AtCoder Beginner Contest 111 / AtCoder Regular Contest 103 解説

writer: nuip 並びに semiexp

2018 年 9 月 29 日

For International Readers: English editorial starts on page 200.

A: AtCoder Beginner Contest 999

与えられる 3 文字をそれぞれ問題文のとおりに置き換えればよいです。

```
#include<iostream>

using namespace std;

int main(){
	string n;
	cin >> n;
	for(char c:n){
		if(c == '1'){
			cout << 9;
		}else{
			cout << 1;
		}
	}
	return 0;
}
```

文字列を相互に置き換える処理は次のようにも書けます。

```
#include<iostream>

using namespace std;

int main(){
	string n;
	cin >> n;
	for(char c:n){
		cout << (char)('1' + '9' - c);
	}
	return 0;
}
```

これを応用して次のようなシンプルなコードでも解けます。

```
#include<iostream>

using namespace std;

int main(){
	int n;
	cin >> n;
	cout << 111 + 999 - n << endl;
	return 0;
}
```

B: AtCoder Beginner Contest 111

x 回目の ABC が黒橋君が初めて参加する ABC としてふさわしいかどうかは，各桁の数字を見ることで確かめられます．なので，$x = N$ から始めて，条件を満たすまで 1 ずつ x を大きくしていけば解くことができます．

ところで，この問題では，$100 \leq N \leq 999$ という条件から，答えとなる x は必ず 3 桁になります．3 桁で条件を満たす x はすべて 111 の倍数（逆もしかり）なので，N 以上の最小の 111 の倍数を求める方法でも解くことができます．

- C++ による解答例: `https://beta.atcoder.jp/contests/abc111/submissions/3279754`
- Java による解答例: `https://beta.atcoder.jp/contests/abc111/submissions/3279759`
- Python 3 による解答例: `https://beta.atcoder.jp/contests/abc111/submissions/3279761`
- Ruby による解答例: `https://beta.atcoder.jp/contests/abc111/submissions/3279762`
- Rust による解答例: `https://beta.atcoder.jp/contests/abc111/submissions/3279784`
- Bash による解答例: `https://beta.atcoder.jp/contests/abc111/submissions/3279843`

C: /\/\/\/

数列 $v_1, v_2, \ldots, v_n$ から偶数番目を取り出した数列 $v_2, v_4, \ldots, v_n$ を e とおき、奇数番目を取り出した数列 $v_1, v_3, \ldots, v_{n-1}$ を o とおきます。問題の条件を e と o を使って言い換えると、次のようになります。

1. e の要素はすべて同じ
2. o の要素はすべて同じ
3. $e \neq o$

みっつめの条件を無視して考えると、e と o それぞれについて、各要素を数列の最頻値に書き換えるのが最適になります。e と o それぞれの最頻値が異なっていれば、その書き換え方でみっつめの条件もクリアできます。そうでない場合はどちらかを最頻値でない値に書き換えなければなりません。

e で一番出現回数の多い要素を E_1 とおき、二番目に出現回数の多い要素を E_2 とおきます（E_1 と E_2 の出現回数が同じということもありえます)。同様に、o で一番出現回数の多い要素を O_1 とおき、二番目に出現回数の多い要素を O_2 とおきます。今、$E_1 = O_1$ であるため、次のふたつの書き換え方のうちどちらかが最適です。

- e を E_1 で書き換え、o を O_2 で書き換える
- e を E_2 で書き換え、o を O_1 で書き換える

この 2 通りを両方試して、書き換える項の数の小さい方が答えです。

D: Robot Arms

まず，ロボットアームを決めると，$(x_m + y_m) \bmod 2$ は，モードの設定によらず一定になります．なので，$(X_j + Y_j) \pmod 2$ が一定でなければ，条件を満たすことは不可能です．

$(X_j + Y_j) \pmod 2$ が一定であるとします．今，$(X_j + Y_j) \pmod 2 = 1$ と仮定します ($(X_j + Y_j) \pmod 2 = 0$ の場合は，長さ 1 の腕を追加することで，$(X_j + Y_j) \pmod 2 = 0$ の場合に帰着できます)．

座標 (x, y) に対して，$u = x + y, v = x - y$ とおいて (u, v) を考えます．すると，(u_i, v_i) と (u_{i+1}, v_{i+1}) の間に次の関係が成り立ちます：

- 腕 i のモードが `L` のとき，$(u_{i+1}, v_{i+1}) = (u_i - d_i, v_i - d_i)$
- 腕 i のモードが `R` のとき，$(u_{i+1}, v_{i+1}) = (u_i + d_i, v_i + d_i)$
- 腕 i のモードが `D` のとき，$(u_{i+1}, v_{i+1}) = (u_i - d_i, v_i + d_i)$
- 腕 i のモードが `U` のとき，$(u_{i+1}, v_{i+1}) = (u_i + d_i, v_i - d_i)$

よって，$\{d_i\}$ を決めたときに $\{u_i\}, \{v_i\}$ を決定する問題を u, v について独立に解けば，各関節のモードも決定できることになります．

この $\{u_i\}$ を決定する問題を考えます．$u_{i+1} = u_i + d_i$ なる $i \in \{1, \ldots, m\}$ の集合を S とすると，$u_m = -\sum_{i=1}^{m} d_i + \sum_{i \in S} 2d_i$ が成り立ちます．$(X_j + Y_j) \pmod 2 = 1$ の仮定より，U_j, V_j はすべて奇数です．また，$-2^{31} < -2 \cdot 10^9 \leq U_j, V_j \leq 2 \cdot 10^9 < 2^{31}$ が成り立っています．よって，$m = 31$, $d_i = 2^{i-1}$ とすると，$\sum_{i \in S} 2d_i = U_j + (2^{31} - 1)$ が成り立つように S を選ぶことができます（$\frac{U_j + (2^{31}-1)}{2}$ の二進展開を考えればよいです）．よって，この問題を解くことができました．

E: Tr/ee

s の i 文字目を s_i で表します。条件を満たす木が存在するためには、以下の条件が必要です。

- $s_1 = 1$
  葉につながっている辺を取るとサイズ 1 の連結成分ができるため
- $s_n = 0$
  どの辺をとっても非連結な頂点対ができるのだから、連結成分は n よりも真に小さくなるため
- $s_i = s_{n-i}$
  木から辺を 1 つとると連結成分はちょうど 2 こできるため。

逆に、以上の条件をすべてみたしている場合、条件を満たす木を構築することができます。直感的には、存在してはいけないサイズの部分木が存在しないように、うまく根付き木を構成していけばよいです。

s の n 文字目を 1 に書き換えた文字列を s' とします。頂点 n を根とした木で、次の条件を満たす木を構成します。

1. 辺 i は頂点 i とその親をつなぐ
2. $s'_i = 1$ であるとき、 頂点 i の部分木のサイズは i
3. $s'_i = 0$ であるとき、頂点 i は葉

このような木は、頂点 i の親を $i < j$ かつ $s'_j = 1$ となる最小の j とすると構成できます。これを確かめます。親となる頂点が $s'_i = 1$ であるような i に限られることから条件 3 が満たされることが分かります。また、$s'_j = 1$ かつ $i \leq j$ である場合、頂点 i から何度か親をたどることで頂点 j に到達することができることから、$1, 2, \ldots, j$ の j 個の頂点は頂点 j の部分木に含まれます。それ以外の頂点が部分木に含まれないことも明らかであるため、条件 2 も満たされます。

- C++ による解答例: `https://beta.atcoder.jp/contests/arc103/submissions/3288872`

F: Distance Sums

$S(v)$ で，頂点 v から他の点までの距離の和を表すことにします．木の中に辺 uv があるとき，辺 uv を除去したときの u, v を含む連結成分の頂点数をそれぞれ n_{uv}, n_{vu} と書くことにすると，$S(u) = S(v) + n_{vu} - n_{uv} = S(v) + N - 2n_{uv}$ が確かめられます．

一般性を失わず，$D_1 < D_2 < \cdots < D_N$ としてよいです．木を頂点 1 を根とする根付き木として考えます．ここで，頂点 1 から他の頂点までのパス $1, p_1, p_2, \ldots, p_k$ を考えます．まず，$S(1) < S(p_1)$ より，$n_{1p_1} > \frac{n}{2}$ です．また，$n_{1p_1} < n_{p_1p_2} < n_{p_2p_3} < \cdots < n_{p_{k-1}p_k}$ より，$n_{p_ip_{i+1}} > \frac{n}{2}$ であるから，$S(p_i) < S(p_{i+1})$ も成り立ちます．よって，頂点 1 以外の任意の頂点について，それより D の値が小さい頂点への辺がちょうど 1 本存在します．

頂点 N から順に，そのような辺を順次決定していくことを考えます．頂点 $i+1, \ldots, N$ までは確定したとして，頂点 i について考えます．頂点 i に隣接する，i より番号の小さい唯一の頂点を j とすると，$S(j) = S(i) + 2n_{ji} - N$ ですが，n_{ji} は頂点 i を根とする部分木の大きさに等しく，この値は計算できるので（辺を追加するときに順次更新すればよいです），$S(j)$ が決まります．D_i はすべて異なるという条件から，j を求めることができます．ここで，そのような j が存在しない（i より大きくなる場合を含む）ときは，条件を満たす木は存在しません．

頂点 2 まですべて決定できた場合は，一つの木ができます．この木について，実際に $S(i)$ の値を計算して（動的計画法などで可能です），D_i に一致していればその木が答えです．一致していない場合は，条件を満たす木は存在しません．

AtCoder Beginner Contest 111 / AtCoder Regular Contest 103 Editorial

writer: nuip and semiexp

September 29th, 2018

A: AtCoder Beginner Contest 999

Some example implementations:

```
#include<iostream>

using namespace std;

int main(){
	string n;
	cin >> n;
	for(char c:n){
		if(c == '1'){
			cout << 9;
		}else{
			cout << 1;
		}
	}
	return 0;
}
```

```
#include<iostream>

using namespace std;

int main(){
	string n;
	cin >> n;
	for(char c:n){
		cout << (char)('1' + '9' - c);
	}
	return 0;
}
```

```
#include<iostream>

using namespace std;

int main(){
	int n;
	cin >> n;
	cout << 111 + 999 - n << endl;
	return 0;
}
```

B: AtCoder Beginner Contest 111

Some example implementations:

- C++: `https://beta.atcoder.jp/contests/abc111/submissions/3279754`
- Java: `https://beta.atcoder.jp/contests/abc111/submissions/3279759`
- Python 3: `https://beta.atcoder.jp/contests/abc111/submissions/3279761`
- Ruby: `https://beta.atcoder.jp/contests/abc111/submissions/3279762`
- Rust: `https://beta.atcoder.jp/contests/abc111/submissions/3279784`
- Bash: `https://beta.atcoder.jp/contests/abc111/submissions/3279843`

C: /\/\/\/

Let E be the sequence $v_2, v_4, \ldots, v_n$, and let O be the sequence $v_1, v_3, \ldots, v_{n-1}$. We want to do the following (and minimize the number of changes):

1. Change all elements of E to x.
2. Change all elements of O to y.
3. $x \neq y$

If we ignore the third condition, the problem is easy: x should be the mode (the most frequent element) of E and y should be the mode of O. Thus, in case the mode of E and the mode of O are different, this is the optimal solution.

Otherwise, we should try two cases. Let e be the mode of E and e_2 be the second most frequent element of E. (Strictly speaking, e_2 is the most frequent element of E except for e. It's possible that e and e_2 appear the same number of times in E.) SImilarly, let o be the mode of O and o_2 be the second most frequent element of O. Then, the optimal solution is one of the following two:

- $x = e, y = o_2$
- $x = e_2, y = o$

We try both and choose the better one.

D: Robot Arms

For a fixed robot arm, the parity of $x_m + y_m$ is a constant (it must match the parity of the sum of d_i). Thus, in case $(X_j + Y_j) \pmod 2$ are not the same for all j, it is impossible to satisfy the conditions.

Assume that $X_j + Y_j$ is odd for all j. (In case they are all even, add a section of length 1 and we can reduce to the "all even" case.) It turns out that a robot arm with lengths $1, 2, 4, 8, \ldots$ can cover all such positions:

- If the robot arm has only one section and its length is 1, the set of positions where the last joint can go is $\{(1,0),(0,1),(-1,0),(0,-1)\}$.
- If the robot arm has two sections and their lengths are $\{1,2\}$, the set of positions where the last joint can go is $\{(x,y) \mid |x|+|y| \leq 3, x \equiv y \pmod 2\}$. In other words, it can cover all integer points (with correct parities) within a "rotated square" whose vertices are at $(3,0),(0,3),(-3,0),(0,-3)$.
- By induction, we can prove that, if the lengths of sections are $\{1,2,\ldots,2^k\}$, we can cover all vertices with correct parities within a "rotated square" whose vertices are at $(M,0),(0,M),(-M,0),(0,-M)$, where $M = 2^{k+1}-1$.

Thus, if we use $k = 30$, we can reach all given points.

To find a way to reach a given point, determine the mode of sections from the longest ones. For example, suppose that we know the position of (x_i, y_i). If we determine the mode of Section i, we can get the coordinates of (x_{i-1}, y_{i-1}). We should choose it such that (x_{i-1}, y_{i-1}) fits within the valid region (i.e., it should be reachable by lengths $\{1,2,\ldots,2^{i-2}\}$).

E: Tr/ee

Clearly, the following conditions are necessary:

- $s_1 = 1$ (If we cut a leaf we can get a component with size 1)
- $s_n = 0$ (If we cut an edge the sizes of components are less than n)
- $s_i = s_{n-i}$ (When we have a component of size i, the other component has size $n-i$)

These conditions are actually sufficient. Suppose that $0 < x_1 < x_2 < \cdots < x_k < n$ are the positions where s is one (i.e., $s_{x_i} = 1$). Then, the following caterpillar (https://en.wikipedia.org/wiki/Caterpillar_tree) satisfies the conditions:

1. The caterpillar has $k+1$ "body" vertices, labelled with $1,\ldots,k+1$. They form a path in this order.
2. For each i $(2 \leq i \leq k)$, the body vertex i has $x_i - x_{i-1} - 1$ "feets". (In other words, prepare $x_i - x_{i-1} - 1$ new vertices and attach them to vertex i.)

F: Distance Sums

Consider a vertex v with the largest value of D_v. Intuitively, v is a vertex that is far from majority of other vertices. We can prove that v is a leaf, and if w is the only vertex adjacent to v, $D_w = D_v - (n-2)$. We can determine one edge this way (vw), and by repeating similar process we can uniquely determine the entire tree. Let's formalize this.

Without loss of generality, we can assume that $D_1 < D_2 < \cdots < D_N$. Consider the tree as a rooted tree whose root is Vertex 1. In the decreasing order of i, we want to determine the parent of Vertex i.

Suppose that the tree contains an edge uv, and when the edge uv is removed from the tree, the sizes (number of vertices) of the two connected components are n_{uv} and n_{vu} (n_{uv} corresponds to the component with u). Then, $D_u = D_v + n_{vu} - n_{uv} = D_v + N - 2n_{uv}$ holds.

Consider a path $1, p_1, p_2, \ldots, p_k$. Since $D_1 < D_{p_1}$, $n_{1p_1} > \frac{n}{2}$. Also, since $n_{1p_1} < n_{p_1p_2} < n_{p_2p_3} < \cdots < n_{p_{k-1}p_k}$, $n_{p_ip_{i+1}} > \frac{n}{2}$ holds for all i and $D_{p_i} < D_{p_{i+1}}$ holds. Thus, if Vertex j is the parent of Vertex i, $j < i$ holds.

Suppose that we know the parents of Vertices $i+1, \ldots, N$. At this point, we know the subtree rooted at Vertex i. If Vertex j is the parent of Vertex i, as we mentioned above, $D_j = D_i + 2n_{ji} - N$ holds. Since n_{ji} is the size of the subtree rooted at i, we can uniquely determine j. (In case such j doens't exist, we get a contradiction.)

We repeat this process until we determine the parents of Vertices $2, \ldots, n$. Then, we have a tree, and check if this tree really satisfies the conditions. If yes, this is the desired tree (otherwise we get a contradiction).