CODE FESTIVAL 2016 Relay A 問題 解説

catupper

$\frac{1}{R_1} + \frac{1}{R_2} = \frac{1}{R_3}$ を式変形すると以下の形になる。

$$R_3 = \frac{R_1 \times R_2}{R_1 + R_2}$$

上式の右辺を計算すると、答えを求めることが出来る。

CODE FESTIVAL 2016 Relay B 問題 解説

catupper

2 つの文字 a, b について以下のいずれかが成り立っていて、かつそのときのみ、a, b は**鏡の関係にある**と呼ぶとする。

- $\{a, b\} = \{$'b', 'd'$\}$
- $\{a, b\} = \{$'p', 'q'$\}$

S の文字数を N とする。全ての $1 \leq i \leq N$ について以下が成り立っていれば S は鏡文字である。

- S_i と S_{N+1-i} が鏡の関係にある。

ループを回して全ての i について上記の条件をみたすかどうかチェックすれば、S が鏡文かどうか判断することが出来る。

CODE FESTIVAL 2016 Relay C 問題 解説

catupper

トーナメントで行われる勝負の数は 2^N 回であり、一つの勝負の結果の計算は $O(1)$ でできるので、実際にトーナメントをシミュレートすれば、答えを求めることができる。

参加者数が 2 のべき乗であるような、トーナメント戦のシミュレーションには、一つ楽な実装がある。

問題文に書かれているトーナメントの形式は以下のように言い換えても、全く同じものを再現することが出来る。

- 1 番目の石を先頭にして、順番に 2^N 個の石を一列に並べる。
- 先頭の 2 つを列から取り出し、勝負させ、勝ち残ったものを列の末尾に挿入する。
- 石の個数が 1 つになるまで同様のことを続ける。

このように実装すれば、 配列一つでトーナメントを実装することが出来る。

セグメントツリーの配列表示のようなデータ構造を用いるアルゴリズムを応用するやり方もあるが、ここでは紹介にとどめておき、詳細は割愛する。

CODE FESTIVAL 2016 Relay D 問題 解説

catupper

穴の空いた魔方陣を補完する問題である。正解となる魔方陣の各マスに入っている数値を以下のように定義する。

A	B	C
D	E	F
G	H	I

タテ・ヨコ・ナナメに連続する 3 マスの和が全て同じなので。以下の式が成り立つ。

$$\begin{aligned} A+B+C &= C+E+G \\ \Leftrightarrow G &= A+B-E \end{aligned}$$

同様にして、以下の式が導かれる

$$\begin{aligned} F &= A+G-E \\ C &= A+E-F \\ D &= B+C-G \\ H &= A+E-G \\ I &= B+E-G \end{aligned}$$

初めに A, B, E が与えられている状態から、上記の式を上から順に解いていくと、全てのマスの値を求めることが出来る。

CODE FESTIVAL 2016 Relay E 問題 解説

catupper

$A = C$ もしくは $B = D$ が成り立つとき、考える線分は一つもマスを横切らないので答えは 0 になる。

よって $A \neq C$ かつ $B \neq D$ を仮定して良い。
ここでさらに、$A < C$ と $B < D$ を仮定する。対称性より、この仮定を認めても問題の一般性は失われない。

マス (A, B) の左下からマス (C, D) の左下まで結んだ線分が横切るマスの集合を S とする。

以下の 3 種類の移動を使って、S に含まれるマスのみを全て通ってマス (A, B) から $(C-1, D-1)$ まで行くことを考える。

- 今いるマスを (X, Y) として $(X+1, Y)$ に移動する。(右のマスに移動する)
- 今いるマスを (X, Y) として $(X, Y+1)$ に移動する。(上のマスに移動する)
- 今いるマスを (X, Y) として $(X+1, Y+1)$ に移動する。(右上のマスに移動する)

このとき移動の仕方は一意に定まる。
また、3 種類目の移動をする回数はちょうど $\gcd(C-A, D-B)-1$ 回になる。
この移動は $X+Y$ の値が常に 1 もしくは 2 増加する。 2 増加する回数は $\gcd(C-A, B-D)-1$ 回なので、$(C-1-A)+(D-1-B)$ からその値を引けば、移動する回数が求まる。
ただし、$\gcd(X, Y)$ は X と Y の最大公約数とする。

移動する回数は $|S|-1$ となる。今求めたいものは $|S|$ である。よって答えは

$$|S| = |C-A| + |D-B| - \gcd(|C-A|, |D-B|)$$

となる。

CODE FESTIVAL 2016 Relay F 問題 解説

catupper

長さ X の紐があるとき、そこから操作を 1 回だけ行ったとき、作れる紐の長さの範囲を考える。これは以下を満たす B の範囲を考えることと同値である。

$$X = A + B + C$$
$$0 < A \leq B \leq C$$

これを解くと B の動ける範囲は

$$1 \leq B \leq \lfloor \frac{X-1}{2} \rfloor$$

となる。$A = B = 1, C = X - 2$ という切り方が左側の等号を満たし、 $A = 1, B = \lfloor \frac{X-1}{2} \rfloor, C = \lceil \frac{X-1}{2} \rceil$ という切り方が右側の等号を満たす。$\lfloor \frac{X-1}{2} \rfloor$ は X に関して単調増加なので、できるだけ長い紐を残すのが、操作の回数を最大化するために最適な戦略である。よって、紐の長さを $\lfloor \frac{X-1}{2} \rfloor$ にする切り方のみ考えれば良い。

以上から、$f(N)$ を計算する方法がわかった。一回の操作で紐の長さは半分以下になるので $f(N)$ は $O(logN)$ で計算することが出来る。

$f(N)$ は N に関して単調増加なので、$f(N) = X$ となる最大の N は二分探索によって求めることが出来る。
なお、$f(N) = X$ となる最大の N は $2^{2+X} - 2$ と一致する。これは帰納法を使って証明することが出来るが、ここでは割愛する。

CODE FESTIVAL 2016 Relay G 問題 解説

catupper

各コップに番号 1 から N までの番号を振る。操作を始める前に左から i 番目にあるコップの番号を i とする。

超能力を一度も使わない場合、玉はずっとコップ 1 の中にいることになる。

超能力を使う場合、玉は超能力を使った時点でコップ 1 の隣にいたコップの中にいることになる。よって、全ての操作を通じて 1 回以上コップ 1 の隣にくるコップの個数を数えれば良い。

i 回目の操作の後に、左から j 番目のコップの番号が何であるか、というものは配列を使って容易に管理することが出来る。また、コップを入れ替える操作に関しても $O(1)$ で対応することが出来る。
各操作が終わるたびにコップ 1 の隣にあるコップの番号を保存して、最後にその種類数を求めれば、その値から答えを出すことが出来る。

CODE FESTIVAL 2016 Relay H 問題 解説

catupper

T を 午前 0 時から午後 23 時 59 分 59 秒まで動かす代わりに、「早起き」となる時間帯を動かすことを考える。

まず $T = 0$ と固定して、高橋君の起床時刻 N 個を求める。その集合を S とする。(この集合は同じ数値を複数持つことがあることに注意)。
そのあと、$A + 10800 = B$ として閉区間 $[A, B]$ の中に含まれる S の要素の個数が最大となるような A を見つければ、その区間に含まれる S の要素数が答えになる。

A の候補としては、S に含まれる時刻だけを考えれば良い。
A が決まれば B も決まる。S に予め適切な処理がされていれば、$O(logN)$ で 閉区間 $[A, B]$ に含まれる要素を数えることが出来る。

A の候補が $O(N)$、各 A に対する数え上げの計算量が $O(logN)$ なので全体で $O(NlogN)$ の計算量で答えを求めることが出来る。

CODE FESTIVAL 2016 Relay I 問題 解説

catupper

N が奇数のとき $A_{i,j} = (i-1+j)\%N+1$ とすればチャレンジに成功する。

$N=2$ のとき、どのようにしてもチャレンジを成功させることは出来ない。

$N=4$ のとき、以下のような見方をすれば、チャレンジに成功する。

$A_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1$	2	3	4
$i=2$	3	4	1
$i=3$	4	2	1
$i=4$	1	3	2

$N=k$ でチャレンジが成功するとき、以下のようにやることで $N=2k$ でもチャレンジを成功させることが出来る。
まず、$N=2k$ 人を奇数番目の人たちと、偶数番目の人たちの 2 グループにわける。
最初の 1 秒間 i 番目の人は $(i\%N)+1$ 番目の人をみつめる。
次の $k-1$ 秒間、偶数番目の人たちが、そのなかで $N=k$ のときと同様のチャレンジを行う。
その間、奇数番目の人たちは、偶数番目の人たちのうちまだ見つめていない $k-1$ 人を好きな順番で見る。
最後の $k-1$ 秒間、奇数と偶数を入れ替えて同様のことを行う。

上記の処理をうまく実装すれば答えを求めることが出来る。

CODE FESTIVAL 2016 Relay J 問題 解説

catupper

黒く塗られたマスが全て連結になれば、条件は満たされる。

まず、ある整数 n について $6n \leq i < 6n+3$ かつ $0 \leq j < N$ となる長方形区間にはいる黒マスを全て連結にする方法を考える。
$i = 6n+1, j =$ **偶**数 となるようなマスを全て黒く塗ればよい。このとき黒く塗るマスの個数はたかだか $\lceil N/2 \rceil$ 個である。

次に、ある整数 n について $6n+3 \leq i < 6n+6$ かつ $0 \leq j < N$ となる長方形区間にはいる黒マスを全て連結にする方法を考える。
$i = 6n+4, j =$ **奇**数 となるようなマスを全て黒く塗ればよい。このとき黒く塗るマスの個数はたかだか $\lceil N/2 \rceil$ 個である。

これらの約 $N^2/6$ 個のマスを黒く塗った後、$j = 0$ となるマスのうち、まだ黒く塗られていないマスを全て黒くぬると、全てのマスが連結になる。

ここまでで黒く塗ったマスの個数は $N^2/6 + N$ 個以下である。$N = 1000$ に対してこの値は 170000 より小さいので、制約を満たす。

CODE FESTIVAL 2016 Relay K 問題 解説

catupper

M が最大となる頂点列を**最適解**と呼ぶことにする。木の頂点 v について、$v \in V$ となる最適解 V があるかどうか、v の次数について場合分けして考える。

まず、次数が 3 以上のときについて考える。
木から頂点 v を取り除くと、いくつかの部分木に別れる。この部分木の集合を S_v とする。
ある最適解 V について $v \in V$ とすると、$V \cap T \neq \emptyset$ となる $T \in S_v$ はたかだか 2 つしか無い。(仮に 3 つ以上あったと仮定すると、V のなかに v が二度現れなければならなくなる。)

v の次数は 3 以上なので S_v の要素数も 3 以上である。
よって、$v \in V$ とする最適解 V があるとき、$V \cap T = \emptyset$ となる $T \in S_v$ が存在する。
T から 1 つ次数が 1 の頂点 (葉) を取ってきて、V の v と交換し、V' を作る。
このとき V' も問題文の条件をみたすので、$v \in V$ としなくても良い。
つまり次数が 2 以下の頂点のみから最適解を構成することが出来る。

次に、v の次数が 1 のときについて考える。
このとき、次数が 1 であるような v をすべて含むような最適解 V が必ず存在する。
もし $v \in V$ ではない最適解 V があったとき、上で行ったような次数が 1 ではない頂点との交換操作によって、$v \in V'$ であるような別の最適解 V' を作ることが出来るからである。

最後に、v の次数が 2 のときについて考える。
もし、ある最適解 V に次数 1 の頂点が全て含まれているならば、V の中の次数が 2 である頂点は全て、かならず一つのパス上に順番に並んでいる。(もしそうでない場合は、次数 3 の頂点についての議論と同様、V が問題文の条件を満たさなくなる。)

また、逆に、もし次数 2 の頂点が同一のパス上に並んでいるならば、それらが全て V に含まれているような最適解が存在する。

よって上記の考察をまとめると、求める答え M は、次数 1 の頂点の個数を A、木上の全てのパスの中で次数 2 の頂点を最も多く含むものの、含む個数を B とすると、$A + B$ となる。
A は $O(N)$ で求めることができ、B は木 DP をすることで $O(N)$ で求めることが出来る。

CODE FESTIVAL 2016 Relay Problem A Editorial

catupper

$\frac{1}{R_1} + \frac{1}{R_2} = \frac{1}{R_3}$ can be transformed into:

$$R_3 = \frac{R_1 \times R_2}{R_1 + R_2}$$

Compute the right side to obtain the answer.

CODE FESTIVAL 2016 Relay Problem B Editorial

catupper

We will say two characters a and b are in a **mirror relation** if and only if one of the following is satisfied:

- $\{a, b\} = \{$'b', 'd'$\}$
- $\{a, b\} = \{$'p', 'q'$\}$

Let the number of characters in S be N. S is a mirror string if the following is satisfied for all $1 \leq i \leq N$:

- S_i and S_{N+1-i} are in a mirror relation.

Iterate through all i and check if the condition above is satisfied, to determine whether S is a mirror string.

CODE FESTIVAL 2016 Relay Problem C Editorial

catupper

Since the number of fights in the tournament is 2^N, and the outcome of each battle can be found in O(1) time, it is enough to actually simulate the process of the tournament.

There is an easy way to implement the simulation of the process of a tournament where the number of participants is a power of 2.

The tournament described in the statement can be rephrased without changing the outcomes:

- Arrange the 2^N stones in a queue, in order from stone 1 through stone N.
- Take out two stones from the front of the queue, throw them each other, and insert the winning stone at the end of the queue.
- Repeat the step above until there is only one stone remaining.

With this implementation, only one array is necessary.

There are also other ways such as maintaining arrays in a segment tree, but here we will only give an introduction and omit the detail.

CODE FESTIVAL 2016 Relay Problem D Editorial

catupper

In this problem, you are asked to fill empty squares in a magic square. We will denote the integer in each square in the complete magic square, as follows:

A	B	C
D	E	F
G	H	I

Since the three integers in each row, in each column, and in each diagonal all add up to the same sum, the following holds:

$$\begin{aligned} A + B + C &= C + E + G \\ \Leftrightarrow G &= A + B - E \end{aligned}$$

In the same manner, the following are derived:

$$\begin{aligned} F &= A + G - E \\ C &= A + E - F \\ D &= B + C - G \\ H &= A + E - G \\ I &= B + E - G \end{aligned}$$

Given A, B and E, the values in all squares can be found, using the formulas above from top to bottom.

CODE FESTIVAL 2016 Relay Problem E Editorial

catupper

When $A = C$ or $B = D$, no square is crossed by the segment and the answer is 0.

Thus, we will assume $A \neq C$ and $B \neq D$ from now on.
Furthermore, by symmetry, we will also assume $A < C$ and $B < D$ without loss of generality.

Let S be the set of the squares crossed by the segment connecting the lower left corner of square (A, B) and the lower left corner of square (C, D).

Let us consider the way to travel from square (A, B) to $(C - 1, D - 1)$ visiting all squares in S and no other squares, using three kinds of moves below:

- Let the current square be (X, Y) and move to $(X + 1, Y)$. (Move right)
- Let the current square be (X, Y) and move to $(X, Y + 1)$. (Move up)
- Let the current square be (X, Y) and move to $(X + 1, Y + 1)$. (Move up right)

Then, the way to travel is uniquely determined. The number of times the third kind of move is performed, is $\gcd(C - A, D - B) - 1$.
Each kind of move increases the value of $X + Y$ by 1 or 2. The number of times $X + Y$ is increased by 2 is $\gcd(C - A, B - D) - 1$, thus the number of total moves can be found by subtracting this value from $(C - 1 - A) + (D - 1 - B)$.
Here, $\gcd(X, Y)$ denotes the greatest common divisor of X and Y.

The number of moves is equal to $|S| - 1$. What we want to find is $|S|$, thus the answer is:

$$|S| = |C - A| + |D - B| - \gcd(|C - A|, |D - B|)$$

CODE FESTIVAL 2016 Relay Problem F Editorial

catupper

We will consider the range of the possible length of the cord after one operation, when there is a cord of length X. This is equivalent to consider the range of the possible value of B satisfying the below:

$$X = A + B + C$$
$$0 < A \leq B \leq C$$

Solving this, the possible value of B is:

$$1 \leq B \leq \lfloor \frac{X-1}{2} \rfloor$$

The equality on the left is satisfied if the cord is cut so that $A = B = 1, C = X-2$, and the equality on the right is satisfied if the cord is cut so that $A = 1, B = \lfloor \frac{X-1}{2} \rfloor, C = \lceil \frac{X-1}{2} \rceil$. Since $\lfloor \frac{X-1}{2} \rfloor$ is monotonically increasing with respect to X, the optimal strategy to maximize the number of times the operation can be performed, is to leave a cord that is as long as possible. Thus, we should cut the cord so that the length of the remaining cord will be $\lfloor \frac{X-1}{2} \rfloor$.

Now we know how to calculate $f(N)$. Since the length of the cord becomes at most half after an operation, $f(N)$ can be calculated in $\mathrm{O}(\log N)$ time.
The maximum value of N such that $f(N) = X$ can be found performing binary search, since $f(N)$ is monotonically increasing with respect to N.
Actually, such value of N is equal to $2^{2+X} - 2$. It can be proved by mathematical induction, but here we will omit the detail.

CODE FESTIVAL 2016 Relay Problem G Editorial

catupper

We will number the cups 1 through N. The cup at the i-th position from the left before the operations is numbered i.

If we do not cast the magic, the ball will stay in cup 1 during the whole process.

If we cast the magic, the ball will be transferred to a cup that is adjacent to cup 1 at that time. Thus, we can count the number of the cups which will become adjacent to cup 1 at least once during the process.

We can easily maintain the index of the cup that is at the i-th position from the left after each operation, using an array, and the operation to swap the positions of cups can be processed in O(1) time.
The answer can be found by preserving the indices of the cups adjacent to cup 1 after each operation, and counting how many kinds of indices there are among them.

CODE FESTIVAL 2016 Relay Problem H Problem

catupper

Instead of setting T as 0:00:00 through 23:59:59, we will move the period during which an awakening is considered as waking up early.

First, fix T to 0, and find the times of day when Takahashi wakes up for each of the N awakenings. Let S be the set of those times. (Note that this set may contain duplicate elements.)
Then, find the value of A such that the closed interval $[A, B]$ where $A + 10800 = B$ covers the maximum number of elements in S. The answer is the number of elements in S covered by such interval.

For the candidates of value of A, only times that is contained in S need to be considered.
The value of B is determined when A is determined. If S is properly processed in advance, the number of elements covered by the closed interval $[A, B]$ can be counted in $\mathrm{O}(\log N)$ time.

Since there are $\mathrm{O}(N)$ candidates for the value of A, and it takes $\mathrm{O}(\log N)$ time to count for each value of A, the answer can be found in $\mathrm{O}(N \log N)$ time in total.

CODE FESTIVAL 2016 Relay Problem I Editorial

catupper

When N is odd, let $A_{i,j} = (i-1+j)\%N+1$ and the challenge will be successful.

When $N = 2$, it is impossible to be successful in the challenge.

When $N = 4$, follow the following strategy and the challenge will be successful:

$A_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1$	2	3	4
$i=2$	3	4	1
$i=3$	4	2	1
$i=4$	1	3	2

If it is possible to be successful in the challenge when $N = k$, it is also possible to be successful when $N = 2k$, as below.
First, divide the $N = 2k$ persons into two groups: the odd-number-th persons and the even-number-th persons.
During the first 1 second, the i-th person will look at the $(i\%N+1)$-th person.
During the following $k-1$ seconds, the even-number-th persons will carry out the challenge when $N = k$, among those N persons.
During that time, the odd-number-th persons will look at the $k-1$ even-number-th persons that they have not looked at, in arbitrary order.
During the last $k-1$ seconds, they will perform similarly, with “odd” and “even” swapped.

The successful strategy can be found by properly implementing these processes.

CODE FESTIVAL 2016 Relay Problem J Editorial

catupper

The condition is satisfied if all black squares are connected.

First, we will consider how to connect all black squares that is contained in a sub-rectangle $6n \leq i < 6n+3$, $0 \leq j < N$ for some integer n.
We can do this by painting all squares where $i = 6n+1, j = \mathbf{a}nevennumber$. Here, the number of squares painted black is at most $\lceil N/2 \rceil$.

First, we will consider how to connect all black squares that is contained in a sub-rectangle $6n+3 \leq i < 6n+6$, $0 \leq j < N$ for some integer n.
We can do this by painting all squares where $i = 6n+4, j = \mathbf{a}noddnumber$. Here, the number of squares painted black is at most $\lceil N/2 \rceil$.

After painting those approximately $N^2/6$ squares, paint the squares where $j = 0$ and not yet painted black, and all black squares will be connected.

During the whole process, the number of squares painted black is at most $N^2/6 + N$, which is less than 170000 for $N = 1000$, satisfying the constraints.

CODE FESTIVAL 2016 Relay Problem K Editorial

catupper

We will call a sequence of vertices with the maximum value of M, as an **optimal solution**. For a vertex v in the tree, we will consider whether there exists an optimal solution V such as $v \in V$. There are several cases according to the degree of v.

First, we will consider the case where the degree is at least 3.
When vertex v is removed from the tree, it will be separated into several subtrees. Let the set of these subtrees be S_v.
If $v \in V$ for some optimal solution V, there exists at most two $T \in S_v$ such that $V \cap T \neq \emptyset$. (If there were three or more, v would occur in V twice.)

Since the degree of v is at least 3, the number of elements in S_v is also at most 3.
Thus, if there exists a optimal solution V such that $v \in V$, there exists $T \in S_v$ such that $V \cap T = \emptyset$.
We will now take a vertex of degree 1 (a leaf) from T, and replace v in V with it to construct V'.
Then, V' will also satisfy the conditions, thus it is not necessary to include v in V.
That is, an optimal solution can be constructed from only vertices whose degree is at most 2.

Next, we will consider the case where the degree is 1.
There always exists an optimal solution V that contains all v whose degree is 1.
This is because, if there exists an optimal solution V such that $v \notin V$, we can construct another optimal solution V' such that $v \in V'$, by replacing a non-leaf vertex with v, as we did previously.

Lastly, we will consider the case where the degree is 2.
If an optimal solution V contains all leafs, all vertices with degree 2 in V always line up along a path. (If not, similarly to the argument on a vertex with degree 3, V would not satisfy the conditions.)
Conversely, for vertices with degree 2 lining up along a path, there exists a optimal solution containing all of them.

To sum up, the value of M we must find will be $A + B$, where A is the number of leafs, and B is the maximum number of vertices with length 2 contained in a path, among all paths in the tree.
A can be found in $\mathrm{O}(N)$ time, and B can also be found in $\mathrm{O}(N)$ time, performing DP on the tree.