd. Let us paint the grid like a chess board. The loop must go through equal number of black and white cells. Therefore it is impossible that $N + M$ is odd.

$N = 0$ and the remainder of M divided by 4 is 2. There is no straight tracks. Let us define a starting point and place the curves two at a time. Everytime there are two options: 'C' and 'S'. But no matter which one you chooses, the parity of the y-coordinate of the end will change (from odd to even or even to odd). It is impossible to place an odd number of double curve tracks to arrive at $y = 0$.



$N = 0$ and $M = 8$ This is a special case.

Solution construction

There are numerous ways to construct the solution. For example:

$N = 0$ and $M = 4$ This is sample two, which can be done by hard coding.

$N = 0$ and $M = 12, 16, \ldots$

```
.r7....
rJL7...
L7.L7..  repeat this (M - 12) / 4 times
.L7rJ..
..LJ...
```

$N = 2$ **and** $M = 8$

```
.r7.
rJL7
L--J
```

Other cases The solution can be constructed using the above cases by extending the loop horizontally using $N/2$ pairs of `-` in the appropriate locations.

Problem H – Pokemon GO

Since there is no need to maximize the number of Pokestop visits, this problem can be solved using a modified version of either depth-first search or breath-first search.

The state of the DFS / BFS can defined by a tuple: (`node`, `distance`), meaning that we have arrived at node `node`, and the distance travelled so far is `distance`. Therefore, our goal is the state (B, K).

Let's say we use DFS. From state (x, d) we perform the following for each edge (y, l) that originates from node x: If $y + l \leq k$, visit state $(y, d + l)$, *if the state has not been visited.*

When the state (B, K) is reached, mark that a solution has been found using a global variable or returning `true`. Output the current node id x when poppoing out of the DFS stack. Note that doing so would give a reversed path, i.e. from B to A. One simple way to fix this is to swap A and B in the input.

Overall time complexity is $O((N+M)K)$. Memory complexity $O(NK+M)$

Problem I — RNG

Now we come to the problem with the least number of submissions (2). The only team that attempted this problem got `Accepted` in their second attempt. *Chapeau!*

Fun Fact. During problem preparation, we tagged this problem as *easy-medium*. It turned out to be one of the hardest problems in the problemset.

Here is the key fact which makes the problem infinitely simpler.

If there exists a solution, there exists a solution with $M = 5$.

Assuming that our key fact is true, you can just try all values of parameters satisfying $0 \le a, b, c, d \le 4$ and $1 \le X_1 \le 4$. That is a mere total of $5 \times 5 \times 5 \times 5 \times 4 = 2500$ possible sets of parameters.

In contest time, one can just convince oneself (and one's teammates) that a certain statement is true, without carefully proving that it indeed is. Therefore, our original thought was that many teams would try exhausting the parameters from small to large and would discover the key fact, or at least find out that the parameters are quite small, during the process. We were wrong.

Now that the contest has ended, let's focus on proving the key fact.

Let the generated sequence be $\{Y_n\}$. Note that Y_{i+1} only depends on Y_i. For each $k = 1, 2, 3, 4$, define $succ(k)$ (the successor of k) to be the value $(ak^3 + bk^2 + ck + d) \bmod 5$. That is, if $Y_i = k$, then $Y_{i+1} = succ(k)$.

Our sequence depends only on Y_1 (which equals X_1) and $succ(k)$ for $k = 1, 2, 3, 4$. If we can produce *all* different values of $succ(1)$, $succ(2)$, $succ(3)$, $succ(4)$, ($1 \le succ(k) \le 4$, so there are a total of $4^4 = 256$ different values), surely we can generate all valid sequences. There are two ways to show that it is possible.

Note: for those not familiar with linear algebra, you are strongly advised to skip **the hard way**. Although the proof of our key fact *may* be done using higher-level mathematics, we certainly *do not expect* contestants to master mathematics up to such a level. On the contrary, we expect contestants to use **the easy way**.

The hard way. Check that the system of linear equations

$$\begin{cases} a(1^3) + b(1^2) + c(1) + d & \equiv succ(1) \pmod 5 \\ a(2^3) + b(2^2) + c(2) + d & \equiv succ(2) \pmod 5 \\ a(3^3) + b(3^2) + c(3) + d & \equiv succ(3) \pmod 5 \\ a(4^3) + b(4^2) + c(4) + d & \equiv succ(4) \pmod 5 \end{cases}$$

has a (unique) solution. Rewrite the above as

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 4 & 2 & 1 \\ 2 & 4 & 3 & 1 \\ 4 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} succ(1) \\ succ(2) \\ succ(3) \\ succ(4) \end{pmatrix}$$

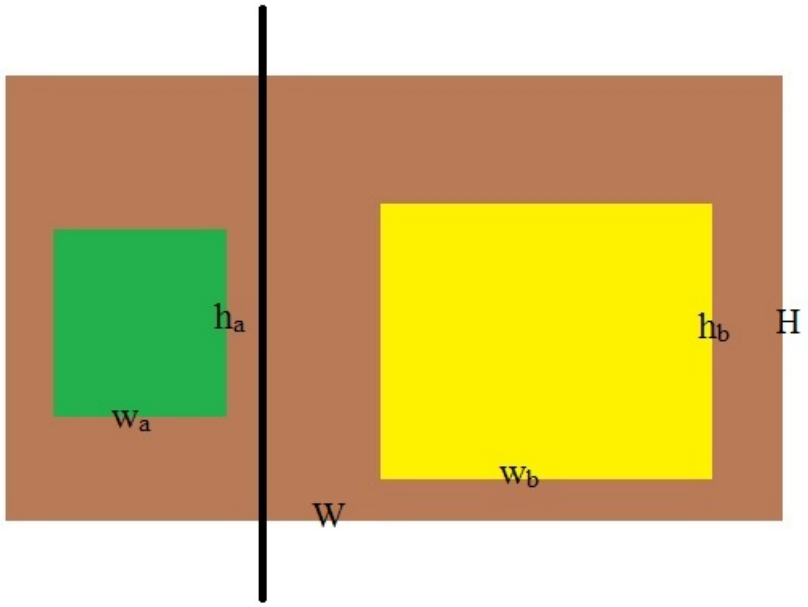
where operations are done in $\mathbb{Z}/5\mathbb{Z}$ (i.e. under mod 5). Since the 4×4 matrix is invertible, the conclusion follows.

The easy easy. ~~Who needs hardcore mathematics if you have computing resources.~~ Write a program to exhaust a, b, c, d and verify that all possible functions $succ()$ can be produced.

Problem J — Posters

1. Plan A

For *Plan A*, order of poster placement doesn't matter. Just consider Alex's task to be to place two posters on the bulletin board while following the rules. For two non-overlapping rectangles placed on a plane in the usual orientation, one can find either a horizontal or vertical line which divides them.



As seen from the picture above, in the case that there exists a horizontal dividing line, we need:

$$\begin{cases} h_a \leq H \\ h_b \leq H \\ w_a + w_b \leq W \end{cases}$$

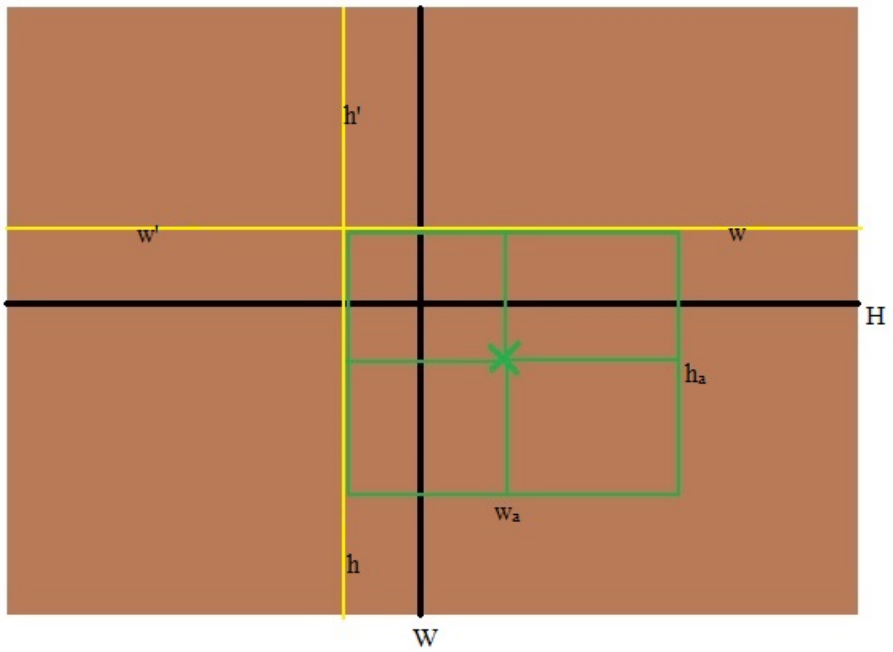
The case in which there exists a vertical dividing line is similar:

$$\begin{cases} w_a \leq W \\ w_b \leq W \\ h_a + h_b \leq H \end{cases}$$

If either set of conditions holds, output `Yes`. Otherwise, output `No`.

2. Plan B

For *Plan B*, order of poster placement matters. Intuitively, Alex may want to place his poster right in the middle of the bulletin board. It turns out to be the best strategy.



If Alex's poster doesn't fit in the bulletin board, output `No`. From now on, assume Alex's poster fits. by reflection, we can assume, without loss of generality, that the center of Alex's poster is located in the bottom right half of the bulletin board. Then, the following holds:

$$\begin{cases} h + h' &= H - h_a \\ w + w' &= W - w_a \\ h' &\geq h \\ w' &\geq w \end{cases}$$

Bob would fit his poster either to the left of the yellow vertical line, or to the top of the yellow horizontal line. To minimize Bob's chance of success, Alex should seek to minimize h' and w'. This is done by taking $h' = h = (H - h_a)/2$ and $w' = w = (W - w_a)/2$.

The rest is, again, simple mathematics. In the case of a horizontal dividing line, Bob succeeds if and only if:

$$\begin{cases} h_b &\leq h' = (H - h_a)/2 \\ w_b &\leq W \end{cases}$$

In the case of a vertical dividing line, Bob succeeds if and only if:

$$\begin{cases} w_b &\leq w' = (W - w_a)/2 \\ h_b &\leq H \end{cases}$$

If *none* of the sets of conditions holds, output `Yes`. Otherwise, output `No`.

Be careful about `/2`, by the way. Direct integer division will round down the results and *may* result in unwanted error.

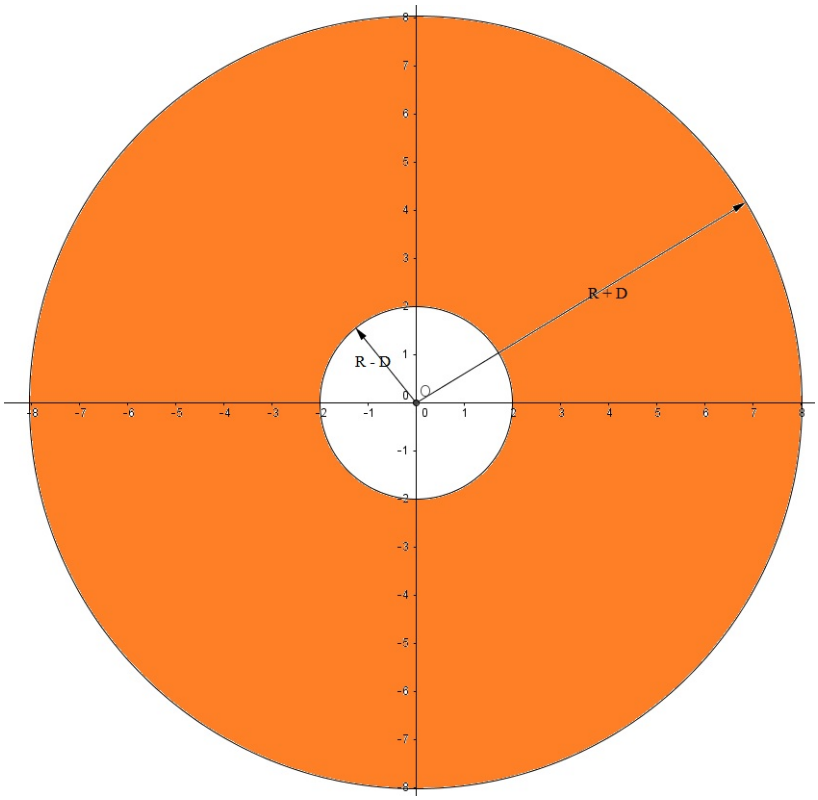
To conclude, in order to solve this problem, you need a good intuition (to guess the best poster placement in each of the cases) and careful mathematical reasoning (to calculate without making error).

Problem K — Lattice Points

Given a circle of radius R centered at the origin, and a point (x, y), what is the distance between the circle and the point? The answer is $|\sqrt{x^2+y^2}-R|$, where $|v|$ denotes the *absolute value* of v. For (x, y) to be within D units from the circle, we need $|\sqrt{x^2+y^2}-R| \leq D$. In other words, $-D \leq \sqrt{x^2+y^2}-R \leq D$.

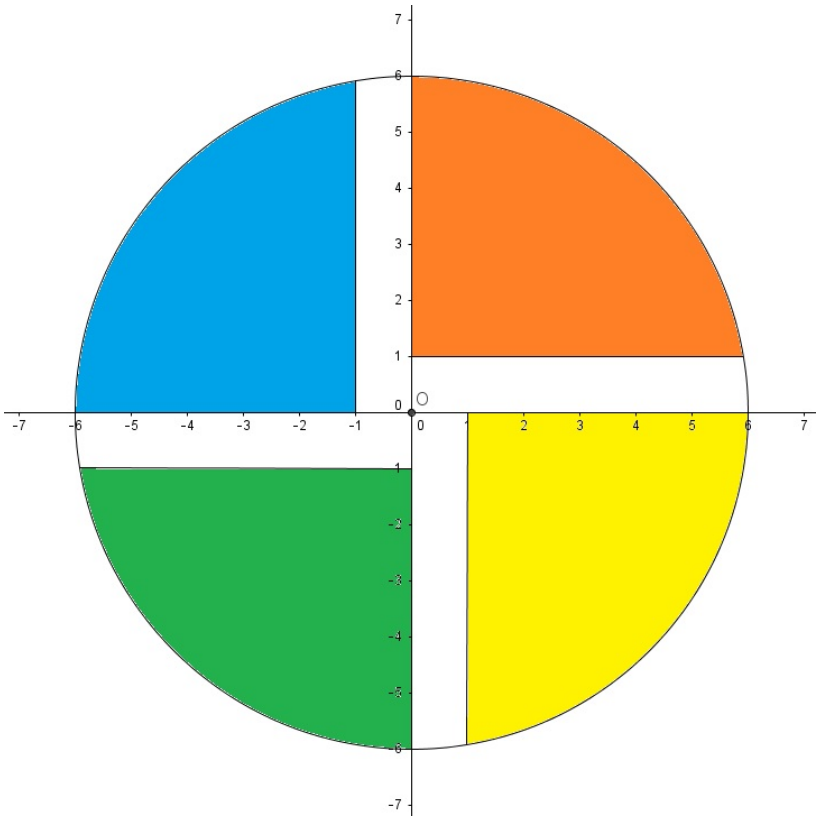
Further simplifying, the above becomes $R-D \leq \sqrt{x^2+y^2} \leq R+D$. This gives us a correct, yet inefficient (verdict will be `Time Limit Exceeded`), solution: consider all points (x, y) such that $-(R+D) \leq x, y \leq (R+D)$. If $R-D \leq \sqrt{x^2+y^2} \leq R+D$, add one to the answer.

Of course, there is a better (faster) solution. We need to make better use of the condition $R-D \leq \sqrt{x^2+y^2} \leq R+D$. The condition precisely means that the points we need to add to our answer are those points within an origin-centered circle of radius $R+D$ (including the boundary) but *not* within an orgin-centered circle of radius $R-D$ (excluding the bonudary). In other words, add the points in the bigger circle and subtract the points in the smaller circle. In pictures:



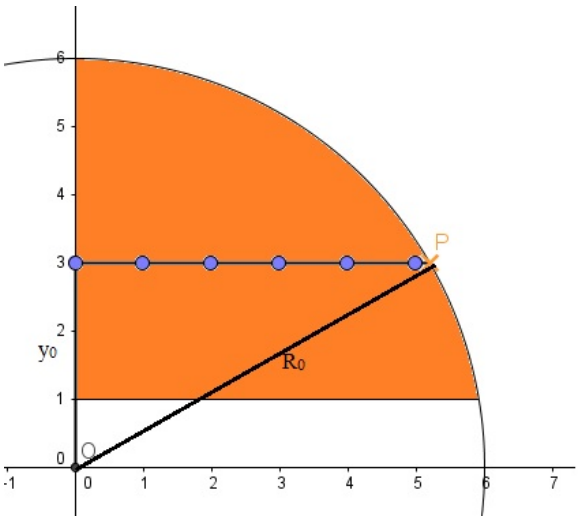
So, our problem is slightly simplified — count the number of points within a certain origin-centered circle, including or excluding the boundary.

Let us include the boundary for now. We divide the circle into four parts, leaving the origin alone, as shown in the picture on the next page:



Notice that the number of points in the four coloured parts are the same. Let the answer in the orange part be C. Then answer equals $4C + 1$ (+1 is for the origin).

How do we calculate C? Consider horizontal lines $y = 1$, $y = 2$, ..., $y = R_0$, where R_0 is the radius of the circle we now consider. Each line contains a certain amount of points that we need to include in the answer.



How many points do we need to include in the horizontal line $y = y_0$? Refer to the picture above. The length of the horizontal line is $\sqrt{R_0^2 - y_0^2}$. The answer is $\left(\left\lfloor \sqrt{R_0^2 - y_0^2} \right\rfloor + 1\right)$.

So now we have an algorithm to the simpler version of the problem. Loop y from 1 to R_0, adding $\left(\left\lfloor \sqrt{R_0^2 - y^2} \right\rfloor + 1\right)$ to C in each iteration. The answer is then $4C + 1$.

As for the similar problem in which the circle boundary is excluded, loop y from 1 to R_0, adding $\left(\left\lceil \sqrt{R_0^2 - y^2} \right\rceil\right)$ to C in each iteration. The answer is then $4C + 1$.

With an efficient solution to the simplified problem, we now then have the solution to the original problem. Finally, `Accepted`.

There is a tricky case which, fortunately, is covered in the samples (sample case 2, to be precise). In the case that $R \leq D$, there is no "smaller circle" to subtract. You only need to calculate the number of points in the bigger circle and output that as answer.

Problem L – Textbook Game

This problem tests contestants about partial sum, a commonly used optimisation technique.

While reading the input array $a[i]$, construct a frequency array freq[0..1000000]. At the end, we would want freq[j] stores the number of times number j appeared. To do this, we add 1 to freq[$a[i]$] for every $i = 1 \ldots N$

Partial sum would be next. Construct the partial array ps[0..1000000] using `ps[i] = ps[i - 1] + freq[i]`. After that, ps[j] stores the number of elements in a having value $\leq j$.

For a spread having x faces, output the following: $\frac{N - \text{ps}[x]}{N}$, $\frac{\text{freq}[x]}{N}$, and $\frac{\text{ps}[x-1]}{N}$.

Since x can be 0, C/C++ users is recommended to offset the partial sum array by 1, otherwise ps[-1] would be accessed. Pascal users can simply declare `ps[-1..1000000]`.

Overall time complexity is $O(N + R)$. Memory complexity $O(N + R)$