Problem Tutorial: "Chess"

Definitions

Let's denote the amount on the i-th square by $b_i = 2^{i-1} \bmod M$. A is sorted sequence B, meaning that there exists a permutation π, such that $\forall i : b_{\pi(i)} = a_i$. We have no efficient way of finding π so far, but we can study the properties of b_i.

First let's see how can we verify that a certain M is a solution (not necessarily the smallest one): we can simply generate B, sort it, and check element by element in time $O(N \log N)$.

It holds that $\forall i : 0 \leq b_i < M$, which translates to a lower bound $M > \max(b_i) = \max(a_i)$.

Let's refer to b_N by the term **terminator**.

For a fixed value of b_i ($i < N$), let's consider the possible values of b_{i+1}:

1. If $2 \cdot b_i < M$, then $b_{i+1} = 2 \cdot b_i > b_i$.
2. If $2 \cdot b_i = M$, then $b_{i+1} = 0$.
3. If $2 \cdot b_i > M$, then $b_{i+1} = 2 \cdot b_i - M < b_i$.

If case 3 holds, then we call number b_i **big**, and number b_{i+1} **small**[1], and we call (b_i, b_{i+1}) a **witness**. For instance, with $M = 11$ and $N = 8$, we have $B = \{1, 2, 4, 8, 5, 10, 9, 7\}$. There are three witnesses: $(8, 5)$, $(10, 9)$ and $(9, 7)$. Here, 8 and 10 are big, 5 and 7 are small, and 9 is both big and small. Note that 7 is not considered big even when it is larger than $\frac{M}{2}$ since it is the terminator and there is no corresponding small value.

If we manage to find a witness where one exists, we can calculate M using the definition of a witness (in our example $M = 2 \cdot 8 - 5 = 2 \cdot 10 - 9 = 2 \cdot 9 - 7 = 11$). We'll see later how to search for witnesses, but first let's crack some edge cases.

0.1 Zero values

What if case 2 holds? Since a_1 is the minimum number, then $a_1 = 0$. This can happen in one of two cases:

1. $b_1 = 0$: This means that all elements necessarily equal to zero, and that can only happen if $M = 1$. This case can be checked by simply looking at a_N in $O(1)$.
2. $b_1 \neq 0$: Denote the last non-zero element of B by b_z. We know that $b_z \equiv 2^{z-1} \pmod{M}$, and also $b_{z+1} = 0 \equiv 2^z \pmod{M}$. The latter can be rewritten as $2^z = k \cdot M$ for some positive integer k. This implies that both k and M have to be a power of two. Furthermore, if $k > 1$ then M divides 2^{z-1}, and this implies $b_z = 0$ which is a contradiction with the selection of z. Hence we know that $k = 1$, $M = 2^z$ and $b_z = 2^{z-1}$. The sequence of B thus contains all powers of two between 1 and 2^{z-1} (inclusive), and the rest are zeroes. Hence b_z is the largest element (i.e., a_N), implying $M = 2 \cdot a_N$.

To summarise, if $a_1 = 0$, then $M = \max(1, 2 \cdot a_N)$. From now on we will assume that there are no zeros in the sequence.

0.2 No witness

If there is no witness, it means that only the case 1 is ever applied, which means that $\forall i : b_i = 2^{i-1}$ and we can obtain such sequence using any $M > 2^{N-1}$. Since we are requested to return the smallest possible M, then $2^{N-1} + 1$ would be the correct answer.

Since B is increasing (hence $\forall i : a_i = b_i$), we can check whether $a_N = 2^{N-1}$ holds[2] in $O(1)$ time. If it does, it is also sufficient condition for non-existence of a witness.

[1]Note that a number can be both **big** and **small**, and can be also neither of them

[2]Within our constraints, this can only happen for $N < \log_2(10^{18})$.

0.3 General case

So far we have resolved the case 2, and also situation where only 1 happens. Let's now assume that there is a witness. If we find one, we can calculate M. Since every pair $a_i > a_j$ can potentially be a witness, we put $M := 2 \cdot a_i - a_j$ and check whether given M generates $\{a_i\}$. There are $O(N^2)$ such pairs and we can check all of them in total time $O(N^3 \log N)$.

0.4 Finding big values

The first improvement is to get a good candidate for the big value. We know that if $i < N$ and $2 \cdot b_i > M$, then b_i is big. However, neither the π (to rule out which a_k is terminator), nor M, is known. A simple argument shows that at least one of the two largest elements (i.e., a_N or a_{N-1}) has to be big: If $\pi(N) \neq N$, then a_N is big. If $\pi(N) = N$, then a_{N-1} is big. We do not know the corresponding small value, but we can try all possibilities. Overall we check $O(N)$ pairs of values, each in $O(N \log N)$, yielding an $O(N^2 \log N)$ algorithm.

0.5 Finding small values

Can we further improve the running time? Let's first assume that we have obtained the set of all small b_i. We know that for each small element it holds $b_i = 2 \cdot b_{i-1} - M$. However, for a fixed M the function $f(x) = 2x - M$ is linear, which means that the largest big value b_{i-1} is paired with the largest small value b_i (amongst all small values). Since we already know that one of a_{N-1} and a_N is big, it follows that one of them is the largest big value. We can simply use the maximum of the set of small values to check constant number of pairs for being a witness. This would yield an $O(N \log N)$ algorithm.

What is left to do is to find the set of all small values. This can be split into two cases:

1. M is odd: Since $2x$ is even for any integer x, then $2x - M$ is odd. Hence, every small value is odd. With the exception of $b_1 = 1$, all odd values are also small. Checking the largest odd value with a_N and a_{N-1} yields the solution.

2. M is even: With the exception of $b_1 = 1$, all b_i are even. Let's denote the count of occurrences of x in $\{a_i\}$ by $c(x)$, and the count of occurrences ignoring the terminator by $c_n(x) \leq c(x)$. Clearly, for every even value we obtain

$$c(x) = c_n\left(\frac{x}{2}\right) + c_n\left(\frac{x+M}{2}\right) \tag{1}$$

However, from the input array we are only able to calculate $c(x)$, not $c_n(x)$, as we do not know terminator. The following theorem provides us a straightforward way of obtaining the set of small numbers.

Lemma 1 *Let M be even. If x is small, then $c_n(\frac{x}{2}) = c\left(\frac{x}{2}\right)$.*

Proof This lemma says that if x is small, then the $\frac{x}{2}$ (which also yields x as next value) cannot be the terminator.

If M is even, then there exist positive integers v, w, such that $M = v \cdot 2^w$ and v is odd. We know that $b_{w+1} = 2^w$. Clearly, for every positive integer k the value $(2 \cdot k \cdot 2^w) \bmod (v \cdot 2^w)$ is divisible by 2^w. In other words, all values after b_{w+1} are divisible by 2^w by induction. Furthermore, all small values are divisible by 2^w, since b_{w+1} is obtained only using case 1.

Furthermore, the first number that is repeated in B (if B is long enough to have repeating values) is 2^w. This can be proven by contradiction - let's assume, that the first repeated number is $k \cdot 2^w$, with indices $i < j$ $(i > w)$. Hence we obtain[3]:

$$\begin{aligned}
2^{i-1} &\equiv 2^{j-1} && \pmod{v \cdot 2^w} \\
2^{i-w-1} &\equiv 2^{j-w-1} && \pmod{v} \\
2^{i-w-1} &\equiv 2^{i-w-1} \cdot 2^{j-i} && \pmod{v} \\
1 &\equiv 2^{j-i} && \pmod{v} \\
2^w &\equiv 2^{w+j-i} && \pmod{v \cdot 2^w}
\end{aligned}$$

which proves that $B_{w+j-i+1}$ is the same as B_{w+1}, but $w + j - i + 1 < j$, hence B_j is not the first repetition.

Let's now have a small value x. If $x = (2k + 1) \cdot 2^w$, then $\frac{x}{2} = (2k + 1) \cdot 2^{w-1}$ never occurs after 2^w, which concludes the proof.

Let's look at case of $x = (2k) \cdot 2^w$. Let's denote the index of second occurrence of 2^w by r. No numbers are repeated between B_{w+1} and B_{r-1}. We know that x occurs after 2^w, which means that it occurs exactly once in this subsequence. This implies that only one of $\frac{x}{2}$ and $\frac{x+M}{2}$ can be member of that periodic part, and it has to be $\frac{x+M}{2}$, since x is small. Hence $c_n(\frac{x}{2}) = c\left(\frac{x}{2}\right) = 0$.

$\square$

Theorem 1 *Let M be even. For every even $x > 0$, x is small if and only if $c(x) > c\left(\frac{x}{2}\right)$.*

Proof $\Longrightarrow$:
Since $\frac{x+M}{2}$ is the corresponding big value to small x, then $c_n(\frac{x+M}{2}) > 0$, and using (1) and lemma 1 we get $c(x) > c_n(\frac{x}{2}) = c(\frac{x}{2})$.
$\Longleftarrow$:

$$c(x) > c\left(\frac{x}{2}\right) \geq c_n\left(\frac{x}{2}\right) =^{(1)} c(x) - c_n\left(\frac{x+M}{2}\right) \implies c_n\left(\frac{x+M}{2}\right) > 0$$

Since $2 \cdot \frac{x+M}{2} \equiv x \pmod{M}$, $\left(\frac{x+M}{2}, x\right)$ is witness and x is small.

$\square$

We have proven that for M even, $c(x) > c\left(\frac{x}{2}\right)$ is sufficient and necessary condition for a value to be small. Getting the set of small numbers can thus be done in $O(N \log N)$ (i.e., using binary search).

Now that we have the set of small numbers, we find its maximum, and with one of a_N and a_{N-1} it has to be a witness.

Problem Tutorial: "Triangle in a Triangle"

The key realisation needed to solve this problem is that there are very few triangles that could be the largest, regardless of the number of lampposts. There are in fact only six lampposts that we need to consider: the lamppost closest to A and the one closest to B on the segment $\overline{AB}$, as well as the two extremal lampposts on each of the segments $\overline{AC}$ and $\overline{BC}$. To prove this we consider any triangle ΔDEF where F is on the segment $\overline{AB}$. If we move F along the segment towards A one of three things can happen:

1. The perpendicular distance between $\overline{DE}$ and F increases $\longrightarrow$ the area of the triangle ΔDEF increases.

[3]Since 2 has an inverse multiplicative element in ring $\mathbb{Z}/c\mathbb{Z}$.

2. The perpendicular distance between $\overline{DE}$ and F stays the same $\longrightarrow$ $\overline{AB}$ and $\overline{DE}$ are parallel, the area of ΔDEF stays the same.
3. The perpendicular distance between $\overline{DE}$ and F decreases $\longrightarrow$ in this case the area decreases, but we can increase it by moving F away from A towards B.

This means that for any triangle ΔDEF, we can find some direction to move F such that the area does not decrease. If we move F to the furthest lampposts in that direction and repeat the process for D and E we get a new triangle that has a larger or equal area to the orinial with corners at some of the extremal lampposts.

So all that is left to do is check all $\binom{6}{3} = 20$ possibile triangles on the six extremal lampposts and return the largest area that is found. The overall runtime will be $\mathcal{O}(n)$ as we need to read all of the input.

Problem Tutorial: "Candy division"

We aim to find 3 divisors of n, namely a, b, c, such that

$$\frac{n}{a} + \frac{n}{b} + \frac{n}{c} = n.$$

In order for this to hold, we need to solve

$$\frac{1}{a} + \frac{1}{b} + \frac{1}{c} = 1$$

for positive integers a, b, c. WLOG assume $a \le b \le c$. This implies

$$1 = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} \le \frac{3}{a},$$

hence $a \le 3$. Let's solve each case separately:

- $a = 1$: Then we have
$$\frac{1}{b} + \frac{1}{c} \le \frac{2}{b} = 0$$
which is clearly impossible.
- $a = 2$: We have
$$\frac{1}{b} + \frac{1}{c} = \frac{1}{2}.$$
By a similar argument as before, we have $2 = a \le b \le 4$. Therefore there are two solutions $(2, 4, 4)$ and $(2, 3, 6)$.
- $a = 3$: Similar argument as above yields $3 = a \le b \le 3$, yielding a solution $(3, 3, 3)$.

There are thus two solutions:

- If n is divisible by three, output $\left(\frac{n}{3}, \frac{n}{3}, \frac{n}{3}\right)$.
- If n is divisible by four, output $\left(\frac{n}{2}, \frac{n}{4}, \frac{n}{4}\right)$.

Otherwise, there is no solution. Note that the solution $\left(\frac{n}{2}, \frac{n}{3}, \frac{n}{6}\right)$ is not needed, as the necessary condition of divisibility by 6 implies divisibility by 3, which is already covered.

Problem Tutorial: "Effective network"

The task first looks intimidating, since there are many subgraphs of G, but it turns out there is a very simple trick.

Denote $diam(G)$ to be the maximum minimum distance between any two vertices u, v. We prove that for if there is a solution, then the full graph is also the solution.

Take a $w \in V(G)$ such that $H := G[V(G) \setminus \{w\}]$ is connected. For any pair $u, v \in G$ we want to bound the distance in G. If none of the u and v equals to w, than clearly the shortest path in H is also a path in G, hence $dist_G(u, v) \leq dist_H(u, v) \leq diam(H)$.

Otherwise WLOG $u = w$. One can easily see that $dist_G(w, v) = 1 + \min_{z \in V(H)}(dist_H(z, v)) \leq 1 + diam(H)$. Hence we obtain $diam(G) \leq diam(H) + 1$.

It follows that

$$|V(G)| - diam(G) \geq |V(H)| - diam(H).$$

By induction, we can extend this argument to arbitrary subgraph of G.

Hence, if $N - diam(G) \geq K$ we output the whole graph, otherwise we output -1. Calculating the diameter can be done in $\mathcal{O}(NM)$ which is fast enough.

Problem Tutorial: "Collection"

The basic solution to this problem is to store each card id in some set datastructure (`set/unordered_set` in C++, `HashSet/TreeSet` in Java) and add one to the duplicate counter every time we find an id that is already present in the set.

Another solution is store all ids in an array and sort it. Then count the duplicates by checking if a position is equal to the previous position in the array.

Problem Tutorial: "Mattress Run"

First, let's only consider qualifying on "nights". This can be done using a standard dynamic programming approach. We will compute for given day, $nights$, $last\_night\_hotel$, what is the minimum cost $C(day, nights, last\_night\_hotel)$ to stay that number of nights by a given day provided that you stay the last night ($day - 1$ to day) in hotel $last\_night\_hotel$. If you aren't staying that night in a hotel, we will assume that $last\_night\_hotel$ will have some special value $\varnothing$ (different from all hotel IDs).

Now, how do we compute C?

$$C(day, nights, \varnothing) = \min_i C(day - 1, nights, i)$$
$$C(day, nights, l \neq \varnothing) = \min_{i \neq l, (h=l, b, e=day, c) \in Rates} C(b, nights - (e - b), i)$$

After we compute all possible values of C we need to take the lowest with $day = Y$ and $nights = N$, and then recover the sequence of used rates.

Finding the cheapest run to qualify on "stays" can be done with the same algorithm, and of course in the end we need to choose the cheaper option out of the two qualification criteria.

Problem Tutorial: "Affine"

Affine functions f map convex sets S to convex sets $f[S]$: Consider any $\vec{x}$ and $\vec{y}$ and $0 \leq \alpha \leq 1$. We want to show that $\alpha \cdot f(\vec{x}) + (1 - \alpha) \cdot f(\vec{y}) \in f[S]$. By definition, $\alpha \cdot f(\vec{x}) + (1 - \alpha) \cdot f(\vec{y}) = f(\alpha \cdot \vec{x} + (1 - \alpha) \cdot \vec{y})$. Because S is convex, $\alpha \cdot \vec{x} + (1 - \alpha) \cdot \vec{y} \in S$ and therefore $f(\alpha \cdot \vec{x} + (1 - \alpha) \cdot \vec{y}) \in f[S]$.

As the m-hypercube is convex for any m, there is no answer if the given polygon is not convex. The polygon is convex if none of the given boundary points P_i is to the left of the line connecting the previous point P_{i-1} and the next point P_{i+1}. This can be checked by computing the determinant of the 2×2 matrix whose columns are $P_i - P_{i-1}$ and $P_{i+1} - P_{i-1}$, as the sign of the determinant of a matrix gives the relative orientation of its column vectors. (We have used the convention $P_0 = P_n$ and $P_{n+1} = P_1$.)

If the sign of the determinant is 0 for some point P_i, we can filter it out without changing the polygon. If we do this for all points, we recover the vertices of the polygon.

Affine functions f map point-symmetric sets S to point-symmetric sets $f[S]$: Let c be the center of symmetry of S. It suffices to show that $f(\vec{x}) \in f[S] \Rightarrow 2 \cdot f(\vec{c}) - f(\vec{x}) \in f[S]$. Using the fact that f is affine,we can rewrite the right hand side as $f(2\vec{c} - \vec{x}) \in f[S]$. Therefore, we need to show that $f(\vec{x}) \in f[S] \Rightarrow f(2\vec{c} - \vec{x}) \in f[S]$. Let $\vec{y} \in S$ be such that $f(\vec{x}) = f(\vec{y})$. It suffices to show that $f(2\vec{c} - \vec{y}) = f(2\vec{c} - \vec{x})$. This is however immediate, as by affinity of f, we have $f(2\vec{c} - \vec{y}) = 2 \cdot f(\vec{c}) - f(\vec{y}) = 2 \cdot f(\vec{c}) - f(\vec{x}) = f(2\vec{c} - \vec{x})$.

As the m-hypercube is point-symmetric, there is no answer if the given polygon is not point-symmetric. We can check symmetry by computing the center of the polygon and checking for each vertex that it has an opposite vertex.

We now show that those two conditions are sufficient: each convex point-symmetric polygon is an image of the m-hypercube for some m. There are two special cases, which we treat separately: If the polygon is a single point, it is an affine image of the 0-hypercube (which is also a single point). If the polygon has only a single edge, it is an affine image of the 1-hypercube.

For the remaining cases, we proceed by induction on the even number of vertices and edges $n \geq 4$ of the polygon, and we show the stronger statement that such a polygon is an affine image of the $n/2$-hypercube.

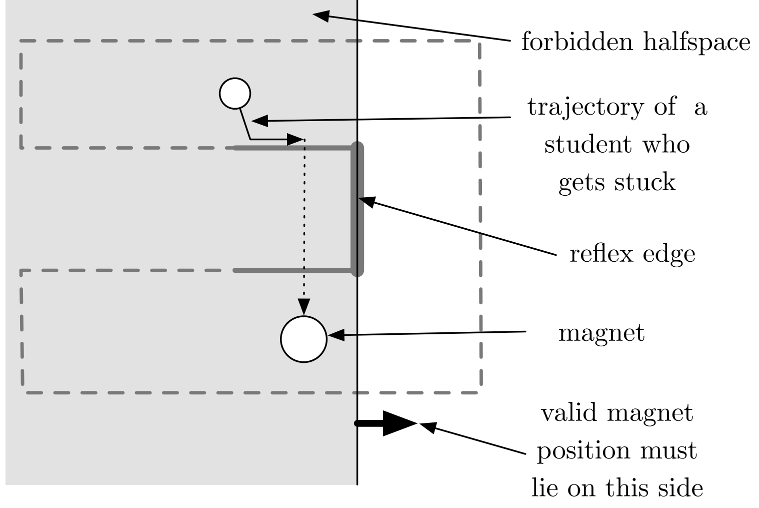
Base ($n = 4$): The polygon is an affine image of the 2-hypercube (the unit square): Take let $\vec{b} = P_1$ and let $A = (\vec{c}_1 \vec{c}_2)$, where the columns $\vec{c}_1$ and $\vec{c}_2$ are given by $\vec{c}_1 = P_2 - P_1$ and $\vec{c}_2 = P_4 - P_1$. Then $f(\vec{x}) = A \cdot \vec{x} + \vec{b}$ maps the unit square to the polygon.

Step ($n-2 \to n$, for $n \geq 2$): Consider any pair of opposite edges (those edges will have the same length and orientation). We can obtain a new center-symmetric polygon with $n-2$ vertices and edges by collapsing those two edges into (non-opposite) incident vertices. By the induction hypothesis, this new polygon is an image of the $(n/2 - 1)$-hypercube. Let $f'(\vec{x}') = A' \cdot \vec{x}' + \vec{b}$ be some affine function mapping the $(n/2 - 1)$-hypercube to the collapsed polygon. We construct a function $f(\vec{x}) = A \cdot \vec{x} + \vec{b}$, where A is obtained from A' by adding one more column: the difference of the two vertices incident to the collapsed edge. Then, f maps the $n/2$-hypercube to the polygon.

This construction is optimal with respect to the dimension of the hypercube: Each edge of the polygon is the image of some edge of the hypercube. As edges of the polygon have a different orientation unless they are opposite, there must be at least $n/2$ different orientations of edges in the hypercube. (Because affine functions map lines of the same orientation to lines of the same orientation.) The m-hypercube has m different edge orientations, hence $m = n/2$ is the smallest possible.

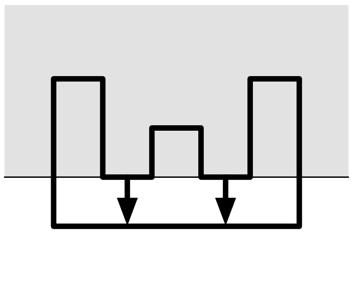
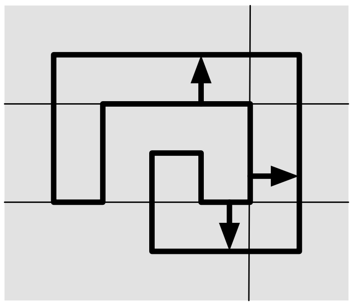
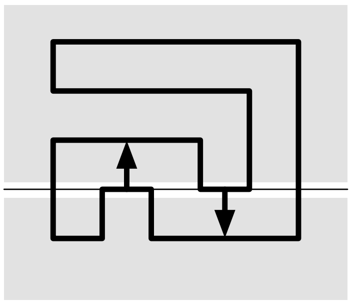
Therefore, the task can be solved by checking whether the given polygon is point-symmetric and convex. If this is the case, print the number of its vertices divided by two, rounded down, and `INFINITY` otherwise.

Problem Tutorial: "Compass"


forbidden halfspace
trajectory of a student who gets stuck
reflex edge
magnet
valid magnet position must lie on this side

Steps to success

- *What determines where the magnet can possibly be placed?* By playing with the examples you can notice that the so-called reflex edges are the critical walls of the university.
- *What is a reflex edge?* If you follow the polygon in clockwise order and you encounter two successive turns in counter-clockwise direction the edge in between the two turns is called a reflex edge.
- *Why are they relevant?* Each reflex edge gives you a condition defining a halfplane where the magnet is not allowed to lie in. *Why?* Take a look at the drawing on top. Placing the magnet to the left of and below the reflex edge (in the gray area) would prohibit anyone coming from the left of and above the reflex edge to reach the magnet as the compass will point straight into the wall and not around the reflex edge.
- *So what's the solution?* Just walk around the polygon once in time $\mathcal{O}(n)$ and intersect all the halfspaces specified by the reflex edges. Simply keeping track of the minimum and maximum coordinates in x- and y-direction that are still admissible is enough for this. In the end, you check whether this results in a rectangle of non-negative size or not.
- *Why is this sufficient? Why don't we have to worry about getting stuck in a corner?* Assume you get stuck somewhere, so the straight line from your position to the magnet leaves and reenters the polygon and cuts the outside area of the polygon into at least two parts. At least one of those parts contains a reflex edge that forbids the magnet's position.
- *How does this look like for the examples on the task statement?*

Problem Tutorial: "The Secret"

Note that there is always an optimal solution where at most one equivalence class is not a singleton. (We can pick any optimal solution and merge all the non-singleton subsets.)

Let $C = \{i | \Pr[y = 1|x = i] \notin [a, b]\}$. This is the set of outcomes that violate the confidentiality of the secret on their own. Clearly, all of those will be in the non-singleton equivalence class. All other outcomes do not contribute any further constraints, but we might need some of them to balance out C in case it does not satisfy $\Pr[y = 1|x \in C] \in [a, b]$.

Wlog, $\Pr[y = 1|x \in C] > b$. (Otherwise we are either done or we can flip the secret and the interval $[a, b]$.)

Our goal is to pick $C' \subseteq \{1, \dots, N\} \setminus C$ such that $|C'|$ is minimal and $\Pr[y = 1|x \in C \cup C'] \in [a, b]$. By definition of C, for each $i \in C'$ we have $\Pr[y = 1|x = i] \geq a$, so in particular, for any $C' \subseteq \{1, \dots, N\} \setminus C$, the constraint $\Pr[y = 1|x \in C \cup C'] \geq a$ is satisfied. This means the only constraint that remains to be satisfied is $\Pr[y = 1|x \in C \cup C'] \leq b$.

Written out, this constraint is

$$\frac{\sum_{i \in C \cup C'} \Pr[y = 1|x = i] \cdot \Pr[x = i]}{\sum_{i \in C \cup C'} \Pr[x = i]} \leq b.$$

We can also write this as

$$\sum_{o \in C \cup C'} (\Pr[y = 1 | x = i] - b) \cdot \Pr[x = i] \leq 0.$$

Now it is easy to see that it is sufficient to repeatedly pick some i from $\{1, \dots, N\} \setminus C$ that minimizes the expression $(\Pr[y = 1|x = i] - b) \cdot \Pr[x = i]$.

This can be achieved by sorting the outcomes outside of C in time $O(n \log n)$ or $O(n)$.

Problem Tutorial: "Box Hedge"

Given some points (attractions) in the half plane, the task was to find a "minimal" subset (box hedge) that separates them with the outside. The subset needs to be minimal regarding circumference, size and covered area.

First, let's look at some solution that minimizes the circumference. It's easy to see that you can't do better than a rectangle. Let x_L be the x-coordinate of the left-most point, x_R the x-coordinate of the right-most point and y_M the y-coordinate of the up-most point. The minimal circumference then looks like this: $(x_L - 1, 0) - (x_L - 1, y_M + 1) - (x_R + 1, y_M + 1) - (x_R + 1, 0)$. Why is it minimal? Because in order to separate the left-most point with the outside, the rays starting from the point and having directions $(-1, 0)$, $(1, 0)$ and $(0, 1)$ need to intersect the circumference at some point. So some part of the circumference needs to have x-coordinate $x_L - 1$. A similar argument can be made for $x_R + 1$ and the y-coordinate $y_M + 1$. As the circumference is measured in manhattan distance it's easy to see that the rectangle from above minimizes the circumference. By definition of x_L,x_R and y_M, the circumference also contains all attractions.

Next, let's minimize the size of the separating subset. Once we know the outside circumference, we can just remove all strictly interior points. It turns out that the size is the same as the circumference minus two. Tiles can have either zero, one or two visible circumference. The sum of all angles in the circumference must be 180°, so we see that those with zero and two cancel each other out, except for two having two circumference.

What about the minimal area? Say we have some solution with minimal circumference. We can fold it without changing the circumference nor the size:

```
............       ............
.BBBBBBBBBBB       .......BBBBB
.B     .....       .......B....
.B     .....   =>  .......B....
.B..........       .BBBBBBB....
.B..........       .B..........
```

We can obtain the minimum area by folding the rectangle repeatedly with folds that keep the attractions contained, until no more such folds can be done. We then have reached the global minimum. Why? Say there is a better solution. Since the circumference length is the same, we have some some angle on the upper part (old solution) and an opposite angle on the lower part (new solution). The area in between those is empty (otherwise the new solution would be invalid). But then we can perform a fold between those two points, which is a contradiction to our assumption that we can't do any more folds.

To solve this task, it was enough to convince yourself how the optimal solution looks like.

For implementation, the easiest way to split the half plane into two quarter planes with the up-most point in both parts. Then you need to find a dominating increasing/decreasing line in the respective parts, which can be done in a linear scan after sorting the points lecicographically. Using `std::sort`, the asymptotic runtime is thus $\mathcal{O}(n \log n)$.

Problem Tutorial: "ACM"

To find out whether a string contains a certain substring, one could use the member function `string::find`, the C function `strstr` or just code it by hand.