ABC 067 / ARC 078 解説

writer : camypaper

2017 年 7 月 15 日

For international readers: English editorial starts from page 7.

A : Sharing Cookies

$A, B, A+B$ のいずれかが 3 で割り切れるかどうかを調べればよい．なお，答えが `Impossible` となるのは $A \equiv B \equiv 1 \bmod 3$ あるいは $A \equiv B \equiv 2 \bmod 3$ の場合のみである．

```
#include <iostream>
using namespace std;
int main(){
    int a,b;
    cin>>a>>b;
    if(a%3 == 0 || b%3 == 0 || (a+b)%3==0)
        cout<< "Possible" <<endl;
    else cout<< "Impossible" <<endl;
}
```

B : Snake Toy

N 個の棒が長さの昇順に並んでいる場合には，末尾から K 個の棒の長さの和が答えとなる．よって，初めに棒たちを長さの昇順に並び替えたのち，上述したような貪欲法を適用すればよい．

C++ などの多くのプログラミング言語では，ソート用の関数が予め用意されていることが多いため，ソート関数を実装する必要はないだろう．

```
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;
int main(){
    int n,k;
    cin>>n>>k;
    vector<int> l(n);
    for(int i=0;i<n;i++){
        cin>>l[i];
    }
    sort(l.begin(),l.end());
    int ans=0;
    for(int i=0;i<k;i++){
        ans+=l[n-1-i];
    }
    cout<<ans<<endl;
}
```

C : Splitting Pile

N 枚のカードに書かれた数の総和を X とする．カードの山の先頭 i 枚のカードに書かれた数の総和が x_i であったとすると，残ったカードたちに書かれた数の和 y_i は $X - x_i$ であり，$|y_i - x_i|$ は $|X - 2x_i|$ となる．

1 から $N - 1$ までの全ての i について $|X - 2x_i|$ を試せばよい．これは $O(N)$ で実行可能であり，十分高速である．

```
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;
int main(){
    int n;
    long long X=0,x=0,ans=1000000000000000000LL;
    cin>>n;
    vector<long long> a(n);
    for(int i=0;i<n;i++){
        cin>>a[i];
        X+=a[i];
    }
    for(int i=0;i<n;i++){
        x+=a[i];
        if(i+1<n)ans=min(ans,abs(X-2*x));
    }
    cout<<ans<<endl;
}
```

D : Fennec VS. Snuke

「マスに色を塗った回数の多いプレイヤーの勝ち，同数の場合は後手の勝ち」というルールのゲームを考えることにする．元のゲームでどちらかのプレイヤーが敗北したあとも色を塗ることが可能である，と考えると新しいゲームの勝敗と元のゲームの勝敗が一致することが分かる．このとき，2 人の目的は，自分が色を塗る回数を最大化し，相手が色を塗る回数を最小化することである．

与えられるグラフが木であることから，2 人の最善戦略は「マス 1 からマス N へのパス上に色が塗られていないマスが存在するならば色を塗る」であることが示される．マス 1 からマス N へのパス上に色が塗られていないマスが存在するにも関わらず，それ以外のマスに色を塗った場合，自分が不利になり相手が有利になることは直感的にも明らかであろう．例えば図 **??** に示されるような配置からゲームを開始したとき，上記の戦略に従ったときのみ先手は勝利可能である．従わなかった場合，後手が最適に行動すると 7 個以上のマスを白く塗ることが可能となり，先手が勝つことは不可能である．

上記の戦略に従うと，マス i と j の距離を $d(i,j)$ として，マス i の色は $d(1,i) \leq d(N,i)$ ならば黒，そうでなければ白となる．結論としてマス 1 とマス N の 2 点から幅優先探索や深さ優先探索などを行うことで $O(N)$ でこの問題を解くことが可能である．

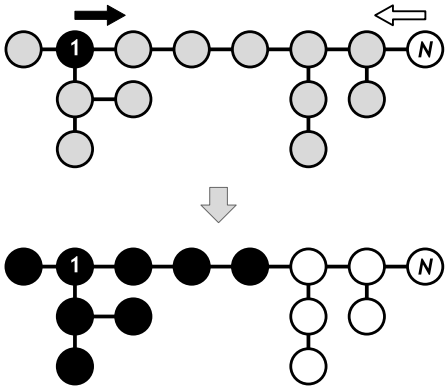

図 1　ゲームの初期状態と最終状態の例

E: Awkward Response

解の一例を説明する．

$q(n)$ を n についての質問の答えが `Yes` ならば 1 を，そうでなければ 0 を返す関数とする．N が L 桁の整数であったと仮定する．このとき，L 桁の整数 n と N の大小は，$\mathrm{str}(n)$ と $\mathrm{str}(N)$ の大小と一致する．ここで，$L+1$ 桁の整数 $10n$ と N の大小を考えたとき，$10n > N$ が必ず成立し，$\mathrm{str}(10n)$ と $\mathrm{str}(N)$ の大小は以下のように表される．

- $n < N$ のとき： $\mathrm{str}(10n) < \mathrm{str}(N)$
- $n \geq N$ のとき： $\mathrm{str}(10n) > \mathrm{str}(N)$

以上より，$q(10n)$ が 1 を返す最小の整数 n が N と一致することが分かる．さらに，$q(10n)$ には単調性が存在するので n について二分法を行うことで $\lg N$ 回程度の質問で N を求めることが可能である．

次に，N の桁数 L を求める方法を 1 つ説明する．

$q(10^{k-1})$ が式 (**??**) で表されることから，$N \neq 10^{L-1}$ の場合は，$q(10^{k-1}) = 1$ を満たす最大の k を求めることで L を求められる．

$$q(10^{k-1}) = \begin{cases} 1 & (k < L) \\ 1 & (k = L) \\ 0 & (k > L \text{ かつ } N \neq 10^{L-1}) \\ 1 & (k > L \text{ かつ } N = 10^{L-1}) \end{cases} \tag{1}$$

$N = 10^{L-1}$ の場合は，$q(2 \times 10^{k-1})$ が式 (**??**) のように表されることから，$q(2 \times 10^{k-1}) = 1$ を満たす最小の k を求めることで，L を求められる．

$$q(2 \times 10^{k-1}) = \begin{cases} 0 & (k < L) \\ 1 & (k = L) \\ 1 & (k > L) \end{cases} \tag{2}$$

解法をまとめると，以下のように表される．

1. $N = 10^{L-1}$ の場合に注意して，N の桁数 L を求める．
2. 二分法を用いて $q(10n) = 1$ なる最小の n を求める．

どのような場合でも質問回数は $\max(2L, L + \lg N)$ 回以下となり，64 回という制限に対しては十分余裕がある．

F : Mole and Abandoned Mine

はじめに，問題設定を以下のように言い換えておく．言い換えた問題設定での答えを x として，$\sum_{i=1}^{M} c_i - x$ が元の問題の答えである．

> N 頂点の辺のないグラフが与えられる．いくつか辺を追加して，頂点 1 から頂点 N への点素パスがただ 1 つ存在するようにしたい．
> 辺を追加する方法は M 個あり，頂点 a_i, b_i 間に辺を追加すると c_i 利得を得る．得られる利得の総和を最大化せよ．

頂点 1 から頂点 N への点素パスがただ 1 つ存在するための条件は，「頂点 1 から頂点 N へのパス上にある辺がいずれも橋である」ことである．頂点 1 から頂点 N へのパスがただ 1 つ存在し，そのパス上に，橋でない辺が存在したとする．このとき，そのような辺を取り除いてもグラフは連結であるから，頂点 1 から頂点 N へのパスは他にも存在するはずである．これは点素パスがただ 1 つ存在することに矛盾する．

ここで，頂点 1 から頂点 N への点素パスがただ 1 つ存在するグラフにおいて，頂点 1 から N へのパス上 (両端を含む) にある頂点をターミナルと呼ぶことにする．最適解においては，ターミナル同士をつなぐ辺を取り除いたとき，各連結成分内においては可能な限り辺が追加されているはずである．そのため，$O(2^N \times M)$ かけて事前に計算をしておけば，各連結成分内での利得の総和は容易に計算可能である．

さて，パスを 1 つ固定して，その他の頂点たちがどのターミナルに属するかを全探索することを考えよう．このままでは，実行時間制限には間に合わない．そこで，頂点 1 から N までのターミナルを順番に決めていきながら，現在着目しているターミナルに属する頂点たちを全て試していくことを考える．S を $\{1, 2, 3, \ldots, N\}$ の部分集合として，$dp(S, t)$ を「S に含まれる頂点たちを使用しており，t が現在着目しているターミナルであるときの，得られる利得の総和の最大値」とする．遷移としては，次のターミナルに接続するか，現在のターミナルに属する頂点集合を試すか，の 2 つの遷移を試せばよい．適切に実装を行うことで，全体として $O(3^N \times N)$ で実行可能である．

ABC 067 / ARC 078 Editorial

writer : camypaper

July 15th, 2017

A, B, C

See the sample codes of Japanese editorial above.

D : Fennec VS. Snuke

Let $v_1 = 1, v_2, \ldots, v_{k-1}, v_k = N$ be the only path from cell 1 to cell N.

Let $d(i, j)$ be the distance between cell i and cell j.

In the optimal strategy, Black should color the vertices in the order $v_2, v_3, \ldots$ (while this is possible), and after that Black can choose arbitrary valid moves. This way, Black can make sure that she can get all vertices v such that $d(1, v) \leq d(N, v)$, no matter how White plays.

On the other hand, White should color the vertices in the order $v_{k_1}, v_{k-2}, \ldots$ (while this is possible), and after that White can choose arbitrary valid moves. This way, White can make sure that he can get all vertices v such that $d(1, v) > d(N, v)$, no matter how Black plays.

To summarize, we can solve this task in $O(N)$ by computing distances from two cells 1 and N.

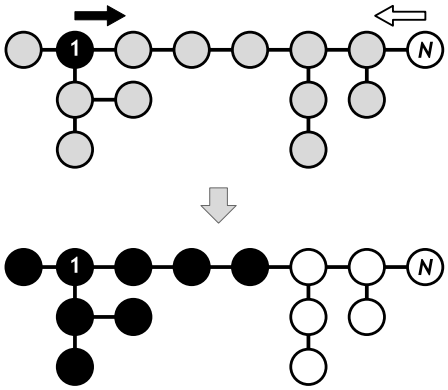

図 1　An example of a game

E: Awkward Response

First, we should query with 10^9. We get 'Y' iff $N = 1, 10, \ldots, 10^9$.
Next, we want to compute the number of digits in N. Let's call it L.
Let $q(n)$ denote the result of the query: 1 means `Yes`, 0 means `No`.
In case $N = 10^{L-1}$, we use the following fact:

$$q(2 \times 10^{k-1}) = \begin{cases} 0 & (k < L) \\ 1 & (k = L) \\ 1 & (k > L) \end{cases} \tag{1}$$

Thus, L is the minimum k that satisfies $q(2 \times 10^{k-1}) = 1$.
In case $N \neq 10^{L-1}$, we use the following fact:

$$q(10^{k-1}) = \begin{cases} 1 & (k < L) \\ 1 & (k = L) \\ 0 & (k > L \mathrm{and} N \neq 10^{L-1}) \\ 1 & (k > L \mathrm{and} N = 10^{L-1}) \end{cases} \tag{2}$$

Thus, L is the maximum k that satisfies $q(10^{k-1}) = 1$.
Next, for a given L-digit number n, we can check if $N < n$ by querying $10n$.

- If $n < N$, $q(10n) = 0$.
- If $n \geq N$, $q(10n) = 1$.

Thus, we can compute N in $\lg N$ queries by binary search.
The number of queries is about $\max(2L, L + \lg N)$.

F : Mole and Abandoned Mine

Suppose that we have a graph that has exactly one path from vertex 1 to vertex N. In this graph, all edges on the unique path from 1 to N must be bridges. (Otherwise, we can find a path from 1 to N even if we remove one of those edges, and this is a contradiction).

Let S be a subset of vertices, and t be a vertex in S. Define $dp(S, t)$ as the minimum cost required to achieve the condition "There is a unique path from 1 to t that only uses vertices in S". We have two transitions:

(Here, $cost(S, T)$ denotes the sum of costs of all edges between two sets S and T.)

- Let $u \notin S$ be a vertex. If there is an edge between t and u, we can reach $dp(S \cup \{u\}, u)$ with cost $dp(S, t) + cost(S \setminus \{t\}, \{u\})$.
- Let T be a set of vertices that is disjoint with S. We can reach $dp(S \cup T, t)$ with cost $(dp(S, t) + cost(S \setminus \{t\}, T)$.

With proper pre-computation, we can compute $cost(S, T)$ in $O(1)$ and this solution works in $O(3^N \times N)$.